Internet Engineering Task Force (IETF)
Request for Comments: 5946
Updates: 2205
Category: Standards Track
ISSN: 2070-1721

F. Le Faucheur
Cisco
J. Manner
Aalto University
A. Narayanan
Cisco
A. Guillou
SFR
H. Malik
Airtel

October 2010

Resource Reservation Protocol (RSVP) Extensions for Path-Triggered RSVP Receiver Proxy

Abstract

Resource Reservation Protocol (RSVP) signaling can be used to make end-to-end resource reservations in an IP network in order to guarantee the Quality of Service (QoS) required by certain flows. With conventional RSVP, both the data sender and receiver of a given flow take part in RSVP signaling. Yet, there are many use cases where resource reservation is required, but the receiver, the sender, or both, is not RSVP-capable. Where the receiver is not RSVP-capable, an RSVP router may behave as an RSVP Receiver Proxy, thereby performing RSVP signaling on behalf of the receiver. This allows resource reservations to be established on the segment of the end-to-end path from the sender to the RSVP Receiver Proxy. However, as discussed in the companion document "RSVP Proxy Approaches", RSVP extensions are needed to facilitate operations with an RSVP Receiver Proxy whose signaling is triggered by receipt of RSVP Path messages from the sender. This document specifies these extensions.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5946.

Table of Contents

1. Introduction

Guaranteed Quality of Service (QoS) for some applications with tight QoS requirements may be achieved by reserving resources in each node on the end-to-end path. The main IETF protocol for these resource reservations is the Resource Reservation Protocol (RSVP), as specified in [RFC2205]. RSVP does not require that all intermediate nodes support RSVP, but it assumes that both the sender and the receiver of the data flow support RSVP. However, there are environments where it would be useful to be able to reserve resources for a flow (at least a subset of the flow path) even when the sender or the receiver (or both) is not RSVP-capable.

Since both the data sender and receiver may be unaware of RSVP, there are two types of RSVP Proxies. In the first case, an entity in the network needs to invoke RSVP on behalf of the data sender and thus generate RSVP Path messages, and eventually receive, process, and sink Resv messages. We refer to this entity as the RSVP Sender Proxy. In the second case, an entity in the network needs to operate RSVP on behalf of the receiver and thus receive Path messages sent by a data sender (or by an RSVP Sender Proxy), and reply to those with Resv messages generated on behalf of the data receiver(s). We refer to this entity as the RSVP Receiver Proxy.

RSVP Proxy approaches are presented in [RFC5945]. That document also discusses, for each approach, how the reservations controlled by the RSVP Proxy can be synchronized with the application requirements (e.g., when to establish, maintain, and tear down the RSVP reservation to satisfy application requirements).

One RSVP Proxy approach is referred to as the Path-Triggered RSVP Receiver Proxy approach. With this approach, the RSVP Receiver Proxy uses the RSVP Path messages generated by the sender (or RSVP Sender Proxy) as the cue for establishing the RSVP reservation on behalf of the non-RSVP-capable receiver(s). The RSVP Receiver Proxy is effectively acting as an intermediary making reservations (on behalf of the receiver) under the sender's control (or RSVP Sender Proxy's control). This somewhat changes the usual RSVP reservation model where reservations are normally controlled by receivers. Such a change greatly facilitates operations in the scenario of interest here, which is where the receiver is not RSVP-capable. Indeed it allows the RSVP Receiver Proxy to remain application-unaware by taking advantage of the application awareness and RSVP awareness of the sender (or RSVP Sender Proxy).

Since the synchronization between an RSVP reservation and an application is now effectively performed by the sender (or RSVP Sender Proxy), it is important that the sender (or RSVP Sender Proxy)

is aware of the reservation state. However, as conventional RSVP assumes that the reservation is to be controlled by the receiver, some notifications about reservation state (notably the error message sent in the case of admission control rejection of the reservation) are only sent towards the receiver and therefore, in our case, sunk by the RSVP Receiver Proxy. Section 3 of this document specifies extensions to RSVP procedures allowing such notifications to be also conveyed towards the sender. This facilitates synchronization by the sender (or RSVP Sender Proxy) between the RSVP reservation and the application requirements, and it facilitates sender-driven control of reservation in scenarios involving a Path-Triggered RSVP Receiver Proxy.

With unicast applications in the presence of RSVP Receiver Proxies, if the sender is notified about the state of the reservation towards the receiver (as enabled by this document), the sender is generally in a good position to synchronize the reservation with the application and to perform efficient sender-driven reservation: the sender can control the establishment or removal of the reservation towards the receiver by sending Path or PathTear messages, respectively. For example, if the sender is notified that the reservation for a point-to-point audio session towards the receiver is rejected, the sender may trigger rejection of the session at the application layer and may issue a PathTear message to remove any corresponding RSVP state (e.g., Path states) previously established.

However, we note that multicast applications do not always coexist well with RSVP Receiver Proxies, since sender notification about reservation state towards each RSVP Receiver Proxy may not be sufficient to achieve tight application-level synchronization by multicast senders. These limitations stem from the fact that multicast operation is receiver driven and, while end-to-end RSVP is also receiver driven (precisely to deal with multicast efficiently), the use of RSVP Receiver Proxies only allows sender-driven reservation. For example, a sender generally is not aware of which receivers have joined downstream of a given RSVP Receiver Proxy, or even which RSVP Receiver Proxies have joined downstream of a given failure point. Therefore, it may not be possible to support a mode of operation whereby a given receiver only joins a group if that receiver benefits from a reservation. Additionally, a sender may have no recourse if only a subset of RSVP Receiver Proxies return successful reservations (even if application-level signaling runs between the sender and receivers), since the sender may not be able to correctly identify the set of receivers who do not have reservations. However, it is possible to support a mode of operation whereby multicast traffic is transmitted if and only if all receivers benefit from a reservation (from sender to their respective RSVP Receiver Proxy): the sender can ensure this by sending a PathTear

message and stopping transmission whenever it gets a notification for reservation reject for one or more RSVP Receiver Proxies. It is also possible to support a mode of operation whereby receivers join independently of whether or not they can benefit from a reservation (to their respective RSVP Receiver Proxy), but do benefit from a reservation whenever the corresponding resources are reservable on the relevant path.

This document discusses extensions to facilitate operations in the presence of a Path-Triggered RSVP Receiver Proxy. As pointed out previously, those apply equally whether RSVP signaling is initiated by a regular RSVP sender or by an RSVP Sender Proxy (with some means to synchronize reservation state with application-level requirements that are outside the scope of this document). For readability, the rest of this document discusses operations assuming a regular RSVP sender; however, such an operation is equally applicable where an RSVP Sender Proxy is used to initiated RSVP signaling on behalf of a non-RSVP-capable sender.

As discussed in [RFC5945], it is important to keep in mind that the strongly recommended RSVP deployment model remains end to end as assumed in [RFC2205] with RSVP support on the sender and the receiver. The end-to-end model allows the most effective synchronization between the reservation and application requirements. Also, when compared to the end-to-end RSVP model, the use of RSVP Proxies involves additional operational burden and/or imposes some topological constraints. Thus, the purpose of this document is only to allow RSVP deployment in special environments where RSVP just cannot be used on some senders and/or some receivers for reasons specific to the environment.

Section 4.1.1 of [RFC5945] discusses mechanisms allowing the RSVP reservation for a given flow to be dynamically extended downstream of an RSVP Proxy whenever possible (i.e., when the receiver is RSVP-capable or when there is another RSVP Receiver Proxy downstream). This can considerably alleviate the operational burden and the topological constraints associated with Path-Triggered RSVP Receiver Proxies. This allows (without corresponding manual configuration) an RSVP reservation to dynamically span as much of the corresponding flow path as possible, with any arbitrary number of RSVP Receiver Proxies on the flow path and whether or not the receiver is RSVP-capable. In turn, this facilitates migration from an RSVP deployment model based on Path-Triggered Receiver Proxies to an end-to-end RSVP model, since receivers can gradually and independently be upgraded to support RSVP and then instantaneously benefit from end-to-end reservations. Section 4 of this document specifies these mechanisms and associated RSVP extensions.

1.1. Conventions Used in This Document

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 [RFC2119].

2. Terminology

The following terminology is borrowed from [RFC5945] and is used extensively in this document:

- RSVP-capable (or RSVP-aware): supporting the RSVP protocol as per [RFC2205].

- RSVP Receiver Proxy: an RSVP-capable router performing, on behalf of a receiver, the RSVP operations that would normally be performed by an RSVP-capable receiver if end-to-end RSVP signaling were used. Note that while RSVP is used upstream of the RSVP Receiver Proxy, RSVP is not used downstream of the RSVP Receiver Proxy.

- RSVP Sender Proxy: an RSVP-capable router performing, on behalf of a sender, the RSVP operations that normally would be performed by an RSVP-capable sender if end-to-end RSVP signaling were used. Note that while RSVP is used downstream of the RSVP Sender Proxy, RSVP is not used upstream of the RSVP Sender Proxy.

- Regular RSVP Router: an RSVP-capable router that is not behaving as an RSVP Receiver Proxy nor as an RSVP Sender Proxy.

Note that the roles of the RSVP Receiver Proxy, RSVP Sender Proxy, and regular RSVP Router are all relative to one unidirectional flow. A given router may act as the RSVP Receiver Proxy for a flow, as the RSVP Sender Proxy for another flow, and as a regular RSVP router for yet another flow.

The following terminology is also used in this document:

- Regular RSVP sender: an RSVP-capable host behaving as the sender for the considered flow and participating in RSVP signaling in accordance with the sender behavior specified in [RFC2205].

- Regular RSVP receiver: an RSVP-capable host behaving as the receiver for the considered flow and participating in RSVP signaling in accordance with the receiver behavior specified in [RFC2205].

3. RSVP Extensions for Sender Notification

This section defines extensions to RSVP procedures allowing sender notification of reservation failure. This facilitates synchronization by the sender between RSVP reservation and application requirements in scenarios involving a Path-Triggered RSVP Receiver Proxy.

As discussed in [RFC5945], with the Path-Triggered RSVP Receiver Proxy approach, the RSVP router may be configured to use receipt of a regular RSVP Path message as the trigger for RSVP Receiver Proxy behavior. On receipt of the RSVP Path message, the RSVP Receiver Proxy:

1. establishes the RSVP Path state as per regular RSVP processing.

2. identifies the downstream interface towards the receiver.

3. sinks the Path message.

4. behaves as if a corresponding Resv message (on its way upstream from the receiver) was received on the downstream interface. This includes performing admission control on the downstream interface, establishing a Resv state (in the case of successful admission control), and forwarding the Resv message upstream, sending periodic refreshes of the Resv message and tearing down the reservation if the Path state is torn down.

Operation of the Path-Triggered Receiver Proxy in the case of a successful reservation is illustrated in Figure 1.

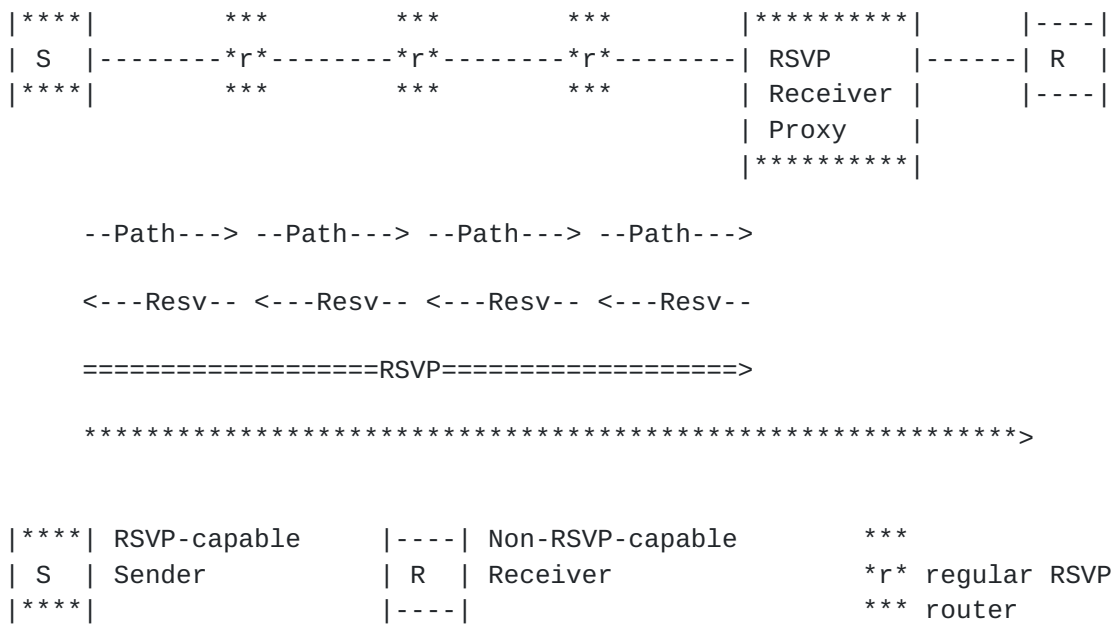


Figure 1: Successful Reservation

We observe that, in the case of successful reservation, conventional RSVP procedures ensure that the sender is notified of the successful reservation establishment. Thus, no extensions are required in the presence of a Path-Triggered RSVP Receiver Proxy in the case of successful reservation establishment.

However, in the case of reservation failure, conventional RSVP procedures ensure only that the receiver (or the RSVP Receiver Proxy) is notified of the reservation failure. Specifically, in the case of an admission control rejection on a regular RSVP router, a ResvErr message is sent downstream towards the receiver. In the presence of an RSVP Receiver Proxy, if we simply follow conventional RSVP procedures, this means that the RSVP Receiver Proxy is notified of the reservation failure, but the sender is not. Operation of the Path-Triggered RSVP Receiver Proxy in the case of an admission control failure, assuming conventional RSVP procedures, is illustrated in Figure 2.

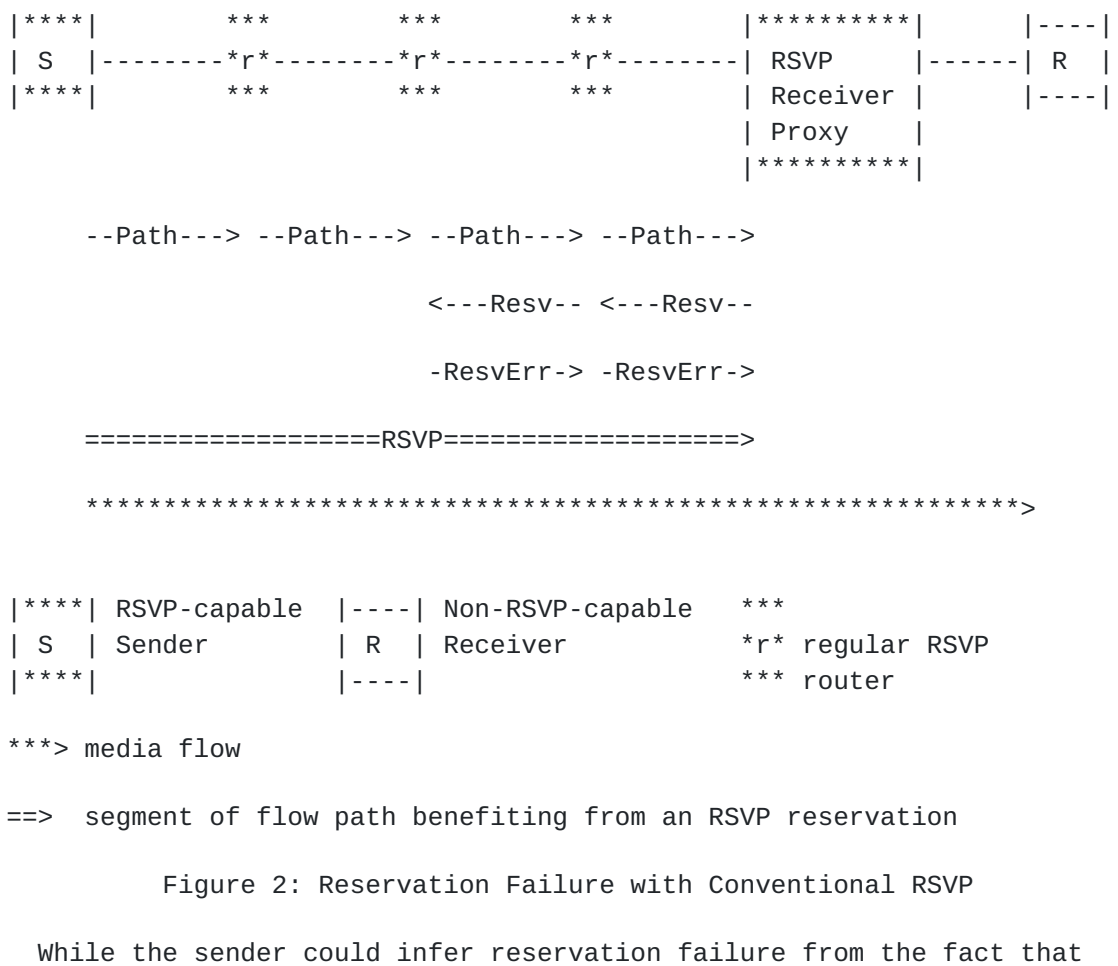


Figure 2: Reservation Failure with Conventional RSVP

While the sender could infer reservation failure from the fact that it has not received a Resv message after a certain time, there are clear benefits to ensuring that the sender gets a prompt, explicit notification in the case of reservation failure. This includes faster end-user notification at the application layer (e.g., busy signal) and faster application-level reaction (e.g., application-level rerouting), as well as faster release of application-level resources.

Section 3.1 defines a method that can be used to achieve sender notification of reservation failure. A router implementation claiming compliance with this document MUST support the method defined in Section 3.1.

Section 3.2 defines another method that can be used to achieve sender notification of reservation failure. A router implementation claiming compliance with this document MAY support the method defined in Section 3.2.

In a given network environment, a network administrator may elect to use the method defined in Section 3.1, the method defined in Section 3.2, or possibly combine the two.

3.1. Sender Notification via PathErr Message

With this method, the RSVP Receiver Proxy MUST generate a PathErr message whenever the two following conditions are met:

1. The reservation establishment has failed (or the previously established reservation has been torn down).

2. The RSVP Receiver Proxy determines that it cannot re-establish the reservation (e.g., by adapting its reservation request in reaction to the error code provided in the received ResvErr in accordance with local policy).

Note that this notion of generating a PathErr message upstream in order to notify the sender about a reservation failure is not completely new. It is borrowed from [RFC3209] where it was introduced in order to satisfy a similar requirement, which is to allow an MPLS Traffic Engineering (TE) Label Switching Router to notify the TE Tunnel head-end (i.e., the sender) of a failure to establish (or maintain) a TE Tunnel Label Switch Path.

Operation of the Path-Triggered RSVP Receiver Proxy in the case of an admission control failure, using sender notification via a PathErr message, is illustrated in Figure 3.

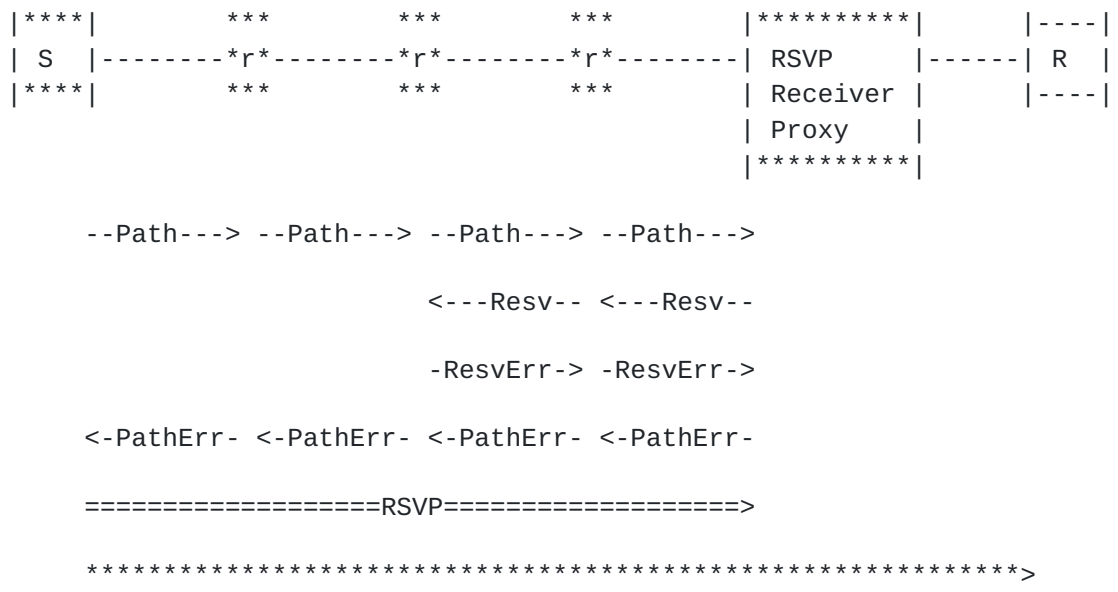


Figure 3: Reservation Failure with Sender Notification via PathErr

The role of this PathErr is to notify the sender that the reservation was not established or was torn down. This may be in the case of receipt of a ResvErr, or because of local failure on the Receiver Proxy. On receipt of a ResvErr, in all situations where the reservation cannot be installed, the Receiver Proxy MUST generate a PathErr towards the sender. For local failures on the Receiver Proxy node, if a similar failure on an RSVP midpoint would cause the generation of a ResvErr (for example, admission control failure), the Receiver Proxy MUST generate a PathErr towards the sender. The Receiver Proxy MAY additionally generate a PathErr upon local failures that would not ordinarily cause generation of a ResvErr message, such as those described in Appendix B of [RFC2205].

The PathErr generated by the Receiver Proxy corresponds to the sender(s) that triggered generation of Resv messages that failed. For FF-style (Fixed-Filter) reservations, the Receiver Proxy MUST send a PathErr towards the (single) sender matching the failed reservation. For SE-style (Shared-Explicit) reservations, the

Receiver Proxy MUST send the PathErr(s) towards the set of senders that triggered reservations that failed. This may be a subset of senders sharing the same reservation, in which case the remaining senders would have their reservation intact and would not receive a PathErr. In both cases, the rules described in Section 3.1.8 of [RFC2205] for generating flow descriptors in ResvErr messages also apply when generating sender descriptors in PathErr messages.

For WF-style (Wildcard-Filter) reservations, it is not always possible for the Receiver Proxy to reliably know which sender caused the reservation failure. Therefore, the Receiver Proxy SHOULD send a PathErr towards each sender. This means that all the senders will receive a notification that the reservation is not established, including senders that did not cause the reservation failure. Therefore, the method of sender notification via a PathErr message is somewhat overly conservative (i.e., in some cases, rejecting reservations from some senders when those could have actually been established) when used in combination with the Wildcard-Filter style (and when there is more than one sender).

The sender notification via the PathErr method applies to both unicast and multicast sessions. However, for a multicast session, it is possible that reservation failure (e.g., admission control failure) in a node close to a sender may cause ResvErr messages to be sent to a large group of Receiver Proxies. These Receiver Proxies would, in turn, all send PathErr messages back to the same sender, which could cause a scalability issue in some environments.

From the perspective of the sender, errors that prevent a reservation from being set up can be classified in two ways:

1. Errors that the sender can attempt to correct. The error code for these errors should explicitly be communicated back to the sender. An example of this is "Code 1: Admission Control Failure", because the sender could potentially resend a Path message with smaller traffic parameters.

2. Errors over which the sender has no control. For these errors, it is sufficient to notify the sender that the reservation was not set up successfully. An example of this is "Code 13: Unknown Object", because the sender has no control over the objects inserted into the reservation by the Receiver Proxy.

The PathErr message generated by the Receiver Proxy has the same format as regular PathErr messages defined in [RFC2205]. The SESSION, ERROR_SPEC, and sender descriptor are composed by the

Receiver Proxy as specified in the following subsections. The Receiver Proxy MAY reflect back towards the sender in the PathErr any POLICY_DATA objects received in the ResvErr.

3.1.1. Composition of SESSION and Sender Descriptor

The Receiver Proxy MUST insert the SESSION object corresponding to the failed reservation into the PathErr. For FF-style reservations, the Receiver Proxy MUST insert a sender descriptor corresponding to the failed reservation into the PathErr. This is equal to the error flow descriptor in the ResvErr received by the Receiver Proxy. For SE-style reservations, the Receiver Proxy MUST insert a sender descriptor corresponding to the sender triggering the failed reservation into the PathErr. This is equal to the error flow descriptor in the ResvErr received by the Receiver Proxy. If multiple flow descriptors could not be admitted at a midpoint node, that node would generate multiple ResvErr messages towards the receiver as per Section 3.1.8 of [RFC2205]. Each ResvErr would contain an error flow descriptor that matches a specific sender. The Receiver Proxy MUST generate a PathErr for each ResvErr received towards the corresponding sender. As specified earlier, for WF-style reservations, the Receiver Proxy SHOULD send a PathErr to each sender.

3.1.2. Composition of ERROR_SPEC

The Receiver Proxy MUST compose the ERROR_SPEC to be inserted into the PathErr as follows:

1. If the Receiver Proxy receives a ResvErr with either of these error codes -- "Code 1: Admission Control Failure" or "Code 2: Policy Control Failure" -- then the Receiver Proxy copies the error code and value from the ERROR_SPEC in the ResvErr into the ERROR_SPEC of the PathErr message. The error node in the PathErr MUST be set to the address of the Receiver Proxy. This procedure MUST also be followed for a local error on the Receiver Proxy that would ordinarily cause a midpoint to generate a ResvErr with one of the above codes.

2. If the Receiver Proxy receives a ResvErr with any error code except the ones listed in item 1 above, it composes a new ERROR_SPEC with error code "36: Unrecoverable Receiver Proxy Error". The error node address in the PathErr MUST be set to the address of the Receiver Proxy. This procedure MUST also be followed for a local error on the Receiver Proxy that would ordinarily cause a midpoint to generate a ResvErr with any error code other than those listed in item 1 above, or if the Receiver Proxy generates a PathErr for a local error that ordinarily would

not cause generation of a ResvErr. In some cases, it may be predetermined that the PathErr will not reach the sender. For example, a node receiving a ResvErr with "Code 3: No Path for Resv", knows a priori that the PathErr message it generates cannot be forwarded by the same node that could not process the Resv. Nevertheless, the procedures above MUST be followed. For the error code "36: Unrecoverable Receiver Proxy Error", the 16 bits of the Error Value field are:

* hhhh hhhh llll llll

where the bits are:

* hhhh hhhh = 0000 0000: then the low order 8 bits (llll llll) MUST be set by Receiver Proxy to 0000 0000 and MUST be ignored by the sender.

* hhhh hhhh = 0000 0001: then the low order 8 bits (llll llll) MUST be set by the Receiver Proxy to the value of the error code received in the ResvErr ERROR_SPEC (or, in case the Receiver Proxy generated the PathErr without having received a ResvErr, to the error code value that would have been included by the Receiver Proxy in the ERROR_SPEC in similar conditions if it was to generate a ResvErr). This error value MAY be used by the sender to further interpret the reason for the reservation failure.

* hhhh hhhh = any other value: reserved.

3. If the Receiver Proxy receives a ResvErr with the InPlace flag set in the ERROR_SPEC, it MUST also set the InPlace flag in the ERROR_SPEC of the PathErr.

3.1.3. Use of Path_State_Removed Flag

[RFC3473] defines an optional behavior whereby a node forwarding a PathErr message can remove the Path state associated with the PathErr message and indicate so by including the Path_State_Removed flag in the ERROR_SPEC object of the PathErr message. This can be used in some situations to expedite release of resources and minimize signaling load.

This section discusses aspects of the use of the Path_State_Removed flag that are specific to the RSVP Receiver Proxy. In any other aspects, the Path_State_Removed flag operates as per [RFC3473].

By default, the RSVP Receiver Proxy MUST NOT include the Path_State_Removed flag in the ERROR_SPEC of the PathErr message. This ensures predictable operations in all environments including those where some RSVP routers do not understand the Path_State_Removed flag.

The RSVP Receiver Proxy MAY support an OPTIONAL mode (to be activated by configuration) whereby the RSVP Receiver Proxy includes the Path_State_Removed flag in the ERROR_SPEC of the PathErr message and removes its local Path state. When all routers on the path of a reservation support the Path_State_Removed flag, its use will indeed result in expedited resource release and reduced signaling. However, if there are one or more RSVP routers on the path of the reservation that do not support the Path_State_Removed flag (we refer to such routers as "old RSVP routers"), the use of the Path_State_Removed flag will actually result in slower resource release and increased signaling. This is because the Path_State_Removed flag will be propagated upstream by an old RSVP router (even if it does not understand it and does not tear its Path state). Thus, the sender will not send a Path Tear, and the old RSVP router will release its Path state only through refresh time-out. A network administrator needs to keep these considerations in mind when deciding whether to activate the use of the Path_State_Removed flag on the RSVP Receiver Proxy. In a controlled environment where all routers are known to support the Path_State_Removed flag, its use can be safely activated on the RSVP Receiver Proxy. In other environments, the network administrator needs to assess whether the improvement achieved with some reservations outweighs the degradation experienced by other reservations.

3.1.4. Use of PathErr by Regular Receivers

Note that while this document specifies that an RSVP Receiver Proxy generates a PathErr upstream in the case of reservation failure, this document does NOT propose that the same be done by regular receivers. In other words, this document does NOT propose modifying the behavior of regular receivers as currently specified in [RFC2205]. The rationale for this includes the following:

- When the receiver is RSVP-capable, the current receiver-driven model of [RFC2205] is fully applicable because the receiver can synchronize RSVP reservation state and application state (since it participates in both). The sender(s) need not be aware of the RSVP reservation state. Thus, we can retain the benefits of receiver-driven operations that were explicitly sought by [RFC2205], which states, "In order to efficiently accommodate large groups, dynamic group membership, and heterogeneous receiver requirements, RSVP makes receivers responsible for requesting a

specific QoS". But even for the simplest single_sender/ single_receiver reservations, the current receiver-driven model reduces signaling load and per-hop RSVP processing by not sending any error message to the sender in case of admission control reject.

- The motivation for adding sender error notification in the case of an RSVP Receiver Proxy lies in the fact that the actual receiver can no longer synchronize the RSVP reservation with application state (since the receiver does not participate in RSVP signaling), while the sender can. This motivation does not apply in the case of a regular receiver.

- There is a lot of existing code and deployed systems successfully working under the current [RFC2205] model in the absence of proxy today. The current behavior of existing deployed systems should not be changed unless there were a very strong motivation.

3.2. Sender Notification via Notify Message

The OPTIONAL method for sender notification of reservation failure defined in this section aims to provide a more efficient method than the one defined in Section 3.1. Its objectives include:

- allowing the failure notification to be sent directly upstream to the sender by the router where the failure occurs (as opposed to first traveling downstream towards the Receiver Proxy and then traveling upstream from the Receiver Proxy to the sender, as effectively happens with the method defined in Section 3.1).

- allowing the failure notification to travel without hop-by-hop RSVP processing.

- ensuring that such a notification is sent to senders that are capable and willing to process it (i.e., to synchronize reservation status with application).

- ensuring that such a notification is only sent in case the receiver is not itself capable and willing to do the synchronization with the application (i.e., because we are in the presence of a Receiver Proxy so that RSVP signaling is not visible to the receiver).

Note, however, that such benefits come at the cost of:

- a requirement for RSVP routers and senders to support the Notify messages and procedures defined in [RFC3473].

- a requirement for senders to process Notify messages traveling upstream but conveying a downstream notification.

[RFC3473] defines (in Section 4.3, "Notify Messages") the Notify message that provides a mechanism to inform non-adjacent nodes of events related to the RSVP reservation. The Notify message differs from the error messages defined in [RFC2205] (i.e., PathErr and ResvErr messages) in that it can be "targeted" to a node other than the immediate upstream or downstream neighbor and that it is a generalized notification mechanism. Notify messages are normally generated only after a Notify Request object has been received.

This section discusses aspects of the use of the Notify message that are specific to the RSVP Receiver Proxy. In any other aspects, the Notify message operates as per [RFC3473].

In order to achieve sender notification of reservation failure in the context of this document:

- An RSVP sender interested in being notified of reservation failure MUST include a Notify Request object (containing the sender's IP address) in the Path messages it generates.

- Upon receiving a Path message with a Notify Request object, the RSVP Receiver Proxy MUST include a Notify Request object in the Resv messages it generates. This Notify Request object MUST contain either:

  * the address that was included in the Notify Request of the received Path message, a.k.a. the sender's address. We refer to this approach as the "Direct Notify" approach, or

  * an address of the Receiver Proxy. We refer to this approach as the "Indirect Notify" approach.

- Upon receiving a downstream error notification (whether in the form of a Notify, ResvErr, or both), the RSVP Receiver Proxy:

  * MUST generate a Notify message with upstream notification to the corresponding sender, if the sender included a Notify Request object in its Path messages and if Indirect Notification is used.

  * SHOULD generate a Notify message with upstream notification to the corresponding sender, if the sender included a Notify Request object in its Path messages and if Direct Notification is used. The reason for this recommendation is that the failure node may not support Notify, so that even if Direct

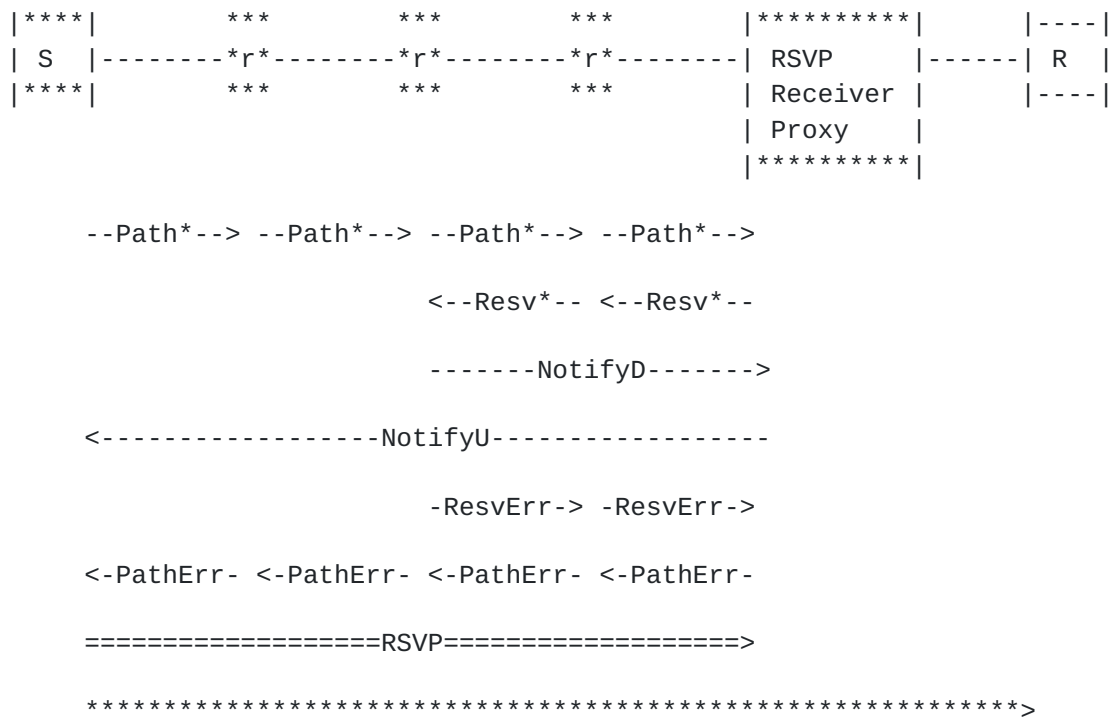
Notification was requested by the RSVP Receiver Proxy, the sender may not actually have received a Notify from the failure node: generating a Notify from the Receiver Proxy will accelerate sender notification, as compared to simply relying on PathErr, in this situation. In controlled environments where all the nodes are known to support Notify, the Receiver Proxy MAY be configured to not generate the Notify with upstream notification when Direct Notification is used, in order to avoid duplication of Notify messages (i.e., the sender receiving both a Notify from the failure node and from the Receiver Proxy).

As a result of these sender and Receiver Proxy behaviors, as per existing Notify procedures, if an RSVP router detects an error relating to a Resv state (e.g., admission control rejection after IP reroute), the RSVP router will send a Notify message (conveying the downstream notification with the ResvErr error code) to the IP address contained in the Resv Notify Request object. If this address has been set by the RSVP Receiver Proxy to the sender's address (Direct Notify), the Notify message is sent directly to the sender. If this address has been set by the RSVP Receiver Proxy to one of its own addresses (Indirect Notify), the Notify message is sent to the RSVP Receiver Proxy that, in turn, will generate a Notify message directly addressed to the sender.

Operation of the Path-Triggered RSVP Receiver Proxy in the case of an admission control failure, using sender notification via Direct Notify, is illustrated in Figure 4.

Figure 4: Reservation Failure with Sender Notification via Direct Notify

Operation of the Path-Triggered RSVP Receiver Proxy in the case of an admission control failure, using sender notification via Indirect Notify, is illustrated in Figure 5.



***> media flow

==> segment of flow path benefiting from RSVP
(but not benefiting from a reservation in this case)

Path* = Path message containing a Notify Request object
with sender IP Address

Resv* = Resv message containing a Notify Request object
with RSVP Receiver Proxy IP address

NotifyD = Notify message containing a downstream notification

NotifyU = Notify message containing an upstream notification

Figure 5: Reservation Failure with Sender Notification
via Indirect Notify

For local failures on the Receiver Proxy node, if a similar failure on an RSVP midpoint would cause the generation of a ResvErr (for example, admission control failure), the Receiver Proxy MUST generate

a Notify towards the sender. The Receiver Proxy MAY additionally generate a Notify upon local failures that would not ordinarily cause generation of a ResvErr message, such as those described in Appendix B of [RFC2205].

When the method of sender notification via a Notify message is used, it is RECOMMENDED that the RSVP Receiver Proxy also issue a sender notification via a PathErr message. This maximizes the chances that the notification will reach the sender in all situations (e.g., even if some RSVP routers do not support the Notify procedure, or if a Notify message gets dropped). However, for controlled environments (e.g., where all RSVP routers are known to support Notify procedures) and where it is desirable to minimize the volume of signaling, the RSVP Receiver Proxy MAY rely exclusively on sender notification via a Notify message and thus not issue sender notification via a PathErr message.

In environments where there are both RSVP-capable receivers and RSVP Receiver Proxies acting on behalf of non-RSVP-capable receivers, a sender does not know whether a given reservation is established with an RSVP-capable receiver or with an RSVP Receiver Proxy. Thus, a sender that supports the procedures defined in this section may include a Notify Request object in Path messages for a reservation that happens to be controlled by an RSVP-capable receiver. This document does not define, nor expect, any change in existing receiver behavior. As a result, in this case, the sender will not receive Notify messages conveying downstream notifications. However, this is perfectly fine because the synchronization between the RSVP reservation state and the application requirement can be performed by the actual receiver in this case as per the regular end-to-end RSVP model, so that in this case, the sender need not care about downstream notifications.

A sender that does not support the procedures defined in this section might include a Notify Request object in Path messages for a reservation simply because it is interested in getting upstream notifications faster. If the reservation is controlled by an RSVP Receiver Proxy supporting the procedures defined in this section, the sender will also receive unexpected Notify messages containing downstream notifications. It is expected that such a sender will simply naturally drop such downstream notifications as invalid. Because it is RECOMMENDED above that the RSVP Receiver Proxy also issue a sender notification via a PathErr message even when sender notification is effected via a Notify message, the sender will still be notified of a reservation failure in accordance with the "sender notification via PathErr" method. In summary, activating the OPTIONAL "sender notification via Notify" method on a Receiver Proxy does not prevent a sender that does not support this method from

relying on the MANDATORY "sender notification via PathErr" method. It would, however, allow a sender supporting the "sender notification via Notify" method to take advantage of this OPTIONAL method.

With Direct Notification, the downstream notification generated by the RSVP router where the failure occurs is sent to the IP address contained in the Notification Request Object of the corresponding Resv message. In the presence of multiple senders towards the same session, it cannot be generally assumed that a separate Resv message is used for each sender (in fact, with WF and SE there is a single Resv message for all senders, and with FF the downstream router has the choice of generating separate Resv messages or a single one). Hence, in the presence of multiple senders, Direct Notification cannot guarantee notification of all affected senders. Therefore, Direct Notification is better suited to single-sender applications.

With Indirect Notification, the RSVP Receiver Proxy can generate Notify messages with the same logic that is used to generate PathErr messages in the "Sender Notification via PathErr" method (in fact, those are conveying the same error information, only the Notify is directly addressed to the sender while the PathErr travels hop-by-hop). Therefore, operations of the Indirect Notify method in the presence of multiple senders is similar to that of the PathErr method as discussed in Section 3.1: with FF or SE, a Notify MUST be sent to the sender or the set of affected senders, respectively. With WF, the RSVP Receiver Proxy SHOULD send a Notify to each sender, again resulting in a somewhat overly conservative behavior in the presence of multiple senders.

4. Mechanisms for Maximizing the Reservation Span

This section defines extensions to RSVP procedures allowing an RSVP reservation to span as much of the flow path as possible, with any arbitrary number of RSVP Receiver Proxies on the flow path and whether or not the receiver is RSVP-capable. This facilitates deployment and operations of Path-Triggered RSVP Receiver Proxies since it alleviates the topological constraints and/or configuration load otherwise associated with Receiver Proxies (e.g., make sure there is no RSVP Receiver Proxy for a flow upstream of a given Receiver Proxy, ensure there is no Receiver Proxy for a flow if the receiver is RSVP-capable). This also facilitates migration from an RSVP deployment model based on Path-Triggered Receiver Proxies to an end-to-end RSVP model, since receivers can gradually and independently be upgraded to support RSVP and then instantaneously benefit from end-to-end reservations.

Section 4.1 defines a method that allows a Path-Triggered Receiver Proxy function to discover whether there is another Receiver Proxy or an RSVP-capable receiver downstream and then dynamically extend the span of the RSVP reservation downstream. A router implementation claiming compliance with this document SHOULD support the method defined in Section 4.1.

Section 4.2 defines a method that allows a sender to control whether or not an RSVP router supporting the Path-Triggered Receiver Proxy function is to behave as a Receiver Proxy for a given flow. A router implementation claiming compliance with this document MAY support the method defined in Section 4.2.

In a given network environment, a network administrator may elect to use the method defined in Section 4.1, or the method defined in Section 4.2, or possibly combine the two.

4.1. Dynamic Discovery of Downstream RSVP Functionality

When generating a proxy Resv message upstream, a Receiver Proxy supporting dynamic discovery of downstream RSVP functionality MUST forward the Path message downstream instead of terminating it (unless dynamic discovery of downstream RSVP functionality is explicitly disabled). If the destination endpoint supports RSVP (or there is another Receiver Proxy downstream), it will receive the Path and generate a Resv upstream. When this Resv message reaches the Receiver Proxy, it recognizes the presence of an RSVP-capable receiver (or of another RSVP Receiver Proxy) downstream and MUST internally convert its state from a proxied reservation to a regular midpoint RSVP behavior. From then on, the RSVP router MUST behave as a regular RSVP router for that reservation (i.e., as if the RSVP router never behaved as an RSVP Receiver Proxy for that flow). This method is illustrated in Figure 6.

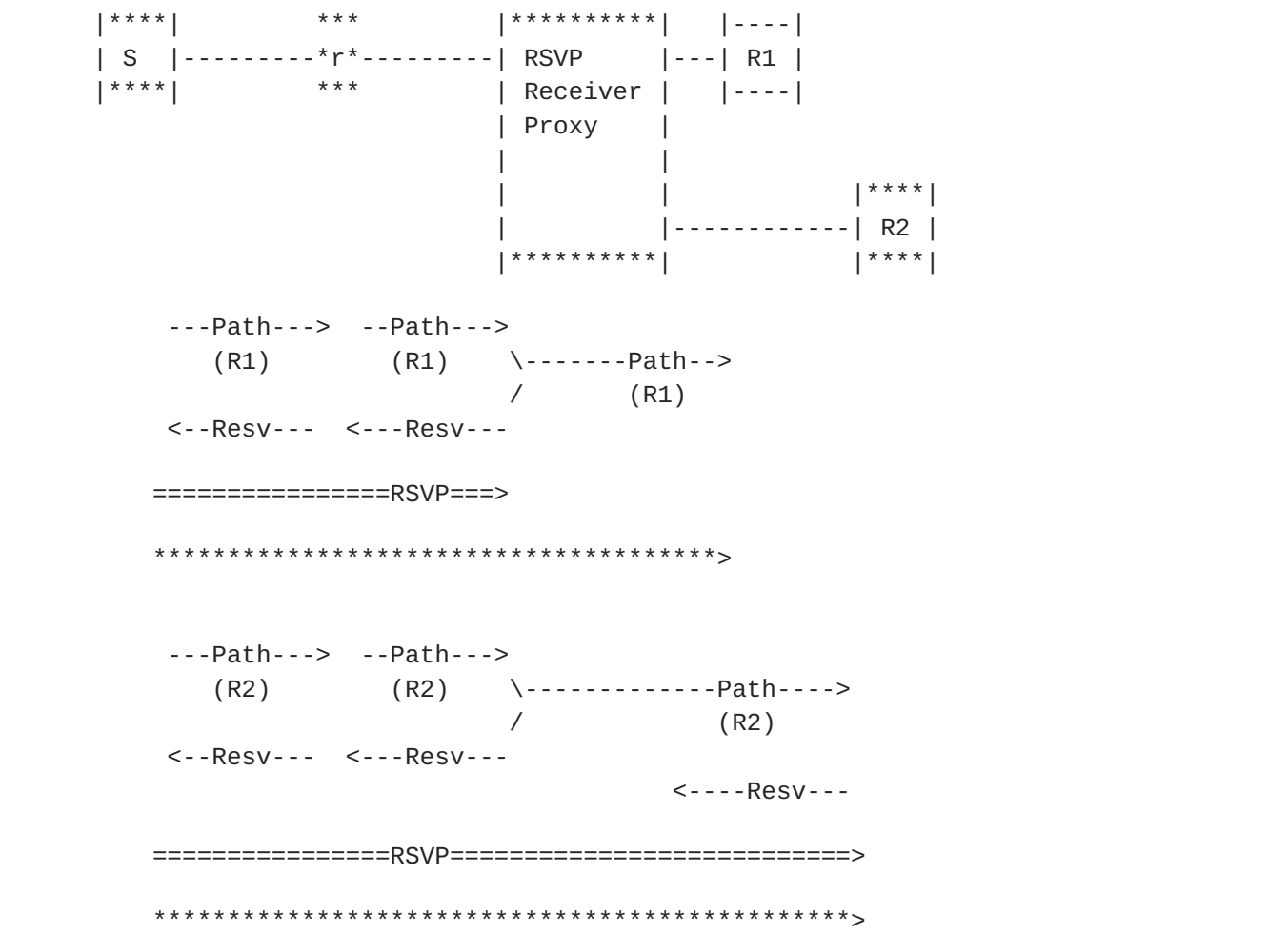


Figure 6: Dynamic Discovery of Downstream RSVP Functionality

If there is no RSVP-capable receiver (or other Receiver Proxy) downstream of the Receiver Proxy, then the Path messages sent by the Receiver Proxy every RSVP refresh interval (e.g., 30 seconds by default) will never be responded to. However, these messages consume a small amount of bandwidth, and in addition would install some RSVP state on RSVP-capable midpoint nodes downstream of the first Receiver Proxy. This is seen as a very minor sub-optimality; however, to mitigate this, the Receiver Proxy MAY tear down any unanswered downstream Path state after a predetermined time (that SHOULD be greater or equal to the Path refresh interval), and MAY stop sending Path messages for the flow (or MAY only send them at much lower frequency).

This approach only requires support of the behavior described in the previous paragraph and does not require any new RSVP extensions.

4.2. Receiver Proxy Control Policy Element

[RFC2750] defines extensions for supporting generic policy-based admission control in RSVP. These extensions include the standard format of POLICY_DATA objects and a description of RSVP handling of policy events.

The POLICY_DATA object contains one or more policy elements, each representing a different (and perhaps orthogonal) policy. As an example, [RFC3181] specifies the preemption priority policy element.

This document defines a new policy element called the Receiver Proxy Control policy element. This document only defines the use of this policy element in Path messages and for unicast reservations. Other usage is outside the scope of this document.

The format of the Receiver Proxy Control policy element is as shown in Figure 7:

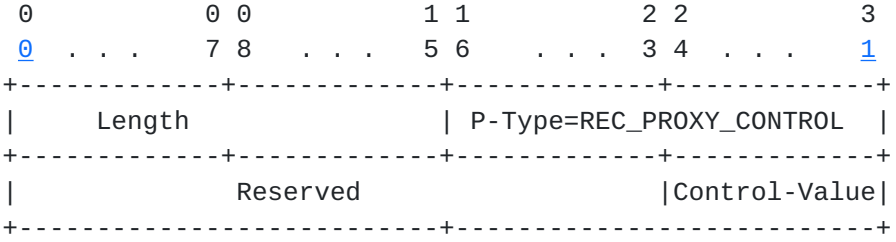
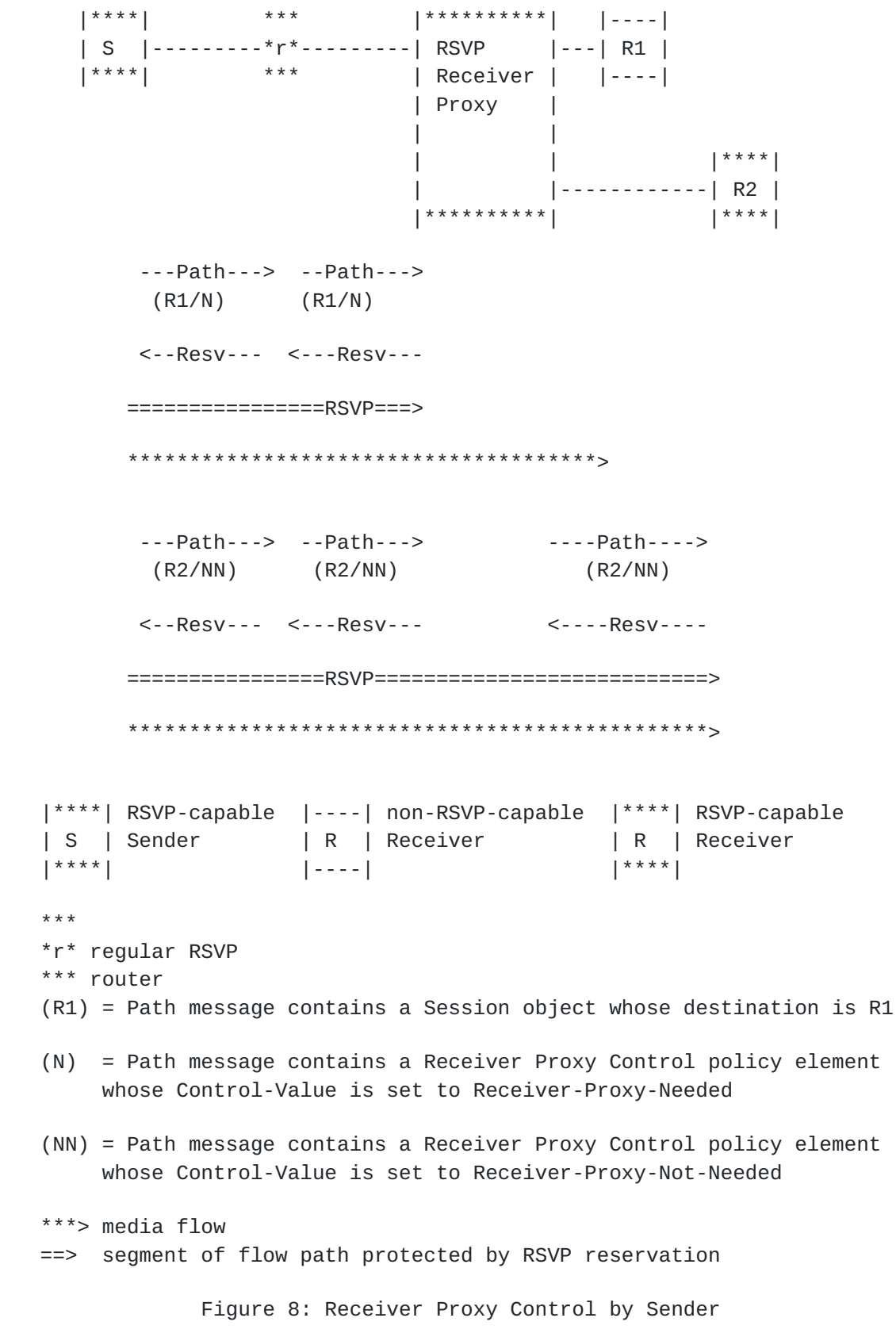


Figure 7: Receiver Proxy Control Policy Element

where:

- Length: 16 bits
  - Always 8. The overall length of the policy element, in bytes.
- P-Type: 16 bits
  - REC_PROXY_CONTROL = 0x07 (see the "IANA Considerations" section).
- Reserved: 24 bits
  - SHALL be set to zero on transmit and SHALL be ignored on reception.
- Control-Value: 8 bits (unsigned)
  - 0 (Reserved): An RSVP Receiver Proxy that understands this policy element MUST ignore the policy element if its Control-Value is set to that value.
  - 1 (Receiver-Proxy-Needed): An RSVP Receiver Proxy that understands this policy element MUST attempt to insert itself as a Receiver Proxy for that flow if the corresponding Path message contains this Control-Value. If the Receiver Proxy also supports dynamic discovery of downstream RSVP functionality as specified in Section 4.1, it MUST still send the Path message downstream and attempt to extend the reservation downstream so that the reservation can be extended to the last Receiver Proxy). An RSVP sender MAY insert the Receiver Proxy Control policy element with this Control-Value when it knows (say, by other means, such as application-level signaling) that the receiver is not RSVP-capable.
  - 2 (Receiver-Proxy-Not-Needed): An RSVP Receiver Proxy that understands this policy element MUST NOT attempt to insert itself as a Receiver Proxy for that flow if the corresponding Path message contains this Control-Value. An RSVP sender MAY insert the Receiver Proxy Control policy element with this Control-Value when it knows (say, by other means, such as application-level signaling) that the receiver is RSVP-capable.

Figure 8 illustrates the method based on the Receiver Proxy Control policy element that allows a sender to control whether or not an RSVP router supporting the Path-Triggered Receiver Proxy function is to behave as a Receiver Proxy for a given flow.



***
r regular RSVP
*** router
(R1) = Path message contains a Session object whose destination is R1

(N) = Path message contains a Receiver Proxy Control policy element whose Control-Value is set to Receiver-Proxy-Needed

(NN) = Path message contains a Receiver Proxy Control policy element whose Control-Value is set to Receiver-Proxy-Not-Needed

***> media flow
==> segment of flow path protected by RSVP reservation

Figure 8: Receiver Proxy Control by Sender

4.2.1. Default Handling

As specified in Section 4.2 of [RFC2750], Policy-Ignorant Nodes (PINs) implement a default handling of POLICY_DATA objects ensuring that those objects can traverse PINs in transit from one Policy Enforcement Point (PEP) to another. This applies to the situations in which POLICY_DATA objects contain the Receiver Proxy Control policy element specified in this document, so that those objects can traverse PINs.

Section 4.2 of [RFC2750] also defines a similar default behavior for policy-capable nodes that do not recognize a particular policy element. This applies to the Receiver Proxy Control policy element specified in this document, so that messages carrying the element can traverse policy-capable nodes that do not support the extensions described in this document.

5. Security Considerations

As this document defines extensions to RSVP, the security considerations of RSVP apply, which are discussed in [RFC2205], [RFC4230], and [SEC-GRP-KEY]. Approaches for addressing those concerns are discussed further below.

The RSVP authentication mechanisms defined in [RFC2747] and [RFC3097] protect RSVP message integrity hop-by-hop and provide node authentication as well as replay protection, thereby protecting against corruption and spoofing of RSVP messages. These hop-by-hop integrity mechanisms can be used to protect RSVP reservations established using an RSVP Receiver Proxy in accordance with this document. [SEC-GRP-KEY] discusses key types and key provisioning methods as well as their respective applicability to RSVP authentication. RSVP authentication (defined in [RFC2747] and [RFC3097]) SHOULD be supported by an implementation of this document.

[SEC-GRP-KEY] also discusses applicability of IPsec mechanisms ([RFC4302], [RFC4303]) and associated key provisioning methods for security protection of RSVP. This discussion applies to the protection of RSVP in the presence of Path-Triggered RSVP Receiver Proxies as defined in this document.

A subset of RSVP messages are signaled with the IP router alert option ([RFC2113], [RFC2711]). Based on the current security concerns associated with the use of the IP router alert option, the applicability of RSVP (and therefore of the RSVP Proxy approaches discussed in this document) is limited to controlled environments

(i.e., where the security risks associated with the use of the IP router alert option are understood and protected against). The security aspects and common practices around the use of the current IP router alert option, and consequences of using the IP router alert option by applications such as RSVP, are discussed in detail in [RTR-ALERT].

When an RSVP Receiver Proxy is used, the RSVP reservation is no longer controlled by the receiver, but rather is controlled by the Receiver Proxy (using hints received from the sender in the Path message) on behalf of the sender. Thus, the Receiver Proxy ought to be trusted by the end-systems to control the RSVP reservation appropriately. However, basic RSVP operation already assumes a trust model where end-systems trust RSVP nodes to appropriately perform RSVP reservations. So the use of an RSVP Receiver Proxy is not seen as introducing any significant additional security threat or as modifying the RSVP trust model.

In fact, there are situations in which the use of an RSVP Receiver Proxy reduces the security risks. One example is where a network operator relies on RSVP to perform resource reservation and admission control within a network and where RSVP senders and RSVP routers are located in the operator's premises, while the many RSVP receivers are located in the operator's customers' premises. Such an environment is further illustrated in Appendix A.1, "RSVP-Based VoD Admission Control in Broadband Aggregation Networks", of [RFC5945]. From the operator's perspective, the RSVP routers and RSVP senders are in physically secured locations and therefore exposed to a lower risk of being tampered with, while the receivers are in locations that are physically unsecured and therefore subject to a higher risk of being tampered with. The use of an RSVP Receiver Proxy function effectively increases the security of the operator's reservation and admission control solution by completely excluding receivers from its operation. Filters can be placed at the edge of the operator network, discarding any RSVP message received from end-users. This provides a very simple and effective protection of the RSVP reservation and admission control solution operating inside the operator's network.

5.1. Security Considerations for the Sender Notification via Notify Message

This document defines, in Section 3.2, an optional method relying on the use of the Notify message specified in [RFC3473]. The Notify message can be sent in a non-hop-by-hop fashion that precludes the use of the RSVP hop-by-hop integrity and authentication model. The approaches and considerations for addressing this issue presented in the Security Considerations section of [RFC3473] apply. In

particular, where the Notify messages are transmitted non-hop-by-hop and the same level of security provided by [RFC2747] is desired, IPsec-based integrity and authentication can be used ([RFC4302] or [RFC4303]). Alternatively, the sending of non-hop-by-hop Notify messages can be disabled. Finally, [SEC-GRP-KEY] discusses the applicability of group keying for non-hop-by-hop Notify messages.

5.2. Security Considerations for the Receiver Proxy Control Policy Element

This document also defines, in Section 4.2, the optional Receiver Proxy Control policy element. Policy elements are signaled by RSVP through encapsulation in a Policy Data object as defined in [RFC2750]. Therefore, like any other policy elements, the integrity of the Receiver Proxy Control policy element can be protected as discussed in Section 6 of [RFC2750] by two optional security mechanisms.

The first mechanism relies on the RSVP authentication discussed above that provides a chain of trust when all RSVP nodes are policy capable. With this mechanism, the INTEGRITY object is carried inside RSVP messages.

The second mechanism relies on the INTEGRITY object within the POLICY_DATA object to guarantee integrity between RSVP Policy Enforcement Points (PEPs) that are not RSVP neighbors. This is useful only when some RSVP nodes are Policy-Ignorant Nodes (PINs). The INTEGRITY object within the POLICY_DATA object MAY be supported by an implementation of this document.

Details for the computation of the content of the INTEGRITY object can be found in Appendix B of [RFC2750]. This states that the Policy Decision Point (PDP), at its discretion, and based on destination PEP/PDP or other criteria, selects an Authentication Key and the hash algorithm to be used. Keys to be used between PDPs can be distributed manually or via a standard key management protocol for secure key distribution.

Note that where non-RSVP hops may exist in between RSVP hops, as well as where RSVP-capable Policy-Ignorant Nodes (PINs) may exist in between PEPs, it may be difficult for the PDP to determine what the destination PDP is for a POLICY_DATA object contained in some RSVP messages (such as a Path message). This is because in those cases, the next PEP is not known at the time of forwarding the message. In this situation, a key shared across multiple PDPs may be used. This is conceptually similar to the use of a key shared across multiple RSVP neighbors, as discussed in [SEC-GRP-KEY]. We also observe that this issue may not exist in some deployment scenarios where a single

PDP (or a low number of PDPs) is used to control all the PEPs of a region (such as an administrative domain). In such scenarios, it may be easy for a PDP to determine what the next-hop PDP is, even when the next-hop PEP is not known, simply by determining what the next region is that will be traversed (say, based on the destination address).

6. IANA Considerations

6.1. RSVP Error Codes

Since, as discussed in Section 3.1.2, this document allows two error codes to be used in PathErr messages while [RFC2205] only specified their use in ResvErr messages, IANA has updated the existing entries for these two error codes under the "Error Codes and Globally-Defined Error Value Sub-Codes" registry. Each entry refers to this document, in addition to referring to [RFC2205]. Specifically, the entry for Error Code 1 and Error Code 2 read:

1 Admission Control Failure [RFC2205] [RFC5946]

2 Policy Control Failure [RFC2205] [RFC5946]

IANA has also allocated a new RSVP Error Code "36;: Unrecoverable Receiver Proxy Error", as discussed in Section 3.1.2. This error code has been allocated under the "Error Codes and Globally-Defined Error Value Sub-Codes" registry. The entry for this error code reads:

36 Unrecoverable Receiver Proxy Error [RFC5946]

The sixteen bits of the Error Value field are defined in [RFC5946]

6.2. Policy Element

This document defines, in Section 4.2, a new policy element called the Receiver Proxy Control policy element. As specified in [RFC2750], standard RSVP policy elements (P-Type values) are to be assigned by IANA as per "IETF Consensus" policy following the policies outlined in [RFC2434] (this policy is now called "IETF Review" as per [RFC5226]).

Thus, IANA has allocated one P-Type to the Receiver Proxy Control policy element from the standard RSVP policy element range.

In Section 4.2, this document defines a Control-Value field inside the Receiver Proxy Control policy element. IANA has created the "Receiver Proxy Control Policy Element (P-Type 0x07) Control-Value field" registry and allocated the following values:

- 0 : Reserved
- 1 : Receiver-Proxy-Needed
- 2 : Receiver-Proxy-Not-Needed

Following the policies outlined in [RFC5226], numbers in the range 3-127 are allocated according to the "IETF Review" policy, numbers in the range 128-240 are assigned on a "First Come First Served" basis, and numbers in the range 241-255 are reserved for "Private Use".

7. Acknowledgments

This document benefited from discussions with Carol Iturralde and Anca Zamfir. Lou Berger, Adrian Farrel, and John Drake provided review and guidance, in particular on the usage of the Path_State_Removed flag and of the Notify message, both borrowed from [RFC3473]. We also thank Stephen Kent, Ken Carlberg, and Tim Polk for their valuable input and proposed enhancements. Finally, we thank Cullen Jennings, Magnus Westerlund, and Robert Sparks for stimulating the work on extensions maximizing the reservation span and facilitating migration from the Proxy model to the end-to-end RSVP model.

8. References

8.1. Normative References

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, October 1999.

[RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, January 2000.

[RFC2750] Herzog, S., "RSVP Extensions for Policy Control", RFC 2750, January 2000.

[RFC3097] Braden, R. and L. Zhang, "RSVP Cryptographic Authentication -- Updated Message Type Value", RFC 3097, April 2001.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3473] Berger, L., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

8.2. Informative References

[RFC3181] Herzog, S., "Signaled Preemption Priority Policy Element", RFC 3181, October 2001.

[RFC4230] Tschofenig, H. and R. Graveman, "RSVP Security Properties", RFC 4230, December 2005.

[RFC5945] Le Faucheur, F., Manner, J., Wing, D., and A. Guillou, "Resource Reservation Protocol (RSVP) Proxy Approaches", RFC 5945, October 2010.

[RTR-ALERT] Le Faucheur, F., "IP Router Alert Considerations and Usage", Work in Progress, October 2009.

[SEC-GRP-KEY] Behringer, M. and F. Le Faucheur, "Applicability of Keying Methods for RSVP Security", Work in Progress, June 2010.

Authors' Addresses

Francois Le Faucheur
Cisco Systems
Greenside, 400 Avenue de Roumanille
Sophia Antipolis 06410
France
Phone: +33 4 97 23 26 19
EMail: flefauch@cisco.com

Jukka Manner
Aalto University
Department of Communications and Networking (Comnet)
P.O. Box 13000
FIN-00076 Aalto
Finland
Phone: +358 9 470 22481
EMail: jukka.manner@tkk.fi
URI: http://www.netlab.tkk.fi/~jmanner/

Ashok Narayanan
Cisco Systems
300 Beaver Brook Road
Boxborough, MA 01719
United States
EMail: ashokn@cisco.com

Allan Guillou
SFR
40-42 Quai du Point du Jour
Boulogne-Billancourt 92659
France
EMail: allan.guillou@sfr.com

Hemant Malik
Bharti Airtel, Ltd.
4th Floor, Plot No. 16
Udyog Vihar, Phase IV
Gurgaon, 122015
India
EMail: Hemant.Malik@airtel.in