TSVWG
Internet-Draft
Intended status: Informational
Expires: March 15, 2008

F. Le Faucheur
Cisco
J. Manner
University of Helsinki
D. Wing
Cisco
A. Guillou
Neuf

September 12, 2007

RSVP Proxy Approaches
draft-ietf-tsvwg-rsvp-proxy-approaches-02.txt

Status of this Memo

Copyright Notice

Abstract

RSVP signaling can be used to make end-to-end resource reservations in an IP network in order to guarantee the Quality of Service 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. This document presents RSVP Proxy behaviors allowing RSVP routers to perform RSVP signaling on behalf of a receiver or a sender that is not RSVP-capable. This allows resource reservations to be established on critical parts of the end-to-end path. This document reviews conceptual approaches for deploying RSVP Proxies and discusses how RSVP reservations can be synchronized with application requirements, despite the sender, receiver, or both not participating in RSVP. This document also points out where extensions to RSVP (or to other protocols) may be needed for deployment of a given RSVP Proxy approach. However, such extensions are outside the scope of this document. Finally, practical use cases for RSVP Proxy are described.

Table of Contents

1. Introduction

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

Since the data sender or receiver may be unaware of RSVP, there are two scenarios. In the first case, an entity in the network must operate on behalf of the data sender, and in particular, generate RSVP Path messages, and eventually receive, process and sink Resv messages. We refer to this entity as the RSVP Sender Proxy. In the latter case, an entity in the network must receive Path messages sent by a data sender (or by an RSVP Sender Proxy), sink those, and return Resv messages on behalf of the data receiver(s). We refer to this entity as the RSVP Receiver Proxy.

The flow sender and receiver generally have at least some (if not full) awareness of the application producing or consuming that flow. Hence, the sender and receiver are in a natural position to synchronize the establishment, maintenance and tear down of the RSVP reservation with the application requirements. Similarly they are in a natural position to determine the characteristics of the reservation (bandwidth, QoS service,...) which best match the application requirements. For example, before completing the establishment of a multimedia session, the endpoints may decide to establish RSVP reservations for the corresponding flows. Similarly, when the multimedia session is torn down, the endpoints may decide to tear down the corresponding RSVP reservations. For instance, [RFC3312] discusses how RSVP reservations can be very tightly synchronized by SIP endpoints with SIP session control and SIP signaling.

When RSVP reservation establishment, maintenance and tearing down is to be handled by RSVP Proxies on behalf of an RSVP sender or receiver, a key challenge for the RSVP Proxy is to determine when the RSVP reservations need to be established, maintained and torn down and to determine what are the characteristics (bandwidth, QoS Service,...) of the required RSVP reservations matching the application requirements. We refer to this problem as the synchronization of RSVP reservations with application level requirements.

The IETF Next Steps in Signaling (NSIS) working group is designing, as one their charter items, a new QoS signaling protocol. This scheme already includes the notion of proxy operation, and terminating QoS signaling on nodes that are not the actual data senders or receivers. This is the same concept as the proxy operation for RSVP discussed in this document. One difference though is that the NSIS framework does not consider multicast resource reservations, which RSVP provides today.

The next section introduces the notion of RSVP Sender Proxy and RSVP Receiver Proxy. The following section defines useful terminology. The subsequent section then presents several fundamental RSVP Proxy approaches insisting on how they achieve the necessary synchronization of RSVP reservations with application level requirements. Appendix A includes more detailed use cases for the proxies in various real life deployment environments.

2. RSVP Proxy Behaviors

This section discusses the two types of proxies; the RSVP Sender Proxy operating on behalf of data senders, and the RSVP Receiver Proxy operating for data receivers. The concepts presented in this document are not meant to replace the standard RSVP and end-to-end RSVP reservations are still expected to be used whenever possible. However, RSVP Proxies are intended to facilitate RSVP deployment where end-to-end RSVP signaling is not possible.

2.1. RSVP Receiver Proxy

With conventional RSVP operations, RSVP reservations are controlled by receivers of data. After a data sender has sent an RSVP Path message towards the intended recipient(s), each recipient that requires a reservation generates a Resv message. If, however, a data receiver is not running the RSVP protocol, the last hop RSVP router will still send the Path message to the data receiver, which will silently drop this message as an IP packet with an unknown protocol number.

In order for reservations to be made in such a scenario, one of the RSVP routers on the data path must somehow know that the data receiver will not be participating in the resource reservation signaling. This RSVP router should, thus, perform RSVP Receiver Proxy functionality on behalf of the data receiver. This is illustrated in Figure 1. Various mechanisms by which the RSVP proxy router can gain the required information are discussed later in the document.

Figure 1: RSVP Receiver Proxy

2.2. RSVP Sender Proxy

With conventional RSVP operations, if a data sender is not running the RSVP protocol, a resource reservation can not be set up; a data receiver can not alone reserve resources without Path messages first being received. Thus, even if the data receiver is running RSVP, it still needs some node on the data path to send a Path message towards the data receiver.

In that case, an RSVP node on the data path must somehow know that it should generate Path messages to allow the receiver to set up the resource reservation. This node is referred to as the RSVP Sender Proxy and is illustrated in Figure 2. This case is more complex than the Receiver Proxy case, since the RSVP Sender Proxy must be able to generate all the information in the Path message (such as the Sender TSpec) without the benefit of having previously received any RSVP message. An RSVP Receiver Proxy, by contrast only needs to formulate an appropriate Resv message in response to an incoming Path message. Mechanisms to operate an RSVP Sender Proxy are discussed later in this document.

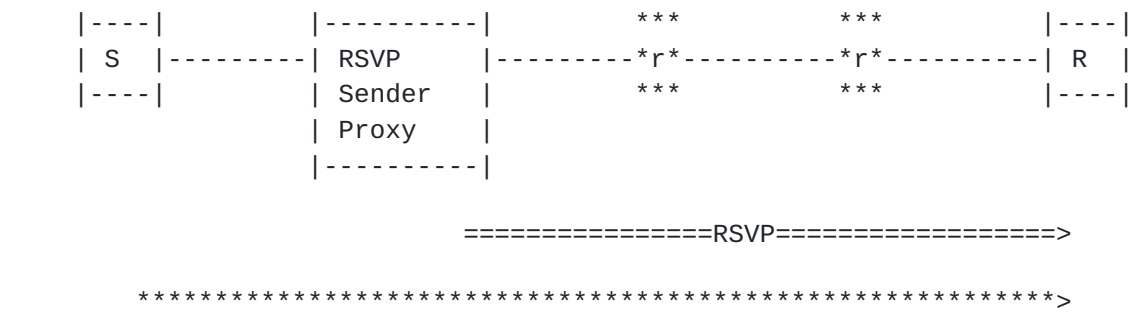


Figure 2: RSVP Sender Proxy

3. Terminology

On-Path: located on the datapath of the actual flow of application data (regardless of where it is located with respect to the application level signaling path).

Off-Path: not On-Path.

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

RSVP Receiver Proxy: an RSVP capable router performing, on behalf of a receiver, the RSVP operations which would normally be performed by an RSVP-capable receiver if end-to-end RSVP signaling was 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 which would normally be performed by an RSVP-capable sender if end-to-end RSVP signaling was 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 which is not behaving as a RSVP Receiver Proxy nor as a RSVP Sender Proxy.

Note that the roles of RSVP Receiver Proxy, RSVP Sender Proxy, 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.

Application level signaling: signaling between entities operating above the IP layer and which are aware of the QoS requirements for actual media flows. SIP and RTSP are examples of application level signaling protocol. RSVP is clearly not an application level signaling.

4. RSVP Proxy Approaches

This section discusses fundamental RSVP Proxy approaches.

4.1. Path-Triggered Receiver Proxy

In this approach, it is assumed that the sender is RSVP capable and takes full care of the synchronization between application requirements and RSVP reservations. With this approach, the RSVP Receiver Proxy uses the RSVP Path messages generated by the sender as the cue for establishing the RSVP reservation on behalf of the receiver. The RSVP Receiver Proxy is effectively acting as a slave making reservations (on behalf of the receiver) under the sender's control. This changes somewhat 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.

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 Resv message (whose details are discussed below) was received on the downstream interface. This includes

performing admission control on the downstream interface, establishing a Resv state (in 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.

In order to build the Resv message, the RSVP Receiver Proxy can take into account information received in the Path message. For example, the RSVP Receiver Proxy may compose a FLOWSPEC object for the Resv message which mirrors the SENDER_TSPEC object in the received Path message.

Operation of the Path-Triggered Receiver Proxy in the case of a successful reservation is illustrated in Figure 3.

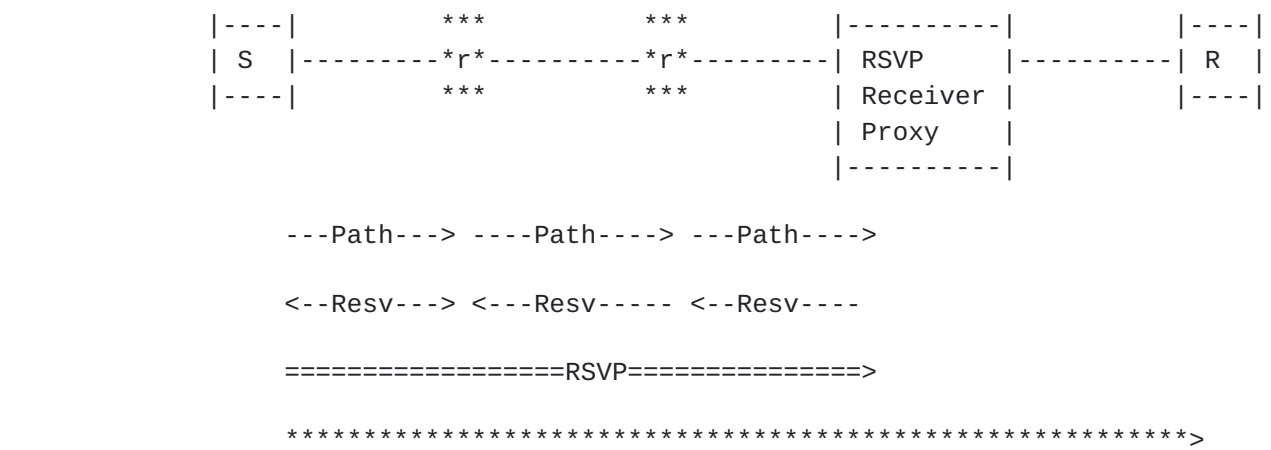


Figure 3: Path-Triggered RSVP Receiver Proxy

In case the reservation establishment is rejected (for example because of an admission control failure on a regular RSVP router on the path between the RSVP-capable sender and the RSVP Receiver Proxy), a ResvErr message will be generated as per conventional RSVP operations and will travel downstream towards the RSVP Receiver Proxy. While this ensures that the RSVP Receiver Proxy is aware of the reservation failure, conventional RSVP procedures do not cater for notification of the sender of the reservation failure. Operation

of the Path-Triggered RSVP Receiver Proxy in the case of an admission control failure is illustrated in Figure 4.

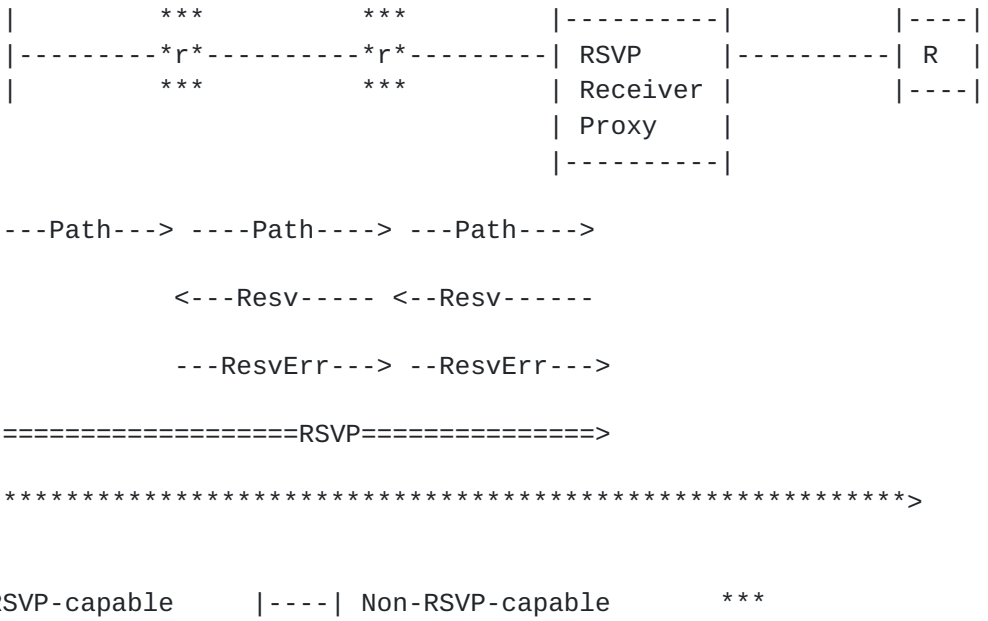


Figure 4: Path-Triggered RSVP Receiver Proxy with Failure

Since, as explained above, in this scenario involving the RSVP Receiver Proxy, synchronization between application and RSVP reservation is generally performed by the sender, notifying the sender of reservation failure is needed. [I-D.ietf-tsvwg-rsvp-proxy-proto] specifies RSVP extensions allowing such sender notification in case of reservation failure in the presence of a Path-Triggered RSVP Receiver Proxy.

4.2. Path-Triggered Sender Proxy for Reverse Direction

In this approach, it is assumed that one endpoint is RSVP capable and takes full care of the synchronization between application requirements and RSVP reservations. This endpoint is the sender for one flow direction (which we refer to as the "forward" direction) and is the receiver for the flow in the opposite direction (which we refer to as the "reverse" direction).

With the Path-Triggered Sender Proxy for Reverse Direction approach,

the RSVP Proxy uses the RSVP signaling generated by the sender as the cue for initiating RSVP signaling for the reservation in the reverse direction. Thus, the RSVP Proxy is effectively acting as a Sender Proxy for the reverse direction under the control of the sender for the forward direction. Note that this assumes a degree of symmetry for the two directions of the flow (as is currently typical for IP telephony, for example). This is illustrated in Figure 5.

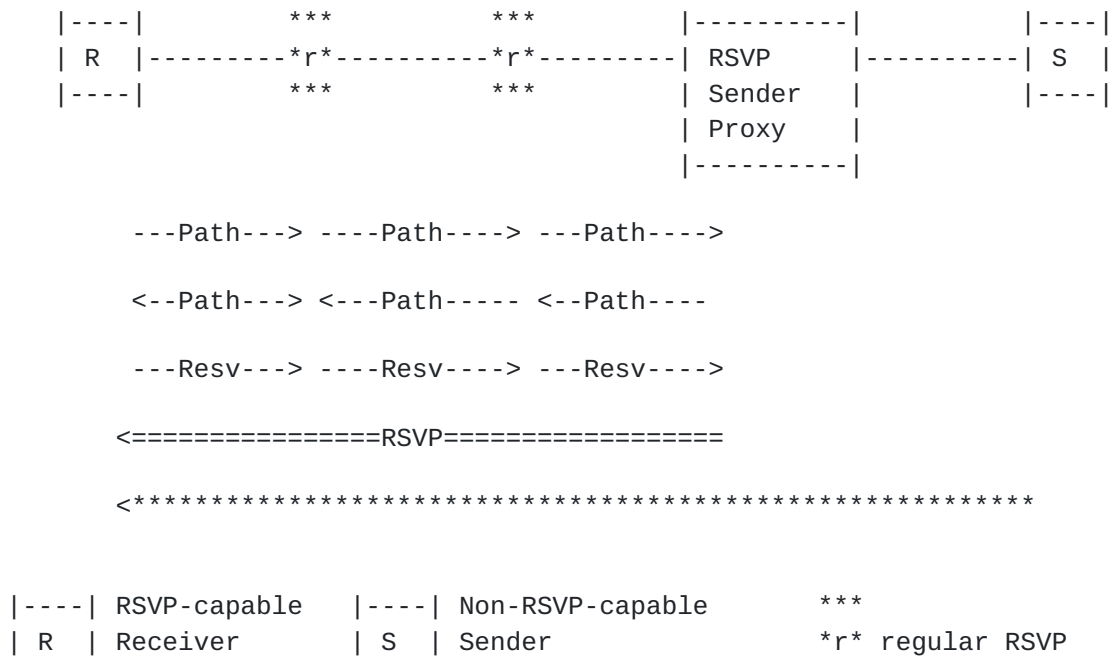


Figure 5: Path-Triggered Sender Proxy for Reverse Direction

Of course, the RSVP Proxy may simultaneously (and typically will) also act as the Path-Triggered Receiver Proxy for the forward direction, as defined in Section 4.1. Such an approach is most useful in situations involving RSVP reservations in both directions for symmetric flows. This is illustrated in Figure 6.

Figure 6: Path Triggered Receiver & Sender Proxy

With the Path-Triggered Sender Proxy for Reverse Direction approach, the RSVP router may be configurable to use receipt of a regular RSVP Path message as the trigger for Sender Proxy for Reverse Direction behavior.

On receipt of the RSVP Path message for the forward direction, the RSVP Sender Receiver Proxy :

1. sinks the Path message

2. behaves as if a Path message for reverse direction (whose details are discussed below) had been received by the Sender Proxy. This includes establishing the corresponding Path state, forwarding the Path message downstream, sending periodic refreshes of the

Path message and tearing down the Path in reverse direction when the Path state in forward direction is torn down.

In order to build the Path message for the reverse direction, the RSVP Sender Proxy can take into account information in the received Path message for the forward direction. For example, the RSVP Sender Proxy may mirror the SENDER_TSPEC object in the received Path message.

We observe that this approach does not require any extensions to the existing RSVP protocol.

4.3. Inspection-Triggered Proxy

In this approach, it is assumed that the RSVP Proxy is on the datapath of "packets of interest", that it can inspect such packets on the fly as they transit through it, and that it can infer information from these packets of interest to determine what RSVP reservations need to be established, when and with what characteristics (possibly also using some configured information).

One example of "packets of interest" could be application level signaling. An RSVP Proxy capable of inspecting SIP signaling for multimedia session or RTSP signaling for Video streaming, can obtain from such signaling information about when a multimedia session is up or when a Video is going to be streamed. It can also identify the addresses and ports of senders and receivers and can determine the bandwidth of the corresponding flows. Thus, such an RSVP Proxy can determine all necessary information to synchronize RSVP reservations to application requirements. This is illustrated in Figure 7.

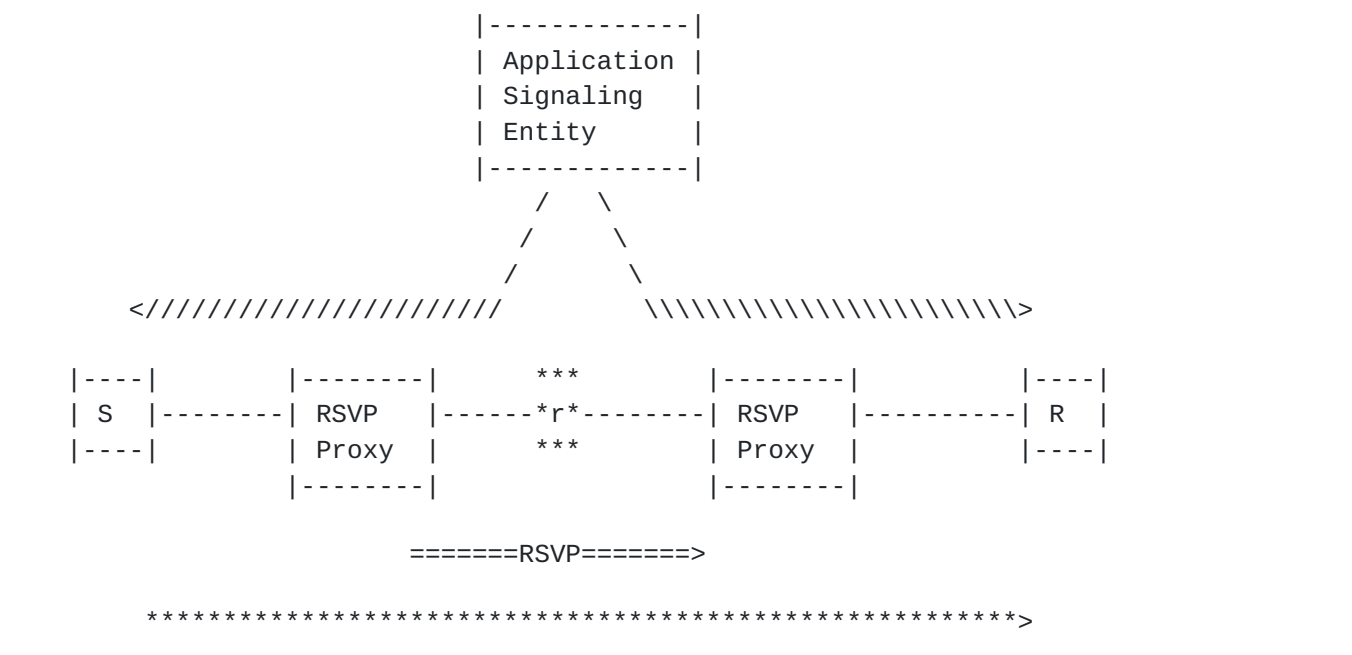


Figure 7: Inspection-Triggered RSVP Proxy

Another example of "packets of interest" could be packets belonging to the application flow itself (e.g. media packets). An RSVP Proxy capable of detecting the transit of packets from a particular flow, can attempt to establish a reservation corresponding to that flow. Characteristics of the reservation may be derived from configuration, flow measurement or a combination of those.

Note however, that in case of reservation failure, the inspection-triggered RSVP Proxy does not have a direct mechanism for notifying the application (since it is not participating itself actively in application signaling) so that the application takes appropriate action (for example terminate the corresponding session). To mitigate this problem, the inspection-triggered RSVP Proxy may mark differently the DSCP of flows for which an RSVP reservation has been successfully proxied from the flows for which a reservation is not in

place. In some situations, the Inspection-Triggered Proxy might be able to modify the "packets of interest" (e.g. application signaling messages) to convey some hint to applications that the corresponding flows cannot be guaranteed by RSVP reservations.

With the inspection-triggered Proxy approach, the RSVP Receiver Proxy is effectively required to attempt to build application awareness by traffic inspection and then is somewhat limited in the actions in can take in case of reservation failure. However, this may be a useful approach in some environments. Note also that this approach does not require any change to the RSVP protocol.

With the "Inspection-Triggered" RSVP Proxy approach, the RSVP router may be configurable to use and interpret some specific "packets of interest" as the trigger for RSVP Receiver Proxy behavior.

4.4. STUN-Triggered Proxy

In this approach, the RSVP Proxy takes advantage of the application awareness provided by the STUN signaling to synchronize RSVP reservations with application requirements. The STUN signaling is sent from endpoint to endpoint. This is illustrated in Figure 8.

```
    |----|        |--------|      ***        |--------|         |----|
    | S  |--------| RSVP   |------*r*--------| RSVP   |----------| R  |
    |----|        | Proxy  |      ***        | Proxy  |         |----|
                  |--------|                 |--------|

         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^>

                          =======RSVP=======>

         *********************************************************>


   |----| Non-RSVP-capable   |----| Non-RSVP-capable      ***
   | S  | Sender             | R  | Receiver              *r* regular RSVP
   |----|                    |----|                       *** router

   ^^^> STUN message flow (over same UDP ports as media flow)

   ==>  segment of flow path protected by RSVP reservation

   ***> RTP media flow
```

Figure 8: STUN-Triggered Proxy

In this approach, a STUN [I-D.ietf-behave-rfc3489bis] message triggers the RSVP Proxy. The STUN message could also include (yet-to-be-specified) STUN attributes to indicate information such as the bandwidth and application requesting the flow, which would allow the RSVP proxy agent to create an appropriately-sized reservation for each flow.

For unicast flows, [I-D.ietf-mmusic-ice] is an already widely-adopted emerging standard for NAT traversal. For our purposes of triggering RSVP Proxy behavior, we rely on ICE's connectivity check -- the exchange of STUN Binding Request messages between hosts to verify connectivity (see section 2.2 of [I-D.ietf-mmusic-ice]). By including new STUN attributes in those connectivity check messages, an RSVP Proxy agent could perform its functions more effectively. Additionally, the RSVP Proxy agent can inform endpoints of an RSVP reservation failure by dropping the ICE connectivity check message or sending ICMP messages back to the endpoint. This provides very RSVP-like call admission control and signaling to the endpoints, without implementing RSVP on the endpoints, and also operates through NATs.

For multicast flows (or certain kinds of unicast flows that don't or can't use ICE), a STUN Indication message [I-D.ietf-behave-rfc3489bis] could indicate the flow's bandwidth,

providing a benefit similar to the ICE connectivity check. STUN Indication messages are not acknowledged by the receiver and have the same scalability as the underlying multicast flow.

The corresponding extensions to ICE and STUN for such a STUN-triggered RSVP Proxy approach are beyond the scope of this document. They may be defined in the future in a separate document.

4.5. Application_Entity-Controlled Proxy

In this approach, it is assumed that an entity involved in the application level signaling controls an RSVP Proxy which is located in the datapath of the application flows (i.e. "on-path"). With this approach, the RSVP Proxy does not attempt to determine itself the application reservation requirements. Instead the RSVP Proxy is instructed by the entity participating in application level signaling to establish, maintain and tear down reservations as needed by the application flows. In other words, with this approach, the solution for synchronizing RSVP signaling with application level requirements is to rely on an application-level signaling entity that controls an RSVP Proxy function that sits in the flow datapath. This approach allows control of an RSVP Sender Proxy, an RSVP Receiver Proxy or both.

Operation of the Application_Entity-Controlled Proxy is illustrated in Figure 9.

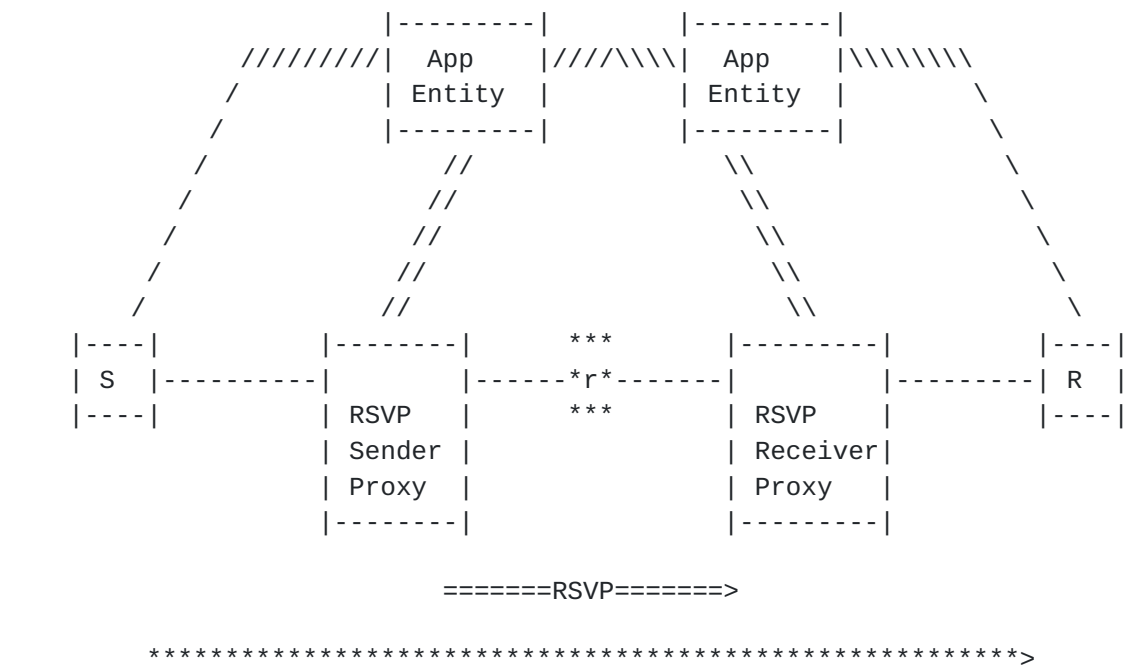


Figure 9: Application_Entity-Controlled Proxy

As an example, the Application_Entity-Controlled Proxy may be used in the context of Session Border Controllers (SBCs) (see [I-D.ietf-sipping-sbc-funcs] for description of SBCs) to establish RSVP reservations for multimedia sessions. In that case, the Application Entity may be the signaling component of the SBC.

This RSVP Proxy approach does not require any extension to the RSVP protocol. However, it relies on an RSVP Proxy control interface allowing control of the RSVP Proxy by an application signaling entity. This RSVP Proxy control interface is beyond the scope of the present document. Candidate protocols for realizing such interface include SNMP, COPS-PR, QPIM, XML and DIAMETER. This interface may

rely on soft states or hard states. Clearly, when hard states are used, those need to be converted appropriately by the RSVP Proxy entities into the corresponding RSVP soft states.

In general, the Application Entity is not expected to maintain awareness of which RSVP Receiver Proxy is on the path to which destination. However, in the particular cases where it does so reliably, we observe that the Application Entity could control the RSVP Sender Proxy and Receiver Proxy so that aggregate RSVP reservations are used between those, instead of one reservation per flow. For example, these aggregate reservations could be of RSVP-AGGREGATE type as specified in [RFC3175] or of GENERIC-AGGREGATE type as specified in [RFC4860]. Such aggregate reservations could be used so that a single reservation can be used for multiple (possibly all) application flows transiting via the same RSVP Sender Proxy and the same RSVP Receiver Proxy.

For situations where only the RSVP Sender Proxy has to be controlled by this interface, the interface may be realized through the simple use of RSVP itself, over a GRE tunnel from the application entity to the RSVP Sender Proxy. This particular case is further discussed in Section 4.5.1. Another particular case of interest is where the application signaling entity resides on the same device as the RSVP Proxy. In that case, this interface may be trivially realized as an internal API. An example environment based on this particular case is illustrated in Section 4.5.2.

4.5.1. Application_Entity-Controlled Sender Proxy using "RSVP over GRE"

This approach is simply a particular case of the more general Application_Entity-Controlled Proxy, but where only RSVP Sender Proxies need to be controlled by the application, and where RSVP is effectively used as the control protocol between the application signaling entity and the RSVP Sender Proxy.

In this approach, the RSVP messages (e.g. RSVP Path message) are effectively generated by the application entity and logically "tunnelled" to the RSVP Sender Proxy via GRE tunneling. This is to ensure that the RSVP messages follow the exact same path as the flow they protect (as required by RSVP operations) on the segment of the end-to-end path which is to be subject to RSVP reservations.

Figure 10 illustrates such an environment.

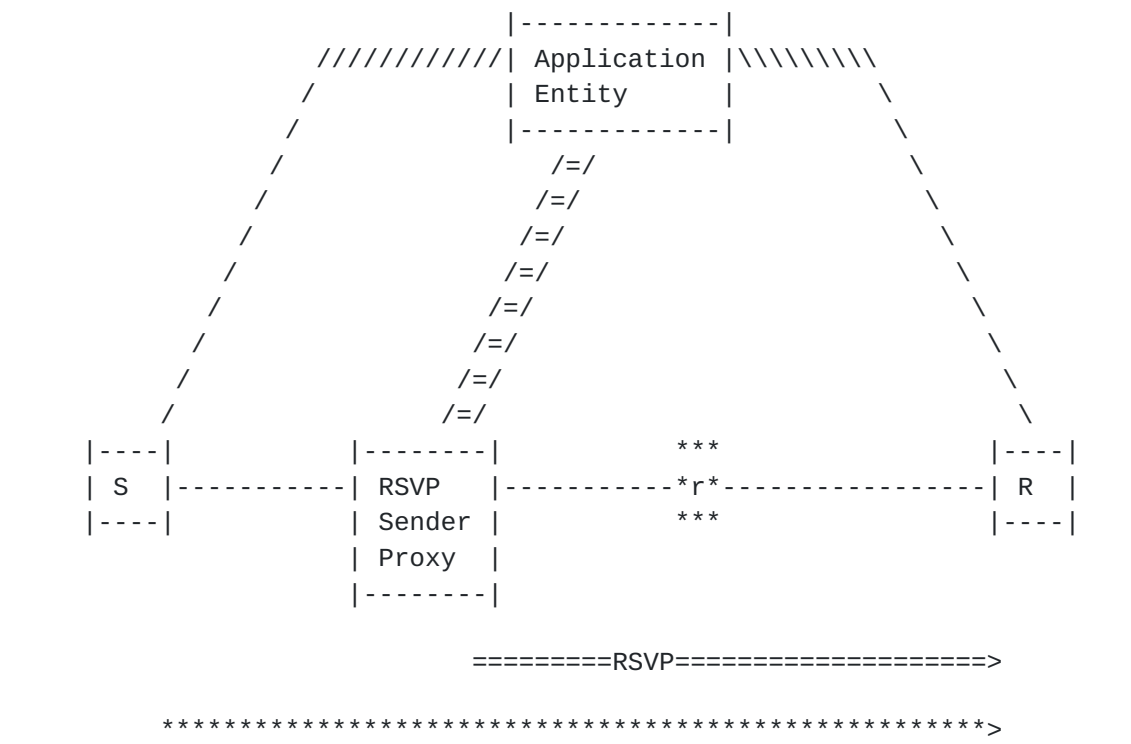


Figure 10: Application-Entity-Controlled Sender Proxy via "RSVP over GRE"

With the Application_Entity-Controlled Sender Proxy using "RSVP Over GRE", the application entity :

- o generates a Path message on behalf of the sender, corresponding to the reservation needed by the application and maintains the corresponding Path state. The Path message built by the application entity is exactly the same as would be built by the actual sender (if it was RSVP-capable), with one single exception which is that the Application Entity puts its own IP address as the RSVP Previous Hop. In particular, it is recommended that the

source address of the Path message built by the application entity be set to the IP address of the sender (not of the application entity). This helps ensuring that, in the presence of non-RSVP routers and of load-balancing in the network where the load-balancing algorithm takes into account the source IP address, the Path message generated by the application entity follows the exact same path that the actual stream sourced by the sender.

- o encapsulates the Path message into a GRE tunnel whose destination address is the RSVP Sender Proxy i.e. an RSVP Router sitting on the datapath for the flow (and upstream of the segment which requires QoS guarantees via RSVP reservation).

- o processes the corresponding received RSVP messages (including Resv messages) as per regular RSVP.

- o synchronizes the RSVP reservation state with application level requirements and signaling.

Note that since the application entity encodes its own IP address as the RSVP PHOP in the Path message, the RSVP Router terminating the GRE tunnel naturally addresses all the RSVP messages travelling upstream hop-by-hop (such as Resv messages) to the application entity (without having to encapsulate those in a reverse-direction GRE tunnel towards the application entity).

4.5.2. Application_Entity-Controlled Proxy via Co-Location

This approach is simply a particular case of the more general Application_Entity-Controlled Proxy, but where the application entity is co-located with the RSVP Proxy. As an example, Session Border Controllers (SBC) with on-board SIP agents could implement RSVP Proxy functions and make use of such an approach to achieve session admission control over the SBC-to-SBC segment using RSVP signaling.

Figure 11 illustrates operations of the Application_Entity-Controlled RSVP Proxy via Co-location.

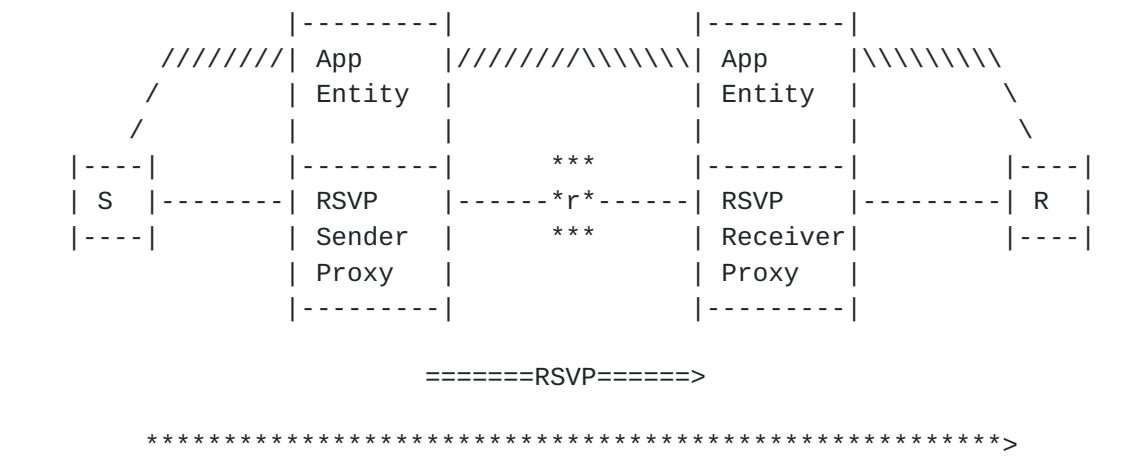


Figure 11: Application_Entity-Controlled Proxy via Co-Location

This RSVP Proxy approach does not require any protocol extensions. We also observe that when multiple sessions are to be established on paths sharing the same RSVP Sender Proxy and the same RSVP Receiver Proxy, the RSVP Proxies have the option to establish aggregate RSVP reservations (as defined in ([RFC3175] or [RFC4860]) for a group of sessions, instead of establishing one RSVP reservation per session.

4.6. Policy_Server-Controlled Proxy

In this approach, it is assumed that a Policy Server, which is located in the control plane of the network, controls an RSVP Proxy which is located in the datapath of the application flows (i.e. "on-path"). In turn, the Policy server is triggered by an entity involved in the application level signaling. With this approach, the RSVP Proxy does not attempt to determine itself the application reservation requirements, but instead is instructed by the Policy Server to establish, maintain and tear down reservations as needed by the application flows. Moreover, the entity participating in application level signaling does not attempt to understand the specific reservation mechanism (i.e. RSVP) or the topology of the network layer, but instead it simply asks the policy server to

perform (or teardown) a reservation. In other words, with this approach, the solution for synchronizing RSVP signaling with application level requirements is to rely on an application level entity that controls a policy server that, in turn, controls an RSVP Proxy function that sits in the flow datapath. This approach allows control of an RSVP Sender Proxy, an RSVP Receiver Proxy or both.

Operation of the Policy_Server-Controlled Proxy is illustrated Figure 12.

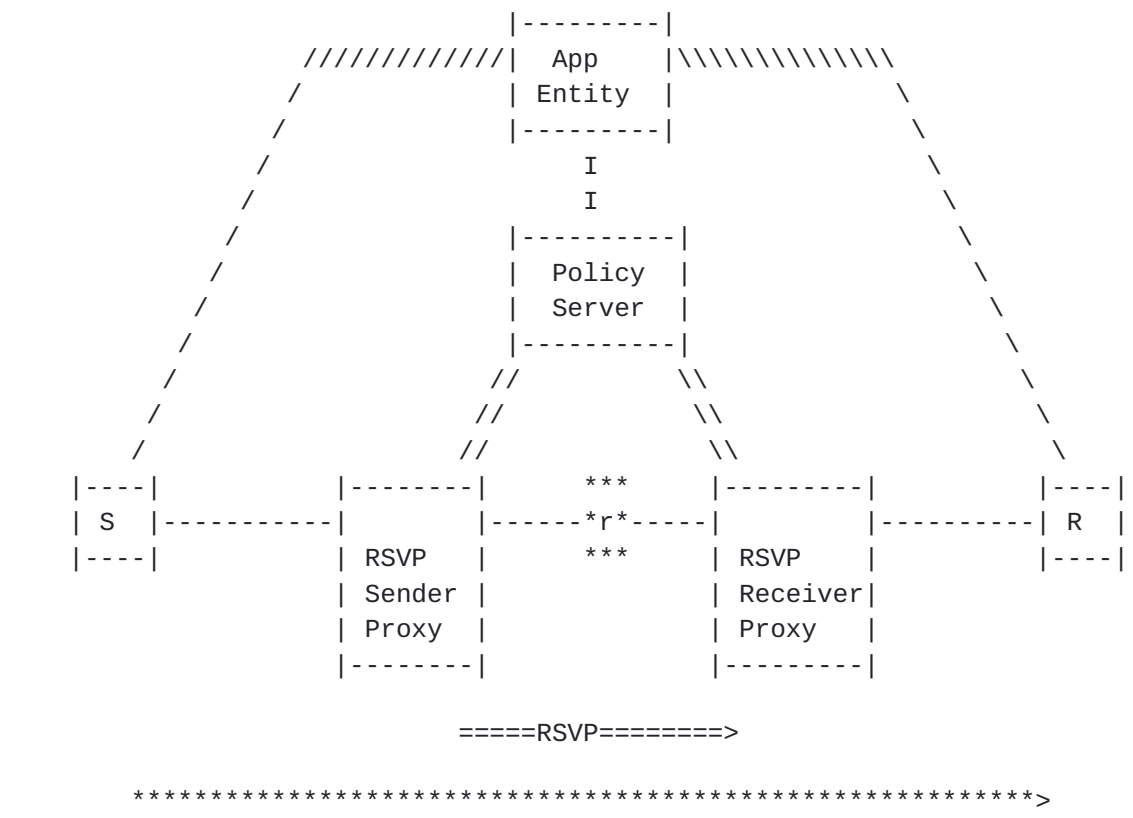


Figure 12: Policy_Server-Controlled Proxy

This RSVP Proxy approach does not require any extension to the RSVP protocol. However, as with the Application_Entity-Controlled Proxy approach presented in Figure 9, this approach relies on an RSVP Proxy control interface allowing control of the RSVP Proxy (by the Policy Server in this case). This RSVP Proxy control interface is beyond the scope of the present document. Considerations about candidate protocols for realizing such interface can be found in Section 4.5.

Again, for situations where only the RSVP Sender Proxy has to be controlled by this interface, the interface may be realized through the simple use of RSVP Itself, over a GRE tunnel from the Policy Server to the RSVP Sender Proxy. This is similar to what is presented in Section 4.5.1 except that the "RSVP over GRE" interface is used in this case by the Policy Server (instead of the application entity).

The interface between the Application Entity and the Policy Server is beyond the scope of this document.

4.7. RSVP-Signaling-Triggered Proxy

An RSVP Proxy can also be triggered and controlled through extended RSVP signaling from the remote end that is RSVP-capable (and supports these RSVP extensions for Proxy control). For example, an RSVP capable sender could send a new or extended RSVP message explicitly requesting an RSVP Proxy on the path towards the receiver to behave as an RSVP Receiver Proxy and also to trigger a reverse direction reservation thus also behaving as a RSVP Sender Proxy. The new or extended RSVP message sent by the sender could also include attributes (e.g. bandwidth) for the reservations to be signaled by the RSVP Proxy.

The challenges in these explicit signaling schemes include:

- How can the nodes determine when a reservation request ought to be proxied and when it should not, and accordingly invoke appropriate signaling procedures?

- How does the node sending the messages explicitly triggering the Proxy know where the Proxy is located, e.g., determine an IP address of the proxy that should reply to the signaling?

- How is all the information needed by a Sender Proxy to generate a Path message actually communicated to the Proxy?

An example of such a mechanism is presented in [I-D.manner-tsvwg-rsvp-proxy-sig]. This scheme is primarily targeted to local access network reservations whereby an end host can request resource reservations for both incoming and outgoing flows only over the access network. This may be useful in environments where the access network is typically the bottleneck while the core is comparatively over-provisioned, as may be the case with a number of radio access technologies. In this proposal, messages targeted to the Proxy are flagged with one bit in all RSVP messages. Similarly, all RSVP messages sent back by the Proxy are also flagged. The use of such a flag allows differentiating between proxied and end-to-end

reservations. For triggering an RSVP Receiver Proxy, the sender of the data sends a Path message which is marked with the mentioned flag. The Receiver Proxy is located on the signaling and data path, eventually gets the Path message, and replies back with a Resv message. A node triggers an RSVP Sender Proxy with a newly defined Path_Request message, which instructs the proxy to send Path messages towards the triggering node. The node then replies back with a Resv. More details can be found in [I-D.manner-tsvwg-rsvp-proxy-sig].

Such an RSVP-Signaling-Triggered Proxy approach would require RSVP signaling extensions (that are outside the scope of the present document). However it could provide more flexibility in the control of the Proxy behavior (e.g. control of reverse reservation parameters) than provided by the Path-Triggered approaches defined in Section 4.1 and Section 4.2.

4.8. Endsystem-Controlled Proxy

In some cases, having a full RSVP implementation running on an end host can be seen to produce excessive overhead. In end-hosts that are low in processing power and functionality, having an RSVP daemon run and take care of the signaling may introduce unnecessary overhead. One article [Kars01] proposes to create a remote API so that the daemon would in fact run on the end-host's default router and the end-host application would send its requests to that daemon. Thus, we can have deployments, where an end host uses some lightweight protocol to communicate with its pre-defined RSVP router - a form of RSVP proxy. Such a lightweight protocol is outside the scope of the present document.

4.9. Reachability Considerations

There may be situations where the RSVP Receiver Proxy is reachable by the sender, while the receiver itself is not. In such situations, it is possible that the RSVP Receiver Proxy is not always aware that the receiver is unreachable, and consequently may accept to establish an RSVP reservation on behalf of that receiver. This would result in unnecessary reservation establishment and unnecessary network resource consumption.

This is not considered a significant practical concern for a number of reasons. First, in many cases, if the receiver is not reachable from the sender, it will not be reachable either for application signaling so that application level session establishment will not be possible in the first place. Secondly, where the receiver is unreachable from the sender but is reachable for application level signaling (say because session establishment is performed through an off-path SIP agent that uses a different logical topology to

communicate with the receiver), then the sender may detect that the receiver is unreachable before attempting reservation establishment. This may be achieved through mechanisms such as ICE's connectivity check ([I-D.ietf-mmusic-ice]). Finally, even if the sender does not detect that the receiver is unreachable before triggering the RSVP reservation establishment, it is very likely that the application will quickly realise this lack of connectivity (e.g. the human accepting the phone call on the receiver side will not hear the human's voice on the sender side) and therefore tear down the session (e.g. hang up the phone) which in turn will trigger RSVP reservation release.

Nonetheless, it is recommended that network administrators consider the above in light of their particular environment when deploying RSVP Proxys.

The mirror considerations apply for situations involving an RSVP Sender Proxy and where the sender cannot reach the destination while the RSVP Sender Proxy can.

5. Security Considerations

In the environments of concern for this document, RSVP messages are used to control resource reservations on a segment of the end-to-end path of flows. To ensure the integrity of the associated reservation and admission control mechanisms, the cryptographic authentication mechanisms defined in [RFC2747]] and [RFC3097] can be used. Those protect RSVP messages integrity hop-by-hop and provide node authentication, thereby protecting against corruption, spoofing of RSVP messages and replay.
[I-D.behringer-tsvwg-rsvp-security-groupkeying] discusses key types, key provisioning methods as well as their respective applicability.

A number of additional security considerations apply to the use of RSVP proxies and are discussed below.

With some RSVP Proxy approaches, the RSVP proxy operates autonomously inside an RSVP router. This is the case for the Path-Triggered Proxy approaches defined in Section 4.1 and in Section 4.2, for the Inspection-Triggered Proxy approach defined in Section 4.3, for the STUN-Triggered Proxy approach defined in Section 4.4 and for the RSVP-Signaling-Triggered approach defined in Section 4.7. Proper reservation operation assumes that the RSVP proxy can be trusted to behave correctly in order to control the RSVP reservation as required and expected by the end systems. Since, the basic RSVP operation already assumes a trust model where end-systems trust RSVP nodes to appropriately perform RSVP reservations, the use of RSVP proxy that

behave autonomously within an RSVP router is not seen as introducing any significant additional security threat or as fundamentally modifying the RSVP trust model.

With some RSVP Proxy approaches, the RSVP proxy operate under the control of another entity. This is the case for the Application_Entity-Controlled Proxy approach defined in Section 4.5 and for the Policy_Server-Controlled Proxy approach defined in Section 4.6. This introduces additional security risks since the entity controlling the RSVP Proxy needs to be trusted for proper reservation operation. The exact mechanisms to establish such trust are beyond the scope of this document, but they may include security mechanisms inside the protocol used as the control interface between the RSVP Proxy and the entity controlling it, as well as security mechanisms for all the interfaces involved in the reservation control chain (e.g. inside the application signaling protocol between the end systems and the application entity, and, in the case of the Policy_Server-Controlled Proxy approach, in the protocol between the application entity and the policy server).

In some situations, the use of RSVP Proxy to control reservations on behalf of end-systems may actually reduce the security risk (at least from the network operator viewpoint). This could be the case, for example, because the routers where the RSVP Proxy functionality runs are less exposed to tampering than end-systems. Such a case is further discussed in section 4 of [I-D.ietf-tsvwg-rsvp-proxy-proto].

6. IANA Considerations

This document does not make any request to IANA registration.

7. Acknowledgments

This document benefited from earlier work on the concept of RSVP Proxy including the one documented by Silvano Gai, Dinesh Dutt, Nitsan Elfassy and Yoram Bernet. It also benefited from discussions with Pratik Bose, Chris Christou and Michael Davenport. Tullio Loffredo and Massimo Sassi provided the base material for Section 4.6.

8. Informative References

[I-D.behringer-tsvwg-rsvp-security-groupkeying]
Behringer, M. and F. Le Faucheur, "A Framework for RSVP Security Using Dynamic Group Keying", July 2007.

[I-D.ietf-behave-rfc3489bis]
Rosenberg, J., Huitema, C., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)", draft-ietf-behave-rfc3489bis-09 (work in progress), August 2007.

[I-D.ietf-mmusic-ice]
Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-17 (work in progress), July 2007.

[I-D.ietf-sipping-sbc-funcs]
Hautakorpi, J., Camarillo, G., Penfield, R., Hawrylyshen, A., and M. Bhatia, "Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments", April 2007.

[I-D.ietf-tsvwg-rsvp-proxy-proto]
Le Faucheur, L., "RSVP Extensions For Path-Triggered RSVP Receiver Proxy", February 2007.

[I-D.ietf-tsvwg-vpn-signaled-preemption]
Baker, F. and P. Bose, "QoS Signaling in a Nested Virtual Private Network", February 2007.

[I-D.manner-tsvwg-rsvp-proxy-sig]
Manner, J., "Localized RSVP for Controlling RSVP Proxies", October 2006.

[Kars01] Karsten, M., "Experimental Extensions to RSVP -- Remote Client and One-Pass Signalling", IWQoS Karlsruhe, Germany, 2006.

[RFC1633] Braden, B., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

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

[RFC2210] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

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

[RFC2961] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F., and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

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

[RFC3175] Baker, F., Iturralde, C., Le Faucheur, F., and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3175, September 2001.

[RFC3312] Camarillo, G., Marshall, W., and J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312, October 2002.

[RFC3525] Groves, C., Pantaleo, M., Anderson, T., and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4860] Le Faucheur, F., Davie, B., Bose, P., Christou, C., and M. Davenport, "Generic Aggregate Resource ReSerVation Protocol (RSVP) Reservations", RFC 4860, May 2007.

Appendix A. Use Cases for RSVP Proxies

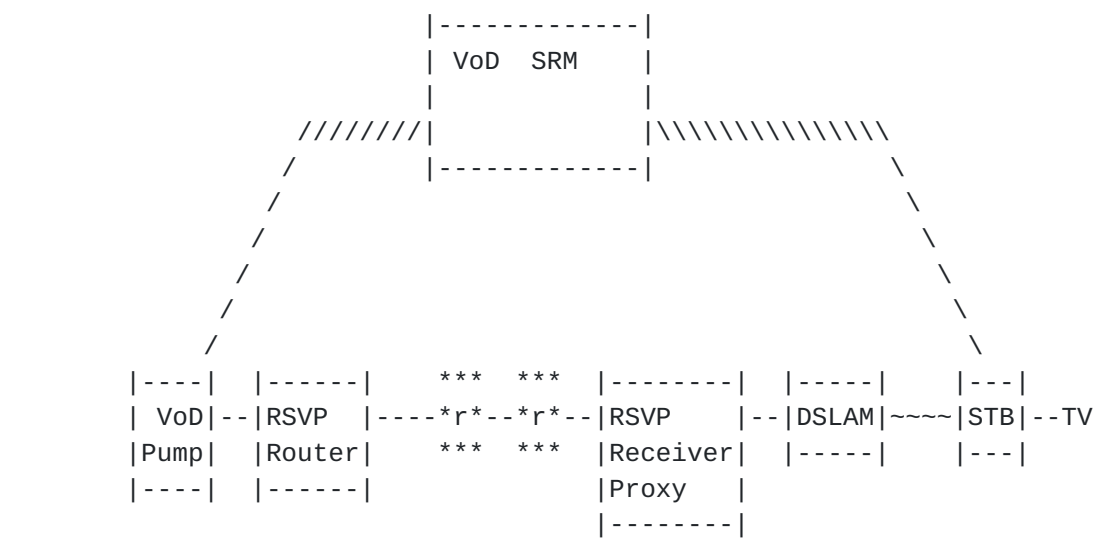
A.1. RSVP-based VoD CAC in Broadband Aggregation Networks

As broadband services for residential are becoming more and more prevalent, next generation aggregation networks are being deployed in order to aggregate traffic from broadband users (whether attached via Digital Subscriber Line technology aka DSL, Fiber To The Home/Curb aka FTTx, Cable or other broadband access technology). Video on Demand (VoD) services which may be offered to broadband users present significant capacity planning challenges for the aggregation network for a number of reasons. First each VoD stream requires significant dedicated sustained bandwidth (typically 2-4 Mb/s in Standard Definition TV and 6-12 Mb/s in High Definition TV). Secondly, the VoD codec algorithms are very sensitive to packet loss. Finally, the load resulting from such services is very hard to predict (e.g. it can vary very suddenly with block-buster titles made available as well as with promotional offerings). As a result, transport of VoD

streams on the aggregation network usually translate into a strong requirement for admission control. The admission control solution protects the quality of established VoD sessions by rejecting the additional excessive session attempts during unpredictable peaks, during link or node failures, or combination of those factors.

RSVP can be used in the aggregation network for admission control of the VoD sessions. However, since Customer Premises equipment such as Set Top Boxes (which behave as the receiver for VoD streams) often do not support RSVP, the last IP hop in the aggregation network can behave as an RSVP Receiver Proxy. This way, RSVP can be used between VoD Pumps and the last IP hop in the Aggregation network to perform accurate admission control of VoD streams over the resources set aside for VoD in the aggregation network (typically a certain percentage of the bandwidth of any link). As VoD streams are unidirectional, a simple "Path-Triggered" RSVP Receiver Proxy (as described in Section 4.1) is all that is required in this use case.

The Figure below illustrates operation of RSVP-based admission control of VoD sessions in an Aggregation network involving RSVP support on the VoD Pump (the senders) and RSVP Receiver Proxy on the last IP hop of the aggregation network. All the customer premises equipment remain RSVP unaware.



SRM Session Resource Manager

*** |---|
r regular RSVP |STB| Set Top Box
*** router |---|

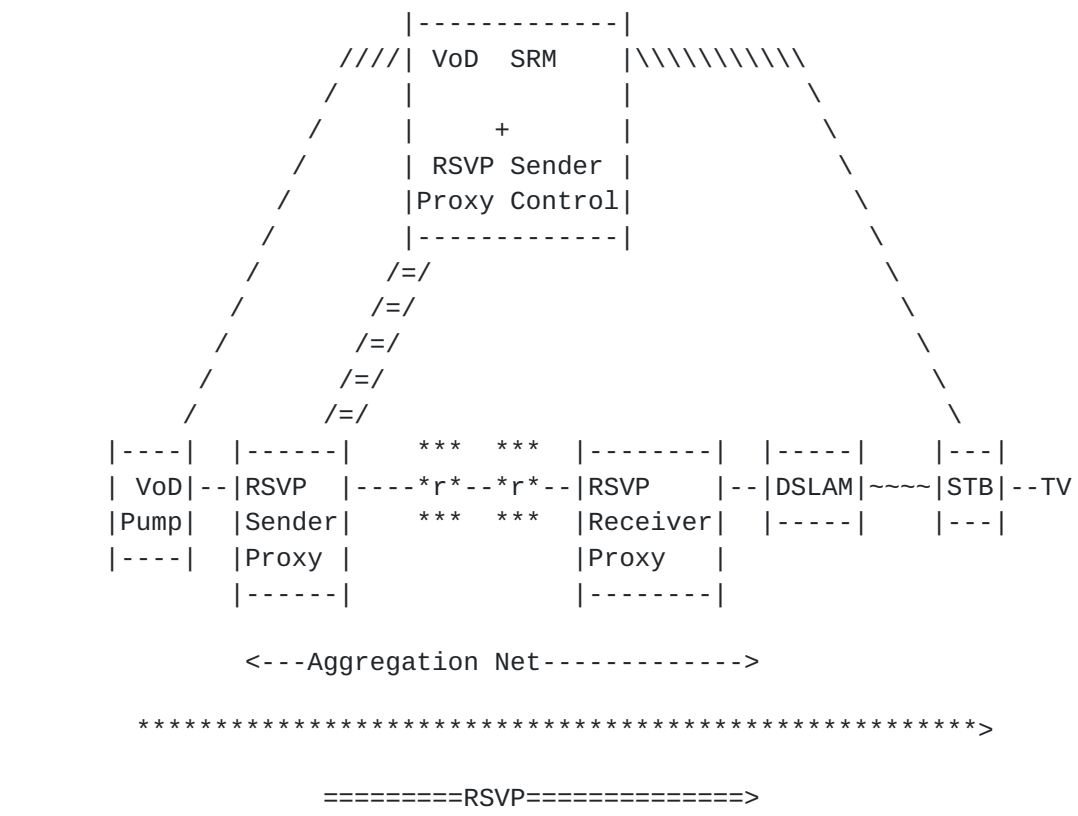
***> VoD media flow

==> segment of flow path protected by RSVP reservation

/\ VoD Application level signaling (e.g. RTSP)

Figure 13: VoD Use Case with Receiver Proxy

In the case where the VoD Pumps are not RSVP-capable, an Application_Entity-Controlled Sender Proxy via "RSVP over GRE" approach (as described in Section 4.5.1) can also be implemented on the VoD Controller or Session Resource Manager (SRM) devices typically involved in VoD deployments. Figure 14 illustrates operation of RSVP-based admission control of VoD sessions in an Aggregation network involving such Application_Entity-Controlled Source Proxy combined with an RSVP Receiver Proxy on the last IP hop of the aggregation network. All the customer premises equipment, as well as the VoD pumps, remain RSVP unaware.



SRM Systems Resource Manager

*** |---|
r regular RSVP |STB| Set Top Box
*** router |---|

***> VoD media flow

==> segment of flow path protected by RSVP reservation

/ VoD Application level signaling (e.g. RTSP)

/=/ GRE-tunnelled RSVP (Path messages)

Figure 14: VoD Use Case with Receiver Proxy and SRM-based Sender Proxy

The RSVP Proxy entities specified in this document play a significant role here since they allow immediate deployment of an RSVP-based admission control solution for VoD without requiring any upgrade to the huge installed base of non-RSVP-capable customer premises equipment. In one mode described above, they also avoid upgrade of non-RSVP-capable VoD pumps. In turn, this means that the benefits of

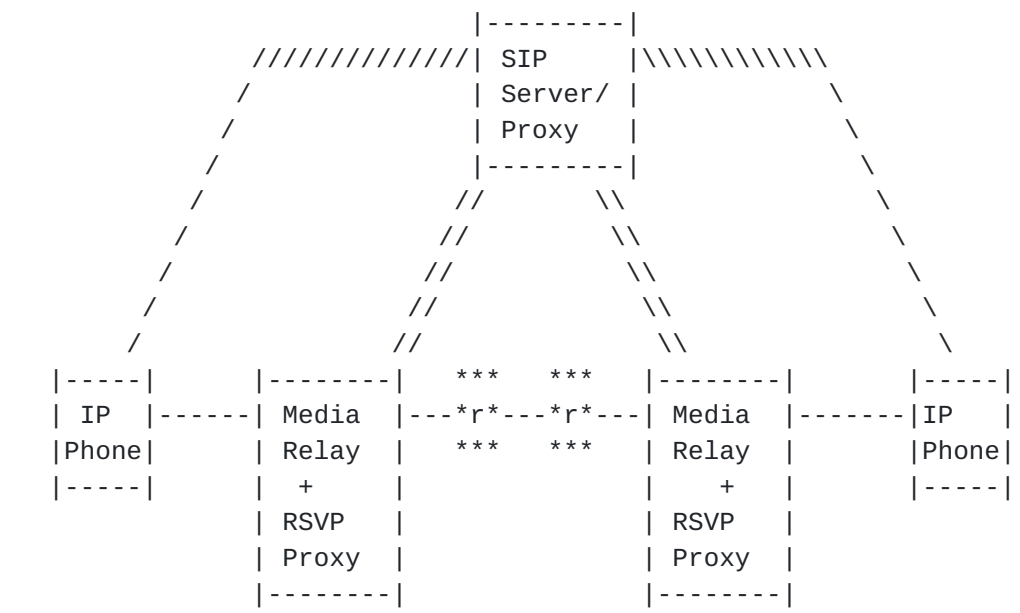
on-path admission control can be offered to VoD services over broadband aggregation networks without network or VoD Pump upgrade. Those include accurate bandwidth accounting regardless of topology (hub-and-spoke, ring, mesh, star, arbitrary combinations) and dynamic adjustment to any change in topology (such as failure, routing change, additional links...).

A.2. RSVP-based Voice/Video CAC in Enterprise WAN

More and more enterprises are migrating their telephony and videoconferencing applications onto IP. When doing so, there is a need for retaining admission control capabilities of existing TDM-based systems to ensure the QoS of these applications is maintained even when transiting through the enterprise's Wide Area Network (WAN). Since many of the endpoints already deployed (such as IP Phones or Videoconferencing terminals) are not RSVP capable, RSVP Proxy approaches are very useful: they allow deployment of an RSVP-based admission control solution over the WAN without requiring upgrade of the existing terminals.

A common deployment architecture for such environments relies on the Application_Entity-Controlled Proxy approach as defined in Section 4.5. Routers sitting at the edges of the WAN network and naturally "on-path" for all inter-campus calls (or sessions) and behave as RSVP Proxies. The RSVP Proxies establish, maintain and tear-down RSVP reservations over the WAN segment for the calls (or sessions) under the control of the SIP Server/Proxy. The SIP Server/Proxy synchronizes the RSVP reservation status with the status of end-to-end calls. For example, the called IP phone will only be instructed to play a ring tone if the RSVP reservations over the corresponding WAN segment has been successfully established.

This architecture allowing RSVP-based admission control of voice and video on the Enterprise WAN is illustrated in Figure 15.



***
r Regular RSVP router
***

<***> media flow

<==> segment of flow path protected by RSVP reservation

/\ SIP signaling

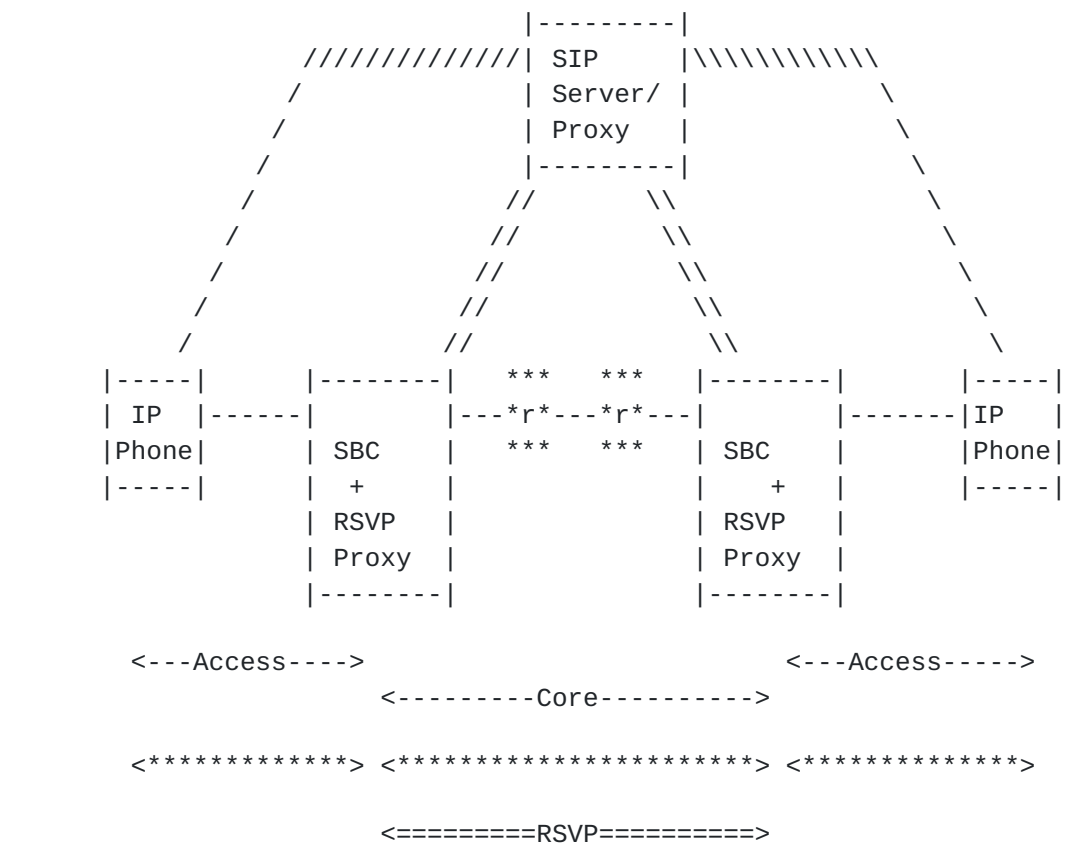
// control interface between the SIP Server/Proxy and RSVP Proxy

Figure 15: CAC on Enterprise WAN Use Case

A.3. RSVP-based Voice CAC in Telephony Service Provider Core

Let us consider an environment involving a Telephony Service Provider (TSP). Let us further assume that end-users are attached to the TSP via Session Border Controllers (SBCs). The SBCs may be remotely controlled by a SIP Server. The SIP Server may control establishment

of RSVP reservations between the SBCs for admission control of sessions over the core. This relies on the Application_Entity-Controlled RSVP Proxy approach presented in Section 4.5. This is illustrated in the Figure below.



***
r Regular RSVP router
***

<***> media flow

<==> segment of flow path protected by RSVP reservation

/\ SIP signaling

// control interface between the SIP Server/Proxy and SBC/RSVP Proxy

Figure 16: Voice CAC in TSP Domain

A.4. RSVP Proxies for Mobile Access Networks

Mobile access networks are increasingly based on IP technology. This implies that, on the network layer, all traffic, both traditional data and streamed data like audio or video, is transmitted as packets. Increasingly popular multimedia applications would benefit

from better than best-effort service from the network, a forwarding service with strict Quality of Service (QoS) with guaranteed minimum bandwidth and bounded delay. Other applications, such as electronic commerce, network control and management, and remote login applications, would also benefit from a differentiated treatment.

The IETF has two main models for providing differentiated treatment of packets in routers. The Integrated Services (IntServ) model [RFC1633] together with the Resource Reservation Protocol (RSVP) [RFC2205] [RFC2210] [RFC2961] provides per-flow guaranteed end-to-end transmission service. The Differentiated Services (DiffServ) framework [RFC2475] provides non- signaled flow differentiation that usually provides, but does not guarantee, proper transmission service.

However, these architectures have potential weaknesses for deployment in Mobile Access Networks. For example, RSVP requires support from both communication end points, and the protocol may have potential performance issues in mobile environments. DiffServ can only provide statistical guarantees and is not well suited for dynamic environments.

Let us consider a scenario, where a fixed network correspondent node (CN) would be sending a multimedia stream to an end host behind a wireless link. If the correspondent node does not support RSVP it cannot signal its traffic characteristics to the network and request specific forwarding services. Likewise, if the correspondent node is not able to mark its traffic with a proper DiffServ Code Point (DSCP) to trigger service differentiation, the multimedia stream will get only best-effort service which may result in poor visual and audio quality in the receiving application. Even if the connecting wired network is over-provisioned, an end host would still benefit from local resource reservations, especially in wireless access networks, where the bottleneck resource is most probably the wireless link.

RSVP proxies would be a very beneficial solution to this problem. It would allow distinguishing local network reservations from the end-to-end reservations. The end host does not need to know the access network topology or the nodes that will reserve the local resources. The access network would do resource reservations for both incoming and outgoing flows based on certain criterion, e.g., filters based on application protocols. Another option is that the mobile end host makes an explicit reservation that identifies the intention and the access network will find the correct local access network node(s) to respond to the reservation. RSVP proxies would, thus, allow resource reservation over the segment which is the most likely bottleneck, the wireless connectivity. If the wireless access network uses a local mobility management mechanism, where the IP address of the mobile

node does not change during handover, RSVP reservations would follow the mobile node movement.

A.5. RSVP Proxies for Reservations in the presence of IPsec Gateways

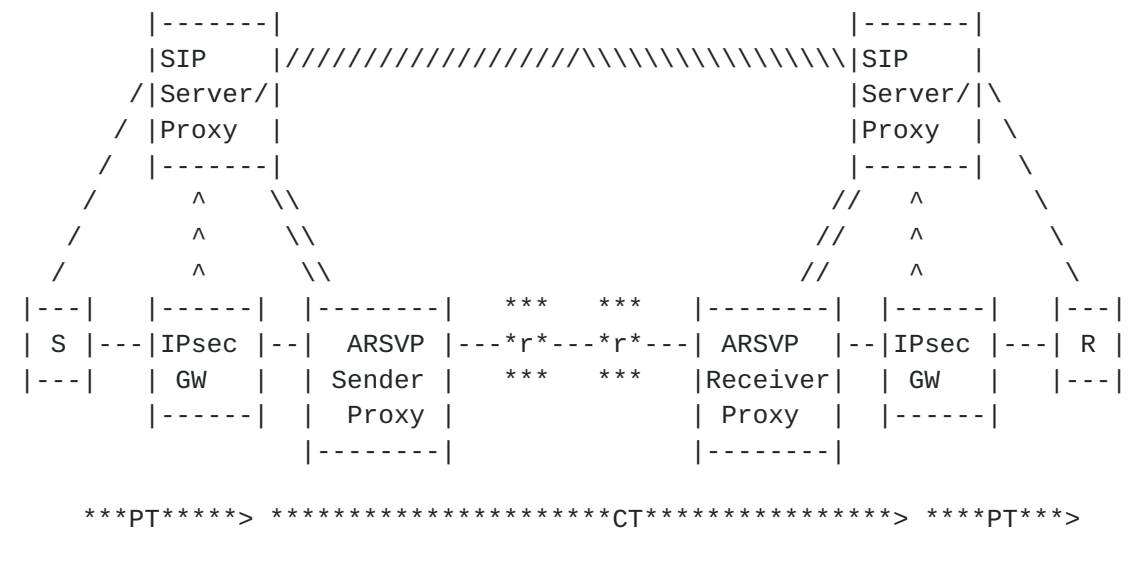
[I-D.ietf-tsvwg-vpn-signaled-preemption] discusses how resource reservation can be supported end-to-end in a nested VPN environment. At each VPN level, VPN Routers behave as [RFC4301] security gateways between a plaintext domain and a cyphertext domain. To achieve end-to-end resource reservation, the VPN Routers process RSVP signaling on the plaintext side, perform aggregation of plaintext reservations, and maintain the corresponding aggregate RSVP reservations on the cyphertext side. Each aggregate reservation is established on behalf of multiple encrypted end-to-end sessions sharing the same ingress and egress VPN Routers. These aggregate reservations can be as specified in [RFC3175] or [RFC4860].

Section 3 of [I-D.ietf-tsvwg-vpn-signaled-preemption] discusses the necessary data flows within a VPN Router to achieve the behavior described in the previous paragraph. Two mechanisms are described to achieve such data flows. Section 3.1 presents the case where the VPN Router carries data across the cryptographic boundary. Section 3.2 discusses the case where the VPN router uses a Network-Guard.

Where such mechanisms are not supported by the VPN Routers, the approach for end-to-end reservation presented in [I-D.ietf-tsvwg-vpn-signaled-preemption] cannot be deployed. An alternative approach to support resource reservations within the cyphertext core is to use the "Application_Entity-Controlled Proxy" approach (as defined in Section 4.5) in the following way:

- the RSVP Proxies are located inside the cyphertext domain and use aggregate RSVP reservations,
- the Application Entity exchange application level signaling with the end systems in the plaintext domain,
- the Application Entity controls the RSVP Proxies in the cyphertext domain via an RSVP Proxy control interface

This is illustrated in Figure 17 in the case where the application is SIP-based multimedia communications.



|----| RSVP-capable Sender
S

|----| RSVP-capable Receiver
R

*** *r* regular RSVP *** router

|------|
|IPsec | IPsec security gateway
GW

ARSVP Aggregate RSVP

***> media flow

==> segment of flow path protected by RSVP reservation

/ \ SIP signaling

^ Network management interface between SIP Server/Proxy and IPsec security gateway

// control interface between SIP Server/Proxy and ARSVP Proxy

PT Plaintext network

CT Cyphertext network

Figure 17: RSVP Proxies for Reservations in the Presence of IPsec

Gateways

Where the sender and receiver are RSVP capable, they may also use RSVP signaling. This achieves resource reservation on the plaintext segments of the end-to-end i.e. :

- o from the sender to the ingress IPsec gateway and
- o from the egress IPsec gateway to the receiver.

In this use case, because the VPN Routers do not support any RSVP specific mechanism, the end-to-end RSVP signaling is effectively hidden by the IPsec gateways on the cyphertext segment of the end-to-end path.

As with the "Application_Entity-Controlled Proxy" approach (defined in Section 4.5), the solution here for synchronizing RSVP signaling with application-level signaling is to rely on an application-level signaling device that controls an on-path RSVP Proxy function. However, in the present use case, the RSVP Proxies are a component of a cyphertext network where all user (bearer) traffic is IPsec encrypted. This has a number of implications including the following:

1. encrypted flows can not be identified in the cyphertext domain so that network nodes can only classify traffic based on IP address and DiffServ Code Points (DSCPs). As a result, only aggregate RSVP reservations (such as those specified in [RFC3175] or [RFC4860]) can be used. This is similar to [I-D.ietf-tsvwg-vpn-signaled-preemption].

2. Determining the RSVP Sender proxy and RSVP receiver Proxy to be used for aggregation of a given flow from sender to receiver creates a number of challenges. Details on how this may be achieved are beyond the scope of this document. We observe that, as illustrated in Figure 17, this may be facilitated by a network management interface between the application entity and the IPsec gateways. For example, this interface may be used by the application entity to obtain information about which IPsec gateway is on the path of a given end-to-end flow. Then, the application entity may maintain awareness of which RSVP Proxy is on the cyphertext path between a given pair of IPsec gateways. How such awareness is achieved is beyond the scope of this document. We simply observe that such awareness can be easily achieved through simple configuration in the particular case where a single (physical or logical) RSVP Proxy is fronting a given IPsec gateway. We also observe that when awareness of the RSVP Receiver Proxy for a particular egress IPsec gateway (or

end-to-end flow) is not available, the aggregate reservation may be signaled by the RSVP Sender Proxy to the destination address of the egress IPsec gateway and then proxied by the RSVP Receiver Proxy.

Different flavors of operations are possible in terms of aggregate reservation sizing. For example, the application entity can initiate an aggregate reservation of fixed size a priori and then simply keep count of the bandwidth used by sessions and reject sessions that would result in excess usage of an aggregate reservation. The application entity could also re-size the aggregate reservations on a session by session basis. Alternatively, the application entity could re-size the aggregate reservations in step increments typically corresponding to the bandwidth requirement of multiple sessions.

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
University of Helsinki
P.O. Box 68
University of Helsinki FIN-00014 University of Helsinki
Finland

Phone: +358 9 191 51298
Email: jmanner@cs.helsinki.fi
URI: http://www.cs.helsinki.fi/u/jmanner/

Dan Wing
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
United States

Email: dwing@cisco.com

Allan Guillou
Neuf Cegetel
40-42 Quai du Point du Jour
Boulogne-Billancourt, 92659
France

Email: allan.guillou@neufcegetel.fr

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).