Developing RSVP-Enabled Firewalls

Author: Jamie Jason
jamie_jason@mailbox.jf.intel.com

1. Overview

Every day, new users make the Internet their own virtual playground. Add to that new, bandwidth-hungry network multimedia applications using an already-congested backbone, and applications begin experiencing higher packet loss rates and increased latencies. Obviously, more backbone capacity would alleviate some of the congestion, but another viable alternative for users who are willing to "pay the price" is to allow applications to request from the network (and receive, if resources are available) assured[1] quality of service (QoS).

As it so happens, the Resource Reservation Setup Protocol Working Group within the Internet Engineering Task Force (IETF) has a proposed standard, the Resource ReSerVation Protocol (RSVP) [RFC2205], for setting up resource reservations in the Internet. Several references in this paper refer to Internet Drafts. Internet Drafts are only valid for six months and are considered "work in progress". Any information referenced by an Internet Draft should be treated as such.

The purpose of this paper is to look at RSVP and present the issues that firewall vendors should be concerned with when determining how to add support to their products. Having said that, the target audience for the paper is the firewall developer/engineering manager who is estimating the effort necessary to add RSVP support to a firewall product. As such, this paper assumes that the reader is familiar with firewall technology and terminology. If not, there are several good treatments of the subject [CHAP95][CHES94][SIYAN95]. This paper is not meant to be a complete description of RSVP. For more details, consult the Request For Comments (RFC) and Internet Drafts submitted by the RSVP Working Group. The RFCs and Internet Drafts can be obtained from *http://www.ietf.org/html.charters/rsvp-charter.html*.

To be consistent with the RSVP specification, any reference to directional terms (for example, upstream, downstream, previous hop, next hop, etc.) are with respect to the direction of data flow.

2. Introduction to RSVP

2.1 What is RSVP?

RSVP is a receiver-oriented resource reservation establishment protocol used by both hosts and routers. A host, on behalf of an application data stream, uses RSVP to request resources for simplex (that is, unidirectional) flows. Routers use RSVP to distribute QoS control requests to the nodes along the path(s) of the flows to establish and maintain state to provide the requested service. Since RSVP requests resources in one direction, it treats a sender as being logically distinct from a receiver, even though an application may be sending and receiving at the same time. Since RSVP reserves resources for simplex flows, a duplex resource reservation requires the establishment of two independent simplex RSVP reservations (one in each direction).

RSVP employs "soft state" to manage reservation states in routers and hosts. This means that RSVP requires periodic refresh messages to maintain the state along the reserved path(s). If refresh messages are not sent, state will automatically time out and will be deleted. If a node times out a state, it will generate the appropriate teardown message(s). However, since RSVP messages are not delivered reliably, it is possible that teardown messages will not be seen. This is not a problem as this will cause one or more of the RSVP nodes that did not receive the teardown to time out its state and generate its own teardown.

[1] The word *assured* is a result of comments from Tim O'Malley, an individual who is active in the RSVP Working Group. The reader should interpret assured to mean that the RSVP-enabled network elements will do their best to deliver the promised QoS. Routing changes or network outages make it virtually impossible to *guarantee* the QoS.

Since it's not possible to enable RSVP on the entire Internet instantaneously, RSVP must be able to function in an environment in which there may be non-RSVP clouds. A non-RSVP cloud is one or more routers along the path that are not able to support RSVP and therefore do not support RSVP QoS guarantees. It is possible that these routers support QoS in another way, or in the worst case only give best effort service to all traffic. This has no impact upon the ability for RSVP-enabled hops to do their jobs. However, if there is congestion in the non-RSVP cloud, this will have an adverse effect upon end-to-end QoS.

2.2 Protocol Overview

RSVP is positioned in the protocol stack at the transport layer, operating on top of IP (either IPv4 or IPv6). However, unlike other transport protocols, RSVP does not transport application data but instead acts like other Internet control protocols (for example, ICMP, IGMP, routing protocols). RSVP messages are sent hop-by-hop between RSVP-capable routers as raw IP datagrams using protocol number 46. It is intended that raw IP datagrams should be used between the end systems and the first (or last) hop router. However, this may not always be possible as not all systems can do raw network I/O. Because of this, it is possible to encapsulate RSVP messages within UDP datagrams for end-system communication. UDP-encapsulated RSVP messages are sent to either port 1698 (if sent by an end system) or port 1699 (if sent by an RSVP-enabled router). For more information concerning UDP encapsulation of RSVP messages, consult Appendix C of the RSVP specification.

2.2.1 RSVP Session

An RSVP session, a data flow with a particular destination and transport-layer protocol, is defined by:

- Destination Address - the destination IP address for the data packets. This may be either a unicast or a multicast address.
- Protocol ID - the IP protocol ID (for example, UDP or TCP).
- Destination Port - a generalized destination port which is used for demultiplexing at a layer above the IP layer. An example is the destination port used by UDP and TCP. It should be noted that the current RSVP specification only supports UDP/TCP and thus makes certain assumptions regarding the "generalized" destination port. This assumption has lead to problems, as described in Section 3.6.

An RSVP session specifies a particular receiver (or receivers in the case of multicast) that may wish to establish a resource reservation.

2.2.2 Flow Descriptors

A simple receiver generated RSVP reservation request consists of the following information:

- A flowspec, which contains the desired QoS (the Reservation Specification, or "Rspec") and a description of the data flow (the Traffic Specification, or "Tspec"). Consult [RFC2210] for information on the format and contents of the Rspec and Tspec objects.
- A filter spec, which specifies a subset of the packets of the data stream in the session. In the current RSVP specification, filter spec is restricted to the sender IP address and optionally the UDP/TCP source port.

Together, these two pieces of information are referred to as the flow descriptor.

A filter spec, together with a session specification as described in the previous section, defines which data flow is to receive the QoS as defined by the flowspec. If data packets addressed for the particular session do not match any of the filter specs for the session, then they are handled as best-effort traffic.

2.2.3 Reservation Styles

A reservation style is a combination of two options:

- How reservations for different senders within the same session are to be treated. The options are to either establish a distinct reservation for each sender or make a single shared reservation for all of the packets of selected senders.

- How senders are selected. An explicit list of senders or a wildcard, that implicitly selects all senders for the session, may be specified. If an explicit list is provided, then each filter spec in the list can match one and only one sender.

The following table presents the different combinations of reservation style options (columns) and sender selection (rows).

	Distinct Reservation	Shared Reservation
Explicit Selection	Fixed-Filter Style	Shared-Explicit Style
Wildcard Selection	None Defined	Wildcard-Filter Style

Table 1 - Reservation Styles

- Wildcard-Filter Style - a combination of a shared reservation and a wildcard sender selection. This creates a single reservation which is shared by all flows from upstream senders to a particular session.
- Fixed-Filter Style - a combination of a distinct reservation and an explicit sender selection. This creates a reservation for a flow from a particular sender to a particular session. Other senders, sending to the same session, do not share the reservation.
- Shared-Explicit Style - a combination of a shared reservation and an explicit sender selection. This creates a single reservation that is shared by explicitly selected (as specified by the filter spec(s)) senders.

2.2.4 Message Integrity

RSVP messages may optionally contain an Integrity object. The purpose of the integrity object is to provide cryptographic data to authenticate the originating node and to verify the contents of the RSVP message. [BAKER97] contains a description of the Integrity object as well as a method for employing it.

2.2.5 Policy Control

RSVP messages may also optionally contain a Policy Data object [HERZOG97], which carries information that a local policy module would use when determining whether or not an associated reservation is administratively permitted.

2.2.6 Reservation Establishment

The general model for the way in which a resource reservation is made in RSVP is as follows:

1. An RSVP sender transmits an RSVP Path message downstream to the receiver(s) of the data stream. The Path message may either be unicast to a single receiver or multicast to a group of receivers (that is, multicast data flows use multicast RSVP Path messages). The source and destination addresses in the Path messages are the same as those that are used for the data flows. The Path message is what defines the RSVP session.
2. Along the way, each RSVP-enabled node stores state for the path. This state information must include at a minimum the IP address of the previous RSVP-enabled hop (this information can be found in the path message).
3. The RSVP receiver transmits via unicast an RSVP Resv (reservation) message upstream to the previous RSVP-enabled hop. RSVP Resv messages are always unicast and must follow exactly the reverse path of RSVP-enabled routers. The flow descriptors in the Resv message define the reservation.
4. As long as the reservation can be satisfied (as ensured by the RSVP-enabled routers), each RSVP-enabled hop along the reverse path unicasts the Resv message to its previous RSVP-enabled hop until either the Resv message reaches the sender or the reservation is merged into an existing reservation. In short, merging is the process by which an RSVP Resv message, forwarded to a previous hop, contains a flowspec that is the "largest" of the flowspecs requested by the next hops to which the data flow is being sent. The topic of merging RSVP reservations is more adequately covered in the RSVP specification.
5. Optionally, an RSVP receiver may request a confirmation message to indicate that the request was (probably) granted. Note: receipt of a confirmation message is not a guarantee that the requested reservation is in place

all the way between the receiver and the sender. See the Sections 1.2 and 2.6 of the RSVP specification for an explanation of this.

2.2.7 Messages

As of version 1 of the RSVP specification, there are seven message types, with Path and Resv being the two fundamental message types. An RSVP packet contains a common header, followed by a variable number of variable-length objects. Consult section 3 and Appendix A of the RSVP specification for the layout of the common header as well as the layout of individual message types. The following table presents each RSVP message type, the purpose of the message, and a list of the fields in the message that firewall designers may care about.

Message	Purpose	Important Objects
Path	Used to store path state in node(s) between the sender and receiver(s).	• An optional RSVP Integrity object (see Section 2.2.4). • The RSVP session (see Section 2.2.1). • The IP and Logical Interface Handle (LIH) from which the RSVP Path message was most recently sent. • The IP and port of the sender of the data stream. • An optional Policy Data object (see Section 2.2.5).
Reservation	Used by a receiver to request a resource reservation.	• An optional RSVP Integrity object (see Section 2.2.4). • The RSVP session (see Section 2.2.1). • The IP and LIH from which the RSVP Resv was most recently sent. • An optional RSVP confirmation which contains the IP address where the RSVP Confirm message should be sent. • The reservation style (see Section 2.2.3). • A flow descriptor list (see Section 2.2.2). • An optional Policy Data object (see Section 2.2.5).
Path Tear[2]	Used to tear down path state along the path from sender to receiver. Note that this also deletes any dependent reservation state along the way. A Path Tear message must be routed exactly like its corresponding Path message.	Same as RSVP Path message (minus the Policy Data object).

[2] Because RSVP messages are not delivered reliably, there is no guarantee that a node will receive this message. However, soft state requires that a node eventually time out RSVP state and initiate its own teardown message.

Message	Purpose	Important Objects
Reservation Tear[2]	Used to tear down reservation state along the reverse path. A Resv Tear message must be routed like its corresponding Resv message.	• An optional RSVP Integrity object (see Section 2.2.4). • The RSVP session (see Section 2.2.1). • The IP and LIH from which the RSVP Resv was most recently sent. • The reservation style (see Section 2.2.3). • A flow descriptor list (see Section 2.2.2).
Path Error	Used to report errors in processing of path messages. The IP destination address of the message is the unicast address of the previous hop.	• An optional RSVP Integrity object (see Section 2.2.4). • The RSVP session (see Section 2.2.1). • The IP and port of the sender of the data stream. • An optional Policy Data object (see Section 2.2.5).
Reservation Error	Used to report errors in processing of reservation messages or the disruption of a reservation (for example, by some sort of administrative preemption). The IP destination address of the message is the unicast address of the next hop.	• An optional RSVP Integrity object (see Section 2.2.4). • The RSVP session (see Section 2.2.1). • The IP and LIH from which the RSVP Resv was most recently sent. • The reservation style (see Section 2.2.3). • Flow descriptor(s) (see Section 2.2.2) in error. • An optional Policy Data object (see Section 2.2.5).
Reservation Confirmation	Used to (probabilistically) acknowledge a reservation request.	• An optional RSVP Integrity object (see Section 2.2.4). • The RSVP session (see Section 2.2.1). • The IP address that originated the message. • The IP address to which the RSVP Confirm message is destined. • The IP and LIH from which the RSVP Resv was most recently sent. • The reservation style (see Section 2.2.3). • Flow descriptor(s) (see Section 2.2.2) for reservations that are being confirmed.

Table 2 - RSVP Message Types

3. Supporting RSVP

Depending upon the type of firewall, adding support for RSVP in a firewall may not be as simple as allowing through RSVP packets (whether they be raw IP datagrams or UDP encapsulated). This section presents issues with supporting RSVP in a firewall, including:

- Security threats posed by RSVP
- Support for multicast RSVP
- Granularity of control when allowing/denying RSVP
- Network Address Translation and RSVP

- Proxies and RSVP
- IPSEC and RSVP
- Encrypted Tunnels and RSVP

In addition to the above issues, thought should be given to whether or not the firewall will attempt to preserve the QoS guarantee as packets are forwarded through the firewall. Having said that, it cannot be stated strongly enough that firewalls should make every attempt to conform to QoS guarantees. This is obviously more of an issue for proxy-based firewalls, as data packets may be required to travel all the way up and then down the TCP/IP stack. However, even proxies should be able to conform to certain QoS bounds (on delay for instance).

3.1 Security threats

While RSVP may not seem to be a serious security threat, there are still some possible ways in which RSVP can be exploited:

1. RSVP messages may contain a NULL object. According to the RSVP specification - "A NULL object may appear anywhere in a sequence of objects, and its contents will be ignored by the receiver." This means that it would be possible to "hide" data in the NULL object and the message would still be considered valid by RSVP-enabled hops. This makes it possible for an outside user, in concert with an inside user, to covertly exchange information. It may also be possible, depending upon the implementation of RSVP, to exploit some bug in the software (for instance, an RSVP implementation may crash when it encounters NULL objects). Firewall designers may wish to strip NULL objects from the RSVP message.
2. In order to allow for new RSVP message object types in the future, and still ensure that older implementations of RSVP will still work with these new message types, it is possible to use an Unknown class object. If the two most-significant bits of the class number are set, then nodes that do not understand the class number are required by the specification to forward the object unexamined and unmodified. As with the NULL object, this could be a mechanism for covertly exchanging information or exploiting an implementation bug. Firewall designers may wish to strip Unknown objects from the RSVP message.
3. It is possible to mount a denial-of-service attack. Since RSVP-enabled hops maintain state about paths, one method would be to flood the network with RSVP Path messages that all have different sender Tspecs, thus preventing the hops from establishing path state information for other valid paths. Once a reservation is in place, sending a large number of reservations, each with a different flowspec, would cause the routers along the reverse path to have to constantly change the reservation. Dependent upon network policy, it would also be possible to attempt to reserve as many network resources as possible in an attempt to deny other valid streams from obtaining reservations. As with other denial-of-service attacks, RSVP denial-of-service attacks are hard to prevent. If a company were to generate revenue by providing QoS, these denial-of-service attacks would negatively impact the company financially. The use of Policy Data and Integrity objects may be able to alleviate some of the impact of denial-of-service attempts.

3.2 Multicast

RSVP is equally well-suited for multicast or unicast data streams. In fact, for multicast data streams, RSVP relies on the same multicast distribution tree for propagating RSVP Path messages. If a firewall cannot support multicast, then it obviously will only be able to support unicast RSVP. Care must be taken when supporting RSVP with multicast. In the section on UDP encapsulation in the RFC specification, it specifies when multicast RSVP messages should be sent to the group address (when using raw IP) or to the well-known RSVP group address (when using UDP encapsulation).

Currently, there are numerous problems just in trying to provide multicast support in firewalls. [CHOU97] presents some of these problems. Other problems, such as supporting multicast with encryption, are still unsolved problems.

3.3 Granularity of control

At the firewall, a decision can be made at several levels as to whether or not an RSVP message should pass through. This can be thought of as the granularity of control that the firewall exerts. Listed are several different levels of granularity:

- All or Nothing. This is obviously the least granular. It either allows or denies all RSVP messages.
- Allow/Deny reservations in a particular direction. For example, a decision may be made to only allow internal hosts to make RSVP reservations, while external users may not. Denying RSVP Resv messages from one direction implies that the firewall should block RSVP Path messages from the other direction (that is, if external hosts are not allowed to make RSVP reservations, then RSVP Path messages generated by internal hosts should probably not be passed through the firewall).
- Allow/Deny reservations from/to particular internal/external hosts.
- Allow/Deny reservations from/to particular users/class of users. This assumes that there is some way in which the firewall is able to make such a distinction.
- Deny reservations above a certain threshold. RSVP-enabled routers are able to make decisions based upon currently available resources as to whether or not to honor a reservation request. This is above and beyond what a firewall does, but this does not preclude a firewall from working in concert with some sort of bandwidth management entity.

The above is by no means an exhaustive list, and obviously some firewall configurations are better suited to support higher levels of granularity than others (for example, a filtering router may be well suited to permit/deny RSVP messages, but is ill-suited to make a decision based upon users). There can be many combinations based upon the level of access control that the firewall provides, and it is expected that the granularity of control a firewall allows is a way to differentiate products. Additionally, firewall designers may wish to use the Policy Data object (if present) to aid in making decisions regarding what is permitted/denied.

3.4 Network Address Translation (NAT)

NAT-based firewalls are presented with unique problems when it comes to supporting RSVP. These problems can be traced back to the purpose of a NAT-based firewall - the ability to present to an external host a different IP address for an internal host than what is used on the internal network. Two things in RSVP conspire against NAT-based firewalls:

1. Several RSVP message objects contain IP addresses. The result is that the firewall must be able to replace the IP addresses based upon the direction and type of the message. For example, if an external sender were to send an RSVP Path message to an internal receiver, the RSVP Session will specify the IP address that the external sender believes is the IP address of the internal receiver. However, when the RSVP Path message reaches the firewall, the RSVP Session must be changed to reflect the IP address that is used internally for the receiver. Similar actions must be taken for all message objects (see Table 2) that contain IP addresses.
2. RSVP provides a means, the RSVP Integrity object, to guarantee the integrity of RSVP messages. The problem is that because of the first point, a NAT-based firewall must be able to change IP addresses within the RSVP messages. However, when this is done, the RSVP Integrity object is no longer valid as the RSVP message has been changed. This means that the firewall must be prepared to verify the old RSVP Integrity object, and to create a new RSVP Integrity object and place it in the RSVP message.

3.5 Proxies

Proxies, both circuit-level and application-level, are in an interesting position when it comes to RSVP. The true sender of a data stream views the proxy as the receiver, whereas the true receiver of the data stream views the proxy as the sender. In effect, what the network sees is a reservation between the true sender and the proxy, and another unrelated reservation between the proxy and the true receiver. As mentioned earlier, a proxy can still conform to some QoS bounds (delay, for example). By adjusting the Adspec parameters based upon the characteristics of the proxy, the proxy can ensure that any reservation takes into account the impact it may have on the data stream and therefore still provide an end-to-end QoS.

3.5.1 Example Proxy Implementation

During the development of an H.323 proxy at Intel, a decision was made to add support for RSVP to the version of the proxy for Microsoft Windows NT* 4.0. The proxy uses the following two policies with respect to RSVP:

1. Internal users (that is, behind the proxy) are allowed/denied the ability to request RSVP reservations.
2. External users (that is, not behind the proxy) are allowed/denied the ability to request RSVP reservations.

H.323 can support multipoint calls, however the proxy supports only point-to-point calls.

A point-to-point H.323 call consists of several UDP data streams:

- two unidirectional audio streams (one from each endpoint).
- a bi-directional audio control stream.
- up to two unidirectional video streams (one from each endpoint). If an endpoint does not have the video capture equipment necessary to send video, it obviously won't be able to send a video stream.
- a bi-directional video control stream (if there is at least one unidirectional video stream).

To either endpoint, the proxy appears to be the sink/source of the data streams. What the proxy must do is maintain a mapping of UDP streams so that when a packet arrives from a particular source stream, it can determine where it is to be forwarded. Since all UDP datagrams flow through the proxy, the proxy must also be prepared to handle RSVP messages, as a receiver may wish to request guaranteed QoS for any of the unidirectional data streams. The proxy does not guarantee any level of QoS as packets are forwarded through the proxy. This does not mean it is impossible guarantee QoS.

As an example, assume that an external host, **E**, is streaming audio, through the proxy, **P**, to an internal host, **I**. **E** uses UDP port ***x*** to send to port ***x*** on **P**. **P** maps incoming port ***x*** to outgoing port ***y***, which is sent to port ***y*** on **I**.

$$(\mathbf{E}{:}x) \rightarrow (\mathbf{P}{:}x) \Rightarrow (\mathbf{P}{:}y) \rightarrow (\mathbf{I}{:}y)$$

The sequence of events to set up an RSVP reservation would be (assuming no errors):

1. **E** generates an RSVP Path message, which is sent to **P**. The RSVP session is (**P**:***x***) and the data stream source is (**E**:***x***).
2. The proxy checks to see if internal hosts are allowed to make RSVP reservations. If not, then the Path message is ignored. Otherwise, processing continues with step 3.
3. **P** generates an RSVP Path message, which is sent to **I**. The RSVP session is (**I**:***y***) and the data stream source is (**P**:***y***). **P** does not change any of the traffic characteristics.
4. **I** receives the Path message and generates an RSVP Resv message. The RSVP session is (**I**:***y***) and the filter spec will contain (**P**:***y***). When **P** receives the Resv message, a resource reservation has been completed between (**P**:***y***) and (**I**:***y***).
5. **P** must now generate a Resv message that will be sent upstream towards **E**. The RSVP session is (**P**:***x***) and the filter spec will contain (**E**:***x***). **P** does not change the flowspec. This step assumes that in addition to the (**E**:***x***) ⇒ (**I**:***y***) mapping, there is a reverse mapping that allows **P** to determine (**E**:***x***) using the RSVP session (**I**:***y***) found in the RSVP Resv message from **I**.
6. When **E** receives the Resv message, there is a resource reservation completed between (**P**:***x***) and (**E**:***x***).

The proxy processes the other RSVP messages in a similar manner. Normally, the proxy would have to do all of the work receiving and parsing RSVP packets. A group at Intel has developed an RSVP WinSock 2.0 Layered Service Provider (LSP) that provides an API designed to make it considerably easier to deal with RSVP messages at the application level. This LSP is available for download from the Intel Architecture Labs (IAL) web site (*http://www.intel.com/ial/rsvp*).

3.6 RSVP and IP Security

The IP Security Protocol Working Group (IPSEC) within the IETF has published two Internet Drafts (these supersede the RFCs 1826 and 1827) that specify mechanisms for providing IP-level security. Two problems arise

when Authentication Header (AH) [KENT97-1] and/or Encapsulated Secure Payload (ESP) [KENT97-2] are used in conjunction with RSVP:

1. Both AH and ESP place headers between the IP header and the transport-layer header, moving the UDP/TCP port numbers from their expected location.
2. In the case of ESP, the transport-layer headers are encrypted, making the port numbers inaccessible to RSVP.

A proposal [RFC2207] has been submitted to the RSVP Working Group to work around this problem. The basic idea is that RSVP should be extended to use, in the case of IPSEC, the Security Parameter Index (SPI) in lieu of UDP or TCP ports.

IPSEC has two different modes of operation. The first mode, Transport Mode, is suitable for peer-to-peer communications. In this mode the TCP/IP stack uses a rule-based approach, using information in the IP header, to determine if the packet needs to be processed (for example, encrypted). RSVP does not have knowledge of what the outgoing packets are going to look like, and therefore will be unable to generate the appropriate RSVP messages for the IPSEC data stream. The net effect is that the data stream will not be able to receive QoS. However, if RSVP could associate an application data stream with its associated IPSEC data stream, then RSVP could generate the appropriate messages (using the SPI of the IPSEC data stream for example). This requires that there be some sort of communication between the IPSEC and RSVP implementations. In addition to the ability to associate an application data stream with an SPI, IPSEC needs to be able to notify RSVP when keys are refreshed (as the SPI will change, and therefore the RSVP session). During the key refresh, it is possible that two SPIs may be active for a short period of time. [RFC2207] contains information regarding IPSEC rekeying and the effects upon RSVP.

The second, and more interesting to the firewall community, mode of operation is IPSEC Tunnel Mode. The two most common uses of tunnels are for Secure Remote Access and Virtual Private Networks (VPNs). This mode utilizes virtual interfaces. All packets that are to be sent through the tunnel are routed to the virtual interface, processed, and then sent back down the TCP/IP stack. Thus, RSVP messages will travel through the same tunnel as the data stream. The problem with this is that while the RSVP message travels inside of the tunnel, RSVP-enabled nodes along the way will not be able to process the messages. The result of this, of course, is that while in the tunnel, the application data stream will not be able to receive QoS. What is needed is a way to obtain QoS for the tunnel data stream as well.

Since RSVP has access to routing information, it can determine when a session describes a data stream that is sent through the tunnel. When an RSVP message is sent, the RSVP implementation should generate the appropriate RSVP messages for the tunnel. While RSVP may have access to the routing information, there may still be a need for communication between the IPSEC and RSVP implementations (as with Transport Mode IPSEC) because RSVP will need to know the other endpoint of the tunnel, as well as the SPI for the tunnel data stream. In Tunnel Mode, two RSVP messages will be generated:

1. The original RSVP message as requested by the application. This message describes the actual data stream and will travel inside of the tunnel.
2. An RSVP message, which RSVP must generate on the fly, which describes the tunnel. This message will obviously travel in the clear.

With this method, the application data stream receives QoS while not inside of the tunnel (because of the application-requested RSVP messages). Additionally, the tunnel data stream also receives QoS (because of the RSVP-generated messages for the tunnel).

This solution has one fundamental problem: if multiple data streams travel inside the same tunnel, there is no way for RSVP-enabled nodes to distinguish between packets of different data streams. The effect is that all data streams will be treated as if they were a single data stream. Applications that request QoS and best-effort traffic will be treated equally. Possible solutions to this problem are:

- Use different tunnels for each data stream. This solution suffers from lack of scalability.
- Provide a “large” enough flowspec for the tunnel such that the reservation can accommodate all of the individual data streams that have requested QoS. This means that the flowspec for the tunnel would need to

be adjusted each time a data stream requests (or relinquishes) QoS. This can be burdensome for RSVP-enabled hops if the reservation is constantly changing. An alternative would be to create an artificially large flowspec for the tunnel - this is a bad idea as it wastes network resources (unused capacity, that would otherwise be given to other resource requests, are instead tied up by the tunnel). This still does not solve the problem of best-effort traffic being treated equally.

- A hybrid of these two solutions is to have at least two tunnels, one that is for best-effort traffic and at least one for traffic requiring QoS.
- [KRAW97] suggests encapsulating the tunneled packets in UDP.

If multiple data streams that require QoS use the same tunnel, then an algorithm needs to be devised for aggregating the QoS requirements of the individual data streams. The net effect being that if the QoS for the tunnel(s) is(are) satisfied, then the QoS of the individual data streams will be satisfied. Determining how to aggregate the QoS requirements is similar to how RSVP-enabled nodes determine how to merge RSVP reservations.

Note: IPSEC is not the only solution for providing tunneling. All tunneling mechanisms share the same problems with respect to RSVP. Currently only IPSEC protocols have a solution. Tunnels, which use other protocols, will appear as non-RSVP clouds.

4. Conclusion

In the case of a simple packet-filtering router, supporting RSVP is as simple as allowing RSVP messages to flow through the router. However, when the firewall becomes more complicated (for example, NAT- and proxy-based firewalls), supporting RSVP becomes more of an issue. The two most important things to keep in mind when designing a solution to allow a firewall product to support RSVP are:

1. Several RSVP message objects contain IP addresses and ports.
2. When changing RSVP messages, it is possible that a new RSVP Integrity object will have to be created (as well as the existing one verified).

Proxies (both circuit and application-level) must also deal with the fact that they appear as endpoints. This implies that some mapping must be preserved so that an RSVP reservation can properly be established on both "sides" of the proxy. Tunneling firewalls (for example, Secure Remote Access or VPNs) must contend with the fact that UDP/TCP port information is lost in the tunnel. Currently, only tunnels that use IPSEC protocols have a solution. Other tunneling mechanisms will appear as non-RSVP clouds.

Finally, firewalls should conform to QoS guarantees. This does not necessarily mean that a firewall is required to participate in the protocol - that is dependent upon the type of firewall. Some firewalls, like filtering routers, may require no additional action to ensure this. Proxies, on the other hand, may require tweaking of RSVP messages to ensure that they can operate within the QoS bounds.

There are a numerous things to keep in mind when designing an RSVP firewall solution. Some of the problems that arise are subtle. RSVP support in a firewall is not impossible (or even terribly hard) - it just takes careful planning and a good understanding of the protocol.

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

[CHAP95] Chapman, D. B. and E. D. Zwicky, Building Internet Firewalls, O'Reilly & Associates, Inc., 1995

[CHES94] Cheswick, W. R. and S. M. Bellovin, Firewalls and Internet Security: Repelling the Wily Hacker, Addison-Wesley Publishing Company, 1994.

[SIYAN95] Siyan, K. and C. Hare, Internet Firewalls and Network Security, New Riders Publishing, 1995.

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

[BAKER97] Baker, F., "RSVP Cryptographic Authentication", <draft-ietf-rsvp-md5-05.txt>, August 1997. Work in Progress

[HERZOG97] Herzog, S., "RSVP Extensions for Policy Control", <draft-ietf-rsvp-policy-ext-02.txt>, April 1997.

[CHOU97] Chouinard, D., "SOCKS V5 UDP and Multicast Extensions to Facilitate Multicast Firewall Traversal", <draft-ietf-aft-mcast-fw-traversal-01.txt>, November 1997.

[KENT97-1] Kent, S. and R. Atkinson, "IP Authentication Header", <draft-ietf-ipsec-auth-header-02.txt>, October 1997.

[KENT97-2] Kent, S. and R. Atkinson, "IP Encapsulated Secure Payload (ESP), <draft-ietf-ipsec-esp-v2-01.txt>, October 1997.

[RFC2207] Berger, L. and T. O'Malley, "RSVP Extensions for IPSEC Data Flows", RFC 2207, September 1997.

[KRAW97] Krawczyk, J. J., "Designing Tunnels for Interoperability with RSVP", <draft-ietf-rsvp-tunnels-interop-00.txt>, March 1997.