Internet Draft
Expires April 29, 1997
File: draft-ietf-rsvp-procrules-00.ps

R. Braden
ISI
L. Zhang
PARC

November 1996

Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules

October 29, 1996

Status of Memo

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

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

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet- Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

Abstract

This memo contains an algorithmic description of the rules used by an RSVP implementation for processing messages. It is intended to clarify the version 1 RSVP protocol specification.

This memo provides a generic description of the rules for the operation of Version 1 of RSVP [RSVPspec96]. It is intended to outline a set of algorithms that will accomplish the needed function, omitting many details.

1 GENERIC DATA STRUCTURES

This memo assumes the generic interface calls defined in [RSVPspec96] and the following data structures. An actual implementation may use additional or different data structures and interfaces. The data structure fields that are shown are required unless they are explicitly labelled as optional.

- PSB – Path State Block

  Each PSB holds path state for a particular (session, sender) pair, defined by SESSION and SENDER_TEMPLATE objects, respectively, received in a *Path* message.

  PSB contents include the following values from a *Path* message:

  - Session
  - Sender_Template
  - Sender_Tspec
  - The previous hop IP address and the Logical Interface Handle (LIH) from a PHOP object
  - The remaining IP TTL
  - POLICY_DATA and/or ADSPEC objects (optional)
  - Non_RSVP flag
  - E_Police flag
  - Local_Only flag

  In addition, the PSB contains the following information provided by routing: OutInterface_list, which is the list of outgoing interfaces for this (sender, destination), and IncInterface, which is the expected incoming interface. For a unicast destination, OutInterface_list contains one entry and IncInterface is undefined.

  Note that there may be more than one PSB for the same (session, sender) pair but different incoming interfaces. At most one of these, which will have the Local_Only flag off, will be the PSB used for forwarding *Path* messages downstream; we will refer to it as the *forwarding PSB* in the following. The other PSB's will have the Local_Only flag on and an empty OutInterface_list. The Local_Only flag is needed to correctly match PSB's against RSB's, by the rules of [RSVPspec96].

- RSB – Reservation State Block

  Each RSB holds a reservation request that arrived in a particular *Resv* message, corresponding to the triple: (session, next hop, Filter_spec_list). Here "Filter_spec_list" may be a list of FILTER_SPECs (for SE style), a single FILTER_SPEC (FF style), or empty (WF style). We define the virtual object type "FILTER_SPEC*" for such a data structure.

  RSB contents include:

  - Session specification
  - Next hop IP address
  - Filter_spec_list
  - The outgoing (logical) interface OI on which the reservation is to be made or has been made.
  - Style
  - Flowspec
  - A SCOPE object (optional, depending upon style)
  - RESV_CONFIRM object that was received (optional)

- TCSB – Traffic Control State Block

  Each TCSB holds the reservation specification that has been handed to traffic control for a specific outgoing interface. In general, TCSB information is derived from RSB's for the same outgoing interface. Each TCSB defines a single reservation for a particular triple: (session, OI, Filter_spec_list). TCSB contents include:

  - Session
  - OI (Outgoing Interface)
  - Filter_spec_list
  - TC_Flowspec, the effective flowspec, i.e., the LUB over the corresponding FLOWSPEC values from matching RSB's. TC_Flowspec is passed to traffic control to make the actual reservation.
  - Fwd_Flowspec, the updated object to be forwarded after merging.
  - TC_Tspec, equal to Path_Te, the effective sender Tspec.
  - Police Flags

    The flags are E_Police_Flag, M_Police_Flag, and B_Police_Flag.
  - Rhandle, F_Handle_list

    Handles returned by the traffic control interface, corresponding to a flowspec and perhaps a list of filter specs.
  - A RESV_CONFIRM object to be forwarded.

- BSB – Blockade State Block

  Each BSB contains an element of blockade state. Depending upon the reservation style in use, the BSB's may be per (session, sender_template) pair or per (session, PHOP) pair. In practice, an implementation might embed a BSB within a PSB; however, for clarity we describe BSB's independently.

  The contents of a BSB include:

  - Session
  - Sender_Template (which is also a filter spec)
  - PHOP
  - FLOWSPEC Qb
  - Blockade timer Tb

The following Boolean Flag variables are used in this section: Path_Refresh_Needed, Resv_Refresh_Needed, Tear_Needed, Need_Scope, B_Merge, and NeworMod. Refresh_PHOP_list is a variable-length list of PHOPs to be refreshed.

2 PROCESSING RULES

MESSAGE ARRIVES

Verify version number and RSVP checksum, and discard message if any mismatch is found.

If the message type is not *Path* or *PathTear* or *ResvConf* and if the IP destination address does not match any of the addresses of the local interfaces, then forward the message to IP destination address and return.

Parse the sequence of objects in the message. If any required objects are missing or the length field of the common header does not match an object boundary, discard the message and return.

Verify the INTEGRITY object, if any. If the check fails, discard the message and return.

Verify the consistent use of port fields. If the DstPort in the SESSION object is zero but the SrcPort in a SENDER_TEMPLATE or FILTER_SPEC object is non-zero, then the message has a "conflicting source port" error; silently discard the message and return.

Processing of POLICY_DATA objects will be specified in the future.

Further processing depends upon message type.

Path MESSAGE ARRIVES

Assume the *Path* message arrives on interface InIf.

Process the **sender descriptor** object sequence in the message as follows. The Path_Refresh_Needed and Resv_Refresh_Needed flags are initially off.

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

  During this search:

  1. If a PSB is found whose session matches the DestAddress and Protocol Id fields of the received SESSION object, but the DstPorts differ and one is zero, then build and send a "Conflicting Dst Port" *PathErr* message, drop the *Path* message, and return.
  2. If a PSB is found with a matching sender host but the SrcPorts differ and one of the SrcPorts is zero, then build and send an "Ambiguous Path" *PathErr* message, drop the *Path* message, and return.
  3. If a forwarding PSB is found, i.e., a PSB that matches the (session, sender_template) pair and whose Local_Only flag is off, save a pointer to it in the variable fPSB. If none is found, set fPSB to NULL.

- If there was no matching PSB, then:

  1. Create a new PSB.
  2. Copy contents of the SESSION, SENDER_TEMPLATE, SENDER_TSPEC, and PHOP (IP address and LIH) objects into the PSB.
  3. If the sender is from the local API, set OutInterface_List to the single interface whose address matches the sender address, and make IncInterface undefined. Otherwise, turn on the Local_Only flag.
  4. Turn on the Path_Refresh_Needed flag.

- Otherwise (there is a matching PSB):

  - If the PHOP IP address, the LIH, or Sender_Tspec differs between the message and the PSB, copy the new value into the PSB and turn on the Path_Refresh_Needed flag. If the PHOP IP address or the LIH differ, also turn on the Resv_Refresh_Needed flag.

- Call the resulting PSB the *current PSB* (cPSB). Update the cPSB, as follows:

  - Start or Restart the cleanup timer for the PSB.
  - If the message contains an ADSPEC object, copy it into the PSB.
  - Copy E_Police flag from SESSION object into PSB.
  - Store the received TTL into the PSB.
    If the received TTL differs from Send_TTL in the RSVP common header, set the Non_RSVP flag on in the PSB.

- If the PSB is new or if there is no route change notification in place, then perform the following routing manipulations, but not if the cPSB is from the local API.

  1. Invoke the appropriate Route_Query routine using DestAddress from SESSION and (for multicast routing) SrcAddress from Sender_Template.
     Call the results (Rt_OutL, Rt_InIf).
  2. If the destination is multicast and Rt_InIf differs from IncInterface in the cPSB, but fPSB points to the cPSB, then do the following.
     - Turn on the Local_Only flag and clear the OutInterface_list of the fPSB. Set the fPSB pointer to NULL.
     - Search for a PSB for the same (session, sender_template) pair whose IncInterface matches Rt_InIf. If one is found, set fPSB to point to it.
  3. If the destination is multicast and Rt_InIf is the same as IncInterface in the cPSB, but fPSB does not point to the cPSB, then do the following.
     - Copy into the cPSB the OutInterface_list from the PSB, if any, pointed to by fPSB. Clear OutInterface_list and turn on the Local_Only flag in the PSB pointed to by fPSB, if any.
     - Turn off the Local_Only flag in the cPSB and set fPSB to point to cPSB.
  4. If Rt_OutL differs from OutInterface_list of the PSB pointed to by fPSB, then:
     - Update the OutInterface_list of the PSB from Rt_OutL, and then execute the PATH LOCAL REPAIR event sequence below.

- If the Path_Refresh_Needed flag is now off, drop the *Path* message and return.

  Otherwise (the path state is new or modified), do refreshes, upcalls, and state updates as follows.

  1. If this *Path* message came from a network interface and not from a local application, make a Path Event upcall for each local application for this session:
     ```
     Call: <Upcall_Proc>( session-id, PATH_EVENT,
                 flags, sender_tspec, sender_template
                 [ , ADSPEC] [ , POLICY_DATA] )
     ```
  2. If OutInterface_list is not empty, execute the PATH REFRESH event sequence (below) for the sender defined by the PSB.
  3. Search for any matching reservation state, i.e., an RSB whose Filter_spec_list includes a FILTER_SPEC matching the SENDER_TEMPLATE and whose OI appears in the OutInterface_list, and make this the 'active RSB'. If none is found, drop the *Path* message and return.
  4. Execute the RESV REFRESH sequence (below) for the PHOP in the PSB.
  5. Execute the event sequence UPDATE TRAFFIC CONTROL to update the local traffic control state if necessary. This sequence will turn on the Resv_Refresh_Needed flag if the traffic control state has been modified in a manner that should trigger a

reservation refresh. If so, execute the RESV REFRESH sequence for the PHOP in the PSB.

- Drop the *Path* message and return.

PathTear MESSAGE ARRIVES

- Search for a PSB whose (Session, Sender_Template) pair matches the corresponding objects in the message. If no matching PSB is found, drop the *PathTear* message and return.
- Forward a copy of the *PathTear* message to each outgoing interface listed in OutInterface_list of the PSB.
- Find each RSB that matches this PSB, i.e., whose Filter_spec_list matches Sender_Template in the PSB and whose OI is included in OutInterface_list.
  1. If the RSB style is explicit, then:
     - Delete from Filter_spec_list the FILTER_SPEC that matches the PSB.
     - if Filter_spec_list is now empty, delete the RSB.
  2. Otherwise (RSB style is wildcard) then:
     - If this RSB matches no other PSB, then delete the RSB.
  3. If an RSB was found, execute the event sequence UPDATE TRAFFIC CONTROL (below) to update the traffic control state to be consistent with the current reservation and path state.
- Delete the PSB.
- Drop the *PathTear* message and return.

PathErr MESSAGE ARRIVES

- Search for a PSB whose (SESSION, SENDER_TEMPLATE) pair matches the corresponding objects in the message. If no matching PSB is found, drop the *PathErr* message and return.
- If the previous hop address in the PSB is the local API, make an error upcall to the application:

  ```
  Call: <Upcall_Proc>( session-id, PATH_ERROR,
                Error_code, Error_value,
                Node_Addr, Sender_Template
                [ , Policy_Data] )
  ```

  Any SENDER_TSPEC or ADSPEC object in the message is ignored.

  Otherwise, send a copy of the *PathErr* message to the PHOP IP address.
- Drop the *PathErr* message and return.

Resv MESSAGE ARRIVES

Initially, Refresh_PHOP_list is empty and the Resv_Refresh_Needed and NeworMod flags are off. These variables are used to control immediate reservation refreshes.

- Determine the Outgoing Interface OI

  The logical outgoing interface OI is taken from the LIH in the NHOP object. (If the physical interface is not implied by the LIH, it can be learned from the interface matching the IP destination address).
- Check the path state
  1. If there are no existing PSB's for SESSION then build and send a *ResvErr* message (as described later) specifying "No path information", drop the *Resv* message, and return.
  2. If a PSB is found with a matching sender host but the SrcPorts differ and one of the SrcPorts is zero, then build and send an "Ambiguous Path" *PathErr* message, drop the *Resv* message, and return.
- Check for incompatible styles.

  If any existing RSB for the session has a style that is incompatible with the style of the message, build and send a *ResvErr* message specifying "Conflicting Style", drop the *Resv* message, and return.

Process the **flow descriptor list** to make reservations, as follows, depending upon the style. The following uses a filter spec list struct Filtss of type FILTER_SPEC* (defined earlier).

For FF style: execute the following steps independently for each **flow descriptor** in the message, i.e., for each (FLOWSPEC, Filtss) pair. Here the structure Filtss consists of the FILTER_SPEC from the flow descriptor.

For SE style, execute the following steps once for (FLOWSPEC, Filtss), with Filtss consisting of the list of FILTER_SPEC objects from the flow descriptor.

For WF style, execute the following steps once for (FLOWSPEC, Filtss), with Filtss an empty list.

- Check the path state, as follows.
  1. Locate the set of PSBs (senders) that route to OI and whose SENDER_TEMPLATEs match a FILTER_SPEC in Filtss.

     If this set is empty, build and send an error message specifying "No sender information", and continue with the next flow descriptor in the *Resv* message.
  2. If the style has explicit sender selection (e.g., FF or SE) and if any FILTER_SPEC included in Filtss matches more than one PSB, build and send a *ResvErr* message specifying "Ambiguous filter spec" and continue with the next flow descriptor in the *Resv* message.

3. If the style is SE and if some FILTER_SPEC included in Filtss matches no PSB, delete that FILTER_SPEC from Filtss.
4. Add the PHOP from the PSB to Refresh_PHOP_list, if the PHOP is not already on the list.

- Find or create a reservation state block (RSB) for (SESSION, NHOP). If the style is distinct, Filtss is also used in the selection. Call this the *active RSB*.
- If the active RSB is new:

  1. Set the session, NHOP, OI and style of the RSB from the message.
  2. Copy Filtss into the Filter_spec_list of the RSB.
  3. Copy the FLOWSPEC and any SCOPE object from the message into the RSB.
  4. Set NeworMod flag on.

- If the active RSB is not new, check whether Filtss from the message contains FILTER_SPECs that are not in the RSB; if so, add the new FILTER_SPECs and turn on the NeworMod flag.
- Start or restart the cleanup timer on the active RSB, or, in the case of SE style, on each FILTER_SPEC of the RSB that also appears in Filtss.
- If the active RSB is not new, check whether STYLE, FLOWSPEC or SCOPE objects have changed; if so, copy changed object into RSB and turn on the NeworMod flag.
- If the message contained a RESV_CONFIRM object, copy it into the RSB and turn on NeworMod flag.
- If the NeworMod flag is off, continue with the next **flow descriptor** in the *Resv* message, if any.
- Otherwise (the NeworMod flag is on, i.e., the active RSB is new or modified), execute the UPDATE TRAFFIC CONTROL event sequence (below). If the result is to modify the traffic control state, this sequence will turn on the Resv_Refresh_Needed flag and make a RESV_EVENT upcall to any local application.

  If the UPDATE TRAFFIC CONTROL sequence fails with an error, then delete a new RSB but restore the original reservation in an old RSB.
- Continue with the next flow descriptor.
- When all flow descriptors have been processed, check the Resv_Refresh_Needed flag. If it is now on, execute the RESV REFRESH sequence (below) for each PHOP in Refresh_PHOP_list.
- Drop the *Resv* message and return.

If processing a *Resv* message finds an error, a *ResvErr* message is created containing flow descriptor and an ERRORS object. The Error Node field of the ERRORS object is set to the IP address of OI, and the message is sent unicast to NHOP.

ResvTear MESSAGE ARRIVES

Processing of a *ResvTear* message roughly parallels the processing of the corresponding *Resv* message

A *ResvTear* message arrives with an IP destination address matching outgoing interface OI. Flag Resv_Refresh_Needed is initially off and Refresh_PHOP_list is empty.

- Determine the Outgoing Interface OI

  The logical outgoing interface OI is taken from the LIH in the NHOP object. (If the physical interface is not implied by the LIH, it can be learned from the interface matching the IP destination address).

- Process the **flow descriptor list** in the *ResvTear* message to tear down local reservation state, as follows, depending upon the style. The following uses a filter spec list struct Filtss of type FILTER_SPEC* (defined earlier).

  For FF style: execute the following steps independently for each **flow descriptor** in the message, i.e., for each (FLOWSPEC, Filtss) pair. Here the structure Filtss consists of the FILTER_SPEC from the flow descriptor.

  For SE style, execute the following steps once for (FLOWSPEC, Filtss), with Filtss consisting of the list of FILTER_SPEC objects from the flow descriptor.

  For WF style, execute the following steps once for (FLOWSPEC, Filtss), with Filtss an empty list.

  1. Find an RSB matching (SESSION, NHOP). If the style is distinct, Filtss is also used in the selection. Call this the *active RSB*. If no active RSB is found, continue with next flow descriptor.
  2. Check the style

     If the active RSB has a style that is incompatible with the style of the message, drop the *ResvTear* message and return.
  3. Delete from the active RSB each FILTER_SPEC that matches a FILTER_SPEC in Filtss.
  4. If all FILTER_SPECs have now been deleted from the active RSB, delete the active RSB.
  5. Execute the UPDATE TRAFFIC CONTROL event sequence (below) to update the traffic control state to be consistent with the reservation state. If the result is to modify the traffic control state, the Resv_Refresh_Needed flag will be turned on and a RESV_EVENT upcall will be made to any local application.
  6. Continue with the next flow descriptor.

- All flow descriptors have been processed.

  Build and send any *ResvTear* messages to be forwarded, in the following manner.

  1. Select each PSB that routes to the outgoing interface OI, and, for distinct style, that has a SENDER_TEMPLATE matching Filtss.

2. Select a flow descriptor (Qj,Fj) (where Fj may be a list) in the *ResvTear* message whose FILTER_SPEC matches the SENDER_TEMPLATE in the PSB. If not match is found, return for next PSB.
   - Search for an RSB (for any outgoing interface) to which the PSB routes and whose Filter_spec_list includes the SENDER_TEMPLATE from the PSB.
   - If an RSB is found, add the PHOP of the PSB to the Refresh_PHOP_list.
   - Otherwise (no RSB is found), add the flow descriptor (Qj,Fj) to the new *ResvTear* message being built, in a manner appropriate to the style.
   - Continue with the next PSB.
3. If the next PSB is for a different PHOP or the last PSB has been processed, forward any *ResvTear* message that has been built.

- If any PSB's were found in the preceding step, and if the Resv_Refresh_Needed flag is now on, execute the RESV REFRESH sequence (below) for each PHOP in Refresh_PHOP_list.
- Drop the *ResvTear* message and return.

ResvErr MESSAGE ARRIVES

A *ResvErr* message arrives through the (real) incoming interface In_If.

- If there is no path state for SESSION, drop the *ResvErr* message and return.
- If the Error Code = 01 (Admission Control failure), do special processing as follows:
  1. Find or create a Blockade State Block (BSB), in the following style-dependent manner.
     For WF (wildcard) style, there will be one BSB per (session, PHOP) pair.
     For FF style, there will be one BSB per (session, filter_spec) pair. Note that an FF style *ResvErr* message carries only one flow descriptor.
     For SE style, there will be one BSB per (session, filter_spec), for each filter_spec contained in the filter spec list of the flow descriptor.
  2. For each BSB in the preceding step, set (or replace) its FLOWSPEC Qb with FLOWSPEC from the message, and set (or reset) its timer Tb to Kb*R seconds. If the BSB is new, set its PHOP value, and set its Sender_Template equal to the appropriate filter_spec from the message.
  3. Execute the RESV REFRESH event sequence (shown below) for the previous hop PHOP, but only with the B_Merge flag off. That is, if processing in the RESV REFRESH sequence reaches the point of turning the B_Merge flag on (because all matching reservations are blockaded), do not turn it on but instead exit the REFRESH sequence and return here.
- Execute the following for each RSB for this session whose OI differs from In_If and whose Filter_spec_list has at least one filter spec in common with the FILTER_SPEC* in the *ResvErr* message. For WF style, empty FILTER_SPEC* structures are assumed to match.

1. If Error_Code = 01 and the InPlace flag in the ERROR_SPEC is 1 and one or more of the BSB's found/created above has a Qb that is strictly greater than Flowspec in the RSB, then continue with the next matching RSB, if any.
2. If NHOP in the RSB is the local API, then:
   - If the FLOWSPEC in the *ResvErr* message is strictly greater than the RSB Flowspec, then turn on the NotGuilty flag in the ERROR_SPEC.
   - Deliver an error upcall to application:

     ```
     Call: <Upcall_Proc>( session-id, RESV_ERROR,
                   Error_code, Error_value,
                        Node_Addr,  Error_flags,
                        Flowspec, Filter_Spec_List
                        [ , Policy_data] )
     ```

   and continue with the next RSB.
3. If the style has wildcard sender selection, use the SCOPE object SC.In from the *ResvErr* message to construct a SCOPE object SC.Out to be forwarded. SC.Out should contain those sender addresses that appeared in SC.In and that route to OI, as determined by scanning the PSB's. If SC.Out is empty, continue with the next RSB.
4. Create a new *ResvErr* message containing the **error flow descriptor** and send to the NHOP address specified by the RSB. Include SC.Out if the style has wildcard sender selection.
5. Continue with the next RSB.

- Drop the *ResvErr* message and return.

RESV CONFIRM ARRIVES

- If the (unicast) IP address found in the RESV_CONFIRM object in the *ResvConf* message matches an interface of the node, a confirmation upcall is made to the matching application:

  ```
  Call: <Upcall_Proc>( session-id, RESV_CONFIRM,
           Error_code, Error_value, Node_Addr,
                LUB-Used, nlist, Flowspec,
                Filter_Spec_List, NULL, NULL )
  ```

- Otherwise, forward the *ResvConf* message to the IP address in its RESV_CONFIRM object.
- Drop the *ResvConf* message and return.

UPDATE TRAFFIC CONTROL

The sequence is invoked by many of the message arrival sequences to set or adjust the local traffic control state in accordance with the current reservation and path state. An implicit parameter of this sequence is the 'active' RSB.

If the result is to modify the traffic control state, this sequence notifies any matching local applications with a RESV_EVENT upcall. If the state change is such that it should trigger immediate *Resv* refresh messages, it also turns on the Resv_Refresh_Needed flag.

- Compute the traffic control parameters using the following steps.
  1. Initially the local flag Is_Biggest is off.
  2. Consider the set of RSB's matching SESSION and OI from the active RSB. If the style of the active RSB is distinct, then the Filter_spec_list must also be matched.
     - Compute the effective kernel flowspec, TC_Flowspec, as the LUB of the FLOWSPEC values in these RSB's.
     - Compute the effective traffic control filter spec (list) TC_Filter_Spec* as the union of the Filter_spec_lists from these RSB's.
     - If the active RSB has a FLOWSPEC larger than all the others, turn on the Is_Biggest flag.
  3. Scan all RSB's matching session and Filtss, for all OI. Set TC_B_Police_flag on if TC_Flowspec is smaller than, or incomparable to, any FLOWSPEC in those RSB's.
  4. Locate the set of PSBs (senders) whose SENDER_TEMPLATEs match Filter_spec_list in the active RSB and whose OutInterface_list includes OI.
  5. Set TC_E_Police_flag on if any of these PSBs have their E_Police flag on. Set TC_M_Police_flag on if it is a shared style and there is more than one PSB in the set.
  6. Compute Path_Te as the sum of the SENDER_TSPEC objects in this set of PSBs.
- Search for a TCSB matching SESSION and OI; for distinct style (FF), it must also match Filter_spec_list.

  If none is found, create a new TCSB.
- If TCSB is new:
  1. Store TC_Flowspec, TC_Filter_Spec*, Path_Te, and the police flags into TCSB.
  2. Turn the Resv_Refresh_Needed flag on and make the traffic control call:

     ```
     TC_AddFlowspec( OI, TC_Flowspec,
                           Path\_Te, police_flags)
                            -> Rhandle, Fwd_Flowspec
     ```

  3. If this call fails, build and send a *ResvErr* message specifying "Admission control failed" and with the InPlace flag off. Delete the TCSB, delete any RESV_CONFIRM object from the active RSB, and return.

4. Otherwise (call succeeds), record Rhandle and Fwd_Flowspec in the TCSB. For each filter_spec F in TC_Filter_Spec*, call:

   ```
   TC_AddFilter( OI, Rhandle, Session, F)
                                 -> Fhandle
   ```

   and record the returned Fhandle in the TCSB.

- Otherwise, if TCSB is not new but no effective kernel flowspec TC_Flowspec was computed earlier, then:

  1. Turn on the Resv_Refresh_Needed flag.
  2. Call traffic control to delete the reservation:

     ```
     TC_DelFlowspec( OI, Rhandle )
     ```

  3. Delete the TCSB and return.

- Otherwise, if TCSB is not new but the TC_Flowspec, Path_Te, and/or police flags just computed differ from corresponding values in the TCSB, then:

  1. If the TC_Flowspec and/or Path_Te values differ, turn the Resv_Refresh_Needed flag on.
  2. Call traffic control to modify the reservation:

     ```
     TC_ModFlowspec( OI, Rhandle, TC_Flowspec,
                             Path_Te, police_flags )
                                 -> Fwd_Flowspec
     ```

  3. If this call fails, build and send a *ResvErr* message specifying "Admission control failed" and with the InPlace bit on. Delete any RESV_CONFIRM object from the active RSB and return.
  4. Otherwise (the call succeeds), update the TCSB with the new values and save Fwd_Flowspec in the TCSB.

- If the TCSB is not new but the TC_Filter_Spec* just computed differs from the FILTER_SPEC* in the TCSB, then:

  1. Make an appropriate set of TC_DelFilter and TC_AddFilter calls to transform the Filter_spec_list in the TCSB into the new TC_Filter_Spec*.
  2. Turn on the Resv_Refresh_Needed flag.

- If the active RSB contains a RESV_CONFIRM object, then:

  1. If the Is_Biggest flag is on, move the RESV_CONFIRM object into the TCSB and turn on the Resv_Refresh_Needed flag. (This will later cause the RESV REFRESH

sequence to be invoked, which will either forward or return the RESV_CONFIRM object, deleting it from the TCSB in either case).

2. Otherwise, create and send a *ResvConf* message to the address in the RESV_CONFIRM object. Include the RESV_CONFIRM object in the *ResvConf* message. The *ResvConf* message should also include an ERROR_SPEC object whose Error_Node parameter is IP address of OI from the TCSB and that specifies "No Error".

- If the Resv_Refresh_Needed flag is on and the RSB is not from the API, make a RESV_EVENT upcall to any matching application:

```
Call: <Upcall_Proc>( session-id, RESV_EVENT,
           style, Flowspec, Filter_spec_list
           [ , POLICY_DATA] )
```

  where Flowspec and Filter_spec_list come from the TCSB and the style comes from the active RSB.

- Return to the event sequence that invoked this one.

PATH REFRESH

This sequence sends a path refresh for a particular sender, i.e., a PSB. This sequence may be entered by either the expiration of a refresh timer or directly as the result of the Path_Refresh_Needed flag being turned on during the processing of a received *Path* message.

- Insert TIME_VALUES object into the *Path* message being built. Compute the IP TTL for the *Path* message as one less than the TTL value received in the message. However, if the result is zero, return without sending the *Path* message.
- Create a **sender descriptor** containing the SENDER_TEMPLATE, SENDER_TSPEC, and POLICY_DATA objects, if present in the PSB, and pack it into the *Path* message being built.
- Send a copy of the *Path* message to each interface OI in OutInterface_list. Before sending each copy:
  1. If the PSB has the E_Police flag on and if interface OI is not capable of policing, turn the E_Police flag on in the *Path* message being built.
  2. Pass the ADSPEC object and Non_RSVP flag present in the PSB to the traffic control call TC_Advertise. Insert the modified ADSPEC object that is returned into the *Path* message being built.
  3. Insert into its PHOP object the interface address and the LIH for the interface.

RESV REFRESH

This sequence sends a reservation refresh towards a particular previous hop with IP address PH. This sequence may be entered by the expiration of a refresh timer, or invoked from the

Path MESSAGE ARRIVES, *Resv* MESSAGE ARRIVES, *ResvTear* MESSAGE ARRIVES, or *ResvErr* MESSAGE ARRIVES sequence.

In general, this sequence considers each of the PSB's with PHOP address PH. For a given PSB, it scans the TCSBs for matching reservations and merges the styles, FLOWSPECs and Filter_spec_list's appropriately. It then builds a *Resv* message and sends it to PH. The details depend upon the attributes of the style(s) included in the reservations.

Initially the Need_Scope flag is off and the new_SCOPE object is empty.

- Create an output message containing INTEGRITY (if configured), SESSION, RSVP_HOP, and TIME_VALUES objects.
- Determine the style for these reservations from the first RSB for the session, and move the STYLE object into the proto-message. (Note that the present set of styles are never themselves merged; if future styles can be merged, these rules will become more complex).
- If style is wildcard and if there are PSB's from more than one PHOP and if the multicast routing protocol does not use shared trees, set the Need_Scope flag on.
- Select each sender PSB whose PHOP has address PH. Set the local flag B_Merge off and execute the following steps.
  1. Select all TCSB's whose Filter_spec_list's match the SENDER_TEMPLATE object in the PSB and whose OI appears in the OutInterface_list of the PSB.
  2. If the PSB is from the API, then:
     - If TCSB contains a CONFIRM object, then create and send a *ResvConf* message containing the object and delete the CONFIRM object from the TCSB.
     - Continue with next PSB.
  3. If B_Merge flag is off then ignore a blockaded TCSB, as follows.
     - Select BSB's that match this TCSB. If a selected BSB is expired, delete it. If any of the unexpired BSB's has a Qb that is not strictly larger than TC_Flowspec, then continue processing with the next TCSB.

     However, if steps 1 and 2 result in finding that all TCSB's matching this PSB are blockaded, then:
     - If this RESV REFRESH sequence was invoked from RESV ERROR RECEIVED, then return to the latter.
     - Otherwise, turn on the B_Merge flag and restart at step 1, immediately above.
  4. Merge the flowspecs from this set of TCSB's, as follows:
     - If B_Merge flag is off, compute the LUB over the flowspec objects. From each TCSB, use the Fwd_Flowspec object if present, else use the normal Flowspec object.

       While computing the LUB, check for a RESV_CONFIRM object in each TCSB. If a RESV_CONFIRM object is found:

      * If the flowspec (Fwd_Flowspec or Flowspec) in that TCSB is larger than all other (non-blockaded) flowspecs being compared, then save this RESV_CONFIRM object for forwarding and delete from the TCSB.
      * Otherwise (the corresponding flowspec is not the largest), create and send a *ResvConf* message to the address in the RESV_CONFIRM object. Include the RESV_CONFIRM object in the *ResvConf* message. The *ResvConf* message should also include an ERROR_SPEC object whose Error_Node parameter is IP address of OI from the TCSB and specifying "No Error".
      * Delete the RESV_CONFIRM object from the TCSB.
    - Otherwise (B_Merge flag is on), compute the GLB over the Flowspec objects of this set of TCSB's.

      While computing the GLB, delete any RESV_CONFIRM object object in any of these TCSB's.
  5. (All matching TCSB's have been processed). The next step depends upon the style attributes.

     Distinct reservation (FF) style

       Use the Sender_Template as the merged FILTER_SPEC. Pack the merged (FLOWSPEC, FILTER_SPEC, F_POLICY_DATA) triplet into the message as a **flow descriptor**.

     Shared wildcard reservation (WF) style

       There is no merged FILTER_SPEC. Merge (compute the LUB of) the merged FLOWSPECS from the TCSB's, across all PSB's for PH.

     Shared distinct reservation (SE) style

       Using the Sender_Template as the merged FILTER_SPEC, form the union of the FILTER_SPECS obtained from the TCSB's. Merge (compute the LUB of) the merged FLOWSPECS from the TCSB's, across all PSB's for PH.
  6. If the Need_Scope flag is on and the sender specified by the PSB is not the local API:
     - Find each RSB that matches this PSB, i.e., whose Filter_spec_list matches Sender_Template in the PSB and whose OI is included in OutInterface_list.
     - If the RSB either has no SCOPE list or its SCOPE list includes the sender IP address from the PSB, insert the sender IP address into new_SCOPE.

- (All PSB's for PH have been processed). Finish the *Resv* message.

  1. If Need_Scope flag is on but new_SCOPE is empty, no *Resv* message should be sent; return. Otherwise, if Need_Scope is on, move new_SCOPE into the message.
  2. If a shared reservation style is being built, move the final merged FLOWSPEC object and filter spec list into the message.
  3. If a RESV_CONFIRM object was saved earlier, move it into the new *Resv* message.
  4. Set the RSVP_HOP object in the message to contain the IncInterface address through which it will be sent and the LIH from (one of) the PSB's.

- Send the message to the address PH.

ROUTE CHANGE NOTIFICATION

This sequence is triggered when routing sends a route change notification to RSVP.

- Each PSB is located whose SESSION matches the destination address and whose SENDER_TEMPLATE matches the source address (for multicast).
  1. If the OutInterface_list from the notification differs from that in the PSB, execute the PATH LOCAL REPAIR sequence.
  2. If the IncInterface from the notification differs from that in the PSB, update the PSB.

PATH LOCAL REPAIR

The sequence is entered to effect local repair after a route change for a given PSB.

- Wait for a delay time of W seconds.
- Execute the PATH REFRESH event sequence (above) for the PSB.

References

[Baker96] Baker, F., *RSVP Cryptographic Authentication*, Work in Progress, February 1996.

[RSVPspec96] Braden, R., Ed, Zhang, L., Berson, S., Herzog, S., and S. Jamin, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*, Work in progress, November 1996.

[IPSEC96] Berger, L. and T. O'Malley, *RSVP Extensions for IPSEC IPv4 Data Flows*, Internet Draft, <draft-ietf-rsvp-ext-04.txt>, Integrated Services Working Group, June 1996.

[RSVP93] Zhang, L., Deering, S., Estrin, D., Shenker, S., and D. Zappala, *RSVP: A New Resource ReSerVation Protocol*, IEEE Network, September 1993.

Security Considerations

Processing the RSVP INTEGRITY object [Baker96] is only mentioned in this memo, because the processing rules are described here only in general terms. The RSVP support for IPSEC [IPSEC96] will imply modifications that have not yet been incorporated into these processing rules.

Authors' Addresses

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

Phone: (310) 822-1511
EMail: Braden@ISI.EDU

Lixia Zhang
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Phone: (415) 812-4415
EMail: Lixia@PARC.XEROX.COM