The Resource Reservation Protocol RSVP

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
December 7, 1999

http://www.cs.unc.edu/~jeffay/courses/comp249f99

The Integrated Services Architecture
Architectural components

- Flow specifications
- Routing
- Resource reservation
- Admission control
- Packet scheduling
Issues in Resource Reservation
Point-to-point communications

- **Goal:** Establish a virtual circuit from H1 to H2
  - Reserve “resources” in routers R1, R2, and R3

- **Resources are...**
  - Link capacity on transmission links
  - Buffer capacity in routers to hold packets in transit
  - CPU capacity at all routers to forward packets from H1 in real-time

---

Issues in Resource Reservation
Point-to-point communications

- **Example:** ST-II — a two-pass reservation protocol
  - H1 sends a *connect* message containing a flowspec towards H2
    - The connect message is modified as needed by R1 - R3
  - Upon receipt of the connect, H2 sends an *accept* message back to H1
    - Reservations are made when routers receive the *accept* message
Issues in Resource Reservation

Point-to-point communications

- What if the route from H1 to H2 changes?
  » How will the application know that the route has changed?
- What level of integration between routing and resource reservation is appropriate?

Issues in Resource Reservation

One-to-many multicast

- Apply the point-to-point method recursively throughout the multicast tree
- How do we handle differing link/router capacities?
  » ST-II — Reduce all connections to the least common denominator
Issues in Resource Reservation

One-to-many multicast

- How do we add/delete new users?
  » ST-II — Source re-executes the reservation protocol with all receivers

Issues in Resource Reservation

Many-to-many multicast

- H1 and H3 independently reserve resources
- How can we avoid over-reserving resources?
**Simple Resource Reservation**

**Summary**

- Guaranteed service requires integration of resource reservation with routing
- Sender-initiated reservations do not scale
  - Protocol overhead at sender becomes a bottleneck
  - Difficult to accommodate heterogeneous receivers
  - Low utilization of network links may result from overly conservative reservations

**RSVP**

A receiver initiated reservation protocol

- Receivers initiate reservations
  - Receivers know what bandwidth they want or can handle
  - Places burden of joining/leaving largely on the involved receiver
  - Admits the possibility of optimizing reservations in routers & switches through aggregation
RSVP Protocol Design

- RSVP is a control protocol
  - RSVP is above IP (like IGMP)

- Reservation is separate from routing
  - Assume only that RSVP and application datagrams sent from a host to a multicast group traverse the same links

- Reservation and admission control are orthogonal processes

- Reservation state in routers is “soft” and must be refreshed
  - Ensures fault tolerance
  - Ensures coupling between reservations and route changes
  - Enables a form of “channel surfing”

RSVP Protocol Design

- Reservations are separate from data transport
  - RSVP reservations do not uniquely specify which packets can use a given reservation
  - Different reservation styles are supported through the use of filters
RSVP

Operation Overview

◆ Senders and receivers join a multicast group
  » Done outside of RSVP
     (Senders need not explicitly join)

◆ Sender’s advertise their existence
  » Sender to network messages
     ✷ Path request — make presence of a sender known to network elements
     ✷ Path teardown — delete path state from routers

◆ Receiver’s subscribe to sender data streams
  » Receiver to network messages
     ✷ Reservation request — reserve resources from sender(s) to receiver
     ✷ Reservation teardown — delete reservations

◆ Network to end system messages
  » Path error
  » Reservation error

RSVP Operation
Sender-to-network messages

◆ Sender’s identify themselves through path messages

◆ Path message contents:
  » session id — a destination or multicast group address, port
  » flowspec — a stream bandwidth & requirements descriptor
  » previous hop — the address of the machine forwarding the message
  » filtering spec — specifies how this source’s packets may be filtered
  » refresh time — time until path information times out

◆ Path message are used by network elements to encode routes from receivers back to senders
  » These routes are part of a router’s state
  » Routers record sender and the name of the previous router that forwarded the message
RSVP Operation
Building up path state

- H1 and H2 are senders, H3, H4, and H5 are receivers

\[ \text{path(TSpec, H1, filter spec, refresh time)} \]

Router State

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound Link</td>
<td>L1[H1⇒H1]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RSVP Operation
Building up path state

- Path state built up as in multicast routing

\[ \text{path(TSpec, R1, filter spec, refresh time)} \]

Router State

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound Link</td>
<td>L1[H1⇒H1]</td>
<td>L6[H1⇒R1]</td>
<td>L7[H1⇒R1]</td>
<td></td>
</tr>
</tbody>
</table>
RSVP Operation
Building up path state

- Path state built up as in multicast routing

RSVP Operation
Complete path state for H1

- When reservations are made, path state is used to build a *multicast sink tree* to forward reservations
RSVP Operation
Receiver-to-network messages

- Receiver’s send reservation messages containing:
  - desired bandwidth
  - filter type
    - no filter — any packets addressed to the multicast group can use the reservation
    - fixed filter — only packets from a specific set of senders can use the reservation
    - dynamic filter — the set of sender’s whose packets can use the reservation can change over time
  - packet filter

- (reservation, source) pairs become part of a router’s state
  - State maintained at all routers along the path from source to receiver

RSVP Operation
Example

- 5 hosts, 3 interconnection elements
- One multicast group containing all hosts
  - Each host will send and receive
RSVP Operation Example
Making *no-filter* reservations

- Each router has received *path* messages from all sources with the *f*-flag off
- No reservations have been made

### No-Filter Reservation Example
Making a reservation

- H1 wants to be able to receive from any sender but only wants 1 stream at a time

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td></td>
</tr>
</tbody>
</table>
No-Filter Reservation Example
Forwarding a reservation

- R1 reserves $b$ units of bandwidth from R1 to H1
- R1 forwards $r_1$ over all links in its PATH database

![Diagram]

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1($b$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No-Filter Reservation Example
Forwarding a reservation

- R2 reserves $b$ units of bandwidth from R2 to R1
- R2 forwards $r_1$ over L5 & L7

![Diagram]

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1($b$)</td>
<td>L6($b$)</td>
<td></td>
</tr>
</tbody>
</table>
No-Filter Reservation Example

Forwarding a reservation

- R3 reserves \( b \) units of bandwidth from R3 to R2
- Finally, \( b \) units of bandwidth are reserved along the path from any host to H1

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1(( b ))</td>
<td>L6(( b ))</td>
<td>L7(( b ))</td>
</tr>
</tbody>
</table>

No-Filter Reservation Example

Reserving bandwidth at a host

- A reservation arriving at a host is an indication that someone is willing to listen to the host
  - The end-system RSVP “client” reserves sufficient internal resources to ensure transmission to the multicast group

\[ r_1(b, \text{no-f}) \]
No-Filter Reservation Example
Complete reservation state for reservation $r_1$

- H1 can receive from any host and any host can send to H1

```
R1          R2          R3
L1[H1], L2[H2], L6[R2] L5[H5], L6[R1], L7[R3] L3[H3], L4[H4], L7[R2]
```

No-Filter Reservation Example
Practical issues

- H1 can receive from any host...
  - Which host does H1 actually receive from?
  - Who decides?

- Any host can send to H1...
  - What if H4’s output stream consumes more than $b$ units of bandwidth?
  - What if H3 & H4 transmit simultaneously?
No-Filter Reservation Example

Policing issues

- If the reserved capacity is exceeded then the flow is policed (shaped) to a conforming flowspec
  - If H3 & H4 transmit simultaneously then their packets are arbitrarily interleaved

No-Filter Reservation Example

Adding additional reservations

- H2 reserves bandwidth for a single stream
  - R1 reserves \( b \) units of bandwidth from R1 to H2

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1(( b )), L2(( b ))</td>
<td>L6(( b ))</td>
<td>L7(( b ))</td>
</tr>
</tbody>
</table>
No-Filter Reservation Example
Adding additional reservations

- R1 forwards $r_2$ over L1 only
  - H1 reserves $b$ units of bandwidth from itself to R1

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1($b$), L2($b$)</td>
<td>L6($b$)</td>
<td>L7($b$)</td>
</tr>
</tbody>
</table>

No-Filter Reservation Example
Adding additional reservations

- H3 reserves $b$ units of bandwidth
  - R3 reserves $b$ units of bandwidth from R3 to H3

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1($b$), L2($b$)</td>
<td>L6($b$)</td>
<td>L7($b$), L3($b$)</td>
</tr>
</tbody>
</table>
No-Filter Reservation Example

Adding additional reservations

- R3 forwards $r_3$ over L7
- R2 reserves $b$ units of bandwidth from R2 to R3

Router State

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1(b), L2(b)</td>
<td>L6(b), L7(b)</td>
<td>L7(b), L3(b)</td>
</tr>
</tbody>
</table>

No-Filter Reservation Example

Adding additional reservations

- R2 forwards $r_3$ over L6
- R1 reserves $b$ units of bandwidth from R1 to R2

Router State

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1], L2[H2], L6[R2]</td>
<td>L5[H5], L6[R1], L7[R3]</td>
<td>L3[H3], L4[H4], L7[R2]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1(b), L2(b), L6(b)</td>
<td>L6(b), L7(b)</td>
<td>L7(b), L3(b)</td>
</tr>
</tbody>
</table>
No-Filter Reservation Example
Complete router path state for the multicast group

- Every host has now reserved sufficient capacity to receive 1 stream

<table>
<thead>
<tr>
<th>Router State</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>L1[H1, L2[H2], L6[R2]]</td>
<td>L5[H5, L6[R1], L7[R3]]</td>
<td>L3[H3, L4[H4], L7[R2]]</td>
</tr>
<tr>
<td>Outbound</td>
<td>L1(b), L2(b), L6(b)</td>
<td>L5(b), L6(b), L7(b)</td>
<td>L7(b), L3(b), L4(b)</td>
</tr>
</tbody>
</table>

No-Filter Reservation Anomaly
Lack of knowledge of senders leads to over-reserving

- Consider the case wherein hosts now reserve bandwidth for 2 streams

- How many streams can ever flow from R1 to R2?
H2, H3, H4, and H5 are members of a multicast group, H1, H4, and H5 are senders.
Each router has received path messages from all sources with the f-flag on.

Filtered Reservation Example
Recording source locations

A router receiving a path message with the f-flag on, must:
» Keep a list of sources that can send on each output link
» Record the previous hop that forwards the source’s data
Filtered Reservation Example

Recording source locations

<table>
<thead>
<tr>
<th>Router State (Output links)</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
</table>
| Source/Previous hop list    | L2[H1⇒H1, H4⇒R2, H5⇒R2]  
                            | L6[H1⇒H1]  |

Filtered Reservation Example

Making a fixed-filter reservation

- H2 reserves bandwidth for one stream from H4
  → R1 reserves bandwidth on L2

<table>
<thead>
<tr>
<th>Router State (Output links)</th>
<th>R1</th>
</tr>
</thead>
</table>
| Source/Previous hop list    | L2[H1⇒H1, H4⇒R2(b), H5⇒R2]  
                            | L6[H1⇒H1]  |
Fixed-Filtered Reservation Example

Forwarding a reservation

- R1 looks up H4 & forwards reservation to R2 only
- R2 reserves $b$ units of bandwidth from R2 to R1

Fixed-Filtered Reservation Example

The complete reservation
Filtered Reservation Example
Making a dynamic-filter reservation

- H5 requests a reservation for bandwidth for any two streams
- R2 reserves 2b over L5
- R2 forwards the reservation...

Filtered Reservation Example
Making a dynamic-filter reservation

- R2 forwards reservation to R1 & R3
- R1 only reserves b bandwidth towards R2
- R3 does nothing

<table>
<thead>
<tr>
<th>Router State (Output links)</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source/Previous hop list</td>
<td>L6<a href="b">H1⇒H1</a> L2<a href="b">...</a> L1[...]</td>
<td>L5<a href="2b">H1⇒R1, H4⇒R3</a> L6<a href="b">...</a> L7[...]</td>
<td>L7[H4⇒H4(b)] L3[...] L4[...]</td>
</tr>
</tbody>
</table>
RSVP Operation

Router soft state

- RSVP state in network elements “times out”
  » Ensures fault tolerance of protocol

- Senders & receivers must periodically send *path/reservation* messages to refresh router state
  » Ensures state is always maintained along path(s) from senders to receivers

- RSVP messages include a *time to live* (TTL) and applications must adhere to a *refresh interval*

RSVP Operation Example III

Soft-state timeout

- H4 crashes or stops sending path/reservation messages
  » Path information for H4 times-out and is deleted

<table>
<thead>
<tr>
<th>Router State (Output links)</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source/Previous hop list</td>
<td>L6[H1⇒H1(b)] L2[H1⇒H1, H5⇒R2, H4⇒R2(b)]</td>
<td>L5[H1⇒R1, H4⇒R3](2b,* H6[H4⇒R3(b), H5⇒H5] L7[H1⇒R1, H5⇒H5]</td>
<td>L7[H4⇒H4(b)] L3[H4⇒H4, H5⇒R2, H1⇒R2] L4[H5⇒R2, H1⇒R2]</td>
</tr>
</tbody>
</table>

![Diagram of RSVP network with nodes H1, H2, H3, H4, R1, R2, R3, and links L1 to L7 with labels b, 2b, and L6, L7 showing the network topology and state transitions.](image-url)
Soft-State Timeout Example

Deleting reservations

- As path information goes away, reservation (refreshes) for H4 are not forwarded

<table>
<thead>
<tr>
<th>Router State (Output links)</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source/Previous hop list</td>
<td>L6[H1⇒H1(b)] L2[H1⇒H1, H5⇒R2]</td>
<td>L5[H1⇒R1(b)] L6[H5⇒H5] L7[H1⇒R1, H5⇒H5]</td>
<td>L3[H5⇒R2, H1⇒R2] L4[H5⇒R2, H1⇒R2]</td>
</tr>
</tbody>
</table>

![Diagram of network showing reservation changes](image)

Soft-State Timeout

Deleting no-filter & fixed-filter reservations

- Note that no-filter and fixed-filter reservations cannot be explicitly deleted
  » A router has no way of knowing if a reservation is shared between multiple receivers

<table>
<thead>
<tr>
<th>Router State (Output links)</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source/Previous hop list</td>
<td>L2[H4⇒R2(b), H5⇒R2]</td>
<td>L6[H5⇒H5, H4⇒R3(b)] L5[H4⇒R3(b)]</td>
<td>L4[H5⇒R2] L7[H4⇒H4(b)]</td>
</tr>
</tbody>
</table>

![Diagram of network showing reservation changes](image)
RSVP as a Protocol

Design issues

- Accommodate heterogeneous receivers
- Adapt to changing multicast group membership
- Exploit the resource needs of different applications to use network resources more efficiently
- Allow users to “change channels” without the possibility of being denied a reservation

Design issues

- Adapt to changes to underlying unicast and multicast routes
- Limit protocol overhead to grow (no worse than) linearly with the size of the group
- Make the protocol modular to accommodate heterogeneous transmission technologies
RSVP as a Protocol

Open issues

- Admission control
- Enforcement of reservations
- Integration with routing
- Appropriate performance semantics
- Making advance reservation
- What to do about legacy LANs