

RSVP Refresh Reduction Extensions

RSVP Working Group Interim Meeting

Lou Berger (lberger@tidalwave.net)

LabN Consulting

April 29, 1999

Draft Information

- “RSVP Refresh Reduction Extensions”
 - draft-berger-rsvp-refresh-reduct-01.txt
 - Authors:Berger, Gan, Swallow
- Derived from “Extensions to RSVP for LSP Tunnels”
 - per discussion in Orlando
 - draft-ietf-mpls-rsvp-lsp-tunnel-01.txt
- Included Extensions
 - RSVP Aggregate Message
 - MESSAGE_ID Extension
 - Hello Extension

Motivation

- MPLS Scalability
 - Must be able to support $O(n^2)$ Label Switched Paths
 - For non-merging solutions
 - N is number of edge routers
 - 100 edge routers $\implies O(10,000)$ reservations
 - 300 edge routers $\implies O(100,000)$ reservations
 - All sessions will be unicast
 - Multicast is for further study
- Other MPLS requirements
 - LSP Merging
 - LSP Re-routing
 - LSP Protection
 - Network resiliency
 - Support for SE and FF style reservations

Motivation (continued)

- Address key RSVP implications
 - Refresh message rate
 - Processing overhead per refresh message
 - Bounded setup/teardown time
 - Rapid failure detection
- Leverage RSVP as signaling protocol
 - Re-use base RSVP capabilities
 - State establishment, teardown, error handling, code
 - Maintain compatibility with existing RSVP (RFC2205) implementations
- Allow for extension implementation on a feature by feature basis

Proposed Approach

- Start with RFC2205 RSVP
- Incrementally reduce raw message rate
 - via message aggregation
 - ==>RSVP Aggregate Message
- Bound state change propagation time
 - via message acknowledgements
 - ==>MESSAGE_ID Extension
- Refresh all state via a single message
 - via refresh message *suppression*
 - ==> Hello Extension and MESSAGE_ID Last_Refresh flag
- Enable arbitrary failure detection interval
 - $n * \text{link RTT}$ to seconds
 - ==> Hello Extension
- Enable implementations to implement only needed extensions
 - Host requirements differ from router requirements

Extension Compatibility Chart

Extension Compatible with:	Message Aggregation	Message Acknowledgement	State Refresh via Single Message
Existing (RFC2205) Implementations	Yes - via Msg Hdr bit	via object class (unknown=ignore)	via object class (unkown=error)*
Unicast Sessions	Yes	Yes	Yes
Multicast Sessions	Yes-When 1 next hop (per oI/F)	Yes - When num next hops known Limited-otherwise	Yes
Multiple Senders	Same as session type	Same as session type	Yes
Reservation Styles	Any	Any	Any

- Single refresh rate limited by link round trip time

(*) - RFC 2205 does not define handling of unknown message types
will either result in no response or Unknown Object Class error

Extensions Overview

- Hello Extension
 - Refreshes all state in a single message
 - Controversy on:
 1. What this message should contain
 2. Handling when out-of-sync state detected
 3. What type of failures should be detected
- Message_ID Extension
 - Provides:
 1. Reliable message delivery
 2. Indication of which state is refreshed by Hello messages
 - Provides reliability for non-Hello supporting neighbors, e.g., hosts
 - Controversy on the size and contents of the message identifier
- Message Aggregation
 - Allows aggregation of RSVP messages
 - Not controversial

Hello Extension

What it does:

- Used to refresh state shared between neighboring RSVP nodes via a single message
 - Refreshes state advertised in messages with MESSAGE_ID last_refresh flag set
- Detects when state may be out-of-sync
- Triggers state resynchronization state when a state *error/failure* is detected
 - Times-out and refreshes all associated state
 - No partial state role-back or resynchronization
- Allows for independent and asymmetric failure detection intervals

Hello Extension

Which state errors are detected:

Objective: Address state synchronization errors that are introduced through specific network operations events

- Errors detectable via the Hello extension:
 - Neighbor node reboots or RSVP state is reset
 - Neighbor node is unreachable
 - Due to single or multi-link failure, can support unnumbered links
 - Neighbor believes it has seen an error
- Errors not covered
 - Lack of system resources
 - Already addressed by existing (RFC2205) protocol mechanisms
 - Non-conformant (to RFC2205) implementations
 - Implementation bugs are not hidden,
They trigger failures, not allocation/state transients

Hello Extension

Discussion Issues

1. Are there other sources of state out-of-sync errors that should be protected against?
 - Non-conformant implementations
 - Improper handling of scarcity/loss of internal resources
 - Other
2. What's the proper handling when state gets out of sync?
 - Partial roll-back / resynchronization
 - Complete time-out and refresh
3. How should node state be represented?
4. What's the real cost of the solution?
 - Limit on failure detection rates
 - CPU cycles & memory required

Hello Extension

How it works:

- Extension composed of two new objects
 - HELLO REQUEST and HELLO ACK objects
 - Both carried in new Hello message
 - Each object carries a value that represents the installed *state* of the sender
 - Value is referred to as the *state instance* value
 - Value is advertised on a per neighbor basis
 - Advertised value may only change on reboot, loss of local RSVP state or detection of a neighbor state synchronization error
- Implementations supporting the Hello extension
 - MUST also support MESSAGE_ID objects
 - MUST be able to properly act on a set last_refresh flag

Hello Extension

How it works: (continued)

- How are errors detected:
 1. A change in neighbor's state instance value
 2. No response from neighbor after a configured period of time
- What's done after an error is detected:

If any state installed or advertised via a message containing a MESSAGE_ID object with the last_refresh flag:

 - Node changes state instance value advertised to neighbor
 - All state installed via a message containing a set last_refresh flag is timed-out
 - All state advertised via a message containing a set last_refresh flag is refreshed
- No partial state role-back or resynchronization

Hello Extension - Formats

- Hello Message format:

```
<Hello Message> ::= <Common Header> [ <INTEGRITY> ] <HELLO>
```

- HELLO REQUEST object

```
Class = HELLO Class, C_Type = 1
```

```

0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Instance                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

- HELLO ACK object

```
Class = HELLO Class, C_Type = 2
```

```

0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Instance                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

- HELLO Class = 22 (Value to be assigned by IANA of form 0bbbbbbb)

Message_ID Extension

What it does:

1. Facilitates identification of non-refresh messages
 - Provides a short hand indication to whether the message represents new state or a state refresh
2. Enables deterministic state propagation
 - Provides mechanism for reliable RSVP message delivery
3. Indicates when Hello is to be used to refresh advertised state
 - Requires that RSVP next hop supports Hello extension
- Supports partial implementation
 - 3 requires 1, 2 and Hello ACK
 - 2 requires 1
 - 1 has no dependencies

Message_ID Extension

How it works:

- Composed of two new objects
 - MESSAGE_ID and MESSAGE_ID ACK
 - Both may be carried in any type of RSVP message
 - MESSAGE_ID ACK Object may also be carried in ACK message
- Each object contains a 24-bit identifier
 - Identifier remains unchanged on refreshes
 - MUST differ on new state advertisement
 - Identifier values have no order requirements (0 is illegal)
 - Sample ways to generate identifier:
 - Global message counter (must cover cross reboot issues)
 - Message CRC or hash (must cover collision for same state)
 - System time based (must cover reboot and collision)
- MESSAGE_ID object contains:
 - A flag for requesting an acknowledgement (for fast retransmit)
 - A flag to indicate that the Hello extension is to be used to refresh associated state*

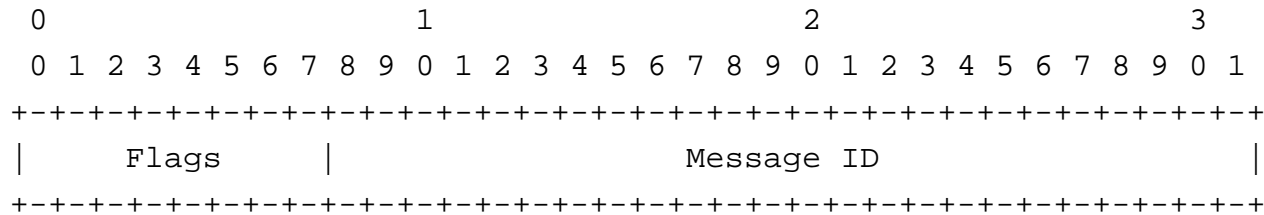
MESSAGE_ID

Multicast Restrictions

- Avoiding ACK implosion
 - Responders wait a random interval prior to acknowledging
- When number of next-hops not known
 - Should only expect a single Ack
 - Means “fast retransmit” until 1st Ack received
 - MUST NOT set last_refresh flag
 - Means using standard RSVP refresh processing
 - Includes most non-RSVP next hop cases
- When new receivers cannot be identified
 - Should only expect a single Ack
 - MUST NOT set last_refresh flag
- When all receivers do not support the Hello extension
 - MUST NOT set last_refresh flag

MESSAGE_ID Format

- Object Format:



- Message_ID field
 - A sender generated value that uniquely identifies message
- Flags
 - ACK_Desired - indicates sender willing to accept an ACK
 - Last_Refresh - indicates that message will not be refreshed once acknowledged
- ACK Flags (for MESSAGE_ID ACK Objects)
 - None

Message_ID Extension

Discussion Issues:

- What should be used as a message identifier?
 - Ordered vs. arbitrary values
 - Time based
 - Monotonically increasing
 - Random
 - Algorithmic
 - How are collisions, wraps and resets/reboots handled?
- What size identifier is needed?
 - 16 bits
 - 24 bits
 - 32 bits
 - 64 bits

Compatibility

Both extensions are fully backward compatible:

- **MESSAGE_ID Class** uses value of form 10bbbbbb
 - Per RFC 2205 classes with values of this form must be ignored and not forwarded by nodes not supporting the class.
 - Non-supporting receivers will silently ignore object
 - Senders will see no ACK and therefore continue with standard RSVP refresh processing
- **Hello related Class** uses values of form 0bbbbbbb
 - Per RFC 2205, this is an “Unknown Object Class”
 - Non-supporting receivers will ignore message or respond with error
 - Senders will see no Hello ACK, and therefore are prohibited from setting No_Refresh flag.