Computer Communications 34 (2011) 1956–1967

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Recovery from control plane failures in the RSVP-TE signaling protocol

Jing Wu *, Michel Savoie

Communications Research Centre Canada, 3701 Carling Avenue, Ottawa, Ontario, Canada K2H 8S2

ARTICLE INFO

Article history:
Received 23 July 2009
Received in revised form 31 May 2011
Accepted 1 June 2011
Available online 17 June 2011

Keywords:
GMPLS control plane
Signaling protocol
RSVP-TE
Control plane failure recovery
Graceful Restart

ABSTRACT

The Resource Reservation Protocol for Traffic Engineering (RSVP-TE) must recover its state after a control plane failure so that the established connections in the data plane continue to be provided full services, are not disrupted by new connection setup requests, and survive any data plane failures. We outline the RSVP-TE Graceful Restart (GR) mechanism defined by the IETF. In this paper, we provide a comprehensive survey of the information that may be carried by RSVP-TE signaling messages when a connection is to be established and the state information that is stored in an RSVP-TE signaling module. We then propose an enhancement for RSVP-TE GR to alleviate the requirement of local recovery of the data plane state. Our proposal includes a two-step RSVP-TE state recovery, which uses a fast recovery to recover the RSVP-TE state in which labels were idle before a control plane failure and a detailed recovery to recover all of the RSVP-TE state. The fast RSVP-TE state recovery is realized as an extension to the RSVP-TE Hello mechanism, allowing a restarting node to process new connection setup requests as quickly as possible without interfering with existing connections. The detailed RSVP-TE state recovery generally follows RSVP-TE GR with minor modifications that can be performed in the background in parallel with normal RSVP-TE operations for connection setup and removal. Our performance evaluations show that our proposal shortens the waiting time for new connection setups being processed by a restarting node.

1. Introduction

Control plane reliability is critical for efficient operation of next-generation networks [1]. New-generation Multi-Protocol Label Switching (MPLS) switches/routers are able to maintain label switching states in the data plane following control plane failures or during their maintenance [2]. In a Generalized Multi-Protocol Label Switching (GMPLS) network, failures in the control and data planes occur more or less independently when they are physically separate [3–7]. In planned control plane maintenance or accidental control plane failures, methods are required to minimize service interruptions to data plane connections.

The Resource Reservation Protocol for Traffic Engineering (RSVP-TE) is a signaling protocol for MPLS and GMPLS networks. IETF defines traffic engineering extensions to the RSVP so that connections can be established, maintained and removed in an MPLS network [8]. The RSVP-TE is also the preferred signaling protocol for GMPLS-controlled circuit-switched networks [9–11]. Between two neighbors, an RSVP-TE session is used to exchange RSVP-TE messages and control the corresponding data plane links.

The RSVP-TE Graceful Restart (GR) mechanism enables a restarting node to recover its lost RSVP-TE state. An RSVP-TE session terminates when its two end nodes cannot communicate for a certain period or when either of the two nodes fails. A failed node may lose its RSVP-TE state completely or lose the synchronization of its RSVP-TE state with its neighbors on restart. RSVP-TE GR requires that all the states of an RSVP-TE session should be redundantly stored in the upstream and downstream neighbors of the session [10,12]. A restarting node can then recover its RSVP-TE state from its neighbors. RSVP-TE GR may recover the RSVP-TE state in any single-point failure.

RSVP-TE GR requires a restarting node to locally recover or retrieve its related data plane state. As a subset of the RSVP-TE state, the data plane state reflects the state of the resources that are visible to neighbors, such as incoming or outgoing ports. The data plane state is modified using RSVP-TE as connections are set up or removed. The key data plane state is defined by forwarding tables in MPLS networks or cross-connection tables in GMPLS networks. An MPLS forwarding table maintains key information specifying the appropriate incoming and outgoing label/port mapping [13]; a GMPLS cross-connection table maintains incoming and outgoing timeslot/wavelength-channel/port mapping.

RSVP-TE GR cannot operate properly if a restarting node does not recover its data plane state first. RSVP-TE GR processes connection setup requests while the RSVP-TE state is being recovered. New connection setups and the RSVP-TE state recovery are conducted independently and in parallel. Without the data plane resource state being recovered, there is a potential risk that a new connection setup may accidentally disrupt an existing connection

* Corresponding author. Tel.: +1 613 9982474; fax: +1 613 9908382.
E-mail addresses: jing.wu@crc.gc.ca (J. Wu), michel.savoie@crc.gc.ca (M. Savoie).

0140-3664/$ - see front matter Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2011.06.001

by using resources allocated to it. RSVP-TE GR verifies the recovered RSVP-TE state by comparing it with the locally recovered data plane state (Section 4.5.2.2 of [12]).

The local recovery or retrieval of the data plane state may be impossible or unfavorable for switching elements. Local recovery without information exchange with neighbors is a challenge and may not be accomplished by all switching elements. First, a node failure may lose or damage the data plane state, even though the hardware in the switching elements keeps existing connections operating. Second, not all switching elements are able to provide the preserved MPLS forwarding tables or GMPLS cross-connection tables to their restarting RSVP-TE module.

Our design goal is to extend the RSVP-TE GR so that the requirement of local recovery of the data plane state is not mandatory. In our proposed enhanced RSVP-TE GR, this requirement becomes optional. If the data plane state is successfully locally recovered, it can be used to assist the RSVP-TE state recovery, for example, for the verification of recovered information from an RSVP-TE neighbor or to speed up recovery. However, our design solely relies on information from neighbors. This is the key contribution of this paper.

We provide an informational clarification and performance evaluation of the procedural process of connection removal requests during the recovery. Because a restarting node may receive connection removal requests during its recovery, it must coordinate the processing of the requests and its recovery. This has not been clearly defined by the IETF and is the second contribution of this paper.

In this paper, we provide a comprehensive survey of the information that may be carried by RSVP-TE signaling messages when a connection is to be established and the state information that is stored in an RSVP-TE signaling module. Over the past 15 years, RSVP and RSVP-TE have been continuously developed and extended. The extensions are scattered in many IETF standards. We provide a survey of the information carried by RSVP-TE signaling messages and an outline of the related RSVP-TE state information so that the scale of RSVP-TE state recovery discussed in this paper can be assessed. This is the third contribution of this paper.

This paper is organized as follow. In Section 2, we discuss recovery time requirements of the GMPLS protocols. In Section 3, we present an overview of the RSVP-TE GR and outline the RSVP-TE operations for GMPLS-controlled optical networks. We propose a two-step RSVP-TE state recovery in Section 4, followed by performance evaluation results in Section 5. We conclude the paper in Section 6. A complete view of the state related to RSVP-TE sessions and *Path* and *Resv* messages is given in the Appendix A.

2. Recovery time requirements of the GMPLS protocols

GMPLS protocols play important roles in data plane failure restoration and shared protection. The biggest value of building a separate GMPLS control plane is to save data plane resources used for protection. Compared with 1 + 1 or 1 : 1 protection, significant savings may be achieved by using restoration and shared protection [14]. However, these approaches need signaling protocols for rapid fault notification and data plane connection setup or switchover. The target time is 200–300 ms and 100–200 ms to restore a failed data plane connection and complete a shared protection, respectively [15].

The failure recovery time of the GMPLS protocols varies. The signaling protocol availability directly impacts the survivability of data plane connections. With an increase of the out-of-service time of signaling protocols due to a control plane failure, there is an increased risk of data plane connections not being protected or restored in time. In this sense, the signaling protocol recovery is service impacting, and data plane connections are affected. In contrast, the only rule of the Link Management Protocol (LMP) in a failure recovery is fault isolation, which is a maintenance action and does not have a strict time requirement [4]. A restoration of a failed data plane connection must search for an alternative route in the Traffic Engineering Database (TED), which is populated by routing protocols [16,17]. Before routing protocols recover from a control plane failure, their link state advertisements are postponed, resulting in the TED not being updated. However, a connection setup tolerates the inaccuracy of the TED even in the normal state because the TED is often out-of-date due to its distributed updating process. Consequently, the time requirement of the routing protocol recovery is not critical.

3. Overview of the RSVP-TE GR mechanism

3.1. Overview of the RSVP-TE operations for GMPLS controlled optical networks

In this paper, our major focus is the RSVP-TE operations for GMPLS-controlled optical networks. RSVP-TE has been chosen as the preferred GMPLS signaling protocol [11]. The original RSVP was proposed to support resource reservation and Quality of Service (QoS) in packet networks [18,19]. Since then, many enhancements have been proposed and standardized. For the latest progress, refer to work in the two IETF working groups: common control and measurement plane [20] and MPLS [21]. Our discussions apply to the RSVP-TE extensions for establishing, maintaining and removing point-to-point unidirectional connections (i.e., lightpaths) in GMPLS-controlled optical networks. Other applications may need modifications, but the proposed concept and framework remain largely valid. We assume the control plane is secured and control nodes trust each other; no RSVP-TE security feature to prevent intrusions is considered.

RSVP-TE uses *Path* and *Resv* messages to establish connections. *Path* messages are sent downstream from the head (i.e., ingress) to the tail (i.e., egress) of a connection, following a computed route [19]. *Resv* messages are sent upstream, following the same route but in the opposite direction. The ingress creates an entry in its state base and then sends out a *Path* message. Upon receipt of a *Path* message, an intermediate control node creates an entry in its state base and then generates another *Path* message to send downstream. The same process repeats until a *Path* message arrives at the egress. The egress makes necessary resource reservations, updates its corresponding state entry, and generates a *Resv* message to send upstream. Upon receipt of a *Resv* message, an intermediate control node makes resource reservations and necessary cross-connections, updates its corresponding state entry, and generates a *Resv* message to send upstream. The same process repeats until a *Resv* message arrives at the ingress. To establish connections in GMPLS-controlled optical networks, *Path* and *Resv* messages carry additional information, such as generalized label objects [10]. A complete list of the up-to-date defined objects in *Path* and *Resv* messages by the IETF is shown in Table 1. The detailed RSVP-TE state is given in the Appendix A.

The entry for a given connection in a state base is identified by the connection's source and destination nodes, and if necessary, a connection ID. The "Session" object carries the destination node (i.e., egress) in all *Path* and *Resv* messages. The "Sender template" object carries the source node (i.e., ingress) in all *Path* messages, while the "Filter spec" object carries the source node in all *Resv* messages. When there are multiple connections between the same ingress-egress pair, a connection ID (i.e., Label Switched Path – LSP ID) may be added in the "Sender template" or "Filter spec" objects [8].

In this paper, our discussions are not limited to any particular information model of the RSVP-TE state. The information model

Table 1
Objects in *Path* and *Resv* messages.

Objects	Descriptions	Mandatory or optional in *Path* messages	Mandatory or optional in *Resv* messages	Reference IETF RFCs
Common header	Message type, total length of this RSVP message, etc.	Mandatory	Mandatory	[19]
Session	An RSVP session's destination (i.e., egress) IP address and port, IP protocol ID	Mandatory	Mandatory	[19,8]
RSVP hop	The IP address and data plane interface of the node that sent this message. It is a "previous hop" object for a downstream message or a "next hop" object for an upstream message.	Mandatory	Mandatory	[19]
Time values	The time period that the sender uses to refresh this message	Mandatory	Mandatory	[19]
Label request	Request a label to establish an LSP tunnel	Mandatory		[8,10]
Sender template	An RSVP session's source (i.e., ingress) IP address and optional information, such as the ingress's port, LSP ID	Mandatory		[19,8]
Sender TSPEC	The traffic characteristics of a data flow	Mandatory		[19,22,23]
Filter spec	Identical to the "sender template"		Mandatory	[19,8]
Flowspec	A desired QoS		Mandatory	[19,22,23]
Style	Reservation style		Mandatory	[19]
RSVP label	Label allocated for this connection		Mandatory	[8,10]
Integrity	Cryptographic data to authenticate the originating node and verify the contents of this message	Optional	Optional	[19]
Message ID; Message ID ACK/NACK; Message ID list	Support acknowledgments and reliable RSVP message delivery, summary refresh extension	Optional	Optional	[24]
Notify request; Alarm spec	Identify where event notifications are to be sent, and what alarm information should be sent	Optional	Optional	[10,25]
Session of interest	Support RSVP session aggregation	Optional	Optional	[26]
Diffserv	Support differentiated services	Optional		[27]
Admin status	Administrative state of a particular LSP	Optional	Optional	[10]
Policy data	Information for a local policy module to decide whether the associated reservation is administratively permitted	Optional	Optional	[19]
Record route	Record a route to collect detailed path information	Optional	Optional	[8]
Explicit route; Exclude route	A particular path that is described as a group of nodes that must or must not be traversed	Optional		[8,28]
Label set; Suggested label	The set of labels that downstream nodes can choose from, or the upstream node's label preference	Optional		[9,10]
Session attribute; LSP attributes; LSP required attributes	Attributes required to support an LSP, such as setup and holding priorities, resource affinities, local protection, and fast re-reroute functionality	Optional		[8,29,30]
ADSPEC	Enable nodes in the path to advertise their service capabilities, resource availability, and transmission characteristics	Optional		[19]
Recovery label	Support the RSVP Graceful Restart by providing the label in corresponding *Resv* messages	Optional		[10]
Upstream label	Support bidirectional LSP setup	Optional		[10]
Protection; Association; Primary path route; Detour; Fast reroute; Secondary record route; Secondary explicit route	Support protection of an LSP, for example, associating a recovery LSP with the LSP it is protecting, indicating the branch and merge nodes of recovery LSPs, etc.	Optional		[10,31,32,29]
Resv confirm	The IP address of the receiver that requested a confirmation		Optional	[19]
Scope	Senders towards which this message is to be forwarded		Optional	[19]

that a node uses to store its RSVP-TE state varies in different implementations. In general, the RSVP-TE state is stored in path and reservation state blocks, as well as in other state storage modules. A relational information model was used to present an algorithmic description of the RSVP message processing rules in [33]. A relational style search was used to find state blocks in [33], although the state blocks were not required to be stored in a relational database. Objects are used to present the processing rules on the basis of object relationships between state blocks in a prototype RSVP implementation [34–36]. The state information was represented as relations, and their functional dependencies were identified. The relationships between objects were explicitly stored. The object relationships are generally more expressive and efficient than finding state blocks on the basis of relational rules.

Additional RSVP-TE messages are defined for confirmation, connection removal, error report and notification. These messages are optional in some RSVP-TE operation modes, but using them improves the RSVP-TE performance. For example, *PathTear* and *ResvTear* messages remove state promptly [19]. *PathErr* and *ResvErr* messages report errors in *Path* and *Resv* messages, such as parameter errors [19]. A *ResvConfirm* message explicitly confirms the success of processing a *Resv* message [19]. A node may send a *Notify* message to another node, targeting either its immediate neighbor or a non-adjacent node [10].

Although RSVP-TE supports a soft-state approach, an increasing trend is to operate RSVP-TE in a hard-state manner for GMPLS-controlled optical networks. When RSVP-TE operates in a soft-state manner, it requires sending periodic refresh *Path* and *Resv* messages. Failure to receive a certain number of consecutive refresh *Path* or *Resv* messages normally results in a timeout and removal of the corresponding state [19]. In GMPLS-controlled optical networks, the removal of established connections due to control plane failures should be prevented [6,7]. A solution is operating RSVP-TE in a hard-state manner by disabling the state timeout mechanism. However, this solution should only be used with the reliable delivery of trigger messages. Otherwise, if a trigger message is lost, the connection may never be established or removed [37]. Reliable delivery of messages can be achieved by using *ACK* messages and proper re-transmission of lost messages [24].

3.2. Standard RSVP-TE GR mechanism

RSVP-TE GR enhances two aspects of RSVP-TE: the Hello mechanism and the restarting and recovery behaviors. In addition to

detecting the reachability of a neighbor control node [8], the Hello mechanism is extended to diagnose the failure type (a control channel or node failure) [10] and announce the GR capability and associated parameters [10,12]. The extended restarting and recovery behaviors are specified for two cases: a restarting downstream node assisted by its upstream neighbor and a restarting upstream node assisted by its downstream neighbor [10,12]. RSVP-TE GR has two phases: restart and recovery. The restart phase is between a failure's occurrence and the completion of a new Hello establishment; the recovery phase ends when the RSVP-TE state is recovered or the recovery times out.

RSVP-TE GR extends the Hello mechanism to diagnose the failure type: a control node failure that results in a control node restart or a control channel failure that does not result in a control node restart [10]. Two nodes exchange Hello messages, in which a sender puts the instance numbers of itself and of the neighbor, namely, the source and destination instance numbers. A node diagnoses a failure type by monitoring changes of the neighbor's instance number or the neighbor's reflection of this node's instance number. More details on how the Hello mechanism is extended to detect and diagnose an RSVP-TE failure are given in Chapter 9 of [2].

The Hello mechanism is extended to announce RSVP-TE GR capability and associated parameters. A node that supports RSVP-TE GR must announce two parameters – restart time and recovery time – in all of its Hello messages [10]. A node's restart time is the total time of its RSVP-TE restart and Hello establishment with a given neighbor. Its recovery time is the maximum time spent recovering its RSVP-TE state from the neighbor. In RSVP-TE GR, a special technique is defined to recover the RSVP-TE state from a downstream neighbor [12]. If a node needs its downstream neighbor's help or if it wants to help its upstream neighbor, it must announce its desire or capability to do so in its Hello messages to the neighbor.

RSVP-TE GR defines the restarting behaviors of a restarting node and its neighbor [10]. A node detects a failure when the number of continuously missing Hello messages that should arrive reaches a set threshold. The node assumes the worst failure case (i.e., the neighbor restarts) and waits for a period equal to the restart time that the restarting neighbor declared before the failure. The node suspends the RSVP-TE state, refreshing to the restarting neighbor, but maintains the established data plane connections. Any incomplete connection establishment associated with the restarting neighbor is removed. If no Hello message arrives within the restart time, the node stops RSVP-TE GR, releases data plane connections and cleans up the state associated with the restarting neighbor. If RSVP-TE GR does not complete within the recovery time, the unrecovered state is cleaned up, and the corresponding data plane connections are released.

When a downstream node restarts, it recovers its RSVP-TE state from its upstream neighbor by receiving the *Path* messages carrying the Recovery Label to provide the most recently received label value in the corresponding *Resv* messages. Handling of a received *Path* message is defined in Section 9.5.2 in [10] and is modified in Section 4.5.2 in [12]. The modification is made to synchronize the recovery of the RSVP-TE state in a transit node (i.e., a node that is neither a non-ingress nor non-egress node), where the recovery from upstream and downstream neighbors must be co-ordinated. The RSVP-TE state in a transit node is considered "re-synchronized" when its recovery from both upstream and downstream neighbors is completed as defined in Section 4.5.2 of [12]. The handling procedure is illustrated in Fig. 1. Note that new connection setup requests could potentially be processed during recovery, for example, in steps 9 or 13 in Fig. 1.

When an upstream node restarts, it recovers its RSVP-TE state from its downstream neighbor by receiving *RecoveryPath* and retransmitted *Resv* messages. The *RecoveryPath* messages copy back the *Path* messages (excluding message ID and a few other objects) that were previously sent out from the restarting node. Before using *RecoveryPath* messages, the restarting node and its downstream neighbor must mutually agree upon their use in the Hello mechanism [12]. The restarting node re-synchronizes its RSVP-TE state related to a data plane connection after it receives the *Path* message with the Recovery Label from its upstream neighbor with respect to the connection and corresponding *RecoveryPath* message from its downstream neighbor. Upon synchronization, the restarting node must send a triggered *Path* message to its downstream neighbor, which in turn triggers a re-transmitted *Resv* message from its downstream neighbor. After receiving the re-transmitted *Resv* message, the restarting node recovers its state that represents the cross-connect for the connection.

3.3. Related work on control plane failure recovery in the GMPLS control plane

Control plane failure recovery has been studied architecturally, analytically, and experimentally for different protocols in the GMPLS control plane. GR has been applied to GMPLS routing and signaling protocols [2]. The details of the architectures and specifications of the GR for different protocols are defined by the IETF (see, for example, [20,21]). An overview of RSVP-TE GR was given in [37]. The performance of RSVP-TE GR was evaluated using analytical models and simulations [38–40]. The performance of general RSVP or RSVP-TE was evaluated with a focus on message loss [41,42], processing and bandwidth requirements [42,43], scalability [36,44], and connection setup time [45]. We presented the concept of RSVP-TE GR enhancement in [46]. Our previous work on the enhancements of a signaling protocol (the Constraint-based Routing Label Distribution Protocol (CR-LDP)) was presented in [47,48].

4. Proposal of a two-step RSVP-TE state recovery

4.1. Description of a two-step RSVP-TE state recovery

In standard RSVP-TE GR, requests for connection setup and removal are processed as soon as the restart phase completes. In the recovery phase, RSVP-TE state recovery messages are mixed with messages for connection setup and removal. In this way, the delay for connection setup and removal is reduced. However, if the data plane state cannot be recovered first, new connections could accidentally take in-use resources and disrupt existing connections.

We propose to recover the RSVP-TE state in two steps, where the first step is to recover the RSVP-TE state in which labels were idle before a control plane failure, and the second step is to recover the complete RSVP-TE state. The first step is realized as an extension to the RSVP-TE Hello mechanism, where a neighbor announces the idle labels to the restarting node with respect to the RSVP session. The second step follows RSVP-TE GR, except that verification with the locally recovered data plane state becomes optional.

Our extensions to the RSVP-TE Hello mechanism for idle label announcement are as follows:

- The capability of transmitting and receiving idle label announcements should be carried by the capability object (i.e., the RestartCap object) in Hello messages. The restarting node must tell its neighbor of its desire to receive idle label announcements or not. If the restarting node can locally recover its data plane state, it should tell its neighbor that it does not want to receive idle label announcements. A neighbor must tell the restarting node whether it is able to provide idle labels. In

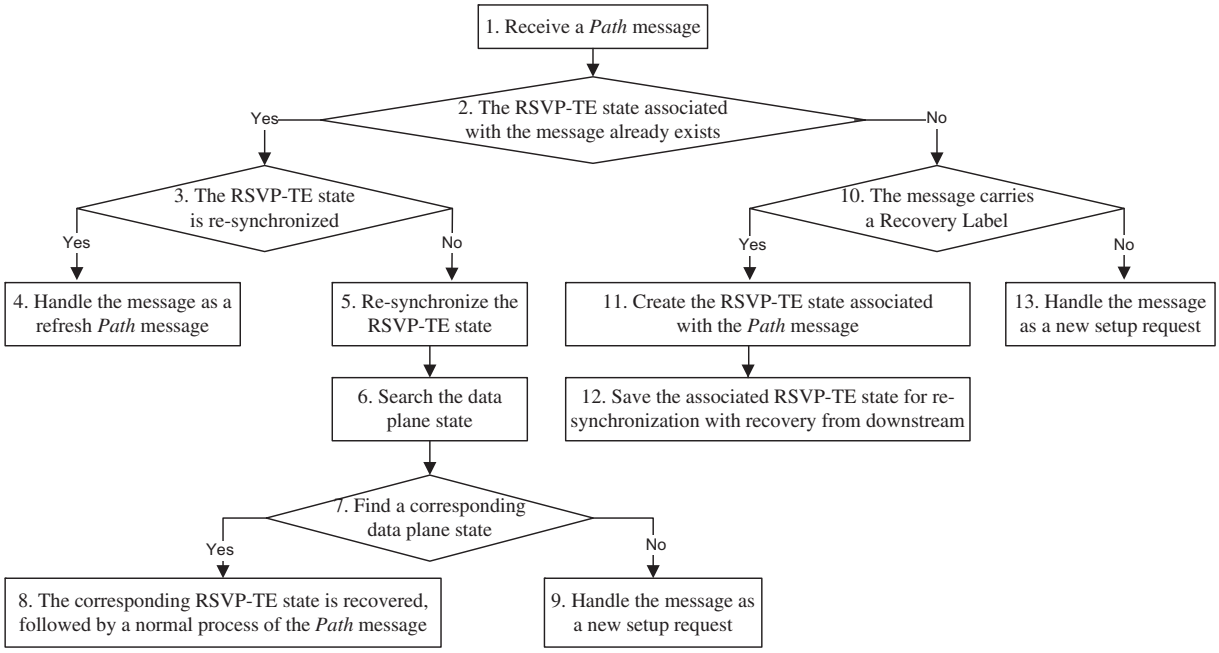

Fig. 1. Handling of a received *Path* message at a restarting node.

the worst case, where the neighbor does not support idle label announcements and the restarting node cannot locally recover its date plane state, the restarting node must wait for the completion of its in-use label recovery in the second step before processing new connection establishment requests.

- The idle label announcements start from the beginning of the recovery phase.
- The idle label announcements stop either when all idle labels are announced or at the end of the recovery phase, when all in-use labels should have already been recovered. Unrecovered labels will be removed at the end of the recovery phase.
- The idle label announcements are delivered on a best-effort basis, and thus some announcements could be lost. Furthermore, the neighbor that sends out the idle label announcements does not know whether they arrive at or are processed by the restarting node, as there is no acknowledgement.
- A neighbor does not repeat its announcement for the same idle label.
- Idle labels may be encoded in two formats [9]: a generalized label (32 bits) and a waveband switching label (96 bits). The generalized label format is used for individual labels; the waveband switching label format is used for a block of consecutive labels (not necessarily in the same waveband), where only the start and end labels are specified.

Coordination of the RSVP-TE state recovery in a transit node is modified so that it does not rely on the data plane state. When recovery from either upstream or downstream completes, the other side of the connection is checked using the attributes that have just been recovered (i.e., the ingress, egress, and LSP ID). Only when both sides of the connection complete the recovery is the RSVP-TE state for the connection considered recovered.

In the recovery phase, the restarting node can only process a new connection setup request when it is certain that there is no risk of disrupting existing connections. That means a connection setup request can be processed when an idle label meeting the requirement is known via idle label announcements. If such a risk cannot be ruled out, the request must wait until an idle label is known or the recovery phase completes.

In the following section, we present a functional description of the operational steps of the proposed two-step RSVP-TE state recovery. The sequence of operational steps is for the purpose of illustration and may not necessarily correspond to a particular implementation. The idle label announcement mechanism is illustrated in Fig. 2. The attributes in a Hello message are the source instance number, destination instance number, restart time (if irrelevant, we use the symbol "..."), recovery time, and idle label blocks (if any exist).

1. The middle node exchanges RSVP-TE Hello messages with its upstream and downstream nodes, informing each other about RSVP-TE GR capabilities. In addition to the two attributes (i.e., restart time and recovery time) defined in the RestartCap object for RSVP-TE GR capabilities, the upstream and downstream nodes announce their capabilities of transmitting idle labels, and the middle node announces its capability of receiving idle labels. In Fig. 2, the middle node sends the upstream and downstream nodes Hello messages with the restart time set to value "P" and recovery time set to value "Q".
2. Connections are established, including connections from (or through) the upstream node towards the middle node and connections from (or through) the middle node towards the downstream node.
3. Assume the control module of RSVP-TE in the middle node fails.
4. Using the RSVP-TE Hello mechanism, the upstream and downstream nodes detect the loss of Hello communication to the middle node.
5. The upstream and downstream nodes detect a failure of the middle node by counting the number of expected but lost Hello messages from the middle node. They then wait at least the amount of time indicated by the restart time (i.e., value "P") in the RestartCap object from the middle node. During this waiting time, they send Hello messages with the destination instance number set to 0 and source instance number unchanged. In Fig. 2, the upstream node sends Hello messages with the destination instance number set to 0

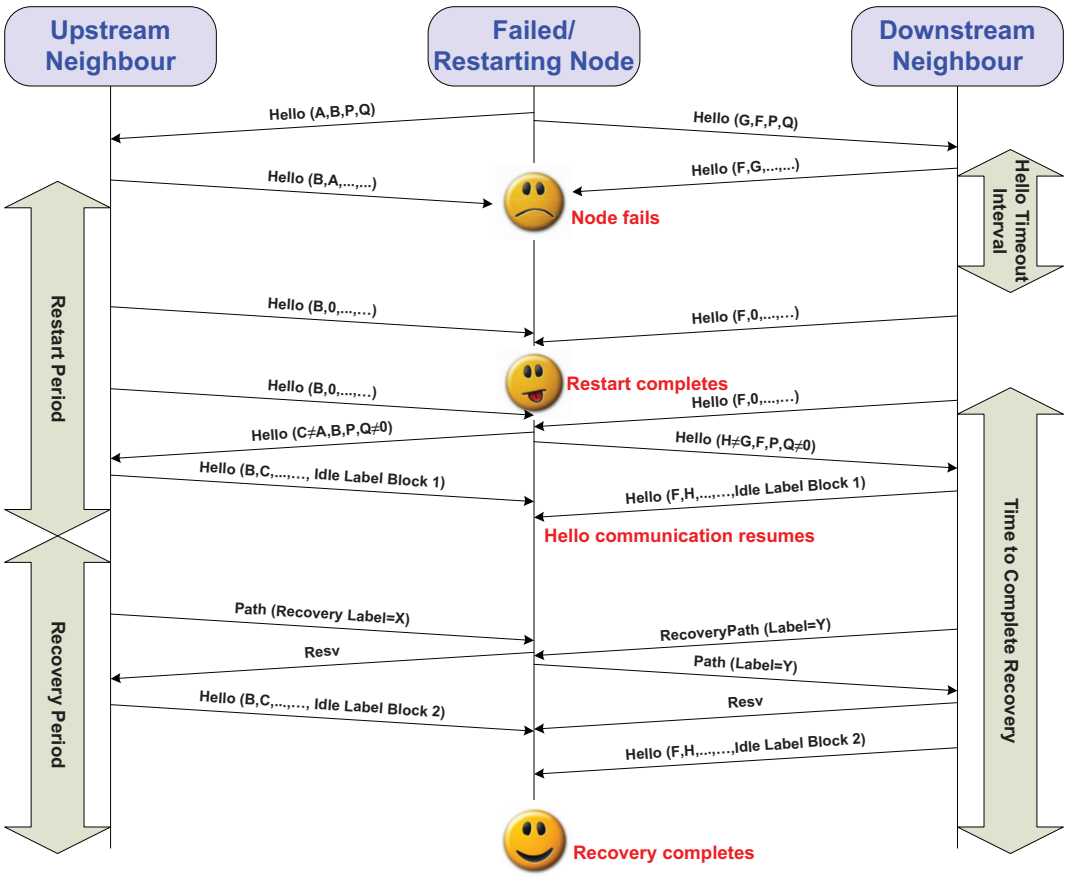

Fig. 2. Idle label announcements in the proposed RSVP-TE Hello mechanism. The timeline starts from the top and progresses towards the bottom. The timeline is not to scale.

instead of its previous value of "A" and the source instance number unchanged as value "B". The downstream node sends Hello messages with the destination instance number set to 0 instead of its previous value of "G" and the source instance number unchanged as value "F". While waiting, the upstream and downstream nodes retain their RSVP-TE state for the established connections that traverse them. They treat the RSVP-TE state of the established connections as if they still received RSVP refresh messages from the middle node. In the waiting period, the upstream and downstream nodes suppress the normal refreshing of the RSVP-TE state.

6. Once the middle node restarts, it initiates the RSVP-TE state recovery process. It advertises its allowable recovery time in the RestartCap object to the upstream and downstream nodes. During the recovery period, the middle node sends Hello messages that contain the following elements: (i) the recovery time set to a non-zero duration; (ii) the destination instance number reflecting the neighbor's source instance number or destination instance number set to 0 if the middle node has not yet received Hello messages from the neighbor; and (iii) the source instance number set to a new value. As shown in Fig. 2, in Hello messages to the upstream node, the middle node sets its source instance number to value "C" (different than the "A" value used previously) and uses the destination instance number to reflect the upstream node's source instance number (i.e., value "B"). In Hello messages to the downstream node, the source instance number is set to value "H" (different than value "G"), and the destination instance number reflects the downstream node's source instance number (i.e., value "F").
7. Once the upstream and downstream nodes receive new Hello messages from the middle node, they know that the middle node has restarted because the source instance number of the middle node's Hello messages has changed. They also know that the middle node wants to recover its lost RSVP-TE state because the middle node indicates a non-zero recovery time (i.e., value "Q"). From one of our proposed new attributes – the capability of receiving idle labels – in the RestartCap object sent by the middle node, the upstream and downstream nodes know that the middle node can facilitate our proposed two-step recovery. The upstream and downstream nodes then mark all of their RSVP-TE states shared with the middle node as stale, retain the stale state, and continue to use the corresponding connections.
8. The upstream and downstream nodes announce their idle labels shared with the middle node in Hello messages to the middle node. They also set one of our proposed new attributes – the capability of transmitting idle labels – in the RestartCap object sent to the middle node. They the initiate RSVP-TE GR as described in the following steps. During the recovery period, they do not refresh their RSVP-TE state shared with the middle node until corresponding *Path* messages have been received.
9. For each connection that travels from the upstream node to the middle node, the upstream node places its outgoing label (the label which was received in a *Resv* message from the middle node before the middle node restarted) in the Recovery Label object of the *Path* message and sends it to the middle node, e.g., in Fig. 2, "Recovery Label = X".

10. In our example, the downstream node does not send a *Path* message to the middle node because no connection travels from the downstream node to the middle node. The downstream node waits until it has received a *Path* message from the middle node to refresh the corresponding RSVP-TE state in the downstream node.
11. The middle node processes the received *Path* message according to the procedure illustrated in Fig. 1. Assume the middle node does not find the RSVP-TE state corresponding to the received *Path* message. The middle node therefore creates an RSVP-TE state and waits for re-synchronization with the downstream node for the created RSVP-TE state. The middle node sends a *Resv* message to the upstream node so that the upstream node can clear its stale flag for the label.
12. For each connection that travels from the middle node to the downstream node, the downstream node places its incoming label (the label assigned by the downstream node and sent to the middle node in a *Resv* message before the middle node restarted) in a *RecoveryPath* message and sends it to the middle node, e.g., in Fig. 2, "Label = Y".
13. The middle node re-synchronizes its RSVP-TE state by associating its incoming and outgoing labels together for pass-through connections. Then, the middle node sends the downstream node a *Path* message, e.g., in Fig. 2, "Label = Y". The downstream node clears its stale flag for the label.
14. The recovery period expires. The upstream node sends *PathTear* messages to the middle node for all stale labels. The downstream node sends *ResvTear* messages to the middle node for all stale labels. The recovery is completed.

4.2. Processing of connection removal requests during the RSVP-TE state recovery

In the recovery phase, the processing of connection removal requests must be coordinated with recovery of the related RSVP-TE state. We present a generic scenario in which a restarting node is an intermediate node of the connection to be removed. The restarting node always releases the resources used by the connection upon receipt of a connection removal request. Additional actions in the restarting node are shown in Table 2. As a general rule, RSVP-TE state recovery can never install a new connection. This rule also applies to the connection removed first, but the recovery for the state of the removed connection arrives later. RSVP-TE state recovery cannot change the state of a resource from idle (i.e., released after the connection removal) to in-use (i.e., used by a previous connection).

5. Performance evaluation

5.1. General assumptions for performance evaluation

We use several rules to avoid overloading a restarting node. First, up to 128 bits can be used in one Hello message for idle label announcements. This capacity includes a 32-bit header of the new object for idle label announcements. Consequently, only one waveband switching label or at most three generalized labels can be announced in one Hello message. Second, the waveband-switching label containing the highest number of idle labels should be announced first. Third, the restarting node sets its restart time and recovery time to big values and uniformly spreads the RSVP-TE GR messages in the time horizon. To cope with unexpected delays and prevent premature termination of the recovery phase, we set a safeguard in the recovery time of approximately 20%, i.e., the RSVP-TE GR messages are spread over the first 80% of the recovery time. However, note that setting the restart and recovery times to large values results in long delays in cleaning up stale labels and releasing corresponding resources.

We assume *Path* and *Resv* messages are 332 bytes. Although the minimum IP packet size is around 120 bytes for *Path* and *Resv* messages, a measurement of a commercial RSVP implementation showed that the *Path* and *Resv* messages are 332 bytes long, which indicates that optional objectives are intensively used [42]. In another commercial implementation, messages for establishing connections are normally more than 300 bytes and sometimes as long as 1000 bytes, while messages for releasing connections are about 200 bytes [43].

Table 2
Process procedure of received connection removal messages at a restarting node.

The RSVP-TE message arrived at the restarting node	Is the recovery with the upstream neighbor completed for the connection?	Is the recovery with the downstream neighbor completed for the connection?	Actions in the restarting node
PathTear	Yes	Yes	Send downstream neighbor a triggered *PathTear* message
PathTear	Yes	No	The triggered *PathTear* message that should be sent to the downstream neighbor is delayed until the downstream neighbor's IP address and interface are recovered
PathTear	No	Yes	Send downstream neighbor a triggered *PathTear* message; if the recovery with the upstream neighbor for the removed connection arrives later, a *PathErr* message should be sent to the upstream neighbor
PathTear	No	No	The triggered *PathTear* message that should be sent to the downstream neighbor is delayed until the downstream neighbor's IP address and interface are recovered; if the recovery with the upstream neighbor for the removed connection arrives later, a *PathErr* message should be sent to the upstream neighbor
ResvTear	Yes	Yes	Send upstream neighbor a triggered *ResvTear* message
ResvTear	Yes	No	Send upstream neighbor a triggered *ResvTear* message; if the recovery with the downstream neighbor for the removed connection arrives later, a *ResvErr* message should be sent to the downstream neighbor
ResvTear	No	Yes	The triggered *ResvTear* message that should be sent to the upstream neighbor is delayed until the upstream neighbor's IP address and interface are recovered
ResvTear	No	No	The triggered *ResvTear* message that should be sent to the upstream neighbor is delayed until the upstream neighbor's IP address and interface are recovered; if the recovery with the downstream neighbor for the removed connection arrives later, a *ResvErr* message should be sent to the downstream neighbor

Table 3
Processing time of received RSVP-TE messages.

RSVP-TE messages	Message processing time (ms)
Path message	50
Resv message	50
RecoveryPath message	50
Hello message with idle label announcement	7
Hello message without idle label announcement	5
Acknowledgement message	5

Table 4
Processing time of generating RSVP-TE messages.

RSVP-TE messages	Processing time for message generation (ms)
Path message	10
Resv message	10
RecoveryPath message	10
Hello message	2
Acknowledgement message	2

The processing time of RSVP-TE messages has a dramatic impact on overall recovery time. Our assumptions regarding the processing time of received messages are shown in Table 3. Our assumptions are consistent with the performance evaluation numbers used in [39], which are based on an experiment on a commercial router [44]. In another experiment, the measured processing time of a *Path* message is about 91 ms, while the measured processing time of a *Resv* message is about 8.6 ms [45]. Our assumptions on the message generation time are shown in Table 4, and these assumptions are consistent with [38]. We assume that there is a single processor for incoming messages at the restarting node, and that all incoming messages share a single input queue. It is assumed that the propagation delay and processing and queuing delays at neighbors are negligible. The refresh period of *Path* and *Resv* messages is 30 s, and the lifetime factor is 3. The Hello interval is 1 s and the hello timeout interval is 3.5 s, although no hello timeout is allowed at the restarting node during the recovery period. Reliable RSVP-TE message delivery is implemented and applied to all *Path*, *Resv* and *RecoveryPath* messages sent during the recovery period; unacknowledged messages are retransmitted up to three times, with the retransmission interval initially set to 0.5 s and then doubling with each retransmission.

We assume that approximately 1/3 of the processor time is allocated to the RSVP-TE signaling protocol, while the rest is used for other protocols. At a middle restarting node, processing of the messages for one connection takes about 170 ms (recovery with both the upstream and downstream neighbors). Therefore, processing the RSVP-TE state recovery messages for two connections takes about 1 s. At a restarting ingress or egress node, processing the RSVP-TE state recovery messages for four connections takes about 1 s.

5.2. Performance evaluation results

We first evaluate the performance in a simple three-node example network. A restarting node has two neighbors: one upstream and one downstream (shown in Fig. 3). Because we assume the restarting node has a single processor and message queue, and the delays at neighbors are negligible, the performance evaluation results should be the same when there are multiple neighbors. We assume the total number of labels on one link is 4096 for multiple fibers and each has multiple wavelength-channels, and there is only one RSVP-TE session for a link. There are 2048 in-use labels randomly distributed within the entire label space.

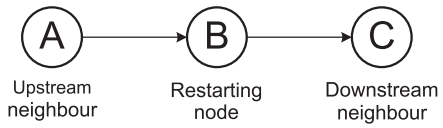

Fig. 3. Simulation example network.

We compare the waiting time to establish a new connection with and without the proposed enhancement. The simulation results of the maximum waiting time for a new connection request are shown in Fig. 4. The results are the average of 100 simulations. At a low load of new connection requests, the waiting time is reduced significantly. At a high load, waiting time increases compared with at a low load but is much better than without the proposed enhancement, demonstrating scalability with respect to the request load. A new connection request may require an idle label that satisfies certain constraints. When the constraints become tighter, the chance of finding a matching idle label becomes smaller, and the waiting time increases.

We evaluated the recovery status of upstream and downstream labels with respect to the load of the connection removal requests. The probability of three-label recovery status is shown in Fig. 5: none, one or both of the upstream and downstream labels is/are recovered. As the removal request load increases, the chance that both its upstream and downstream labels are already recovered becomes smaller. However, if the recovery with both upstream and downstream neighbors starts at approximately the same time, the connection can be removed immediately.

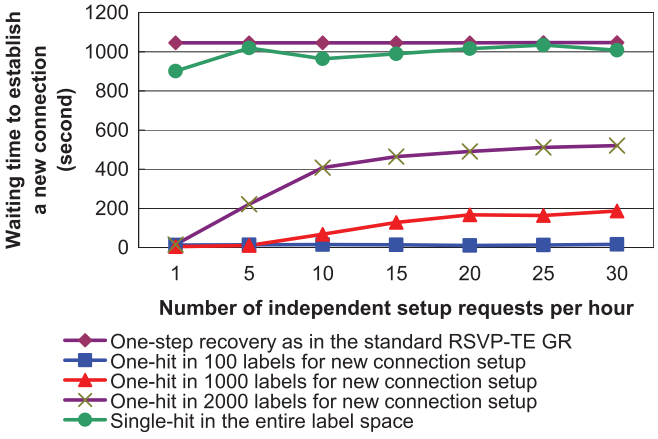

Fig. 4. Waiting time to establish a new connection.

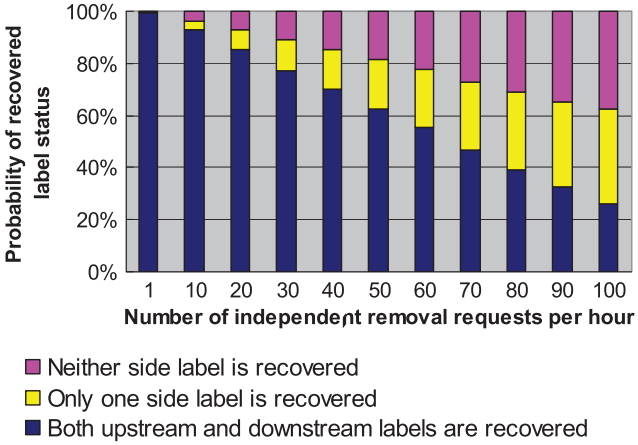

Fig. 5. Recovery status for the upstream and downstream labels.

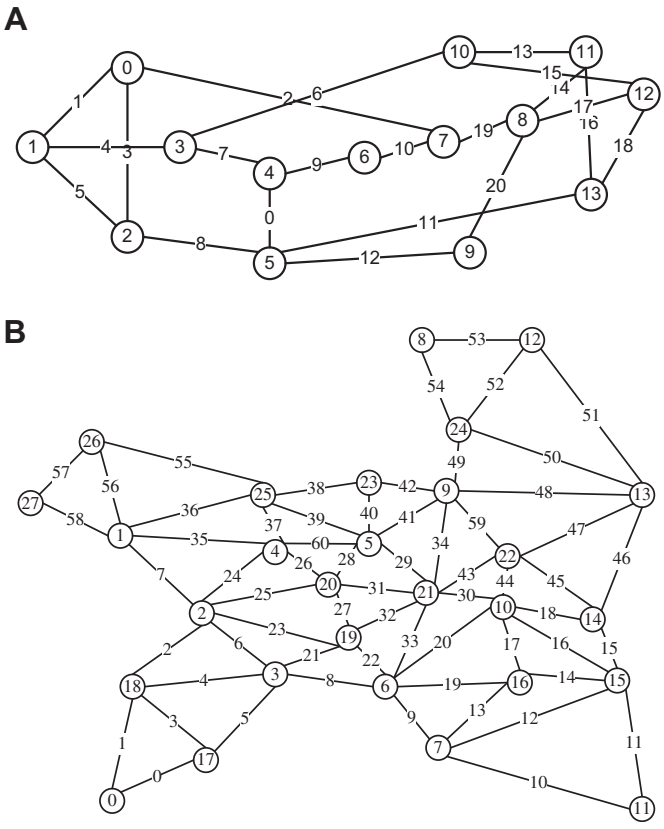

Fig. 6. Example networks, (a) NSFNET with 14 nodes and 21 links; (b) Pan-European network with 28 nodes and 61 links.

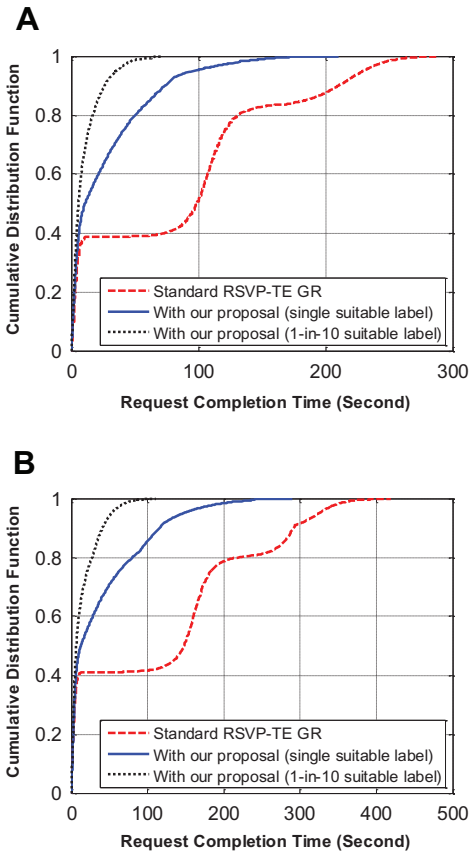

Fig. 7. Cumulative distribution function of the request completion time (no message loss), (a) NSFNET; (b) Pan-European network.

We extend our performance evaluations to mesh topology networks. We use two example networks (shown in Fig. 6): NSFNET and the Pan-European network. We assume each link has a pair of fibers, one for each direction, having 100 channels, and there is only one RSVP-TE session for each link. We also assume that RSVP-TE sessions only run between physically adjacent neighbors, although RSVP-TE could be applied to logical TE links between non-adjacent nodes. We create a baseline scenario of network usage by randomly loading a certain number of unidirectional connections into the networks. Routes do not necessarily follow the shortest paths, to emulate constraint-based routing and routing policy control, and the labels that connections use on each link are randomly assigned. The baseline scenario for NSFNET has 1500 connections loaded, while the baseline scenario for the Pan-European network has 3200 connections loaded. The network utilization is about 70–75% in both examples. From the baseline scenario, we introduce a single control plane failure at a random place, either a control channel or control node. Although the introduced control plane failure is repaired quickly, the associated RSVP-TE session is affected and starts recovery. During RSVP-TE state recovery, we generate connection setup requests, whose source and destination nodes, as well as routing, are random. Every time a request arrives at the network in the baseline scenario, it is used as a "probing" method to check the network reactions. The time from the start of RSVP-TE recovery to request arrivals is random, following a distribution of request arrival statistics.

The time for requests to be completed was measured. Requests rejected or routed over paths unaffected by the failure were not included. In Fig. 7, we present the cumulative distribution function of the request completion time for a certain number of requests successfully routed over affected RSVP-TE control nodes (1000 requests for NSFNET and 2000 requests for the Pan-European network). We made an assumption that there is no message loss. To emulate constraint-based routing, we assume that one suitable label is found in 10 idle labels for new connection setups on average. We also compare with the case in which only one single label in the entire label space is suitable for the new connection setup. Without our proposal (i.e., standard RSVP-TE GR), the new connection setup must wait until all in-use labels are recovered. Depending on the failure type and location, the standard RSVP-TE GR may need a significant amount of time for completion, introducing a delay for new connection setups. Our proposal significantly reduces the request completion time. In NSFNET, our proposal is able to complete approximately 71% connection setups (1-in-10 suitable label) within 12 s instead of 39% within 12 s using the standard RSVP-TE GR. In the Pan-European network, our proposal is able to complete approximately 65% connection setups (1-in-10 suitable label) within 13 s instead of 41% within 13 s using the standard RSVP-TE GR.

The control overhead of the proposed enhancement mainly results from additional idle label announcements in Hello messages. We introduced policy rules to avoid overloading a restarting node, which were described in Section 5.1. The additional control overhead was considered in our simulations by using a longer processing time for Hello messages with idle label announcements. (described in Table 3).

6. Conclusions

With RSVP-TE GR, a restarting RSVP session may recover the state before a control plane failure. Such a capability enables the service continuation of the established data plane connections.

With our proposed two-step recovery, the requirement of local recovery of the data plane state is alleviated. Our approach allows the restarting control node to learn the state of the related data plane from its neighbor control nodes. With our proposal, we shorten the length of time that service is affected when the data plane state cannot be recovered or retrieved in the affected control node. Our performance evaluations show significant time savings for a new connection establishment request compared with full state recovery in the standard RSVP-TE GR.

Our proposed RSVP-TE recovery complements a wide range of fault-tolerance and system failure recovery techniques, although different techniques may need to be engineered to achieve the best performance for specific applications. In particular, because our proposed RSVP-TE recovery is only able to recover from a single-point failure, the network management system would be required to handle more complicated scenarios. The initial target of RSVP-TE GR is for scheduled maintenance, as is the recovery technique presented in this proposal. We expect that the concept may be applied to unplanned outages once reliability has been demonstrated.

Acknowledgement

We thank Dr. Claire Callender from the Communications Research Centre (CRC) Canada for her editorial revisions of this paper.

Appendix A. State information related to RSVP-TE sessions and path and resv messages

The purpose of this section is to provide a complete view of the state related to RSVP-TE sessions and *Path* and *Resv* messages. It is derived from a centralized virtual information store, known as the Management Information Base (MIB). The RSVP-TE signaling protocol must maintain the same state but in a distributed manner, i.e., each node maintains the state related to itself. For example, the state related to an RSVP-TE session (shown in Table 5) is maintained at two end nodes of the session. The state related to a *Path* message (shown in Table 6) is maintained at the sending and receiving nodes, as is the state related to a *Resv* message (shown in Table 7). Although MIB objects for RSVP are fully developed

Table 5
State information related to an RSVP-TE session.

MIB Objects	Descriptions
rsvpSessionNumber	The number of this session (for indexing purposes only)
rsvpSessionType	The type of session (IP v4, IP v6, IP v6 with flow information, etc.)
rsvpSessionDestAddr	The destination address used by all senders in this session
rsvpSessionDestAddrLength	The prefix length of the session destination IP address
rsvpSessionProtocol	The IP protocol used by this session
rsvpSessionPort	The UDP or TCP port number used as a destination port for all senders in this session
rsvpSessionSenders	The number of distinct senders currently known to be part of this session
rsvpSessionReceivers	The number of reservations being requested of this system for this session
rsvpSessionRequests	The number of reservation requests this system is sending upstream for this session

Table 6
State information related to *Path* messages.

MIB Objects	Descriptions
rsvpSenderNumber	The number of this sender (for indexing purposes only)
rsvpSenderType	The type of session (IP v4, IP v6, IP v6 with flow information, etc.)
rsvpSenderDestAddr	The destination address used by all senders in this session
rsvpSenderAddr	The source address used by this sender in this session
rsvpSenderDestAddrLength	The length of the destination address in bits
rsvpSenderAddrLength	The length of the sender's address in bits
rsvpSenderProtocol	The IP protocol used by this session
rsvpSenderDestPort	The UDP or TCP port number used as a destination port for all senders in this session
rsvpSenderPort	The UDP or TCP port number used as a source port for this sender in this session
rsvpSenderFlowId	The flow ID that this sender is using, if this is an IPv6 session
rsvpSenderHopAddr	The address used by the previous RSVP hop (which may be the original sender)
rsvpSenderHopLih	The logical interface handle used by the previous RSVP hop (which may be the original sender)
rsvpSenderInterface	The index that points to the interface on which this *Path* message was most recently received
rsvpSenderInterval	The interval between refresh messages as advertised by the previous hop
rsvpSenderRSVPHop	If TRUE, the node believes that the previous IP hop is an RSVP hop; if FALSE, the node believes that the previous IP hop may not be an RSVP hop
rsvpSenderLastChange	The time of the last change in this *Path* message; this is either the first time it was received or the time of the most recent change in parameters
rsvpSenderPolicy	The contents of the policy object
rsvpSenderStatus	Active or not. This object may be used to install or delete path information.
Other objects related to TSPEC	The TSPEC specifies parameters available for the flow from a bit rate point of view, such as average and peak bit rates, etc.
Other objects related to ADSPEC	The ADSPEC is an optional object descriptor that the sender may include in its generated *Path* messages to advertise to receivers the characteristics of the end-to-end communications path. The ADSPEC enables network elements in the path between sender and receiver to advertise their service capabilities, resource availability, and transmission characteristics.
Additional objects related to RSVP-TE and specific to GMPLS-controlled optical networks	In addition to the previous objects defined for RSVP, objects related to RSVP-TE and GMPLS controlled optical networks are necessary. For example, explicit route, recorded route, label set, suggested label, required link protection, etc.

Table 7
State information related to *Resv* messages.

MIB Objects	Descriptions
rsvpResvNumber	The number of this reservation request (for indexing purposes only)
rsvpResvType	The type of session (IP v4, IP v6, IP v6 with flow information, etc.)
rsvpResvDestAddr	The destination address used by all senders in this session
rsvpResvSenderAddr	The source address of the sender selected by this reservation
rsvpResvDestAddrLength	The length of the destination address in bits
rsvpResvSenderAddrLength	The length of the sender's address in bits
rsvpResvProtocol	The IP protocol used by this session
rsvpResvDestPort	The UDP or TCP port number used as a destination port for all senders in this session
rsvpResvPort	The UDP or TCP port number used as a source port for this sender in this session
rsvpResvHopAddr	The address used by the next RSVP hop (which may be the ultimate receiver)
rsvpResvHopLih	The logical interface handle received from the previous RSVP hop (which may be the ultimate receiver)
rsvpResvInterface	The index that points to the interface on which this *Resv* message was most recently received
rsvpResvService	The QoS Service classification requested by the receiver
rsvpResvInterval	The interval between refresh messages as advertised by the next hop
rsvpResvScope	The contents of the scope object
rsvpResvShared	If TRUE, a reservation shared among senders is requested; if FALSE, a reservation specific to this sender is requested
rsvpResvExplicit	If TRUE, individual senders are listed using Filter Specifications; if FALSE, all senders are implicitly selected. The Scope Object will contain a list of senders that must receive this reservation request for the purpose of routing the *Resv* message
rsvpResvRSVPHop	If TRUE, the node treats the previous IP hop as an RSVP hop; if FALSE, the node treats the previous IP hop as a non-RSVP hop
rsvpResvLastChange	The time of the last change in this reservation request; this is either the first time it was received or the time of the most recent change in parameters
rsvpResvPolicy	The contents of the policy object
rsvpResvStatus	Active or not. This object may be used to install or delete reservation information
rsvpResvTTL	The time-to-live value in the RSVP header last received
rsvpResvFlowId	The flow ID that this receiver is using if this is an IPv6 session
Other objects related to TSPEC	The TSPEC specifies parameters available for the flow from a bit rate point of view, such as average and peak bit rate, etc.
Other objects related to RSPEC	The RSPEC specifies parameters for a reservation, such as guaranteed bit rate or delay
Additional objects related to RSVP-TE and specific to GMPLS-controlled optical networks	In addition to the previous objects defined for RSVP, objects related to RSVP-TE and GMPLS-controlled optical networks are necessary

[49], the complete MIB objects for RSVP-TE and GMPLS-controlled optical networks are still under development. The following information is mainly for RSVP. Modifications and extensions are necessary for RSVP-TE and GMPLS-controlled optical networks.

The scope and framework are defined to store the state information related to the received *Path* messages [49]. The same scope and framework apply to the *Path* messages sent out, where all instances of "sender" in the table should be replaced by "receiver" in Table 6.

References

[1] G. Li, J. Yates, D. Wang, C. Kalmanek, Control plane design for reliable optical networks, IEEE Communications Magazine 40 (2) (2002) 90–96.
[2] I. Hussain, Fault-Tolerant IP and MPLS Networks, Cisco Press, Indianapolis, IN, USA, 2005.
[3] A. Banerjee, J. Drake, J.P. Lang, B. Turner, K. Kompella, Y. Rekhter, Generalized multiprotocol label switching: an overview of routing and management enhancements, IEEE Communications Magazine 39 (1) (2001) 144–150.
[4] J. Lang (Ed.), Link Management Protocol (LMP), IETF RFC 4204, October 2005.
[5] S. Tomic, B. Statovci-Halimi, A. Halimi, W. Muellner, J. Fruehwirth, ASON and GMPLS – Overview and Comparison, Photonic Network Communications 7 (2) (2004) 111–130.
[6] J. Sadler (Ed.), Control Plane Initial Establishment, Reconfiguration and Recovery, ITU-T Recommendation G.7716.
[7] A. Jajszczyk, P. Rozycki, Recovery of the control plane after failures in ASON/GMPLS networks, IEEE Network 20 (1) (2006) 4–7.
[8] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: Extensions to RSVP for LSP tunnels, in: IETF RFC 3209, December 2001.
[9] L. Berger, Generalized multi-protocol label switching (GMPLS) signaling functional description, in: IETF RFC 3471, January 2003.
[10] L. Berger, Generalized multi-protocol label switching (GMPLS) signaling resource reservation protocol-traffic engineering (RSVP-TE) extensions, in: IETF RFC 3473, January 2003.
[11] L. Andersson, G. Swallow, The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols, in: IETF RFC 3468, February 2003.
[12] A. Satyanarayana, R. Rahman (Ed.), Extensions to GMPLS Resource Reservation Protocol (RSVP) Graceful Restart, in: IETF RFC 5063, October 2007.
[13] C. Srinivasan, A. Viswanathan, T. Nadeau, Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base (MIB), in: IETF RFC 3813, June 2004.
[14] H.T. Mouftah, P.H. Ho, Optical Networks: Architecture and Survivability, Kluwer Boston, Inc., 2002.
[15] A. Farrel, Standardising for Control Plane Resilience, Workshop on GMPLS Performance Evaluation: Control Plane Resilience, June 2007, Glasgow, UK. Available at <www.olddog.co.uk/Farrel-ICC2007-CPResilience.ppt>.
[16] A. Banerjee, J. Drake, J.P. Lang, B. Turner, K. Kompella, Y. Rekhter, Generalized multiprotocol label switching: an overview of routing and management enhancements, IEEE Communications Magazine 39 (1) (2001) 144–150.
[17] K. Kompella, Y. Rekhter (Ed.), Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS), in: IETF RFC 4202, October 2005.
[18] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, RSVP: a new resource reservation protocol, IEEE Communications Magazine 40 (5, Anniversary Part) (2002) 116–127.
[19] L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP): Version 1 Functional Specification, in: R. Braden (Ed.), IETF RFC 2205, September 1997.
[20] IETF working group Common Control and Measurement Plane (CCAMP). Available at <www.ietf.org/html.charters/ccamp-charter.html>.
[21] IETF working group Multiprotocol Label Switching (MPLS). Available at <www.ietf.org/html.charters/mpls-charter.html>.
[22] E. Mannie, D. Papadimitriou, Generalized Multi-Protocol Label Switching (GMPLS) Extensions for Synchronous Optical Network (SONET) and Synchronous Digital Hierarchy (SDH) Control, in: IETF RFC 4606, August 2006.
[23] D. Papadimitriou, (Ed.), Generalized Multi-Protocol Label Switching (GMPLS) Signalling Extensions for G.709 Optical Transport Networks Control, in: IETF RFC 4328, January 2006.
[24] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, RSVP Refresh Overhead Reduction Extensions, in: IETF RFC 2961, April 2001.
[25] L. Berger, (Ed.), GMPLS - Communication of Alarm Information, in: IETF RFC 4783, December 2006.
[26] F. Le Faucheur, B. Davie, P. Bose, C. Christou, M. Davenport, Generic Aggregate Resource ReSerVation Protocol (RSVP) Reservations, in: IETF RFC 4860, May 2007.
[27] F. Le Faucheur, (Ed.), L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, J. Heinanen, Multi-Protocol Label Switching (MPLS) Support of Differentiated Services, in: IETF RFC 3270, May 2002.
[28] C.Y. Lee, A. Farrel, S. De Cnodder, Exclude Routes – Extension to Resource ReserVation Protocol-Traffic Engineering (RSVP-TE), in: IETF RFC 4874, April 2007.

[29] P. Pan, G. Swallow, A. Atlas, (Ed.), Fast Reroute Extensions to RSVP-TE for LSP Tunnels, in: IETF RFC 4090, May 2005.
[30] A. Farrel, (Ed.), D. Papadimitriou, J.P. Vasseur, A. Ayyangar, Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE), in: IETF RFC 5420, February 2009.
[31] L. Berger, I. Bryskin, D. Papadimitriou, A. Farrel, GMPLS Segment Recovery, in: IETF RFC 4873, May 2007.
[32] J.P. Lang, Y. Rekhter, D. Papadimitriou, (Ed.), RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery, in: IETF RFC 4872, May 2007.
[33] R. Braden, L. Zhang, Resource ReSerVation Protocol (RSVP): Version 1 Message Processing Rules, in: IETF RFC 2209, September 1997.
[34] M. Karsten, Design and Implementation of RSVP Based on Object-Relationships, Broadband Communications, High Performance Networking, and Performance of Communication Networks, in: 2000 Proceedings of the IFIP-TC6/European Commission International Conference on Networking, Paris, France, Lecture Notes in Computer Science (LNCS), Springer, vol. 1815, 2000, pp. 325–337.
[35] M. Karsten, Design and Implementation of RSVP based on Object-Relationships, Technical University of Darmstadt, Department of Electrical Engineering and Information Technology, Industrial Process and System Communications (KOM), Technical Report TR-KOM-2000-01, May 2000.
[36] M. Karsten, J. Schmitt, R. Steinmetz, Implementation and Evaluation of the KOM RSVP Engine, in: Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), vol. 3, April 2001, pp. 1290–1299.
[37] O. Komolafe, J. Sventek, Overview of Enhancements to RSVP-TE to Increase Control Plane Resilience, in: Workshop on GMPLS Performance Evaluation: Control Plane Resilience, Glasgow, UK, June 2007.
[38] O. Komolafe, J. Sventek, Analysis of RSVP-TE graceful restart, in: Proceedings of the 2007 IEEE International Conference on Communications (ICC 2007), Glasgow, UK, June 2007, pp. 2324–2329.
[39] O. Komolafe, J. Sventek, Evaluating and improving the performance of RSVP-TE graceful restart, in: Proceedings of the 5th International Conference on Broadband Communications, Networks and Systems, (BROADNETS 2008), London, UK, September 2008, pp. 665–672.
[40] Z. Zhou, K. Chen, L. Zheng, GMPLS RSVP-TE signaling recovery with graceful restart in optical user network interface, in: Proceedings of the Optical Fiber Communication and the National Fiber Optic Engineers Conference 2007 (OFC/NFOEC 2007), Anaheim, California, USA, March 2007.
[41] O. Komolafe, J. Sventek, Impact of GMPLS control message loss, IEEE/OSA Journal of Lightwave Technology 26 (14) (2008) 2029–2036.
[42] D. Morató, J. Aracil, J.P. Fernández-Palacios, Ó.G. Dios, J.L. Poyo, On capacity planning for the GMPLS network control plane, Photonic Network Communications 15 (2) (2008) 159–169.
[43] J. Li, F. Gao, J. Gao, S. Shi, Analysis of bandwidth requirements for ASON/ASTN signaling networks, in: S.J. Ben Yoo, G.K. Chang, G. Li, K.W. Cheung (Eds.), Network Architectures, Management, and Applications II, Proceedings of SPIE 5626, February 2005, pp. 690–694.
[44] A. Neogi, T. Chiueh, P. Stirpe, Performance analysis of an RSVP capable router, IEEE Network 13 (5) (1999) 56–63.
[45] X. Zhu, X. Zheng, M. Veeraraghavan, Experiences in implementing an experimental wide-area GMPLS network, IEEE Journal on Selected Areas in Communications 25 (3, Supplement Part) (2007) 82–92.
[46] J. Wu, M. Savoie, GMPLS control plane failure recovery, in: W. Hu, S.K. Liu, K. Sato, L. Wosinska (Ed.), Network Architectures, Management, and Applications VI, Proceedings of the SPIE 7137, November 2008.
[47] J. Wu, M. Savoie, Distributed Failure Recovery of the LDP Signalling Protocol, in: Workshop on GMPLS Performance Evaluation: Control Plane Resilience, Glasgow, UK, June 2007.
[48] J. Wu, M. Savoie, H. Mouftah, Recovery from control plane failures in the LDP signalling protocol, Optical Switching and Networking 2 (3) (2005) 148–162.
[49] F. Baker, J. Krawczyk, A. Sastry, RSVP Management Information Base using SMIv2, in: IETF RFC 2206, September 1997.