A Survey on the Evolution of RSVP

Flavius Pana and Ferdi Put

Abstract—The Resource Reservation Protocol (RSVP) represents one of the most important protocols used for resource reservation in the Internet. Developed initially by the Internet Engineering Task Force (IETF) to be used within the Integrated Services (IntServ) mechanism, this protocol undergoes over time several alterations. These alterations come either to respond to some applicability and functionality problems, or to extend the use of RSVP and to make it compatible with other mechanisms like Differentiated Services (DiffServ) or Multiprotocol Label Switching (MPLS). This work presents a survey on the evolution of RSVP illustrating the different alterations introduced over time for this protocol and explaining how each adaptation affects RSVP in functional terms.

Index Terms—RSVP, IntServ, DiffServ, RSVP-TE, survey.

I. INTRODUCTION

OVER the years several attempts have been made to define an efficient reservation protocol or a generic signalling method for the Internet. Some examples of these approaches are the Internet Stream Protocol 2 (ST2) [1], the Resource Reservation Protocol (RSVP) [2], the Yet another Sender Session Internet Protocol (YESSIR) [3], or Boomerang [4]. A survey of Internet signalling mechanisms can be found in [5]. Among all of these proposals RSVP captured the most attention from the research community, making it one of the most persistent and most altered protocols.

Historically, RSVP was the successor of ST2. Both protocols were intended to support multicast communications. This being one of the key requirements for a resource reservation protocol, since it was considered that demand for multicast-capable real-time teleconferencing was going to grow dramatically in the Internet [6]. ST2 was developed for a sender initiated point-to-multipoint communication. The problem with the ST2 protocol was that since it is sender initiated it does not scale with the number of receivers in a multicast group [6]. This in turn triggered the development of RSVP, which was developed from the beginning as a receiver-based resource reservation protocol which can support multipoint-to-multipoint reservation setup. An architectural comparison of ST2 and RSVP can be found in [7].

RSVP is standardized by the IETF in [2] and it was designed to work under the IntServ mechanism. Introduced in order to guarantee a bounded delay to intolerant applications, RSVP represents a means through which resources can be reserved for one traffic flow in all the nodes from the sender host to the receiver host. Among some of the most important characteristics of RSVP we can enumerate: the receiver based

resource reservation, the soft state approach in maintaining reservations, and the use of an enhancement of a one pass reservation model called One Pass with Advertising (OPWA).

In turn RSVP did not escape criticism and was accused of being overly complex while suffering from a severe scalability problem at the same time. However, the RSVP design is not abandoned, and over the years different extensions which rely on the original RSVP solution have been proposed. Some of these extensions provide features that were not available in the original design, while others are concentrating on alienating the scalability problem.

One of the most adopted implementations that is based on the original RSVP design is the RSVP extension for traffic engineering (RSVP-TE). This extension was widely accepted for the Multiprotocol Label Switching (MPLS) capable networks.

RSVP has proven to be extremely important in the Quality of Service (QoS) field. Extensions have been created that allow the coupling of RSVP with each of the three QoS mechanisms developed by IETF (IntServ, DiffServ and MPLS). An overview of the Internet QoS and the role of RSVP in this field is presented in [8]–[11].

In this paper we will try to inventory extensions of RSVP introduced by IETF. The main goal of this paper is to present in a systematic way the functionality introduced by the most important standards that defined or altered RSVP across time. The paper is tutorial in nature and presents a comprehensive study on the evolution of RSVP.

The rest of the paper is organized as follows. In Section II we present the basic RSVP design and the layout of the underlying components. In Section III we illustrate the use of RSVP under IntServ. In Section IV are presented different extensions of the RSVP design like the cryptographic authentication or the extension for policy control. In Section V we present different proposals introduced to reduce processing overhead: the refresh reduction, the RSVP reservation aggregation and the generic RSVP aggregation. Section VI describes the core extensions of RSVP for MPLS and Generalized MPLS (GMPLS). In Section VII we present the proposed extensions of RSVP-TE. A discussion of other research problems related with RSVP that are not tackled directly by IETF is presented in Section VIII. And finally in Section IX we conclude our work.

II. ORIGINAL RSVP DESIGN

A. General description of RSVP

Defined in RFC 2205, the Resource Reservation Protocol was designed for an IntServ Internet [2]. RSVP is positioned at the transport layer in the OSI protocol stack, being able to operate on top of IPv4 or IPv6. RSVP does not transport

Manuscript received May 25, 2012; revised November 10, 2012.

F. Pana and F. Ferdi are with the Research Centre for Management Informatics, Katholieke Universiteit Leuven, Leuven, Belgium (e-mail: {flavius.pana; ferdi.put}@econ.kuleuven.be).

Digital Object Identifier 10.1109/SURV.2013.021313.00107

any data itself, operating in this sense as an Internet control protocol (ICMP or IGMP) or as a routing protocol. However, RSVP itself is not a routing protocol but is designed to work with all the types of routing protocols.

The RSVP reservation model is an enhancement of a One Pass reservation model called One Pass with Advertising (OPWA) [12]. In a One Pass reservation model a receiver sends a reservation request upstream and each node in the path either accepts or rejects the request [2]. A detailed description of the OPWA approach and a discussion of the features of One Pass and Two Pass mechanisms can be found in [13]. In the RSVP case, control packets are sent from the sender to the receiver hosts on the exact path used by the data flow. The packets collect information about the network, which afterwards is used to predict the end-to-end QoS. These advertisements can be used by the receiver to construct or adjust appropriate reservation requests.

RSVP is a receiver-oriented reservation protocol, meaning that the receivers are the ones which originate a reservation request. After the request is generated, this is forwarded upstream towards the sender. The process in charge of the reservation passes the request to admission control and policy control decisions modules. Here, the decision is based on the availability of the resources, in the case of admission control and on the credentials and rights of the user in the case of policy control. If the reservation is rejected an error message is sent back to the receiver. However, if the reservation is granted, the reservation is propagated upstream to the appropriate senders. An important note here is that the reservation propagated upstream can differ from the one received, in the sense that the downstream branches of a multicast tree originating from the same sender must be merged as the reservation is propagated towards the sender.

RSVP operates on a per session level, providing uni-directional reservations and treating each session separately. A session represents a data flow with a particular destination and transport layer protocol. Sessions are identified by the IP destination address, the IP protocol ID and an optional parameter which represents the TCP/UDP destination port.

A basic RSVP reservation, as this is defined in RFC 2205 consists of a flow descriptor which is composed of a FLOWSPEC and a FILTER_SPEC. The FLOWSPEC specifies the desired QoS which is used by the nodes to set the packet scheduler parameters. The FILTER_SPEC is used together with the session specification to define the set of data packets (or the flow) that will receive the QoS defined by the FLOWSPEC, specifying the necessary parameters to the packet classifier of the node.

The FLOWSPEC includes two sets of parameters: an RSpec which is used in defining the desired QoS and a TSpec used to describe the data flow. The content and format of these parameters will be detailed in the next section when we will present the use of RSVP with Integrated Services. The mentioned parameters are not a direct concern for RSVP since the protocol operates in a way that is opaque to these parameters.

Three types of reservation styles are available to be used with RSVP. The first type called Fixed Filter (FF) is used when the reservation is distinct and the sender selection is

explicit. This means that a distinct reservation is created for data packets from a particular sender. The reservation, in this case is not shared with other senders. In contrast, the second and third type, namely Shared Explicit (SE) and respectively Wildcard Filter (WF), create shared reservations. The difference between the two shared reservation models is that in the SE case the receiver is allowed to explicitly specify the set of senders to be included. On the other hand when the WF style is used the reservation is automatically extended to new senders as they appear. The shared reservations WF and SE are appropriate for multicast applications where multiple sources are unlikely to transmit simultaneously.

So as to deliver information from one node to another, RSVP uses different types of messages, depending on the information that has to be delivered and on the actions that each message triggers. These messages are composed of different standardized entities named objects. In general each object carries specific standardized information, like the IP address of the destination node carried in the SESSION object.

Two fundamental RSVP messages are defined, the Resv message, a reservation request that travels from the receiver to the sender(s), and a Path message that travels downstream from the sender host along the uni/multi-cast routes towards the receiver.

The Path message follows the route provided by the routing protocol, using the same path as the data flow. Each Path message stores a path state in all the intermediate nodes along the way. The path state includes information retrieved from the Path message itself, or from other processes specific to that node, as we will see later.

The Resv message is sent exactly on the reverse path used by the Path message. If the Path message is sent using the same source and destination address as the data, the Resv message is sent hop-by-hop using the path state information stored in the intermediate nodes.

RSVP uses a soft state approach in order to manage the reservation state in the hosts and intermediate nodes. Whereas the path state represents the state which is created when receiving a Path message, accordingly the reservation state is the state created when receiving a Resv message. The Path and Resv messages are used to create but also to refresh the reservation and path state in the nodes. In this way if a route changes, because of routing updates or node failure, the new Path message will create a new Path state in the nodes of the new route and the Resv message will establish reservation state in these nodes for the new route. The old reservation state will be automatically deleted if no matching refresh message will arrive before a cleanup timeout interval expires.

This type of soft state approach makes RSVP reservations dynamic, where in order to change the QoS parameters or to modify the set of senders, it is sufficient to transmit updated Resv/Path messages. Reservations can be removed explicitly by end hosts using teardown messages. Two types of teardown messages are defined: the PathTear and ResvTear. The PathTear message travels downstream towards the receiver(s) and deletes path states and depended reservation states in all the intermediate nodes for that reservation. The ResvTear can be seen as the reverse sense Path message and is in charge of deleting reservation states as it travels towards the sender(s).

Accordingly to the way in which the main Resv and Path messages travel in the RSVP reservation process, two error messages are defined: ResvErr and PathErr. The PathErr simply travels upstream towards the sender that created the error and does not change any Path states along the way. Only a few possible causes are defined which can trigger the sending of a PathErr message, as we will later see. The ResvErr message on the other hand has a much more complex behavior. Because of the merging capabilities of the RSVP reservation model, a failed request can be caused by the merger of a number of different requests, this in turn meaning that the reservation error must be reported to all the involved receivers.

The merging of reservations in RSVP can create even more complicated problems. These problems are called killer reservations, where one request could cause a denial of service to another request.

Two killer reservation problems were identified and presented in [2]. The first one might happen when a reservation is already in place (Q0) and a new receiver wants to make an additional reservation (Q1). The merging of these reservations could be rejected by the admission control procedures in some nodes, due to a lack of resources. This rejection should not affect the already existing reservation Q0. The solution is that whenever a reservation request is rejected by admission control any existing reservation should be left in place.

The second killer reservation problem appears when a receiver who wants to make a reservation (Q1) is persistent even if admission control is rejecting Q1 in some nodes. This type of behavior should not prevent another receiver who wants to make a reservation (Q2) that might succeed if not merged with Q1. With the aim of addressing this problem the ResvErr message creates an additional state named blockade state. This state modifies the merging procedures so as to omit the offending Q1 reservation FLOWSPEC from the merge, while still allowing smaller reservations to be created.

B. RSVP messages and objects

In terms of RSVP functional specifications, RSVP is built in a very flexible way. As we have already seen, RSVP consists of different message types which are in charge of providing information to the end nodes and intermediate nodes.

An RSVP message consists of a common header, where the version of the protocol and the message type are specified. The body of the message is composed of a variable number of variable length items, called objects. Each of the defined message types further has specifications on what type of objects should be included in the body of the message and in what order. However, as we will see, other objects and even message types have been developed over the years for RSVP. Each extension of an object or a message was meant to enhance the functionality of the RSVP protocol or to expand the ability of RSVP. The common header and the general object format can be seen in Fig. 1.

In total seven types of messages were defined in [2]. These are Path, Resv, PathErr, ResvErr, PathTear, ResvTear and ResvConf. Also, 15 different classes of objects were introduced in the same RFC. Every object consists of a multiple of 32 bit words, starting again with a common header

Fig. 1. The common RSVP header and the general object format.

which specifies the length in bytes of the object, a Class-Num field and a C-Type field as shown in Fig.1.

The Class-Num value is used to uniquely identify a class of objects. In total 15 Class-Num values were specified in RFC 2205. Within a class the C-Type field identifies a precise object type. These types and their values are unique within a Class-Num. The 15 object classes defined in RFC 2205 are presented in Table I, illustrating also the information that is included in each object. From these 15, five objects classes were not described in this RFC, but were defined later in other RFCs.

In order to ensure compatibility with future defined object classes for RSVP, the Class-Num values are divided in three groups, depending on what is desired from the RSVP implementation regarding an object that belongs to an unknown class. A Class-Num of the form [0bbb bbbb] will cause the node to reject the entire message and to return an Unknown Object Class error. A Class-Num of the form [10bb bbbb] will trigger the node to ignore the object without forwarding it or sending an error message. And a Class-Num that has the form [11bb bbbb] will cause the node to ignore the object, but to forward it unexamined and unmodified.

In Fig. 2 we illustrate the components of the Path and Resv messages using the Bachus-Naur Form (BNF).

The Path message originates at each sender of a data flow and travels towards the receiver using the same path as the data packets. The Path message uses the IP source address of the sender and the IP destination address of the session. This ensures that the messages will be correctly routed by non-RSVP nodes or domains. A non-RSVP node is a node that does not understand RSVP messages and is unable to follow procedures defined in RFC [2]. The message includes the SENDER_TEMPLATE object to identify the IP address and source port of the sender, and the SENDER_TSPEC object to define the traffic characteristics of the sender's data flow. Optionally an ADSPEC object can be included, which is used for the advertising information (OPWA) of the data flow, as we will see later.

All the RSVP capable nodes along the way capture the Path message(s) and create a path state for the sender. The path state is identified by the tuple composed of SESSION and SENDER_TEMPLATE objects. If a SENDER_TEMPLATE object is used to identify the sender with an IP address and source port, the SESSION object indicates the IP destination address, the IP protocol number and the destination port. Also, the POLICY_DATA, SENDER_TSPEC, ADSPEC and RSVP_HOP are saved in the path state.

TABLE I
OBJECTS DEFINED BY RFC 2205

Object Name	Class-Num	Information carried in the object
SESSION	1	IP unicast destination address
		IP protocol identifier
		Flags (E_Police flag)
		UDP/TCP destination port
"Not defined"	2	Not defined/reserved
RSVP_HOP	3	IP of the last RSVP node
		Logical interface handle (LIH)
INTEGRITY	4	N/A- defined later in RFC 2747
TIME_VALUES	5	Refresh timeout period R
ERROR_SPEC	6	IP of the node in which the error was detected
		Error code
		Error value
		Flags (InPace and NotGuilty)
SCOPE	7	IP addresses used for routing messages with wildcard scope
STYLE	8	Flags
		Option vector which identifies the style of the reservation
FLOWSPEC	9	N/A- defined later in RFC 2210
FILTER_SPEC	10	IP source address
		TCP/UDP source port
SENDER_TEMPLATE	11	IP source address
		TCP/UDP source port
SENDER_TSPEC	12	N/A- defined later in RFC 2210
ADSPEC	13	N/A- defined later in RFC 2210
POLICY_DATA	14	N/A- defined later in RFC 2750
RESV_CONFIRM	15	IP receiver address

The RSVP_HOP is used to transmit the IP address of the previous IP capable node together with the logical outgoing interface handle (LIH). As we will see later, this address will be used by the Resv message to travel hop-by-hop on the reverse path.

The INTEGRITY object is used to carry cryptographic data in order to authenticate the originating node and to provide end-to-end integrity. However, this object and the associated procedures were standardized only later in [14]. Also, the POLICY_DATA object was standardized afterwards by RFC 2750 [15]. This object is used for generic policy based admission control.

The Path messages are forwarded (and replicated in case of multicast) by each intermediate node using routing information received from the appropriate routing process. Once the Path message arrives at the receiver node, this sends back a Resv message which uses the same path as the Path message in the reverse direction, towards the sender of the message. The components of the Resv message are illustrated in Fig. 2.

The RSVP_HOP object contains in this case the IP address of the node which sent the Resv message. The Resv message travels hop-by-hop from one RSVP capable node to another, using the information which is stored in the path state on each intermediate router so as to determine the next hop.

Path message format

```
<Path Message>::=<Common Header>[<INTEGRITY>]<SESSION>
<RSVP_HOP><TIMES_VALUES>[<POLICY_DATA>]
[<sender description>];
```

Where,

```
<Sender description>::=<SENDER_TEMPLATE><SENDER_TSPEC>
[<ADSPEC>].
```

Resv message format

```
<Resv Message>::=<Common Header>[<INTEGRITY>]<SESSION>
<RSVP_HOP><TIMES_VALUES>[<RESV_CONFIRM>]
[<SCOPE>][<POLICY_DATA>]<STYLE>
<flow descriptor list>;
```

Where,

```
<flow descriptor list>::=<empty>|<flow descriptor list>
< flow descriptor>.
```

Fig. 2. Path and Resv message format.

For the FF style:

```
<flow descriptor list>::=<FLOWSPEC><FILTER_SPEC>I
<flow descriptor list><FF flow descriptor>;
```

Where,

```
<FF flow descriptor>::=[<FLOWSPEC>]<FILTER_SPEC>.
```

For the SE style :

```
<flow descriptor list>::=<SE flow descriptor>;
<SE flow descriptor>::=<FLOWSPEC><filter spec list>;
<filter spec list>::=<FILTER_SPEC><filter spec list><FILTER_SPEC>;
```

For the WF style :

```
<flow descriptor list>::=<WF flow descriptor>;
<WF flow descriptor>::=<FLOWSPEC>.
```

Fig. 3. The flow descriptor list format.

The flow descriptor list present in the Resv message is style dependent, meaning that for each of the three styles, a different format of the flow descriptor list is expected. The indication of which style is in use is contained in the STYLE object. In Fig. 3 we present the format of the flow descriptor list for each of the three styles.

Each reservation which uses the FF style is defined by the FLOWSPEC and FILTER_SPEC object pairs. Multiple requests may be packed in a single flow descriptor list, where the FLOWSPEC object that appears in the FF flow descriptor can be omitted if this is identical to the first FLOWSPEC object used.

In the SE style the sender selection is based on matching the FILTER_SPEC object with the SENDER_TEMPLATE object from the existing path state stored in the intermediate nodes. The PathTear messages are triggered explicitly by senders or they are initiated by nodes whose path state times out. The message is routed like a Path message having the IP destination address of the SESSION object and as the source address the IP source address of the sender of the path state that is being torn down. The path state is removed by matching the SESSION, SENDER_TEMPLATE and RSVP_HOP objects from the PathTear message with the values stored in the path state of the node. If no corresponding match is found the message is discarded.

The removal of a path state should maintain consistency in the node in what concerns the style of the related reservation state. If for example the style is WF the overall reservation should be removed only if the sender that initiated the message is the last sender of that session. The PathTear message uses a sender description component that has the same meaning like the one defined in the Path message format.

Using the same logic as for the PathTear messages the ResvTear messages are initiated explicitly by receivers or by intermediate nodes where the reservation state has timed out. These messages are sent in the same way as the Resv messages traveling hop-by-hop towards the senders and deleting matching reservation states in each node. The matching in this case is based on the SESSION, STYLE, FILTER_SPEC and RSVP_HOP objects. If no matching is found the message is discarded.

The PathErr messages can be generated by each node along the way of a reservation which detected an error. The node IP address which generated the message is included in the ERROR_SPEC object along with the code of the detected error. The message is sent hop-by-hop towards the sender using the information stored in the path state. The message does not modify any state along the way being only reported to the sender application.

Just like the PathErr message, the ResvErr can be generated by any node along the path that discovers an error. The address of the node and the code of the error are included in the ERROR_SPEC object. The message in this case travels hop-by-hop towards all the receivers, notifying all the nodes and merging points of the reservation on the way.

The ResvConf message is sent as a response of receiving a Resv message which has integrated a RESV_CONFIRM object. The message is transmitted to the address obtained from the RESV_CONFIRM object on a hop-by-hop basis in order to accommodate the hop-by-hop integrity check mechanism [2]. The ERROR_SPEC object is used in this case only to carry the address of the node which originated the message.

In Fig. 4 we present an overview of a RSVP messaging flow as this is illustrated in [2].

In what follows we are going to present in more detail how the RSVP messages are being sent, what is the behavior associated with the blockade state and the interfaces that are involved in the RSVP process.

Most of the RSVP messages are sent hop-by-hop between RSVP capable routers as raw IP datagrams with the protocol number 46 [2]. However, because some end systems might not support raw network I/O, UDP encapsulation can also be used.

For RSVP messages that are not sent on a hop-by-hop basis, like the Path and PathTear messages, but also for the ResvConf message, the Router Alert IP option must be activated in their IP header. This option will permit the routers to do special processing for these datagrams. In order to avoid problems associated with IP fragmentation, each RSVP message should occupy exactly one IP datagram.

RSVP messages are sent in an unreliable way, and rely on periodic refresh messages to recover from possible packet loss. However, under network overload conditions the increased

packet loss of RSVP messages can cause a failure of reservation. Therefore it is recommended that routers should be configured to offer a preferred treatment to RSVP messages in order to prevent this type of behavior.

In what concerns the usage of the blockade state, when a Resv refresh message is created, normally the FLOWSPECs of the reservation requests are merged by calculating their Lower Upper Bound (LUB). However, this rule is modified by the existence of a blockade state associated with one of the reservations to be merged. Reservations which are not blockaded are merged by computing their LUB, while blockaded reservations are ignored. If all the reservation requests are blockaded then they are merged by computing the Greatest Lower Bound (GLB).

The default value of the refresh period R is suggested to be 30 seconds. The impact of having a lower value of R would mean a speed up of the adaptation to the routing changes but at the same time will cause increasingly routing overhead. A higher value would trigger the inverse effect. The value of the refresh timer period is recommended to be randomly selected in the range $[0.5R; 1.5R]$. This recommendation is due to the disruption that a synchronized signaling can do to the network.

Besides the refresh period (R) for generating refresh messages, there is also a local state lifetime (L). The value of L is determined by the R value, and both of these can vary from one node to another. The L value has to satisfy the condition: $L \geq (K + 0.5) * 1.5 * R$. Where, K symbolizes how many refresh messages can be lost before the state times out. A value of 3 is suggested for K , but this value depends on the node characteristics (if a high loss rate is expected, K should have a higher value).

C. RSVP states

The information that is transmitted by RSVP through messages is stored in nodes along the way in what is defined as states. These states are stored in nodes using generic data structures named state blocks. In total four state blocks are defined to be used by RSVP: the Path State Block (PSB) that corresponds to the Path state, the Reservation State Blocks (RSB) corresponding to the Reservation state, the Blockade State Block (BSB) corresponding to the blockade state and a Traffic Conditioning State Block (TCSB) which is responsible for holding specifications of different reservations for a precise outgoing interface.

In Table II we present the content that is stored in the PSB. Most of the information from the PSB comes directly from the Path message that created that state. Besides the RSVP objects the PSB captures also data from the IP header, for example the remaining IP TTL, and from the routing process, for example the list of outgoing interfaces (OutInterface_list) and the incoming interface (InInterface) for this state.

The RSB holds a reservation request derived from a Resv message and identified by the SESSION, RSVP_HOP and a virtual object called Filter_spec_list. This object is style dependent and can be either a list of FILTER_SPECs for the SE style, a single FILTER_SPEC for the FF style or empty for the WF style. We present in Table III the content stored in the RSB. Contrary to the PSB case, all the information that is stored in the RSB comes from RSVP messages.

Fig. 4. RSVP signaling.

TABLE II
PATH STATE BLOCK FORMAT

Stored Information	Description
SESSION	Identifiers
SENDER_TEMPLATE	
Previous IP address and LIH	
Remaining IP TTL	-
Policy_Data and/or ADSPEC	Optional
Non_RSVP flag	Signal the presence of a non-RSVP capable node on the way
E_Police flag	Indicates to the edge policing capable device that policing should be done
Local_only flag	Indicates the forwarding PSB
OutInterface_list	The list of outgoing interfaces for the (sender, destination) pair
IncInterface	Indicates the expected incoming interface

The TCSB holds information for a specific outgoing interface. The TCSB information is derived from the RSB for that outgoing interface [16]. Each TCSB identifies a single reservation using a tuple composed of the SESSION, Outgoing Interface (OI) and the Filter_spec_list. The format of the TCSB is illustrated in Table IV. The TC_FlowSpec represents the LUB over the FLOWSPEC values in all the matching RSBs. The TC_FlowSpec value is used to make the actual reservation, being passed to traffic control [16].

The format of the BSB is presented in Table V. Depending on the style utilized, the BSB can use the pair composed of SESSION and Previous Hop (PHOP) as identifier or the pair composed of SESSION and SENDER_TEMPLATE. Other

TABLE III
RESERVATION STATE BLOCK FORMAT

Stored Information	Description
SESSION	Identifiers
Next hop IP address	
Filter_spec_list	
Outgoing (logical) interface	The outgoing logical interface on which the reservation is/ has been made
STYLE	-
FLOWSPEC	-
SCOPE	Optional, depending on style
RESV_CONFIRM object	Stores the received object (optional)

elements that compose the BSB are the FLOWSPEC Qb and the Blockade Timer Tb, where, the first parameter represents the FLOWSPEC of the offending reservation, and the second one the time that the flow has to stay in the BSB.

III. THE USE OF RSVP UNDER INTSERV

In this section we are going to illustrate the required specifications for the two service delivery models introduced by Integrated Services (IntServ): Controlled Load Services (CLS) and Guaranteed Services (GS). We present these specifications distinct from the original design since these represent an addition to the RSVP scheme and because these are treated by IETF as two different parts introduced in separate RFCs.

The use of RSVP in IntServ was specified in RFC 2210 [17]. This standard presents how Controlled Load Services and Guaranteed Services proposed by IntServ can be implemented using RSVP. It is important that a modality exists which can

TABLE IV
TRAFFIC CONTROL STATE BLOCK FORMAT

Stored Information	Description
SESSION	Identifiers
OI (Outgoing Interface)	
Filter_spec_list	
TC_Flowspec	The effective FLOWSPEC: i.e. the LUB over corresponding FLOWSPEC values from matching RSB's
Fwd_Flowspec	The updated object to be forwarded after merging
TC_Tspec	The effective sender Tspec
Police Flags	E_Police_Flag, M_Police_Flag and B_Police_Flag
Rhandle, F_handle_list	Handles returned by traffic control
RESV_CONFIRM object	The RESV_CONFIRM object to be forwarded

be used to communicate application requirements to different nodes in the network.

RFC 2210 defined the way in which objects concerned with QoS control services should be used and what the exact format of these objects in RSVP is. Three such objects were described: FLOWSPEC, ADSPEC and SENDER_TSPEC.

The information that describes the traffic flow for which the reservation is requested (Receiver TSPEC) and the parameters necessary to invoke the service (Receiver RSPEC) are included in the FLOWSPEC object. This information is carried from the receiver to the intermediate nodes and finally to the sender. The information contained in the FLOWSPEC object can be modified on its way to the sender by intermediate nodes. The modification can be caused by merging flows or by some other factors.

The information generated by each sender, describing the data flow is carried by the SENDER_TSPEC object. This information is never modified by the intermediate nodes from the network. The information which is generated or modified by the network elements regarding specific QoS control services parameters (i.e.: available services, delay, bandwidth estimate) is carried towards the receivers in ADSPEC objects. ADSPEC represents a summary of these parameters calculated or modified at each node on the path. The values of the parameters describe properties of the path without taking in consideration what is the requested QoS. This information is needed by the receivers so as to determine the necessary reservation specifications.

Examples of parameters included in the ADSPEC object are the Path bandwidth (BW) estimate that provides information about the estimated bandwidth available along the path and the minimum Path Latency parameter that records the absolute smallest value for latency. ADSPEC includes also one or more Break bits. One of the most important bits is the Global Break bit (GBb). This bit is set originally to 1' when an ADSPEC object is created. If a node exists along the path that does not support RSVP, the bit is set to 0' by another network element that has been aware of the gap (for example by comparing the IS hop count parameter with the IP TTL

TABLE V
BLOCKADE STATE BLOCK FORMAT

Stored Information	Description
SESSION	Identifiers
SENDER_TEMPLATE	
PHOP	
Flowspec Qb	"offending" FLOWSPEC
Blockade Timer Tb	Count down timer

value from the IP header). The fact that a non-RSVP capable node is encountered indicates that all the other parameters of ADSPEC can be invalid.

If the ADSPEC object is responsible for discovering path properties in terms of available QoS, the SENDER_TSPEC and FLOWSPEC objects are used to carry the parameters of the traffic that will flow on that path.

The SENDER_TSPEC describes the sender view of the parameters for the generated traffic under the form of a token bucket. Besides the bucket rate (r) and the bucket depth (b) also a peak rate (p), a minimum policed unit (m) and a maximum packet size (M) are specified in the SENDER_TSPEC object. This information can be used afterwards by either Guaranteed Service or Controlled Load Services to set the appropriate parameters in the FLOWSPEC object.

The FLOWSPEC object carries information necessary for the reservation request. The format of the FLOWSPEC object like the format of the ADSPEC object depends on the type of service that is required by the receiver. In both the CLS and the GS case the traffic parameters are expressed in token bucket specifications similar to the ones from the SENDER_TSPEC object.

The specification for GS includes in addition to the TSpec token bucket also present in the CLS case, the RSpec parameters. RSpec introduces additional QoS specifications in the case that GS is used. The terms included in RSpec are the R term, which indicates the desired reserved rate expressed in bytes per second and the slack term S which is expressed in microseconds [18]. The slack term is used in order to express the difference between the desired delay and the delay obtained using the rate R .

The specifications on what is the expected network element behaviour that supports CLS were presented in [19]. The QoS received under CLS approximates the QoS that the flow would receive from an unloaded network element. The end-to-end behaviour offered to an application subject to CLS will approximate the service received by the application in a lightly loaded best-effort network.

In order to ensure this type of behaviour, the network elements are provided with an estimation of the data traffic that will be generated. Each hop along the way of a data flow that uses CLS must ensure that adequate bandwidth and packet processing resources are available, as these are specified in TSpec, before accepting the request.

The amount of data sent should not exceed $rT + b$ where T is the length of the time period while r and b are the token bucket parameters. Packets that are bigger than the $rT + b$ bound or that are bigger than the outgoing link MTU are considered nonconforming. Nonconforming packets will be forwarded on

a best effort basis if sufficient resources are available. The m parameter is used as the minimum measuring unit for the token bucket. Even if the size of a datagram is less than m , it will be counted as being of size m .

The p value parameter exists only for compatibility purposes. No specific use of the peak-rate parameter was defined in [19].

The network element behaviour required for the delivery of guaranteed services is presented in [20]. Since GS is employed by traffic that requires strict guarantees on delay, the delay bound represents an important concern in this case. It is important to notice here that the delay is composed of two main parts: a fixed delay and a queuing delay. While the fixed delay is a property of the chosen path, the queuing delay is determined by the time that a packet has to stay in different queues in order to get service.

The queuing delay is expressed primarily as a function of two parameters: a bucket size (b) and a data rate (R). These two values are under application control, and an application can estimate apriori what is the queuing delay that guaranteed service will be able to assure.

The end-to-end behaviour conforms to a delivered queuing delay that does not exceed the fluid model delay by more than the specified error bounds. The fluid model represents the service that would be provided by a dedicated wire between the source and receiver [20].

The end-to-end delay bound is determined by the function:

$$- [(b - M)/R * (p - R)/(p - r)] + (M + C_{tot})/R + D_{tot}$$
 for the case when the data rate (R) is smaller than the peak data rate (p) but bigger than the token bucket rate ($r \leq R < p$)

$$- (M + C_{tot})/R + D_{tot}$$
 when the data rate (R) is bigger than both the token bucket data rate and peak rate ($r \leq p \leq R$)

Where, b (token bucket size), r (Token bucket rate), p (Peak data rate) and M (Maximum datagram size) are defined as before.

C_{tot} and D_{tot} are error terms which represent how the network elements deviate from the fluid model. These terms are computed end-to-end along the entire path from the rate dependent (C) and rate independent (D) error terms incorporated in the ADSPEC object.

In terms of policing the token bucket and peak rate parameters are the ones that are used to ensure that the amount of data to be sent is less than $M + \min[pT, rT + b - M]$. If this level is exceeded datagrams will be considered non conformant and will be treated as best-effort traffic.

IV. RSVP EXTENSIONS

Several extensions have been introduced over time by IETF in order to provide additional features to the protocol. We are going to illustrate in this section some of the most important additions to the protocol: the IP tunnelling enhancement mechanism, the cryptographic authentication, the extension introduced for policy control and the RSVP interfaces enhancements. Other extensions, like the Diagnostic facility [21] are omitted from this overview.

A. RSVP IP tunneling enhancement

One of the problems associated with RSVP was that in the case of IP tunnels, RSVP signalling was not possible. This

is due to the fact that RSVP packets which enter a tunnel are encapsulated with an outer IP header, where the protocol number is not 46 (4 for IP in IP encapsulation) and where the router alert option is turned off. This type of configuration makes the RSVP packets invisible to RSVP capable routers between the endpoints of the tunnel.

With the purpose of allowing RSVP operations over IP tunnels, two new objects were introduced in RFC 2746 [22], the SESSION_ASSOC object and the NODE_CHAR object. The first object was introduced in order to associate an end-to-end session with a tunnel session, by including this object in the end-to-end Path message at the tunnel entry point.

The tunnel entry point encapsulates and sends end-to-end Path messages through the tunnel to the end point of the tunnel where the message gets decapsulated and sent downstream. The same procedure is followed also by the end point of the tunnel upon the receipt of a Resv message. Taking in consideration the information present in the end-to-end Path, the FLOWSPEC in the end-to-end Resv and local policies, the tunnel entry point decides how to map the end-to-end session to a tunnel session. The tunnel entry point sends a Path message for a new tunnel session containing the SESSION_ASSOC object which associates the end-to-end session to the tunnel session. The tunnel end point responds to the Path message received transmitting a Resv message and reserving resources inside the tunnel.

The second object NODE_CHAR was introduced in order to allow the tunnel end point to inform the tunnel entry point that there is a tunnel end point supporting the RSVP operations over IP tunnels.

Basically the RSVP IP tunneling mechanism enables RSVP to make reservations over IP-in-IP tunnels by recursively applying RSVP for the tunnel portion of the path.

A more detailed approach on how the RSVP IP tunneling mechanism works is offered in [22].

B. RSVP Cryptographic Authentication

The ability of RSVP to offer protection against corruption and spoofing was introduced later by IETF with the publication of RFC 2747 RSVP Cryptographic Authentication. Two new objects were defined in [14]: INTEGRITY, and CHALLENGE. Even if the existence of the INTEGRITY object was mentioned in the original RFC defining RSVP, only later this object was defined together with the messages association rules.

The Hash-based Message Authentication Code Message-Digest (HMAC-MD5) was presented in [14] as the preferred cryptographic algorithm used for RSVP. Other cryptographic algorithms may be supported as well, but HMAC-MD5 is required as a baseline to be universally included in RSVP implementations. A performance evaluation of the MD5 and other three commonly used hash algorithms in the context of RSVP is presented in [23]. It is found that depending on the authentication key parameters and on the algorithm used the connection setup time can fluctuate [23].

The inclusion of the INTEGRITY object in RSVP messages slightly modifies the basic processing rules of the messages, both in the creation and on the reception of an RSVP message. This addition, only introduces a way in which protection

against forgery or message modification can be offered. The rest of RSVP operations as these have been presented beforehand are not affected by this process.

The second object, the CHALLENGE object, was introduced in order to create the integrity handshake, at the restart or initialization of the receiver. This object will be used in the context of two new defined messages types: the Integrity Challenge and the Integrity Response message. These messages are used in the handshake procedure for obtaining a live Authentication Key. Each key obtained has a limited lifetime and no key can be used outside its lifetime.

The keyed message digest present in the INTEGRITY object is computed using the Authentication Key obtained, in conjunction with the HMAC-MD5 keyed hash algorithm.

C. RSVP Extensions for Policy Control

The ability of RSVP to support policy based admission control was conceptually introduced in the first specification of the protocol. However RFC 2205 only states the existence of the POLICY_DATA object which should be used for policy control. The format of this object and the actual actions that have to be triggered in the nodes were presented later in RFC 2750 [15]. The introduction of the policy control mechanism is explained in [15] as a normal characteristic of a protocol which by definition has to discriminate between users. The policy control information is carried by the POLICY_DATA object situated in the RSVP messages.

Not all the nodes are required to support policy enforcement. The nodes that support this type of enforcement are named Policy Enforcement Points (PEP) and the ones that are considered non-trusted to handle policy information are named Policy Ignorant Nodes (PIN). Even if a PIN does not handle policy information it is still allowed to process RSVP messages. The general assumption made in RFC 2750 was that the PEPs are more likely to be at the border between different autonomous systems [15]. Even if the senders or the receivers are not aware of the policy objects, the PEP can generate, modify or remove these objects. This type of behavior supports the generation of consistent end-to-end policies [15].

The format of the POLICY_DATA object corresponds to the general format associated with RSVP objects. Present in this object are two special fields: the option list and the policy element list. Both lists have variable length, depending on the number of items included in the lists.

All the items in the option list are in fact normal RSVP objects using the same format but having a slightly different meaning. For example FILTER_SPEC, SCOPE and the RSVP_HOP are used to indicate to the PEP: the sender(s) associated with the POLICY_DATA object (in the case of FILTER_SPEC or SCOPE objects), the neighboring policy-capable node and the destination node (for the RSVP_HOP object). The INTEGRITY object is used to create a secure direct connection between PEPs excluding in this way the PIN nodes.

The policy element list is populated with policy elements. These policy elements are understood only by PEPs and are opaque to RSVP. The Internet Assigned Numbers Authority (IANA) is responsible for allocating and maintaining a registry

of the code points and the associated meaning of the policy elements.

Policy control is enforced only for four types of messages: Path, Resv, PathErr and ResvErr. It is generally assumed that teardown messages are received only from the same nodes that sent the installation, based on the integrity verification.

As a continuation of the specification presented in [15], RFC 2752 Identity Representation for RSVP defines the representation of identity information in POLICY_DATA [24]. The intent of the identity representation is to allow a secure identification of the owner and of the application of the communication process which are requesting network resources. For this purpose an Authentication Data (AUTH_DATA) policy element was defined.

With the intention of offering a way by which flows are admitted based on their importance, and not on a first come first served manner, a preemption policy element was defined in RFC 3181 [25]. This element uses the notions of preemption priority and defending priority to indicate the relative importance of a flow within a set of flows that are competing to be admitted into the network. The preemption priority field indicates the priority of the new flow. Complementary the defending priority is used to be compared with the preemption priority of new flows in order to decide if an existing flow should be preempted or not.

Normally the admission control mechanism that grants resources is based on user or application identity. RFC 3520 however proposed that it can be valuable to provide the ability of per-session admission control, and introduced a session authorization policy element which can be included in RSVP messages and verified by the network [26]. The goal of this Session Authorization Policy Element (AUTH_SESSION) is to enable the exchange of information between nodes in the network, so as to authorize the use of resources for a service and to co-ordinate actions between the signaling and the transport planes [26]. The host must obtain an AUTH_SESSION element and encapsulate this in the POLICY_DATA object within the RSVP Path message.

As a complementary specification of the signaled preemption priority policy element defined in [25], RFC 6401 introduced an extension for RSVP in order to support admission priority at the network layer admission control level. Two new RSVP policy elements were introduced, allowing the admission priority to be incorporated in RSVP signaling messages. This ensures that a selective bandwidth admission control can be enforced in RSVP nodes based on the session admission priority [27]. The new policy elements introduced are called Application-Level Resource Priority (ALRP) Policy Element and Admission Priority Policy Element.

The Application-Level Resource Priority Policy Element is used to convey application level information in RSVP messages. The ALRP Policy Elements are processed in a local Policy Decision Point (PDP). A PDP represents a node that controls the PEP at the periphery of a policy area. After processing the ALRP Policy Element, the PDP includes the application level information in the new defined Admission Priority Policy Element.

The Admission Priority Policy Element was designed to be simple, stateless and lightweight. This Policy Element is

easily processed in PEPs and allows the use of the bandwidth allocation models introduced for DiffServ-aware MPLS-TE networks. These bandwidth allocation models are the Maximum Allocation Model (MAM) standardized in [28], the Russian Dolls Model (RDM) introduced in [29] and the Maximum Allocation Model with Reservation (MAR) specified in [30]. Also a new simpler bandwidth allocation model, the Priority Bypass Model (PrBM) is introduced in [27]. Further description on how the bandwidth allocation model functions can be found in [31] and [32].

The discussion on the policy control elements and mechanisms does not represent the main focus of this paper. However, these were presented in order to illustrate the capability and modularity of RSVP in enforcing these control mechanisms. Further information regarding the policy control mechanisms can be found in the RFC that introduces the Common Open Policy Service (COPS), specifications on COPS usage for RSVP [33] and updating RFCs like [15] [34] [27] and [35].

D. RSVP interface enhancements

Besides the presented alterations of RSVP, also some interface enhancements of the protocol have been developed in order to make RSVP compatible or usable for other mechanisms or protocols. In what follows we are going to briefly present three RSVP interface additions: the RSVP extension for Internet Protocol Security (IPsec) data flows [36], the IEEE 802-style LAN interface [37] and the DiffServ interface [38]. These do not represent the full spectrum of RSVP interface extensions, but cover only the main extension proposed by IETF. The IPsec interface extension enables RSVP to be used with the two security headers introduced by IPsec. These two headers are the Authentication Header (AH) [39] and the Encapsulating Security Payload (ESP) [40] and are added by IPsec between the packet IP header and the transport protocol header. In RFC 2207 RSVP was extended so it can use the Security Parameter Index (SPI) introduced by IPsec instead of the TCP/UDP port numbers. New formats of the existing FILTER_SPEC, SENDER_TEMPLATE and SESSION objects were defined.

RSVP sessions are identified by the tuple composed of the destination address, protocol ID and the destination port. However, IPsec does not make use of the UDP/TCP destination port, this field being changed in the new defined SESSION object to a virtual destination port (vDestPort). This vDestPort allows the differentiation of multiple IPsec sessions destined to the same IP address.

The FILTER_SPEC object was changed so as to include the SPI. The SPI is used to allow the control of multiple independent flows between the same source and destination IP address. Traffic can be so classified to a reservation based on the SPI from the FILTER_SPEC. Conforming to the modification of the FILTER_SPEC object, the SENDER_TEMPLATE will have the same format as the new modified object.

As a consequence of the specifications of the modified objects, Path and Resv processing will require some modification as well. A session will be identified by the tuple composed of destination address, protocol ID and the vDestPort and

will be classified to reservations based on the SPIs from the FILTER_SPEC.

Introduced in [37] is the Subnet Bandwidth Management (SBM) signaling protocol which uses RSVP based admission control over IEEE 802-style networks. This standard describes how RSVP can be used to support reservations on link-layer devices.

The SBM protocol uses the concept of Designated SBM (DSBM), which represents an entity that resides and manages resources in a specific layer 2 (L2) segment. The procedure for selecting the DSBM is composed of an election process from all the SBM capable devices of that segment, as this is explained in [37].

In order to support the use of RSVP with L2 devices, the way in which RSVP functions has to be altered, and also some new objects have to be introduced.

The DSBM is in charge for allocating resources over a specific segment. As a consequence, every DSBM client on that segment must communicate its resource reservation requests to the DSBM. According to this procedure, each DSBM client forwards the Path messages to the DSBM instead of sending it to the RSVP session destination address. After processing and updating the ADSPEC object (if this is the case) the Path message is forwarded towards its destination. Normal processing and standard RSVP rules apply to the Resv messages, which are sent to the sender hop-by-hop on the reverse path of the Path message (including also the DSBM node).

However, because the DSBM node is not necessarily a layer 3 (L3) capable device (in which case it does not have routing information), new RSVP objects were introduced to allow correct operation and routing for RSVP messages.

Four new types of objects were introduced; these are RSVP_HOP_L2, LAN_NHOP, LAN_LOOPBACK and the TCLASS object.

The LAN_NHOP object is used to indicate to the DSBM what is the IP and MAC address of the next hop. The RSVP_HOP_L2 was introduced to convey the MAC address of the previous hop, complementary to the IP address found in the RSVP_HOP object. The MAC address is stored also in the path state of the node. This is necessary for L2 devices which might not have an Address Resolution Protocol (ARP) cache or ARP capability.

In order to facilitate the detection of loops that might happen in a segment, the LAN_LOOPBACK object was introduced. This object simply carries the IP address of an interface. Each DSBM client must overwrite the LAN_LOOPBACK object with its IP address. If the DSBM client finds a Path message with its own IP address in the LAN_LOOPBACK object, then it can discard the message, as being a duplicate.

The IEEE 802.3 Ethernet standard does not provide any isolation of traffic flows that require different services as this is requested in IntServ. However, IEEE 802.1p defined a way in which user-priority values can be used for differentiation. The TCLASS object carries the traffic classes based on the specifications of IEEE 802.1p. Only the last three bits are used from this object to indicate the user-priority value.

To some extent, the TCLASS object is treated like an ADSPEC object. The L3 devices should remove and store the

TCLASS object as part of the Path state for a specific session. When the Resv RSVP message is forwarded towards the sender, the TCLASS object must be included in this message. At the reception of the Resv message the sender should use the user-priority value to override the traffic class in the outgoing packets.

All four defined SBM specific objects are carried in RSVP messages in addition to the other original RSVP objects.

To support the use of RSVP in a DiffServ (DS) domain IETF standardized in RFC 2996 a new object called DCLASS object [38].

The basic idea adopted by IETF was that when RSVP is used in a DS domain it can be useful that the reservation protocol carries the Differentiated Service Code Points (DSCP) in RSVP messages. The main use of this new defined object is to carry the DSCP information from a DS network to nodes that may want to mark packets with DSCP values.

The DCLASS object may contain multiple DSCPs, enabling the host to discriminate sub flows within a behavior aggregate. The DSCP values that have to be associated with a particular flow are determined based on the resources required by RSVP requests and on the type of traffic. There is no specification on where the actual marking will be made. This could be done by an agreement or by a negotiation protocol. The standard itself presents only the format of the object and the availability of RSVP to carry this kind of information, leaving a lot of details to be implementation specific.

V. RSVP SCALABILITY IMPROVEMENTS

One of the problems associated with RSVP is the scalability issue of the protocol. Mainly, the protocol is accused that with an increase of the number of flows, the associated overhead in nodes that support RSVP also increases. For example, an analysis of the overhead introduced by RSVP on a commercial IP router can be found in [41]. It is found that the processing overhead becomes considerable when a large number of sessions are present. In extreme cases it is possible that the performance guarantee of some flows may not be kept and that some best-effort packets are dropped even if the total bandwidth requirements are smaller than the available bandwidth [41].

RFC 2961 identified as the main cause for the scalability issue the frequency at which refresh messages are generated by RSVP [42]. These messages are needed in order to maintain and synchronize RSVP states.

One way to address this problem will be to increase the refresh period R . However, increasing the refresh period will cause the protocol to take more time in order to synchronize states, thus increasing latency and decreasing reliability. This type of behaviour can be unacceptable for certain types of applications.

With the purpose of remedying this problem IETF developed diverse solutions over time. In what follows we are going to present the Refresh Overhead Reduction Extensions introduced in [42], the RSVP Reservation Aggregation standardized in [43] and the Generic Aggregate RSVP enhancement introduced in [44].

A. Refresh Overhead Reduction Extensions

So as to address the aforementioned problem, RFC 2961 standardized an RSVP extension which is referred to as Refresh Overhead Reduction. In fact three mechanisms were proposed in order to attenuate the scalability problem and reliability issue. The proposed mechanisms are the Message Bundling, the Message ID extension and the Summary Refresh extension. For the Message Bundling strategy a new bundle message was defined. This message uses a header identical with the RSVP common header, but has in the message type field the value 12 corresponding to the Bundle message.

A Bundle message further consists of a variable number of standard RSVP messages encapsulated or bundled within. This type of message is used to put together different RSVP messages in a single Protocol Data Unit (PDU). These messages are transmitted only from one RSVP capable node to another and only if the node is indicating its availability to use this refresh overhead reduction extension.

A Bundle message can include any other type of message, with the exception of another Bundle message. Each Bundle message cannot exceed the size of one IP datagram so that IP fragmentation is avoided.

A problem associated with this type of message is the time with which a normal RSVP message is delayed in order to be bundled. No exact specifications are presented in this case, but a differentiation is made between triggered messages and refresh messages. Triggered messages are messages that contain information transmitted for the first time. Refresh messages contain the same information and objects, as the corresponding triggered message and are transmitted on the same path, so as to refresh reservations and path states in nodes. It is specified that triggered messages should be delayed as little as possible and that refresh messages can be delayed at most until the next refresh interval of that message.

The second extension is the Message ID extension which enables acknowledgement and reliable RSVP message delivery, but also supports the Summary Refresh extension that we will present later. For this enhancement three new object types and one new message were introduced (MESSAGE_ID, MESSAGE_ID_ACK and MESSAGE_ID_NACK).

All three objects have a similar format, consisting of an 8 bit field for Flags, a 24 bit field named Epoch and a 32 bit field named Message_Identifier.

Only one flag was defined for the MESSAGE_ID object. The ACK_Desired flag is used to indicate to the receiver that the sender wants an acknowledgment of the message.

The Message_Identifier coupled with the IP address of the generator is used to uniquely identify a message. The value in the Message_Identifier field changes only in an incremental manner, and decreases just in two cases: when the value wraps or when the Epoch changes. The Epoch field indicates when the Message_Identifier field was reset and is generated randomly when the node reboots or the RSVP agent is restarted.

The MESSAGE_ID object can be included in any type of message except in the previous defined Bundle message and in the ACK message that we will present later. This object is always used on a per RSVP hop basis.

Whenever the MESSAGE_ID object is present in an RSVP refresh message, the value from the Message_Identifier field should be the same as the value from the RSVP message that first advertised the state that is being refreshed. When a triggered message is sent by a node, the Message_Identifier should have a greater value than other previously used values with the same Epoch value.

So as to ensure reliable delivery of RSVP messages, the ACK_Desired flag can be set in order to indicate the fact that the sender wants a MESSAGE_ID_ACK object sent in response. The MESSAGE_ID_ACK and MESSAGE_ID_NACK can be sent in any RSVP message where the IP destination address matches the address of the node that generated the MESSAGE_ID. When no such message is available, the new ACK message type can be used. The only functionality of the ACK message is to carry one or more MESSAGE_ID_ACK or MESSAGE_ID_NACK objects. The Summary Refresh extension utilizes all the previously defined Message ID extensions and introduces a new message type, the Srefresh message and three new objects MESSAGE_ID LIST, MESSAGE_ID SRC_LIST and the MESSAGE_ID MCAST_LIST.

The basic idea behind the summary refresh message is that instead of sending Path or Resv refresh messages between two nodes that implement this extension, a Srefresh message is sent instead. The node that receives the Srefresh message associates each listed Message_Identifier with the already installed Path or Resv state. The identified states are then updated as if a normal Path or Resv refresh procedure has taken place. The extension cannot be used for Triggered messages. The MESSAGE_ID LIST object is used to carry all the Resv and Path states from different unicast sessions. In the case of a multicast session the other two objects are used, since the node needs source and group information contained in these objects to refresh states.

The big advantage of this enhancement is that it reduces the amount of information that must be transmitted and processed in order to maintain RSVP synchronization.

Whenever a node cannot match the state received from the Srefresh message, the sender is notified using a refresh NACK [42]. Upon receiving a Message_ID_NACK object, the node has to make a Resv or Path state lookup based on the Epoch and Message Identifier values from the Message_ID_NACK object. If a state is found, a refresh message will be transmitted following normal Path and Resv procedures. If no matching state is found no further action is required. Specifications on how the Srefresh message should be triggered are not clearly presented. This is defined as being implementation specific.

An overview of RSVP signalling used for the RSVP Summary Refresh extension is presented in Fig. 5. A performance analysis of the Refresh Overhead Reduction Extensions for each of the proposed mechanisms is presented in [45]. The performance of the extensions is analysed taking in consideration three parameters: CPU load, throughput and signalling messages usage. It is found that better results are obtained for each of the three parameters when the proposed extensions are implemented compared with the original RSVP design [45]. A more detailed investigation covering also different implementation aspects of RSVP but also a decomposition of the execution overhead can be found in [46].

B. Aggregation of RSVP Reservations

The extensions introduced in RFC 2961 are not the only addendums proposed by IETF in order to enhance the scalability of the protocol. The introduction of the DiffServ mechanism has facilitated a new extension of RSVP, standardized in RFC 3175. The new addition proposes aggregating in a single RSVP reservation multiple reservations across a transit domain or a routing region [43].

One of the primary reasons why RSVP did not manage to aggregate reservations was that it didn't have a clear way of classifying an aggregate [43]. The development of the DiffServ mechanism managed to solve this problem. DiffServ traffic that belongs to a particular class can be associated with a specified DSCP and so classified.

The basic idea introduced in [43] was that several end-to-end (E2E) reservations crossing a common set of RSVP capable nodes could be aggregated into one larger reservation from ingress to egress. The edge node that implements the aggregation of reservations is called the Aggregator node, while the node that deregates the reservation at the end of the common region is called the Degregator node.

The Aggregator hides E2E RSVP messages from the RSVP capable routers in the aggregation region. This is done by changing the IP protocol number in the common region for the aggregated flow from RSVP (46) to a new introduced IP protocol number RSVP-E2E-IGNORE (134). The protocol number is restored to RSVP at the Degregator point. The change of the protocol number is done only for E2E Path, PathTear and ResvConf messages. Resv and other messages do not require this modification since these are unicast and travel from one RSVP hop to another.

The token bucket parameters of the E2E reservations are summed up into corresponding information elements in aggregate Path and Resv messages. The aggregated Path message is sent from the Aggregator to the Degregator using the normal IP protocol number (46). Correspondingly, the aggregated Resv message is sent from the Degregator to the Aggregator creating an aggregate reservation for a set of E2E flows in this common domain.

The DiffServ mechanism is used for classification and scheduling of the aggregated reservation(s). One or more DSCP values could be used in this case, so as to differentiate among different classes of traffic that might be mapped between the same Aggregator and Degregator.

In order to better understand how the proposed extension works we will take a step by step approach. We present in Fig. 6 the RSVP operations for the RSVP Reservation Aggregation extension.

At the receipt of an E2E Path message, the Aggregator has to determine whether the message is going to traverse the common region or not. If the latter case applies, the message is sent in the regular way using regular RSVP procedures. In the first case reservations can be aggregated. Because the Degregator is unknown at this point, the node does not know in which aggregated reservation the flow should be included. Therefore, the E2E Path message is sent with the RSVP-E2E-IGNORE protocol number (134) following for the rest normal RSVP procedures. The change in the protocol number will

Fig. 5. RSVP Summary Refresh extension.

Fig. 6. RSVP signaling for the Reservation Aggregation extension.

cause all the nodes between the Aggregator and Degregator to simply forward the message as a normal IP datagram.

The receipt of the E2E Path message by the Degregator triggers several actions. First, before forwarding the E2E Path message to the receiver, the ADSPEC object should be updated in order to reflect the impact of the aggregation. This is done by inspecting the ADSPEC of the aggregated Path message which travels from the Aggregator to the Degregator. However, the Degregator should check beforehand if an aggregated path message exists for the corresponding DSCP onto which the E2E flow will be mapped. If this exists, the ADSPEC from this aggregated path is used to update the received E2E Path message. If this does not exist the Degregator requests the establishment of the appropriate aggregated path.

This request for establishing an aggregated path is done by sending an E2E PathErr message towards the Aggregator with a new introduced code (NEW-AGGREGATE-NEEDED) and with the desired DSCP encoded in the DCLASS object. When the Degregator receives the aggregated Path message matching the desired DSCP, the ADSPEC of the received aggregated Path is used to update the ADSPEC of the E2E Path.

The generation of the mentioned E2E PathErr does not trigger in this case the removal of any Path state. Instead, it should invite the Aggregator to initiate the necessary aggregated Path message.

In the end the Degregator should forward the E2E Path message to the intended destination after changing the protocol number back to RSVP.

Upon receiving the E2E Resv message for the session, it is the responsibility of the Degregator to ensure that enough resources exist in the aggregated domain to support the new E2E reservation. At this point two cases are possible.

First, if no aggregated reservation exists for the desired DSCP, a new aggregated reservation should be initiated. The Degregator should look up the aggregated path state, which was used earlier in order to retrieve the ADSPEC information, and create a new aggregated Resv message as a response to the received aggregated Path message. The new aggregated Resv message includes a FLOWSPEC of a value that is not smaller than of the required E2E reservation. The aggregated Resv is using new Aggregated_RSVP C-Types defined under the SESSION, SENDER_TEMPLATE and FILTER_SPEC Class-Num introduced in [2].

On the other hand, if an aggregated reservation already exists for the desired DSCP and has enough bandwidth to support the new flow, the Degregator sends the E2E Resv message using the IP protocol number for RSVP to the Aggregator including also the DSCP object. This message represents the final confirmation requested by the Aggregator to map the E2E flow to the aggregated reservation.

However, if the aggregated reservation has not enough resources for the E2E flow, the Degregator will try to increase the bandwidth for the aggregated reservation. This is done by including in the aggregated Resv message an increased size of the FLOWSPEC. If this request fails the normal RSVP procedures are followed for a reservation with insufficient resources. The E2E reservations are removed as usual, as an effect of a PathTear or a ResvTear message or as a result of an error or timeout. When the reservations are removed these should also be removed from the aggregated reservation.

As we have mentioned, three new objects were defined for the introduction of RSVP aggregation. These objects are SESSION, SENDER_TEMPLATE and FILTER_SPEC. These objects were defined under the same Class-Nums introduced in [2], but under different C-Type values.

The SENDER_TEMPLATE and FILTER_SPEC objects were defined to carry the IP address of the Aggregator while, the SESSION object was defined to carry the IP address of the aggregated session destination and the DSCP that will be used for the aggregated reservation.

An analysis of the aggregated RSVP extension is presented in [47]. Unsurprisingly it is found that the aggregated RSVP alienates some of the limitations of RSVP by reducing the overhead on the aggregated region. However this does not mean that the RSVP scalability problem is completely solved. As it is explained in [47] aggregated RSVP flows are merely fatter RSVP reservations and increasing the number of flows will induce the same scalability problem as the original RSVP design.

C. Generic Aggregate RSVP

The RSVP aggregated reservation extension introduced in [43] allows the establishment of separate aggregated reservations across a domain under different DSCP values. The DSCP value is used to classify each packet for a specific Per Hop Behavior (PHB) at every network node along its

Generic_Aggregate_IPv4 SESSION		
Length (Bytes)	Class=1	C-Type=17
Ipv4 DestAddress (4 bytes)		
Reserved	Flags	PHB-ID
Reserved		vDstPort
Extended vDstPort		

Generic_Agg_IPv4_SOI		
Length (Bytes)	Class= 132	C-Type=1
Content of a Generic_Aggregate_IPv4 SESSION object		

Fig. 7. Format of the Generic_Agg_IPv4_SOI and Generic_Aggregate_IPv4 SESSION objects.

path. However, only one aggregated reservation can be established for a given PHB between a certain sender IP address and the corresponding IP destination address. This drawback represents a problem for scenarios where multiple aggregated reservations are needed for the same PHB. Situations where this type of malleable behavior is desired are presented in [44].

As a solution of the presented problem, a more flexible type of aggregated reservations was introduced in RFC 4860 [44]. The standard uses the notions of Virtual Destination Port (VDstPort) introduced in [36] and Extended Destination Port which represents a generalized version of the Extended Tunnel ID presented in [48]. These notions were added to the RSVP SESSION object in order to narrow the scope of the session to the ingress and egress pair.

Two new objects were defined under the existing SESSION class, the Generic_Aggregate_IPv4 SESSION object for IPv4 addresses and the Generic_Aggregate_IPv6 SESSION object for IPv6 addresses. Also two other objects, Generic_Agg_IP4_SOI and Generic_Agg_IP6_SOI were defined under a new introduced class called Session_Of_Interest. The format of these objects is presented in Fig. 7.

The VDstPort in the new SESSION object represents a 16-bit identifier that remains constant over the lifetime of the generic aggregate reservation. The new SESSION object uses the PHB ID, introduced in [49] for identifying a PHB or a set of PHBs, instead of using DSCP. This is due to the fact that such a PHB ID allows conveying of PHBs, independently from the DSCP that is used locally.

In the case of the Session_Of_Interest class the field called content of a Generic_Aggregate_IP4 SESSION object represents a copy of the SESSION objects. Most of the procedures previously presented for aggregated reservation apply also for the generic aggregated RSVP reservation extension. However, some small alterations are required. The Degregator must include in the E2E PathErr message a SOI object that contains the Generic_Aggregate SESSION that will be used for establishing the requested generic aggregated reservation. In this situation the DCLASS object is no longer necessary since the PathErr message contains a SESSION object with the PHB ID.

Thus a sharing scenario different from the one in the aggregate reservation case can be supported by modifying the VDstPort and the Extended VDst Port field. Separate reservations from a given Aggregator to a given Degregator can be made by specifying distinct VDstPort and Extended

VDst Port values. If sharing is desired between multiple Aggregators to a certain Degregator, the same values can be specified in the two fields.

At a receipt of an E2E PathErr message with the code `New_Aggregate_Needed` and that contains a SOI object, the Aggregator is responsible for initiating the establishment of a new generic aggregate reservation. This is done by sending an aggregated Path message with the `Generic_Aggregate_IPv4` Session object found in the SOI object.

The Degregator will use the same procedures as the ones defined in [43], but using the `Generic_Aggregate` Session object presented earlier.

When the E2E Resv message is processed, the Degregator must include a SOI object that indicates to the Aggregator what is the generic aggregate session used to map the E2E reservation onto. As mentioned earlier, the DCLASS object is no longer necessary in this case, since the information about the PHB should be applied is contained in the new SESSION object which is included in the SOI object.

The Aggregator will be responsible for interpreting the SOI object and for mapping the E2E reservations to a generic aggregate reservation session. The SOI object should be removed from the message when the E2E Resv is sent upstream towards the sender.

Reservations are removed in the same way as in the aggregated RSVP reservation case.

VI. RSVP EXTENSIONS FOR MPLS AND GMPLS

The introduction of the Multiprotocol Label Switching architecture in [50] has triggered the extension of RSVP in order to support the establishment of label-switched paths (LSPs) and to incorporate traffic engineering features. This extension for which several additional objects were introduced is called RSVP Traffic Engineering (RSVP-TE) [51].

However, in order to allow the MPLS control plane to support Synchronous Optical Network (SONET), Synchronous Digit Hierarchy (SDH), wavelengths (optical lambdas) and spatial switching, the original MPLS architecture was extended to Generalized MPLS (GMPLS) in [52]. Together with this enhancement, correspondingly an extension of RSVP-TE was introduced to support RSVP-TE signalling over GMPLS.

In this section we will present the RSVP Traffic Engineering extension for MPLS and subsequently the GMPLS RSVP-TE extension.

A. RSVP Traffic Engineering extension for MPLS

MPLS is based on the concept of labels. In this architecture the IP header is checked only once, when the packet first enters the MPLS network. Afterwards, the packet is associated with a label, which is further used for directing it to the correct destination. In MPLS terminology the term LSP is used to define the path that the labeled packet has to follow in a network. The label can be viewed as an index in a table that specifies the next hop of the packet and what actions the network element has to perform. A network element that can recognize a label and that can perform specific actions on that label is called a Label Switch Router (LSR). All the LSRs that reside in a MPLS network have to inform each other about

the meaning of particular labels. This information exchange is done by what is defined in the MPLS architecture as the label distribution protocol.

RSVP was extended in [51] to behave as a label distribution protocol and to implement traffic engineering features. Traffic engineering as defined in [53] is concerned with the performance optimization of operational networks, having as a main goal to facilitate efficient and reliable network operations while simultaneously optimizing network resources utilization and traffic performance [53].

This new extension of RSVP supports the creation of explicitly routed LSPs with or without resource reservation [51]. A LSP created by RSVP can be used to carry an aggregation of traffic flows of the same class [54]. Because traffic flows along a LSP are identified by the label which is associated with it, such a path can be treated as a tunnel. Tunneling in this case, is realized below the normal IP routing and the associated filtering mechanism. LSPs that are used in this way are referred to as LSP tunnels.

In order to support the LSP tunnel feature, new RSVP SESSION, SENDER_TEMPLATE and FILTER_SPEC objects were defined through new C-Types (`LSP_Tunnel_IPv4` and `LSP_Tunnel_IPv6`). Besides these, five other new objects were introduced: LABEL_REQUEST, LABEL, EXPLICIT_ROUTE, RECORD_ROUTE and the SESSION_ATTRIBUTE object. The LABEL_REQUEST object indicates that a label binding is requested but also provides information about the network layer protocol used over specific paths. The LABEL object conveys the label associated with the outgoing traffic for a specific LSP tunnel. The EXPLICIT_ROUTE object carries the route that has to be followed by a LSP as a sequence of nodes. The RECORD_ROUTE is used to inform the sender node about the actual route that an LSP tunnel traverses and the SESSION_ATTRIBUTE object is used for aid in session identification and diagnostics [51].

So as to create a LSP tunnel, the first node in the MPLS network generates a RSVP Path message with the new defined SESSION object and LABEL_REQUEST object included. If this node has information about what path will meet the tunnel QoS requirements, or satisfies some policy criteria, an EXPLICIT_ROUTE object is introduced in the Path message as well. This object can be changed if a better route is found later, creating in this way the possibility to reroute the session in a dynamic way. Also, a RECORD_ROUTE and a SESSION_ATTRIBUTE object can be optionally inserted in the Path message.

The destination node responds to the LABEL_REQUEST object, by including in the RSVP Resv message a LABEL object. As in the normal RSVP case the Resv message is sent back upstream following the path state created by the Path message. Each node along the path receiving a Resv message that contains a LABEL object will use that label for the outgoing traffic associated with that LSP tunnel. If the receiving node of the Resv message is not the sender, a new label is allocated and placed in the LABEL object of the Resv message which is sent further upstream [51]. This new label is used to identify the incoming traffic associated with that LSP.

```

<Path Message>::=<Common Header>[<INTEGRITY>]<SESSION>
<RSVP_HOP><TIMES_VALUES>[<EXPLICIT_ROUTE>]
<LABEL_REQUEST>[<SESSION_ATTRIBUTE>]
[<POLICY_DATA>...]<sender descriptor>;

<Sender descriptor>::=<SENDER_TEMPLATE><SENDER_TSPEC>
[<ADSPEC>][<RECORD_ROUTE>].

```

Fig. 8. Format of the Path message in RSVP-TE.

The new defined objects are optional with regard to RSVP. However, the LABEL_REQUEST and LABEL objects are mandatory for the new traffic engineering extension of RSVP (RSVP-TE) [51]. The extended format of the Path and Resv messages and the format of the new defined objects will be presented next.

As we can see in Fig. 8, the format of the Path message is extended from the original specifications of [2] to include the LABEL_REQUEST and optionally the EXPLICIT_ROUTE and SESSION_ATTRIBUTE objects. Also, the RECORD_ROUTE object can be specified in the sender descriptor in order to explicitly include the route that has been followed by the Path message.

Even if the SESSION and the SENDER_TEMPLATE objects seem to be identical to the ones used by the original specifications of RSVP, these objects have a new format defined under the same Class-Num but with new C-Type values. We are going to present the format of these new objects below.

The format of the Resv message is presented in Fig. 9. Optionally the RECORD_ROUTE object can be included in the Resv message. An observation that can be made here is that each LABEL and RECORD_ROUTE object are coupled to a FILTER_SPEC object that precedes them in the flow descriptor list. Only one LABEL and one RECORD_ROUTE can exist for one FILTER_SPEC object.

The LABEL object contains only a single label encoded in 4 octets. The downstream node is the one responsible for selecting a label for the flow. If a label is not available then the node sends a PathErr message indicating label allocation failure [51]. When the label is allocated, a new LABEL object is created and is forwarded to the upstream node in a Resv message [51]. The LABEL object is stored in the Reservation State Block, and the node should be prepared to forward packets carrying the assigned label.

The upstream node uses the label from the LABEL object as the label associated with the outgoing interface for that sender. Also, a new label is allocated in the same node and is linked to the incoming interface for this session/sender. The interface is the same as the one used to forward Resv messages to previous hops.

The LABEL_REQUEST object was defined in the context of three possible C-Types. The first type is named Label Request without Label Range and is employed to indicate the layer 3 protocol that uses the path. The other two types are specific for Asynchronous Transfer Mode (ATM) and Frame Relay technologies.

The LABEL_REQUEST object is stored in each node along the path in the Path State Block. Each node that accepts a

```

<Resv Message>::=<Common Header>[<INTEGRITY>]<SESSION>
<RSVP_HOP><TIME_VALUES>[<RESV_CONFIRM>]
[<SCOPE>][<POLICY_DATA>...]<STYLE>
<flow descriptor list>;

<flow descriptor list>::=<FF flow descriptor list>|
<SE flow descriptor list>;

<FF flow descriptor list>::=<FLOWSPEC><FILTER_SPEC><LABEL>
[<RECORD_ROUTE>]|<FF flow descriptor list>
<FF flow descriptor>;

<FF flow descriptor>::=[<FLOWSPEC>]<FILTER_SPEC><LABEL>
[<RECORD_ROUTE>];

<SE flow descriptor>::=<FLOWSPEC><SE filter spec list>;

<SE filter spec list>::=<SE filter spec>|<SE filter spec list>
<SE filter spec>;

<SE filter spec>::=<FILTER_SPEC><LABEL>[<RECORD_ROUTE>].

```

Fig. 9. Format of the Resv message in RSVP-TE.

LABEL_REQUEST object must include a LABEL in the Resv message that responds to that Path message. Every node that sends a LABEL_REQUEST object must be ready to receive and handle the LABEL object in the resulting Resv message [51].

If a router knows that it has a neighbour which is not RSVP capable, then the LABEL_REQUEST object must not be advertised when sending the message that passes through non-RSVP nodes. The router should also send a PathErr message back with the error value MPLS being negotiated, but a non-RSVP capable node stands in the path [51].

As we already presented, the EXPLICIT_ROUTE object (ERO) is used to indicate a route that has to be followed by a LSP. This object is intended to be used only for unicast traffic flows. The explicit route is specified by the subobjects included in the EXPLICIT_ROUTE object.

Each subobject indicates the IP address of a node that has to be included in the path, or the Autonomous System (AS) number to which that node should belong. Also, each subobject must indicate whether it represents a strict or a loose hop in the explicit route. The process that is used by the LSR in evaluating and handling the ERO is composed of the following six steps:

1. The node evaluates the first subobject from the ERO. If the node detects that it does not belong to the specified AS number or that it is not the node with the IPv4 or IPv6 address indicated by the subobject, it sends back an error message.
2. After evaluating the first subobject, it is checked if a second subobject exists. If no second subobject exists it means that this is the end of the explicit route and the ERO object should be removed from the Path message.
3. The node verifies if it has an address that belongs to the second subobject. If this is the case the first subobject is removed and the procedure goes back to step 2. If however, the node does not belong to the address space indicated by the second subobject the procedure continues with the 4th step.
4. The node has to determine if it is attached to the node described by the second subobject. If that node is an abstract node, for example representing an AS with multiple hops, then a next hop is chosen that is a member of that abstract node. An abstract node represents a group of nodes whose

internal topology is opaque to the ingress node of the LSP [51]. Afterwards the node erases the first subobject from ERO and has the ability to add new subobjects in the ERO (if this is necessary).

5. If the node determines that it is not attached to the second subobject from ERO, then it selects a next hop from the abstract node of the first subobject (subobject to which also this node belongs) which is on the way towards the second subobject from ERO. If the node cannot find such a path, then two possible causes exist:

a. The second subobject indicates a strict node (not an AS), case in which an error message is returned.

b. The subobject indicates a loose node to which no path can be found. In this case also an error message is returned.

6. The node replaces the first subobject with an abstract node that contains the next hop. This action is necessary so as to the next hop will accept the reservation.

The explicit route extension does not provide support for RSVP routers that do not recognize ERO. If such a node is encountered on the path, a PathErr message with the error code Unknown object class is send towards the sender [51].

The RECORD_ROUTE object (RRO) enables RSVP to inform the sender of the actual path followed. This object has the same format as the ERO, being also composed of a list of subobjects. Three types of subobjects are defined to be used within RRO in [51]. From these, the IPv4 address and the IPv6 address have the same format as the ones specified for ERO. The third subobject is a Label, and it allows RRO to record labels. This Label subobject cannot be used singly. It must always be associated with the IP address of the node and is pushed in the RRO prior of pushing the node's IP address. In a RRO the subobjects are recorded in a last-in-first-out stack, meaning that a subobject is always added to the top of the stack.

The SESSION_ATTRIBUTE object was defined in order to aid in session identification, diagnostics, and to provide additional control like setup and hold priorities. This object was defined with two C-Types: the LSP_Tunnel and the LSP_Tunnel_RA (with Resources Affinities). Both C-Types have in common the Setup Priority, the Holding Priority, Flags and the Name Length fields.

The Setup priority is used in order to indicate what the priority of the session in taking resources is. This is expressed as a value from 0 to 7, 0 representing the highest priority. Correspondingly, the Holding Priority is used in deciding if the session can be preempted by another session. When a flow asks for resources, and all the resources are already reserved, the setup priority of that flow is compared with the holding priority of the already established flows. If the setup priority is higher than the holding priority of a defending flow, then the defending flow is preempted and resources are granted to the new incoming flow.

The Name Length field simply indicates the length of the display string. Three types of flags were defined for the SESSION_ATTRIBUTE object: the Local Protection Desired flag, the Label Recording Desired flag and the SE Style Desired flag. The Local Protection Desired flag indicates to the intermediate nodes on the path that a form of local repair mechanism can be used that can violate or alter the explicit

route expressed by ERO. The Label Recording Desired flag indicates that the Label subobject should also be included in the RRO when doing route recording. The SE Style Desired specifies that the ingress node of the tunnel may choose to reroute the tunnel without tearing it down [51].

The LSP_Tunnel_RA C-Type includes also three 32-bits fields indicating conditions that have to be fulfilled by a link in order to be considered valid. The Exclude-any field renders a link unacceptable if the link has any of the attributes in the set. The Including-Any validates a link if this has any of the attributes in this set. Accordingly, a link is considered acceptable if all the attributes that are present in the Include-All set are properties of that link. More specifications about the attributes sets and processes associated with the LSP_Tunnel_RA are presented in [51].

Two new C-Types were introduced for the SESSION object Class-Num specified in [2]. These new C-Types are intended for the RSVP-TE extensions with IPv4 addresses (LSP_Tunnel_IPv4) and IPv6 addresses (LSP_Tunnel_IPv6).

The SESSION object includes an IPv4 tunnel end point address which represents the egress node of that tunnel, a Tunnel ID and an Extended Tunnel ID. Both types of tunnel IDs have to remain unchanged over the lifetime of the tunnel. The Extended Tunnel ID can be used in order to limit the scope of the SESSION to a specific ingress-egress pair by putting the IP address of the ingress node in this field [51].

The new SENDER_TEMPLATE and FILTER_SPEC objects have an identical format. As in the SESSION object case two new C-Types were introduced for each of these objects, one for IPv4 and one for IPv6.

The new C-Types specify the value of the IP address of the tunnel sender node and a LSP ID. The LSP ID represents an identifier of the LSP that can be chosen in order to permit the sender to have multiple LSPs sharing resources.

Besides the RSVP extension for traffic engineering and support for LSP creation, also an RSVP Hello extension was introduced in [51]. This extension provides RSVP nodes with the ability to detect the reachability of neighbouring nodes. The extension consists of a new Hello message and two new objects: a Hello Request object and a Hello Acknowledgment (ACK) object. Both objects, have the same format and belong to the same class, but have different C-Types values.

The Hello message is to be used by two immediate RSVP neighbours. Each node has the possibility to issue a Hello request, which in turn is answered with a Hello ACK. Neighbor failure detection is done by collecting and storing a neighbor's instance value. The instance represents a unique identifier associated with a neighbor. The value of this identifier should not be changed while the node is exchanging Hello messages with other nodes. Further specifications on how the Hello extension operates are presented in [51].

To conclude, it can be observed that the soft state mechanism used by RSVP-TE is very similar to the one used by RSVP. Refresh messages that manage LSP connectivity must be transmitted periodically. As a consequence the scalability problem of the RSVP-TE protocol exists also in MPLS networks [55]. An analysis on how efficient the refresh overhead reductions are for RSVP-TE is presented in [55]. The processing overhead is evaluated by measuring the CPU

load of an LSR as the number of LSPs increases. It is found that the summary refresh mechanism manages to reduce the processing overhead by reducing the number of sending and receiving messages” [55]. However, it is argued that even if RSVP uses the summary refresh mechanism “the scalability problems still remains for larger numbers of LSPs” [55].

B. RSVP-TE extension for GMPLS

Four types of interfaces are supported in GMPLS. The first type, which is already supported in MPLS, is the ATM VPI/VCI interface that recognizes the packet/cell boundaries and forwards traffic based on the content of the packet/cell header. This type of interface is referred to as Packet_Switched Capable (PSC). The second type is the interface that forwards data based on a time slot in a repeating cycle, like in the SONET/SDH Cross-Connect. This type of interface is called Time-Division Multiplex Capable (TDM). The third type is the Lambda Switch Capable (LSC) interface which forwards traffic based on the wavelength on which the data is received. The fourth type is referred to as Fiber_Switch Capable (FSC), and forwards data based on the position of the data in real word physical spaces (e.g. an interface on an Optical Cross-Connect that operates at the level of a single or multiple fibers).

In order to be able to communicate with these interfaces, RSVP-TE was extended and new forms of label objects were introduced: Generalized Label Request, Generalized Label, Generalized Label with support for waveband switching, Suggested Label and Label Set. These are referred to collectively as a generalized label.

An observation that has to be made here is that since the nodes which are sending and receiving the new form of label know what kinds of link they are using, the introduced labels do not contain a type field. The nodes are required to know from the context what type of label to expect [52].

The format of the Generalized Label Request object is presented in Fig. 10. The LSP Encoding Type is used to indicate the type of encoding that the LSP is requesting. Eleven values were defined in [52]. Eight values were defined for the Switching Type. They are used to indicate the type of switching that should be performed on a particular link. This indication is necessary for interfaces that advertise more than one type of switching capability. The eight values comprise four values for PSC (PSC from 1 to 4) a value for Layer-2 Switching Capable (L2SC), a value for TDM, one value for LSC and one value for FSC.

The Generalized PID (G-PID) field is used to identify the payload carried by an LSP. The values that were specified for this field are presented in [52].

The Generalized Label Request object is set by the ingress node and localized in the Path message. This object has a class type of 19 and is named RSVP_LABEL. A node that receives a Path message containing a Generalized Label Request has to verify that the parameters requested can be assured by the interface where the incoming label is to be allocated. The ability of the node to create or use tunnels is dictated by the local policies applied at that node. In the event that the requested LSP Encoding Type cannot be supported a PathErr

Fig. 10. Format of the “generalized label” objects.

message must be generated with the Unsupported encoding indication. The G-PID value is only verified at the egress point. If it cannot be supported a PathErr message with Unsupported L3PID must be generated.

The non-PSC types used by GMPLS can perform bandwidth allocation only in discrete units [52]. Typical values that can be used for requested bandwidth are presented in [52]. These values are carried in the SENDER_TSPEC and FLOWSPEC objects defined in [17]. Only the Peak data rate field is used in this case, such that bandwidth and service parameters in the object can be ignored and carried transparently [56].

The format of the Generalized Label object is presented in Fig. 10. The Generalized Label is used to extend the traditional label used in [51] by allowing also the representation of labels that identify time-slots, wavelengths and space division multiplexing. The Generalized Label object allows the transportation of generic MPLS labels which are encoded right justified in the label field in 32 bits. In the case of ATM labels, these are encoded with the VPI right justified in the first 16 bits and the VCI right justified in the last 16 bits. For the use of FSC and LSC the label indicates the data channel/link to be used for a LSP. If we talk about a port and a Waveband label, the information contained in the label field indicates the port/fiber or lambda to be used. As we already presented, the Generalized Label does not indicate the class in which it resides, this is assumed to be implicit in the multiplexing capability of the link [52]. A special case that can appear in lambda switching is waveband switching. An optical cross connect should be able to switch multiple wavelengths as units. For this purpose the Generalized Label with support for Waveband Switching was introduced. The waveband ID represents the waveband identifier and the Start and End Label indicates the channel identifier, delimitating the highest and respectively the lowest values of the wavelength making up the waveband.

In order to reduce setup latency for nodes that take a considerable amount of time in establishing a label in hardware, a Suggested_Label object was introduced in [52]. This pro-

vides to a downstream node, information about the upstream node preferences, enabling the upstream node to initialize the hardware configuration before the label is communicated by the downstream node. The format of this object is the same one used by the RSVP_LABEL object. The Class-Num of this object is 129 (or of the form 10bb bbbb ignore without forwarding) and the C-Type corresponds to the label that is suggested.

Also, in order to limit the choice of a downstream node to a specific set of labels, the Label_Set object was introduced. The format of this object is presented in Fig. 10. The receiver of the Label_Set object must restrict the choice of the label to one that is in the label set. The Label Type indicates the type and format of the labels carried in the object. The values used for this field being the C-type of the appropriate RSVP_Label object previously presented. The subchannels indicate the labels that can be used for allocation. The format that is used here depends on the C-Type used for the RSVP-LABEL object. A label set is defined using one or more Label_Set objects. Depending on the value of the action field, specific labels included in the Label_Set object are added (action field 0) or excluded (action field 1). Also, whole ranges of labels, subchannels can be included (action field 2) or excluded (3) from the label set. In normal MPLS procedures, a bidirectional path can only be established by initiating two independent LSPs. However, in GMPLS the support for bidirectional LSPs is directly provided. This is due to the fact that many optical network service providers require bidirectional optical LSPs to reduce the setup latency and control overhead.

A bidirectional LSP is indicated by the inclusion of an Upstream_Label object in the Path message. This new Upstream_Label object has the same format as the Generalized Label object and has a C-Type that indicates the label being used.

The node receiving the Path message processes it in the normal way, with the exception that the upstream label can be used immediately to transport traffic affiliated with that LSP towards the initiator. Each intermediate node has to allocate a label on the outgoing interface before filling in an outgoing upstream label and forwarding the Path message. Four notification extensions were introduced in order to support the use of RSVP-TE for GMPLS. The first extension defines the Acceptable_Label_Set object, introduced in order to indicate from one node to another which labels would be acceptable. The format of this object is identical to the one of the Label_Set object.

The second and third extensions (the Notify Request object and the Notify message) enable the notification of failures and other events to LSR responsible for restoring failed LSPs. The Notify message represents a generalized notification message that is targeted towards the node address included in the Notify Request object. Further specifications on the actual format and on the way the Notify message is used can be found in [56].

The fourth notification extension introduced allows the removal of Path states while handling PathErr messages. The utilization of explicit routes creates the premises that errors on the path can only be corrected by source nodes or some specific nodes further upstream. In order to eliminate the idle time that resources have to be held until a PathTear message

Length (Bytes)	Class=3	C-Type=3
Ipv4 Next/Previous Hop Address		
Logical Interface Handle		
TLVs		

Fig. 11. Format of the IF_ID RSVP_HOP IPv4 object.

is received, it is suggested that the ability to rapidly converge can be enhanced if states can be removed on certain error conditions. In order to facilitate this, a new Path_State_Removed flag was defined for the ERROR_SPEC object. This flag simply indicates that the node which is forwarding the error message has removed its associated Path state. If the node that sets the flag is not the session endpoint, then also a PathTear message should be generated so as to trigger the removal of the corresponding Path state in the downstream direction. New ERO and RRO subobjects were introduced in order to support the previously presented bidirectional LSP. Here the label field has the same format as the one used in the Generalized Label object, but upstream labels must be explicitly marked using an additional bit.

The Protection object was introduced as an optional object in charge of indicating the link related protection attributes of a requested LSP. This object is used to indicate if the requested LSP is the primary or secondary LSP, where the secondary LSP represents a backup of the primary LSP. The Protection object indicates also the desired protection of the link. The decision about the specific type of protection to be used is based on a local policy.

Another new object, called Admin_Status was introduced to indicate the administrative state of a LSP, or to request to the ingress node a change in the administrative state of an LSP. Multiple bits are used to indicate whether the LSP is in "listening mode" whether the LSP is "up" or "down" or whether the LSP is being deleted among other things. The detailed procedures associated with the Admin_Status object are presented in [57].

Another difference between GMPLS and normal MPLS is that in the MPLS case there is a one-to-one association of the control channel to the data channel. When this type of association exists no additional information is required to associate a specific LSP setup transaction with a particular data channel [52]. In GMPLS, there is no explicit association between control channel and data channel. In this case it is necessary to provide additional information in signalling to identify the particular channel being controlled. For this purpose two new IF_ID RSVP_HOP C-Types are presented for the RSVP_HOP object, one for IPv4 and one for IPv6. The format of this object is presented in Fig. 11. The Next/Previous Hop Address and the Logical Interface fields are used as they were originally defined for the RSVP_HOP object in [2]. The Type Length Value (TLV) field is defined in [52] as we will see in the next section.

Five different types were defined to be used within the type field of the TLV in RFC 3471, depending on the interface that is being used. The IF_ID RSVP_HOP is used when there is not a one-to-one association of a control channel to a

data channel. The interface specified in the TLV field of the IF_ID RSVP_HOP is used to identify the data channel that is associated with an LSP.

The separation of the control and data plane raises also the problem of correctly identifying an interface where an error happened. In order to support this, the IF_ID ERROR_SPEC C-Type object was introduced with the same Class-Num as the ERROR_SPEC object defined in [2]. This new object includes a TLV field also following the definition of TLVs presented in [52]. Two new faults are possible if the control channel is independent from the data channel. The first type of fault can occur if the ability of the neighbouring node to pass control messages becomes limited for example because a link failure occurred. The second type of fault can happen if the node's control plane is restarted and almost all the state configuration is lost. In order to handle these faults a new Restart_Cap object was defined to be used under the Hello message. It must be indicated in milliseconds how much time it takes for the node to restart the RSVP-TE component and the communication channel and how big the period of time is in which the sender of the object wants the receiver to resynchronize RSVP and GMPLS forwarding state with the sender.

The introduction of the Restart_Cap object alters also how Hello messages are processed. All the nodes that support state recovery have to advertise this capability by including the Restart_Cap object in all the exchanged Hello messages.

If a control channel fault is detected, then the Summary Refresh extension [42] can be used to refresh the shared state with the neighbour. For node failures a new Recovery_Label object was introduced. If a node fault is detected, then the Recovery_Label object is used in the recovery process. The format of this object is identical to the one of the Generalized Label object, with class number 34 and C-Type depending on the label that is being used. The detailed recovery process is presented in [56].

VII. RSVP-TE EXTENSIONS

Similar to RSVP, RSVP-TE was also altered over time, and a number of extensions were introduced for this expansion of RSVP as well. In this section we are going to present these enhancements of the protocol, but concentrating only on the original extension of RSVP-TE and not on extensions of RSVP-TE for GMPLS. Extensions for GMPLS either dealt with specific interface problems that might appear in GMPLS or copied some functionality already available in RSVP or RSVP-TE. In what follows we are going to present the extension that introduces support for unnumbered links in RSVP-TE, the fast re-route enhancement, the introduction of the TLV format, the exclude route addition, the aggregation extension of RSVP-TE, the enhancement of the protocol to support point-to-multipoint LSPs, the inter-domain RSVP-TE extension and the non-penultimate hop popping behaviour and out of band mapping enrichment.

A. RSVP-TE support for unnumbered links

MPLS-TE did not support unnumbered links that do not have an IP address (like for example PPP links). In order to

respond to this problem a new object and two subobjects were introduced.

The basic idea is that when an unnumbered link is discovered, the LSRs at each end of the link assign an identifier for that link [58]. The choice of the identifier has to be agreed with the Interior Gateway Protocol (IGP) used for that link (like IS-IS or OSPF). No apriori knowledge exists of the identifiers assigned by the LSRs at each end of the link. The LSRs exchange with each other the identifiers they assigned to the unnumbered link. Also, in order to uniquely identify an unnumbered link the term Router ID is used. In this context the Router ID means a stable IP address of an LSR, address that is reachable at any time as long as connectivity with the LSR exists. The Router ID is usually implemented by a loopback interface and has the advantage that the address does not become unusable if a link on that LSR goes down.

The unnumbered link becomes uniquely identified by the tuple composed of the LSR's Router ID and the interface ID associated with the interface where the unnumbered link exists. The object introduced to carry this information is named LSP_Tunnel_Interface_ID. This new object can appear in either a Path message or in a Resv message.

In order to specify the unnumbered links in ERO and RRO two new subobjects were introduced. Basically they use the Router ID and the Interface ID of the unnumbered link in order to uniquely identify that link.

B. RSVP-TE fast reroute extension

In order to allow LSP-TE tunnels to redirect traffic very fast (less than 100ms) in the event of a failure, RFC 4090 introduced the Fast Reroute Extension for RSVP-TE. The standard extended RSVP with the capability to establish backup LSP tunnels for the local repair of LSP tunnels. The document applies only to explicitly routed LSPs that are provided with protection. Two methods were proposed, the One-to-One Backup and the Facility Backup [59].

In the One-to-One backup, a LSP is constructed for each LSR on the path of the original LSP. These LSPs intersect the original LSP somewhere downstream the point of failure. Fig. 12 illustrates what are the LSPs constructed for a theoretical topology.

In the presented example the protected LSP runs from R1 to R4. Each of the LSR on the path can provide user traffic protection by creating a partial backup LSP that merges with the original path downstream the point of failure. This type of partial One-to-One backup LSP is referred to as a detour. In order to protect a LSP that crosses N nodes, there can be as much as $N-1$ detours

The Facility Backup works in a different way. In this case a single LSP is created to back-up a set of LSPs. The LSP created for this purpose is referred to as a bypass tunnel. The bypass tunnel path must intersect the original protected LSP somewhere downstream of the point of local repair (PLR). Where, the PLR represents the head-end LSR of a bypass tunnel or a detour. The set of LSPs that can be protected is constrained by the fact that all LSPs must share the PLR and the common downstream node. We present in Fig. 13 an example of the facility backup technique, as this is illustrated in [59].

Fig. 12. One-to-one backup.

Fig. 13. Facility Backup.

This method has the advantage of providing scalability improvements. The LSPs from R1 to R5, R8 to R4 and R2 to R9 can be protected, using the same bypass tunnel. However, as in the One-to-One backup case, in order to fully protect an LSP with N nodes there can be as many as N-1 bypass tunnels.

To implement the RSVP-TE fast reroute extension, two new objects and two new flags were defined for the SESSION_ATTRIBUTE and the RECORD_ROUTE objects. The two new objects are the FAST-REROUTE and DETOUR objects. We illustrate in Fig. 14 the format of these objects.

The FAST-REROUTE object is used to control the backup used for the protected LSP. This object includes most of the fields that were defined for the LSP_Tunnel_RA object. However, some additional fields are included as well. The bandwidth is used to indicate what is the necessary bandwidth for the backup LSP and is expressed in bytes/ second. The hop limit indicates the number of extra hops that the backup path is allowed to take.

Two new flags were defined for the FAST-REROUTE object. The One-to-One Backup Desired flag which requests protection using the One-to-One Backup method and the

Fig. 14. Format of the FAST-REROUTE and DETOUR objects.

Facility Backup Desired flag which requests protection using the Facility Backup method.

The DETOUR object is used for the One-to-One backup method in order to identify the detour LSPs. For each detour, this object specifies the IP addresses of the PLRs that represent the beginning of the detour, and the address of the node that is trying to be avoided.

Two new flags were introduced for the Session_Attribute object. The new flags are Bandwidth protection desired and Node protection desired. These flags are used to indicate to the PLR along a protected LSP that a backup path with a bandwidth guarantee and node protection is desired. The same flags were defined also for the RRO, in order to correctly report what are the parameters of the LSP. In the case of bandwidth guarantee, the bandwidth that has to be protected is the one that is required for the original LSP (if no FAST-REROUTE object is included in the message) or the one specified in the FAST-REROUTE object (if a FAST-REROUTE object is included).

The decision on whether an LSP should receive local protection as well as the parameters associated and the protection method used, is decided by the head-end LSR of that LSP. To indicate that the LSP needs local protection, the head-end LSR either sets the local protection desired flag of the SESSION_ATTRIBUTE object or includes a FAST-REROUTE object in the Path message.

C. Introduction of the Type Length Value

Another extension of RSVP-TE was standardized with the introduction of RFC 4420 [60]. The problem addressed by this extension is the fact that the flags which were defined for the SESSION_ATTRIBUTE object can only carry a limited amount of options (eight bits = eight options). In order to allow RSVP-TE to carry more attribute parameters and in order to make it easily extendable for new sets of requirements which might follow, new objects were introduced. This extension is equally applicable to both RSVP-TE for MPLS-TE and the extension of RSVP-TE for GMPLS. Two LSP attributes objects were defined in [60].

The format that is used for defining the new objects introduced is referred to as the Type Length Value (TLV) format.

The type field is used to identify the TLV. In [60] the Length field was described as indicating the length of the value field only. This was altered in a subsequent RFC to indicate the length of the whole TLV object [61]. The value field includes the data of the TLV padded.

Only one TLV type was defined in [60], identified by the Type 1 value and indicating the Attribute Flags TLV. The Attribute Flags TLV value represents an array of 32 flags. Two objects were defined on the basis of the Attribute Flags TLV in [60]. These are the LSP_Attributes object and the LSP_Required_Attributes object. The format of the two new introduced objects is identical, and the encoding used for these objects is the one specified by the Attribute Flags TLV. Both objects are used to signal attributes that are required in support of an LSP or to indicate the nature of the LSP. The difference between these objects is to what types of nodes they are addressed.

The LSP_Attributes object with a Class-Num value of 197 is of the form 11bb bbbb, thus ensuring that a LSR that does not recognize this object will forward it without any modifications. On the other hand, the LSP_Required_Attributes has a Class-Num of 67, which is of the form 0bbb bbbb. This will cause LSRs that do not recognize the object to reject the entire message, rejecting also the LSP setup. This distinction extends RSVP-TE capabilities and provides a good way to ensure that only the LSR that understand this object will examine it. Also, in order to maintain accurate route recording, a new subobject was defined within RRO. This subobject enables RRO to record the implemented attributes. The RRO attributes subobject has the standard format of the RRO subobjects, where the content stored is the same as the Attribute Flags TLV, registering a series of bit flags. A one to one correspondence exists between the bits from the Attribute Flags TLV and the ones in the RRO attributes subobject.

D. Aggregation of RSVP-TE over MPLS-TE/DS-TE tunnels

Similar to the aggregation of RSVP introduced in [43], RFC 4804 provides the means by which RSVP end-to-end reservations can be aggregated over MPLS-TE or MPLS DiffServ-aware Traffic Engineering (DS-TE) tunnels [62]. The approach is based on the procedures defined in [43] and just adapts the corresponding procedures for MPLS-TE tunnels instead of RSVP tunnels.

This approach has the advantage to be scalable in achieving admission control over a large number of flows. The scalability in this case is a consequence of the fact that the devices in the core of the network are managing only MPLS-TE tunnels and they are not aware of the RSVP flows. A number of similarities exist between aggregated RSVP reservations and TE tunnels. First, both TE and aggregated RSVP reservations are signalled using the RSVP protocol (or some extension of it). Second, both are controlled by intelligent devices on the edge of the aggregation core. Third, a TE tunnel is subject to admission control just like aggregated RSVP reservations.

The procedures involved in aggregating RSVP reservations over MPLS-TE tunnels are presented in [62], by exemplifying these operations over a TE pre-established tunnel. In what

follows we illustrate the model presented in [62] and the step-by-step actions involved in the process. The first step represents the arrival of the E2E Path message at the Aggregator, who attempts to map the E2E reservation to a TE tunnel. The mapping procedure takes into consideration the available routing information and the local policy information. Also the E2E RSVP reservation pre-emption priority and the MPLS-TE tunnel setup and hold priorities are taken in to account.

Next the Aggregator updates the ADSPEC object in the Path message of the E2E flow in order to indicate the impact of the MPLS-TE tunnel. After the update is done, the Path message is sent directly to the Degregator, by modifying the IP address in the IP header of the Path message. The Router Alert IP option is not set, and the RSVP protocol number is not changed. At the receipt of the Path message by intermediate nodes, this will be processed as normal IP traffic, since the Router IP alert option is not set, and the IP address is that of the Degregator.

At the receipt of the E2E Path message by the Degregator, normal RSVP processing will take place since the protocol number indicates RSVP. The Degregator then forwards the E2E Path message towards the receiver by modifying the destination address of the IP header with the address found in the session object. Also the Router Alert option is now set. Upon receiving the Path message the receiver performs normal RSVP operations and sends back a Resv message hop-by-hop towards the sender. The Degregator receives the Resv message of the E2E flow and performs normal RSVP operations. This means that the Resv message is sent directly to the Aggregator, since this is the PHOP signalled in the Path message. Similar to traditional Resv RSVP procedures, the Router Alert option is not set in this case. This in turn means that the Resv message is hidden from intermediate nodes, which handle it as a normal IP packet.

Upon receiving the Resv message the Aggregator checks the availability of the resources in the indicated TE tunnel. If enough resources exist the reservation is mapped on the tunnel and the Aggregator must update the internal understanding of how much of the TE tunnel is used. If no resources are available, normal RSVP procedures are followed and a Resv Error is sent towards the receiver.

Now, on receiving traffic that belongs to a mapped E2E reservation over a TE tunnel, the traffic is encapsulated by the Aggregator in that TE tunnel. The E2E reservation can be removed in this case by following normal procedures like PathTear and ResvTear messages or as a result of a timeout or an error condition that appeared.

E. Exclude Routes, extension to RSVP-TE

The original RSVP-TE extension for MPLS, and the extension of RSVP-TE for GMPLS allow abstract nodes to be included in the setup. However, in [63] it is argued that some systems might find it useful to explicitly exclude some abstract nodes and resources from the path setup as well. RFC 4874 provides the means to do this by introducing a new object and subobject. As an example, the exclude route extension could be helpful in the case when two non-overlapping LSPs are required between two nodes [63]. In this scenario, a possibility

will be to construct the first LSP using the route recording object. The second LSP is constructed afterwards, excluding the nodes from the first path.

The new introduced object is called Exclude_Route object (XRO), and it can be used to exclude a set of abstract nodes on the whole path. The new introduced subobject, Explicit Exclusion Route Subobject (EXRS), was designed to be used in the ERO, and indicates the exclusion of certain abstract nodes or resources between a defined pair of abstract nodes that are present in ERO.

The knowledge of the link that constitutes a shared link group (SRLG) as this is defined in [64] may be used in this case to be excluded from the path of two abstract nodes. The XRO represents a container for a series of variable set of subobjects. Five types of subobjects were defined in [63]. These subobjects are the IPv4 prefix, the IPv6 prefix, the Autonomous System number, the Unnumbered Interface ID and the SRLG subobject. The first three subobjects have the same format as the one indicated for the corresponding ERO subobjects defined in [51].

The Unnumbered Interface ID subobject has the same format as the one presented in [58]. The SRLG subobject is a new subobject defined by RFC 4874. However this is formatted as a general ERO subobject defined in [51]. The difference is that, for example with the IPv4 prefix subobject instead of indicating the IP address, a 32-bit identifier of the SRLG is used.

An additional bit must be used to indicate if an abstract node must be excluded (when value is 0) or if this should be avoided (value is 1).

This extension modifies some of the procession rules used for establishing explicit routes as these are defined in [51]. Accordingly, when a node receives a Path message, it must check if it is not a member of an abstract node specified in the XRO. If the node is a member of any of the abstract nodes from XRO, and if the bit is set to zero, then a PathErr message should be returned with the error value Local node in Excluded Route.

The subobjects specified in XRO should not contradict the ones present in the ERO. However if a Path message is received with contradicting subobjects, then the XRO subobject that has the bit not set takes precedence over the ERO subobjects and such Path message should be rejected sending a PathErr message with the error value Routing Blocked by Exclude route.

Subobjects in the XRO with the bit set do not take precedence over ERO subobjects, and an implementation can choose to either reject such a Path message or to continue the setup of the LSP.

The Class-Num of the XRO is of the form 11bb bbbb, meaning that nodes that do not support XRO will forward the object without inspecting it.

The EXRS is ERO subobject and must not be included in the XRO. The EXRS is used to indicate abstract nodes or resources (links, unnumbered interfaces or labels) that should not be used between two successive abstract nodes in the explicit route. Just like XRO, the EXRS represents a vessel that can include one or more EXRS subobjects, where the format of these sub-subobjects is exactly the same as the ones

specified for the XRO subobjects. All the previously presented subobjects for XRO can be included in an EXRS.

If an LSR finds both XRO and EXRS in the ERO, it should exclude all the routes indicated by the XRO and EXRS. If elements appear in both lists then the stricter exclusion rule should apply.

F. Extension of RSVP-TE to support P2MP

The TE extension for RSVP introduced in [51] and GMPLS support for RSVP-TE in [56] defined mechanisms for setting point-to-point (P2P) TE LSPs but did not provide specifications on how these mechanisms can be used for establishing Point-to-Multipoint (P2MP) LSPs.

This support was defined later with the introduction of RFC 4875. A P2MP LSP is comprised of multiple source-to-leaf (S2L) sub-LSPs which are set up between the ingress and egress LSRs [65]. In turn a P2MP TE Tunnel can be composed of one or more P2MP LSPs. Overall a P2MP TE Tunnel is classified by using a new SESSION object, the P2MP_LSP_Tunnel_IPv4 object. Further, a P2MP LSP can be uniquely identified through the new introduced SESSION object coupled with a new P2MP_SENDER_TEMPLATE object. Additionally, in order to uniquely distinguish a S2L sub-LSP a new S2L_SUB_LSP object was introduced in [65]. The only information that is provided by this object is the S2L Sub-LSP destination address. Using this address together with the P2MP_SENDER_TEMPLATE object and the P2MP_SESSION object, an S2L sub-LSP can be uniquely recognized in the context of a P2MP TE Tunnel.

The extension of RSVP to support P2MP LSP has to be reflected also in the explicit route functionality of RSVP-TE. In order to implement this, a new object was introduced, the P2MP_Secondary_Explicit_Route object (SERO). SERO was defined as identical to ERO, but using a new C-Type (value 2). Accordingly, in order to support the Route recording functionality, the P2MP_Secondary_Record_Route Object (SRRO) was introduced. This is identical to RRO but uses a new C-Type (value 2). Both SERO and SRRO use the same subobjects that are defined for ERO and respectively RRO.

The path of each S2L sub_LSP is encoded in a SERO. Each S2L sub-LSP is represented by the tuple composed of the SERO and the S2L_SUB_LSP object. Only the path from a branch LSR to the egress LSR of an S2L sub_LSP is included in the SERO. The absence of an SERO must be interpreted as requiring normal RSVP hop-by-hop routing for that particular sub-LSP.

G. Inter-domain MPLS and GMPLS-TE extensions for RSVP-TE

When network operators started to use MPLS-TE more and more, it was desired to extend the capabilities of the mechanism for inter-area traffic engineering. Because the original design of MPLS was limited to a single Interior Gateway Protocol (IGP) area, some expansion of the three main components of the MPLS-TE control plane was needed [66]. These components are: the routing component, which is assured by the extension of the link state IGP (ISIS-TE

and OSPF-TE), the path computation component which depending on the TE topology and LSP constraint calculates the placement of the LSP, and the signaling component (ensured by RSVP-TE) which takes care of the label distribution and resource reservation.

The difficulty of extending the applicability of MPLS-TE to inter-area traffic engineering comes more from the routing and path computation components than from the signalling component [66].

The problem is caused by the desire to maintain the IGP hierarchy concept, which limits topology visibility of the head-end LSR to a single IGP area. This limitation makes the computation of the end-to-end path not feasible. In what follows we are going to present only the required extension for the signalling component. The extensions for the routing and path computation components are outside the scope of this paper. More information on the requirements and framework proposed for Inter-area MPLS-TE can be found in [67] [64] [66] [68].

In what concerns the signalling component, three different methods have been identified, that can be used for RSVP-TE signalling and LSP setup. The first method represents the establishment of a contiguous LSP, or normal LSP that can be achieved by following procedures presented in [51], or in [56]. This method creates one end-to-end LSP that crosses all the areas (or a part of them).

The second method is the nested LSP, which can be implemented by using the procedures presented in [69]. The last method is the Stitched LSP which means that the end-to-end LSP is constructed of a concatenation of distinct LSPs, where each individual LSP spreads over a domain. The procedures that are used for the Stitched LSP are presented in [70]. The area border routers are the ones that restrict the utilization of the signalling method used. The main constraints that apply here are the policies enforced in that domain. An operator may choose to allow only LSP stitching because it gives the operator the chance to re-optimize the domain under its own control.

In network environments where policies decide which inter-area LSP should be applied, no extension of the RSVP-TE protocol is needed. However, in cases where more than one method for inter-area LSP setup is supported, the ingress node must be able to constrain the choice made by the domain border nodes.

In order to do this, a new bit was defined in [68] that can be set in the LSP_Attribute object. This bit was defined in the Attributes Flags TLV and can be set in order to restrict intermediate nodes to install contiguous LSP.

H. RSVP message format for LSP attributes objects

IETF introduces in RFC 6510 [71] a clarification statement that specifies how LSP_Attributes introduced in [61] can be carried in RSVP Resv and Path messages. Also some clarifications were introduced concerning the processing of the LSP attributes object in the case of the P2MP extension introduced in [65]. Only the formats of the Path and Resv messages were affected by the LSP attributes object.

For example, in the Path message, the LSP_Attributes and LSP_Required_Attributes objects are to be placed immediately

after the Session_Attribute object or after the Label_Request object if the Session_Attribute is not present.

If an LSR wishes to report the operational status of an individual S2L sub_LSP, the LSP attributes object has to be included in the S2L sub_LSP flow descriptor after the S2L_SUB_LSP object.

If an LSP_Attributes object is present before the first S2L_SUB_LSP object, the LSP_Attributes represents the operational status of all the S2L sub_LSP identifiers in the message. Further specifications on the actual format of the messages are presented in [71].

I. Non Penultimate Hop Popping Behavior and out-of-band Mapping for RSVP-TE LSPs

The introduction of Multicast Virtual Private Network (MVPN) in [72] and the Virtual Private LAN Service in [73] has triggered specific requirements for RSVP-TE. For these types of services, the egress LSR receives a binding of the LSP to a specific application by using an out-of-band (OOB) mechanism like the Border Gateway Protocol (BGP). Until this information is received, the egress LSR cannot make correct forwarding decisions. And even if this information is available the egress LSR must know which incoming LSP will be used. In this case a Non-Penultimate Hop Popping (Non-PHP) behaviour is requested in order to apply the OOB binding mechanism. The Non-PHP behaviour means that the egress LSRs have to assign a non-Null label to the LSP that is being signalled. This capability is needed, for example by a leaf LSR in a P2MP LSPs scenario where LSPs are used to carry IP multicast traffic in order to identify the P2MP LSP on which traffic is received.

Using the LSP_Attributes object defined in [61] two flags were defined by [74]. The Non-PHP behaviour flag and the OOB mapping flag were introduced. The Non-PHP behaviour flag, which is set by the ingress LSR, is used to request Non-PHP for an RSVP-TE LSP and must be ignored by all LSRs except the egress LSR. When an egress LSR recognizes the LSP attributes object and the Non-PHP behaviour flag in the Attribute Flags TLV, it must allocate a non-Null local label to that LSP. The OOB mapping flag is used by the ingress LSR to signal to the egress LSR that the binding of an RSVP-TE LSP to an application and payload identifier is being signalled out of band [74].

VIII. OTHER RSVP EXTENSIONS

The previously presented extensions of RSVP introduced by IETF do not represent the full spectrum of addendums suggested over time for RSVP. The research community has also been very active in this field and a lot of research papers have been published over time focusing either on analysing the functionality of different elements in RSVP or by proposing multiple other RSVP extensions or improvements. The efforts of the research community can be categorized in three distinct research areas: scalability improvements, reservation efficiency and mobility enhancements. In this section we will present in brief some of the propositions regarding RSVP in each of the three areas.

A. Scalability Proposals

One of the specific problems of RSVP is still the scalability of the protocol. Besides the extensions introduced by IETF, other extensions have been suggested by the research community. Each of these extensions is focusing on a specific part of RSVP. For example in [75] and [76] an alteration of the One Pass With Advertising (OPWA) principle used by RSVP was proposed. The motivation and the suggestions of the two papers differ. If in [75] a two-pass with advertising mechanism is presented in order to solve the killer reservation problems (presented in Section II), in [76] a one-pass reservation model was illustrated with the goal to reduce the signalling overhead.

In [77] the soft state approach is improved by introducing timer values that adapt dynamically to available bandwidth and to the amount of control traffic on a link [77]. The interaction between RSVP timing parameters and network performance is treated in [78] as a multi-objective optimization problem. Insights about the relative importance of different timing parameters are presented in [78]. Similarly, based on adaptive timers, the extension proposed in [79] uses a feedback mechanism coupled with dynamic timers in order to reduce the signalling overhead. A different approach that also, is concentrated on the reduction of the overhead introduced by RSVP is presented in [80]. In this scenario, as opposed to soft state per flow used by RSVP a soft state per neighbour was proposed. Here, control messages have to be generated at a fixed rate, independent of the number of flows, alienating in this way the scalability problem [80].

A more drastic approach in offering a scalable version of RSVP was presented in [81]. Here a light-weight RSVP for generic signalling was recommended. Basically [81] suggested a new version of RSVP that is more light-weight and where the signalled data is separated by the signalling messages and the multicast capability is removed [81]. The signalling messages in this new RSVP flavour represent a scaled down version of the original Path/Resv messages. For example the Path message that is used in this context captures only a part of the functionality of the original Path message, while the actual signalled data is to be defined by separate protocols (client protocols). The distinction here between signalled data and signalling messages is important because [81] did not proposed only a scalable version of RSVP but also a generic signalling protocol.

In [82] the scalability of the RSVP-TE is improved not by proposing an alteration of the protocol itself, but by suggesting a different architecture for next generation routers. The basic idea is that the scalability, resilience and robustness of the protocol can be improved by off-loading the current components of the RSVP-TE module into the line cards [82].

B. Reservation Efficiency

Other researchers are concentrating on improving the TCP performance over RSVP. One of the problems of TCP working over RSVP is that RSVP is unidirectional and it cannot provide any type of protection for the ACK packets on the reverse path of the reservations. Example of such suggestions can be found in [83] and [84] where symmetrical and respectively asymmetrical RSVP bi-directional resource reservation enhancements

were recommended.

Some proposals are not concentrating on RSVP operations overall, but treat only specific types of reservations. For example [85] discussed the efficiency of semi-elastic reservations. It is presented in [85] that it can be beneficial if the sender (server) would be capable to directly modify or release a reservation. In normal RSVP operations the server cannot make changes directly, but has to indicate the changes first to the receivers. These receivers will finally modify or change the reservations. The extensions needed in order to allow the sender to do direct reallocation of the bandwidth and a performance evaluation are presented in [85].

In [86] an extension of RSVP was proposed that allows better coexistence of sensitive video traffic and elastic traffic. Sensitive traffic, which requires very strict QoS guarantees, leads to a non-optimal use of network resources [86]. Basically [86] advocated that it can be beneficial to dynamically change the reservations based on the current demand for network resources. The suggested extension is applicable for aggregated traffic flows but also for individual streams [86].

C. Mobility Enhancements

A completely different area that is not covered by IETF is the mobility support of RSVP. The fact that mobile IP was chosen as the de facto management mechanism for wireless LAN and advanced cellular networks, although in its basic form it does not provide QoS guarantees, has triggered the development of many RSVP extensions that provide mobility support [87]. Some of these extensions are: Mobile RSVP (MRSVP) [88], the Hierarchical Mobile RSVP (HMRSVP) [89], the RSVP Mobility Proxy (RSVP-MP) [90], the Location RSVP [91] or the On Board RSVP presented in [92]. These however represent only a small part of the proposals that enhance the capability of RSVP for mobility support. Other papers that are focused on RSVP mobility extensions can be found in [93]–[96]. It is out of the scope of this paper to analyse all RSVP mobility extensions introduced over time. A survey of the different enhancements of RSVP that support mobility is presented in [87].

IX. CONCLUDING REMARKS

This paper presented a survey on the evolution of RSVP as the main resource reservation protocol in the Internet. It illustrated, starting from the basic RSVP design up to the RSVP-TE extension and subsequent modifications, the signalling involved and the messages exchanged. For each extension, the motivation which triggered this extension and in what way it altered the original design of RSVP was presented.

RSVP has evolved a long way since the standardization of the protocol in [2]. As we have presented throughout this paper, multiple extensions have been proposed or adopted over time to make RSVP interoperable with a large number of technologies.

RSVP proves to be very up to date, receiving a lot of attention from IETF. The format of RSVP has demonstrated to be extremely flexible, allowing the definition of new functionality within the protocol in a very modular way. Although not explicitly stated, this format is built in a hierarchal way, which

allowed the additions introduced over time to focus exactly on the functionality that had to be introduced. The functional specifications of RSVP allow the introduction of new capabilities by defining new messages, new objects (using either new Class-Num or C-Types values), new subobjects or by using new policy elements. The flexibility and adaptability of RSVP is illustrated also by multiple extensions that introduced additional interfaces.

RSVP has proven to be extremely significant with respect to QoS. The importance of RSVP in the QoS field is demonstrated by the introduced additions of the protocol that allow it to be used with all the major QoS mechanisms. Also, the widely adopted RSVP extensions for MPLS and GMPLS have proven the importance of RSVP for resource reservations. Unfortunately no thorough evaluation exists of the actual use of RSVP in the Internet. This is mainly due to the fact that ISPs are very reluctant to reveal any details about the technologies that are implemented in their networks. Further research that deals with this problem will help not only a better understanding of the usability of RSVP but also the applicability of all the QoS mechanisms in general.

One of the main concerns regarding RSVP is its scalability problem. In this respect the IETF introduced a number of alterations of RSVP with the aim to reduce the processing overhead caused by the protocol. This paper presented the modifications introduced by these extensions. Further research should concentrate on a scalability analysis that compares the processing overhead introduced by each of the presented RSVP formats.

Judging from a CISCO forecast published in 2012 [97] which specifies that the Internet traffic is expected to grow threefold from 2011 to 2016, and that new QoS sensitive applications will represent an important part of this growth, it is not unlikely that the significance of RSVP will grow even more. The advantages presented by RSVP in term of resource reservation and QoS are clear "it provides guaranteed quality of service and supports a wide range of services" [11]. However the scalability and complexity of RSVP still remain the main issues of the protocol. In this scenario where sensitive traffic will grow considerably it is expected that other extensions of the protocol that treat scalability will appear in the future. On the other hand taking into consideration the fast pace in which CPU power and memory become more and more affordable it is possible that the advantages offered by RSVP will outweigh the disadvantages.

A closer look at all the papers that analyse or evaluate RSVP reveals the fact that none of these is able to offer a quantifiable measure of the scalability problem of RSVP. In other words, what is exactly defined as scalable for the RSVP resource reservation, what is the allowed increase of memory and CPU for an additional flow in order to be able to label RSVP as scalable. This lack of measurability of scalability represents an open research question.

As demand for mobile data applications grows dramatically and more users rely on mobile devices for connecting to the Internet, it is highly probable that a lot of research effort will be concentrated on mobility enhancements in the future.

To conclude we can say that the large number of extensions, resulting in a growing applicability and functionality, tightly

linked with the modularity and flexibility of this protocol, make RSVP a very important protocol for resource reservation.

REFERENCES

- [1] L. Delgrossi and L. Berger, "Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+," RFC 1819 (Experimental), Internet Engineering Task Force, August 1995. [Online]. Available: <http://www.ietf.org/rfc/rfc1819.txt>
- [2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification," RFC 2205 (Proposed Standard), Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2205.txt>
- [3] P. Pan and H. Schulzrinne, "YESSIR: a simple reservation mechanism for the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 89-101, Apr 1999. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=505733.505740>
- [4] G. Fehér, K. Németh, M. Maliosz, I. Cselényi, J. Bergkvist, D. Ahlrad, and T. Engborg, "Boomerang - A Simple Protocol for Resource Reservation in IP Networks," Vancouver, Canada, June 1999. [Online]. Available: <http://boomerang.ttt.bme.hu/rtas99.html>
- [5] D. Vali, S. Paskalis, L. Merakos, and A. Kaloxylas, "A survey of internet QoS signaling," *IEEE Communications Surveys & Tutorials*, vol. 6, no. 4, pp. 32-43, 2004. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5342297>
- [6] J. Manner and X. Fu, "Analysis of Existing Quality-of-Service Signaling Protocols," RFC 4094 (Informational), Internet Engineering Task Force, May 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4094.txt>
- [7] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang, "An architectural comparison of ST-II and RSVP," in *INFOCOM '94. Networking for Global Communications., 13th Proc. IEEE*, Jun 1994, pp. 716 -725 vol.2.
- [8] X. Xiao and L. Ni, "Internet QoS: a big picture," *Network, IEEE*, vol. 13, no. 2, pp. 8 -18, Mar 1999.
- [9] W. Zhao, D. Olshefski, and H. Schulzrinne, "Internet Quality of Service: an Overview," Tech. Rep., 2000.
- [10] A. Meddeb, "Internet QoS: Pieces of the puzzle," *IEEE Commun. Mag.*, vol. 48, no. 1, pp. 86-94, Jan 2010. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5394035>
- [11] P. Flavius and P. Ferdi, "Internet Service Delivery Models: Evolution and Current Issues," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*. IEEE, Oct. 2011, pp. 146 -153.
- [12] P. White and J. Crowcroft, "The integrated services in the Internet: state of the art," *Proc. IEEE*, vol. 85, no. 12, pp. 1934 -1946, Dec 1997.
- [13] S. Shenker and L. Breslau, "Two issues in reservation establishment," in *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communication*, ser. SIGCOMM '95. New York, NY, USA: ACM Press, 1995, pp. 14-26.
- [14] F. Baker, B. Lindell, and M. Talwar, "RSVP Cryptographic Authentication," RFC 2747 (Proposed Standard), Internet Engineering Task Force, January 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2747.txt>
- [15] S. Herzog, "RSVP Extensions for Policy Control," RFC 2750 (Proposed Standard), Internet Engineering Task Force, January 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2750.txt>
- [16] R. Braden and L. Zhang, "Resource ReSerVation Protocol (RSVP) - Version 1 Message Processing Rules," Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2209.txt>
- [17] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210 (Proposed Standard), Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2210.txt>
- [18] S. Shenker and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements," RFC 2215 (Proposed Standard), Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2215.txt>
- [19] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211 (Proposed Standard), Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2211.txt>
- [20] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," RFC 2212 (Proposed Standard), Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2212.txt>
- [21] A. Terzis, B. Braden, S. Vincent, and L. Zhang, "RSVP Diagnostic Messages," RFC 2745 (Proposed Standard), Internet Engineering Task Force, January 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2745.txt>

- [22] A. Terzis, J. Krawczyk, J. Wroclawski, and L. Zhang, "RSVP Operation Over IP Tunnels," RFC 2746 (Proposed Standard), Internet Engineering Task Force, January 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2746.txt>
- [23] J. Zhi, C.-H. Lung, X. Xu, A. Srinivasan, and Y. Lei, "Securing RSVP and RSVP-TE signaling protocols and their performance study," in *Information Technology: Research and Education, 2005. ITRE 2005. 3rd International Conference on*, June 2005, pp. 90 – 94.
- [24] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, and S. Herzog, "Identity Representation for RSVP," RFC 2752 (Proposed Standard), Internet Engineering Task Force, January 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2752.txt>
- [25] S. Herzog, "Signaled Preemption Priority Policy Element," RFC 3181 (Proposed Standard), Internet Engineering Task Force, October 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3181.txt>
- [26] L.-N. Hamer, B. Gage, B. Kosinski, and H. Shieh, "Session Authorization Policy Element," RFC 3520 (Proposed Standard), Internet Engineering Task Force, April 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3520.txt>
- [27] F. Le Faucheur, J. Polk, and K. Carlberg, "RSVP Extensions for Admission Priority," RFC 6401 (Proposed Standard), Internet Engineering Task Force, October 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6401.txt>
- [28] F. Le Faucheur and W. Lai, "Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering," RFC 4125 (Experimental), Internet Engineering Task Force, June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4125.txt>
- [29] F. L. Faucheur, "Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering," RFC 4127 (Experimental), Internet Engineering Task Force, June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4127.txt>
- [30] J. Ash, "Max Allocation with Reservation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering & Performance Comparisons," RFC 4126 (Experimental), Internet Engineering Task Force, June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4126.txt>
- [31] T. Shan and O. Yang, "Bandwidth Management for Supporting Differentiated Service Aware Traffic Engineering," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 9, pp. 1320 – 1331, Sep 2007. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4288130>
- [32] D. Zhang and D. Ionescu, "QoS Performance Analysis in Deployment of DiffServ-aware MPLS Traffic Engineering," *IEEE*, Jul 2007, pp. 963–967. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4287988>
- [33] S. Herzog, J. Boyle, R. Cohen, D. Durham, R. Rajan, and A. Sastry, "COPS usage for RSVP," RFC 2749 (Proposed Standard), Internet Engineering Task Force, January 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2749.txt>
- [34] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, S. Herzog, and R. Hess, "Identity Representation for RSVP," RFC 3182 (Proposed Standard), Internet Engineering Task Force, October 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3182.txt>
- [35] F. Le Faucheur, J. Manner, A. Narayanan, A. Guillo, and H. Malik, "Resource Reservation Protocol (RSVP) Extensions for Path-Triggered RSVP Receiver Proxy," RFC 5946 (Proposed Standard), Internet Engineering Task Force, October 2010. [Online]. Available: <http://www.ietf.org/rfc/rfc5946.txt>
- [36] L. Berger and T. O'Malley, "RSVP Extensions for IPSEC Data Flows," RFC 2207 (Proposed Standard), Internet Engineering Task Force, September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2207.txt>
- [37] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer, "SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks," RFC 2814 (Proposed Standard), Internet Engineering Task Force, May 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2814.txt>
- [38] Y. Bernet, "Format of the RSVP DCLASS Object," RFC 2996 (Proposed Standard), Internet Engineering Task Force, November 2000. [Online]. Available: <http://www.ietf.org/rfc/rfc2996.txt>
- [39] R. Atkinson, "IP Authentication Header," RFC 1826 (Proposed Standard), Internet Engineering Task Force, August 1995. [Online]. Available: <http://www.ietf.org/rfc/rfc1826.txt>
- [40] —, "IP Encapsulating Security Payload (ESP)," RFC 1827 (Proposed Standard), Internet Engineering Task Force, August 1995. [Online]. Available: <http://www.ietf.org/rfc/rfc1827.txt>
- [41] T. Chiueh, A. Neogi, and P. Stirpe, "Performance analysis of an RSVP-capable router," in *Real-Time Technology and Applications Symposium, 1998. Proc.. Fourth IEEE*, Jun 1998, pp. 39 – 48.
- [42] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini, "RSVP Refresh Overhead Reduction Extensions," RFC 2961 (Proposed Standard), Internet Engineering Task Force, April 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc2961.txt>
- [43] F. Baker, C. Iturralde, F. L. Faucheur, and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations," RFC 3175 (Proposed Standard), Internet Engineering Task Force, September 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3175.txt>
- [44] F. L. Faucheur, B. Davie, P. Bose, C. Christou, and M. Davenport, "Generic Aggregate Resource ReSerVation Protocol (RSVP) Reservations," RFC 4860 (Proposed Standard), Internet Engineering Task Force, May 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4860.txt>
- [45] F. Tommasi, S. Molendini, and S. Zaccchino, "Measurements of the performance of the RSVP protocol," in *Workshop on Architectures for Quality of Service in the Internet, 2003. Proc.*, ser. Art-QoS, 2003, pp. 24–25.
- [46] M. Karsten, "Collected experience from implementing RSVP," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 4, pp. 767–778, Aug 2006.
- [47] I. Sebuktekin, J. Haluska, D. Chee, J. Giacomelli, Y.-Z. Lee, K. Adams, and J. Pulliam, "Aggregate RSVP implementation experience and performance analysis for applicability in tactical IP networks," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*. IEEE, Nov 2008, pp. 1 –7. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4753478>
- [48] C. Adams, P. Sylvester, M. Zolotarev, and R. Zuccherato, "Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols," RFC 3029 (Experimental), Internet Engineering Task Force, February 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3029.txt>
- [49] D. Black, S. Brim, B. Carpenter, and F. L. Faucheur, "Per Hop Behavior Identification Codes," RFC 3140 (Proposed Standard), Internet Engineering Task Force, June 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3140.txt>
- [50] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031 (Proposed Standard), Internet Engineering Task Force, January 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3031.txt>
- [51] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209 (Proposed Standard), Internet Engineering Task Force, December 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3209.txt>
- [52] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description," RFC 3471 (Proposed Standard), Internet Engineering Task Force, January 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3471.txt>
- [53] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702 (Informational), Internet Engineering Task Force, September 1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2702.txt>
- [54] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering," RFC 3272 (Informational), Internet Engineering Task Force, May 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3272.txt>
- [55] Y. W. Lee, S. Kim, J. Park, and S. H. Kim, "A lightweight implementation of RSVP-TE protocol for MPLS-TE signaling," *Computer Communications*, vol. 30, no. 6, pp. 1199–1204, Mar 2007. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0140366406004646>
- [56] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions," RFC 3473 (Proposed Standard), Internet Engineering Task Force, January 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3473.txt>
- [57] P. Ashwood-Smith and L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions," RFC 3472 (Proposed Standard), Internet Engineering Task Force, January 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3472.txt>
- [58] Y. Kompella, K. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)," RFC 3477 (Proposed Standard), Internet Engineering Task Force, January 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3477.txt>
- [59] P. Pan, G. Swallow, and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels," RFC 4090 (Proposed Standard), Internet Engineering Task Force, May 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4090.txt>
- [60] A. Farrel, D. Papadimitriou, J.-P. Vasseur, and A. Ayyangar, "Encoding of Attributes for Multiprotocol Label Switching (MPLS) Label

- Switched Path (LSP) Establishment Using Resource Reservation Protocol-Traffic Engineering (RSVP-TE)," RFC 4420 (Proposed Standard), Internet Engineering Task Force, February 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4420.txt>
- [61] A. Farrel, D. Papadimitriou, J. Vasseur, and A. Ayyangars, "Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE)," RFC 5420 (Proposed Standard), Internet Engineering Task Force, February 2009. [Online]. Available: <http://www.ietf.org/rfc/rfc5420.txt>
- [62] F. L. Faucheur, "Aggregation of Resource ReSerVation Protocol (RSVP) Reservations over MPLS TE/DS-TE Tunnels," RFC 4804 (Proposed Standard), Internet Engineering Task Force, February 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4804.txt>
- [63] C. Lee, A. Farrel, and S. D. Cnodder, "Exclude Routes - Extension to Resource Reservation Protocol-Traffic Engineering (RSVP-TE)," RFC 4874 (Proposed Standard), Internet Engineering Task Force, April 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4874.txt>
- [64] R. Zhang and J.-P. Vasseur, "MPLS Inter-Autonomous System (AS) Traffic Engineering (TE) Requirements," RFC 4216 (Informational), Internet Engineering Task Force, November 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4216.txt>
- [65] R. Aggarwal, D. Papadimitriou, and S. Yasukawa, "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)," RFC 4875 (Proposed Standard), Internet Engineering Task Force, May 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4875.txt>
- [66] J.-L. L. Roux, J.-P. Vasseur, and J. Boyle, "Requirements for Inter-Area MPLS Traffic Engineering," RFC 4105 (Informational), Internet Engineering Task Force, June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4105.txt>
- [67] A. Farrel, J.-P. Vasseur, and A. Ayyangar, "A Framework for Inter-Domain Multiprotocol Label Switching Traffic Engineering," RFC 4726 (Informational), Internet Engineering Task Force, November 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4726.txt>
- [68] A. Farrel, A. Ayyangar, and J. Vasseur, "Inter-Domain MPLS and GMPLS Traffic Engineering - Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions," RFC 5151 (Proposed Standard), Internet Engineering Task Force, February 2008. [Online]. Available: <http://www.ietf.org/rfc/rfc5151.txt>
- [69] K. Kompella and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)," RFC 4206 (Proposed Standard), Internet Engineering Task Force, October 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4206.txt>
- [70] A. Ayyangar, K. Kompella, J. Vasseur, and A. Farrel, "Label Switched Path Stitching with Generalized Multiprotocol Label Switching Traffic Engineering (GMPLS TE)," RFC 5150 (Proposed Standard), Internet Engineering Task Force, February 2008. [Online]. Available: <http://www.ietf.org/rfc/rfc5150.txt>
- [71] G. Berger, L. Swallow, "Resource Reservation Protocol (RSVP) Message Formats for Label Switched Path (LSP) Attributes Objects," RFC 6510 (Proposed Standard), Internet Engineering Task Force, February 2012. [Online]. Available: <http://www.ietf.org/rfc/rfc6510.txt>
- [72] E. Rosen and R. Aggarwal, "Multicast in MPLS/BGP IP VPNs," RFC 6513 (Proposed Standard), Internet Engineering Task Force, February 2012. [Online]. Available: <http://www.ietf.org/rfc/rfc6513.txt>
- [73] K. Kompella and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling," RFC 4761 (Proposed Standard), Internet Engineering Task Force, January 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4761.txt>
- [74] Z. Ali, G. Swallow, and R. Aggarwal, "Non-Penultimate Hop Popping Behavior and Out-of-Band Mapping for RSVP-TE Label Switched Paths," RFC 6511 (Proposed Standard), Internet Engineering Task Force, February 2012. [Online]. Available: <http://www.ietf.org/rfc/rfc6511.txt>
- [75] T.-L. Sheu and G.-Y. Pao, "A bandwidth allocation model for a two-pass RSVP setup mechanism," *Computer Communications*, vol. 26, no. 14, pp. 1662–1672, Sep. 2003.
- [76] M. Karsten, "Experimental Extensions to RSVP - Remote Client and One-Pass Signalling," in *Proc. of the 9th International Workshop on Quality of Service*, ser. IWQoS '01. London, UK: Springer-Verlag, 2001, pp. 269–274.
- [77] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson, "Scalable timers for soft state protocols," in *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, *Proc. IEEE*, vol. 1, Apr 1997, pp. 222–229 vol.1.
- [78] O. Komolafe and J. Sventek, "RSVP performance evaluation using multi-objective evolutionary optimisation," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proc. IEEE*, vol. 4, March 2005, pp. 2447–2457 vol. 4.
- [79] S. Hwang and B. Yoon, "An adaptive and dynamic timer design to maintain soft state in RSVP," in *Parallel and Distributed Systems, 2001. ICPADS 2001. Proc.. Eighth International Conference on*. IEEE Comput. Soc, 2001, pp. 774–779. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=934897>
- [80] L. Mathy, D. Hutchison, S. Schmid, and S. Simpson, "A performance study of RSVP with proposed extensions," *Computer Communications*, vol. 25, no. 18, pp. 1782–1798, Dec. 2002.
- [81] X. Fu and C. Kappler, "Towards RSVP Lite: light-weight RSVP for generic signaling," in *Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on*, March 2003, pp. 619–622.
- [82] K.-K. Nguyen and B. Jaumard, "A distributed and scalable RSVP-TE architecture for next generation IP routers," *IEEE*, Jun 2009, pp. 1–6. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5307434>
- [83] L. Zhang and Y. Hao, "An Asymmetry Bi-directional RSVP for improving the network performance," in *Computer Science and Service System (C3SS), 2011 International Conference on*. IEEE, Jun 2011, pp. 170–173. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5972035>
- [84] L. Jun, Y. ZhiHong, and L. Zhenming, "Bi-directional resource reservation for TCP performance improvement," in *Info-tech and Info-net, 2001. Proc.. ICII 2001 - Beijing. 2001 International Conferences on*, vol. 2, 2001, pp. 373–377 vol.2.
- [85] M. Postigo-Boix and J. L. Melús-Moreno, "Performance evaluation of rsvp extensions for a guaranteed delivery scenario," *Computer Communications*, vol. 30, no. 9, pp. 2113–2121, Jun 2007. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S014036640700182X>
- [86] R. Choderek and M. Leszczuk, "QoE validation of a RSVP protocol extension enabling efficient resource reservation for aggregated traffic in heterogeneous IP networks," in *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*. IEEE, Jun 2010. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5518309>
- [87] A.-E. Taha, H. Hassanein, and H. Mouftah, "Extensions for Internet QoS paradigms to mobile IP: a survey," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 132–139, May 2005. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1453434>
- [88] A. K. Talukdar, B. R. Badrinath, and A. Acharya, "MRSVP: a resource reservation protocol for an integrated services network with mobile hosts," *Wireless Networks*, vol. 7, no. 1, pp. 5–19, Jan. 2001. [Online]. Available: <http://link.springer.com/10.1023/A:1009035929952>
- [89] C.-C. Tseng, G.-C. Lee, R.-S. Liu, and T.-P. Wang, "HMRSVP: a hierarchical mobile RSVP protocol," *Wireless Networks*, vol. 9, no. 2, pp. 95–102, Mar. 2003. [Online]. Available: <http://link.springer.com/10.1023/A:1021833430898>
- [90] S. Paskalis, A. Kalokylos, and E. Zervas, "An efficient QoS scheme for mobile hosts," in *Local Computer Networks, 2001. Proc.. LCN 2001. 26th Annual IEEE Conference on*. IEEE Comput. Soc, 2001, pp. 630–637. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=990844>
- [91] G. A. R. Ali, Z. Swallow, "Localized RSVP," Internet Engineering Task Force, February 2006. [Online]. Available: <http://tools.ietf.org/id/draft-manner-tsvwg-lrsvp-00.txt>
- [92] M. A. Malik, S. S. Kanhere, M. Hassan, and B. Benatallah, "On-board RSVP: an extension of RSVP to support real-time services in on-board IP networks," in *Proc. of the 6th international conference on Distributed Computing*, ser. IWDC'04. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 264–275.
- [93] N.-C. Wang, J.-W. Jiang, and Y.-F. Huang, "RSVP extensions for real-time services in heterogeneous wireless networks," *Computer Communications*, vol. 30, no. 10, pp. 2248–2257, Jul 2007. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0140366407001892>
- [94] G. Kamel, A. Mihailovic, and A. Aghvami, "A Cost-Optimal QoS Aggregation Policy for Network Mobility: Analysis and Performance Comparisons," *IEEE Trans. Veh. Technol.*, vol. 58, no. 7, pp. 3547–3557, Sep 2009.
- [95] S. Adibi and S. Erfani, "Mobile ad-hoc networks with QoS and RSVP provisioning," in *Electrical and Computer Engineering, 2005. Canadian Conference on*. IEEE, May 2005, pp. 2069–2072. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1557394>
- [96] E. Mirzamany, A. Lasebae, and O. Gemikonakli, "Using aggregated RSVP in nested HMIPv6," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th*

International. IEEE, Aug 2012, pp. 716 –721. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6314292>

- [97] “Cisco visual networking index: Forecast and methodology, 2011-2016,” White Paper, Cisco Inc., May 2012.

Flavius Pana is a Ph.D. student in the Research Centre for Management Informatics at KU Leuven in Belgium. He received his Engineering degree in Electronics and Telecommunications in 2009 from the Polytechnic University of Timisoara (Universitatea Politehnica Timisoara) in Romania, and a master degree in Information Management in 2010 from KU Leuven. His research interests focus on Quality of Service (QoS) in the Internet, adaptive QoS provisioning and signaling protocols.

Ferdi Put is professor in the Research Centre for Management Informatics at KU Leuven. He received a master degree in Business Engineering from KU Leuven in 1980, an MBA from Cornell University in 1981, and a PhD in Applied Economics from KU Leuven in 1988. His research interests are in Simulation and Networks.