H.323/RSVP Synchronization for Video over IP

Engineering White Paper

Subha Dhesikan

Cisco Systems

December 5, 2002

ENG-177305

Version 1.0

Table of Contents

LIST OF FIGURES

LIST OF TABLES

TERMINOLOGY

ACF	Admission confirm message. A RAS message issued by the gatekeeper indicating the acceptance of the ARQ.
Admission Control	This is the mechanism, which decides whether the network device has sufficient resources to supply the requested QoS.
ARQ	Admission request message. A RAS message that is sent by the endpoint to the gatekeeper requesting admission for a proposed call.
BE	Best effort. This indicates a type of service in the network, which does not involve any priority.
COPS	Common Open Policy Service. Client/Server protocol to support policy control. It is described in [6].
DSCP	DiffServ Code Point. Described in [2]
Endpoints	In this document, endpoints are referred to as any entity that generates or receives media traffic in a videoconference. Endpoints include terminals, gateways, MCUs, proxies etc.
Flow	A flow is a set of packets belonging to one instance of the application identified by some combination of source address, source port, destination address and port, and protocol identifier.
H.225	ITU standard for call signaling protocol and media stream packetization for packet-based multimedia communication systems.
H.245	ITU standard for control protocol for multimedia communications.
H.323	ITU standard for packet based multi media communication systems. It provides the architecture for multimedia communication over a packet switched network using various protocols such as H.225 and H.245.
Logical channel	The channel that is used to carry media between two endpoints. The logical channel may or may not correspond with a physical channel.
MCU	Multipoint Control Unit. This is an endpoint that enables three or more endpoints to join together in a conference.
Network device	This refers to a device in the network that handles traffic. Routers and switches are examples of network devices.
OLC	H.245's Open Logical Channel message. This describes the media that the channel will carry.
Policing	This is the process of enforcing the policies, which could result in delaying or dropping packets.
Policy	This is a set of rules that define the criteria for allowing access to a network resource.

	network resource.
Policy control	The application of policies to make a decision whether to allow access to a resource.
Policy Decision Point (PDP)	A COPS acronym. This is the device where the policy decisions are made. The PDP has usually a global knowledge of all the network polices that pertain to one administrative domain.
Policy Enforcement Point (PEP)	This is the device where the policy decisions are enforced.
Quality of Service (QoS)	This refers to the type of service provided by the network devices.
RAS	Registration, Admission and status. This is a part of the H.225 specification, which is used for communications between the gatekeeper and the endpoints.
Resource	This refers to all the factors in the network device that affect the forwarding of packets such as bandwidth on an interface, queues, processing power etc.
RSVP	Resource ReSerVation Protocol [1].
Traffic	Traffic refers to one or more flows that traverse through the network.

CONVENTIONS

This document uses the following conventions:

1. All message names used in this document have the first letter capitalized. Example: Setup or Path.
2. All parameter names used in this document are in bold typeface and have the first letter capitalized. Some of the parameter names may consist of multiple concatenated words, in which case the first letter of each word is capitalized. Example: **QosMode**, **OpenLogicalChannel, SenderTemplate** etc.

1 Introduction

Quality of service is an important factor that affects the success and wide spread deployment of IP videoconferencing. While quality means different things to different people, this document uses Quality of Service (QoS) to refer to the treatment of the videoconferencing traffic as it traverses through the network devices between the caller and the called endpoint. This can be measured in terms of jitter, delay and loss.

The document "QoS for IP videoconferencing" recommended the use of RSVP for obtaining QoS from the network. It also recommends the synchronization of RSVP with the H.323 call signaling. Synchronization is necessary so that the call can react and change based on the results of the RSVP reservation. This document takes the subject a step further by providing the details necessary to implement RSVP and to synchronize it with H.323 call signaling in an IP videoconferencing solution.

Appendix II of the H.323 protocol provides a high level description of the resource reservation procedures using RSVP. This document goes further by providing the details necessary to implement RSVP and to synchronize it with the different H.323 call procedures. Section 2 is a short description of the RSVP protocol. Section 3 contains the summary of the H.323 call signaling. The details of RSVP synchronization with the different call signaling methods are described in Section 5. The document also covers various other QoS features such as Differentiated Service marking, policy and authentication as it relates to RSVP. Resource reservation within an MCU and interoperability issues are also briefly discussed.

2 RSVP Tutorial

This section is a brief overview of the RSVP specifications as documented in RFC 2205 [1]. If you are familiar with RSVP, you may skip this section. It is recommended that you read the RSVP specification, but for most readers, this summary should suffice.

RSVP is a resource reservation setup protocol that is used by end systems to transmit Quality of Service (QoS) requests for their traffic flow to devices in the network. It is a signaling protocol that is used to request, establish and maintain resource reservations. RSVP messages travel between the end systems using the same path as that of the application's traffic and therefore are able to reserve resources in the same devices that service the traffic. Though RSVP can be used for a variety of control services, this document refers to RSVP only as a signaling protocol for obtaining QoS from the network.

Some of the important features of RSVP are:

- RSVP reservations are made for a particular session. A session is defined as a flow that has a particular destination address, destination port, and a protocol Identifier. RSVP reservations treat each session as one independent unit.
- RSVP messages travels along the same path as that of the media flow.
- RSVP supports both unicast and multicast flows. RSVP dynamically adapts to changing membership in case of a multicast flow.
- RSVP sessions are simplex, i.e., flows are reserved in one direction only.
- RSVP maintains soft state. This helps RSVP to adjust to changing routes and membership.
- RSVP is receiver-oriented. The receiver of the stream requests the reservation.
- RSVP messages flows transparently through non-RSVP routers and switches.

- RSVP does not directly control the behavior of the network devices. The network devices' treatment of the traffic flows depends upon the traffic conditioning mechanisms installed and configured in any particular device.

The remaining of this section is organized as follows: First is the description of the RSVP objects, followed by a description of the RSVP messages. The RSVP operation is subsequently illustrated with examples of unicast and multicast sessions. Finally, some of the other RSVP features are presented.

2.1 RSVP Objects

Some of the RSVP objects are explained in this section. These objects are carried in a RSVP message.

2.1.1 Session

A RSVP **Session** is defined by a destination address, destination port and a protocol identifier. In case of a unicast session, the destination address is that of the unicast receiver. In case of a multicast session, the destination address is the multicast address used by all receivers. The destination port mentioned here usually refers to a UDP/TCP port. They could also refer to a "generalized destination port" i.e. an equivalent field in another transport protocol or any other application specific information. All RSVP messages carry their session information and all messages with the same session refer to a single reservation.

2.1.2 Sender Template

The **SenderTemplate** is an RSVP object that contains information to identify the sender of the traffic flow. It is made up of the sender's IP address, sender's port, and the protocol identifier.

2.1.3 Style

The **Style** object contains the requested reservation style. A session may have a single sender or multiple senders. The reservation style specifies how the reservation is to be setup for all the flows in the session. It indicates whether the reservation should be shared among the senders or be distinct between senders. For a shared reservation, the style also indicates whether the senders are to be explicitly listed or not. There are 3 reservation styles that are defined in the RSVP specification. They are:

- Fixed Filter (FF): This indicates that a distinct and a separate reservation is required for each sender in the session.
- Shared Explicit (SE): This indicates that a common reservation should be shared by the explicit list of senders that is provided.
- Wildcard (WC): This indicates that a common reservation should be made for all the senders in that session.

2.1.4 Sender Traffic Specification

- The **TSpec**[1] object is a description of the traffic that is generated by the sender and is transported through the network to all the intermediary routers and the destination endpoint. The intermediate routers do not change this object and it is delivered unchanged to the

[1] The information in this object is opaque to RSVP and therefore not described in the RSVP specification. The Integrated Service model, RFC 2210, determines the format and content of these objects.

ultimate receiver(s). It is usable with either controlled load or guaranteed service type. This object consists of a token bucket specification along with the peak rate of the traffic, minimum policed unit and maximum packet size. The token bucket specification consists of the traffic rate and bucket size.

- Traffic rate (r): This refers to the average rate of traffic that the sender is generating. It is measured as the number of bytes per second and must be a positive value.
- Bucket Size (b): This is the burst in the traffic generated by the sender. It is measured in bytes and must be positive.
- Peak rate (p): This refers to the peak rate of traffic generated by the host. It is measured as the number of bytes per second of traffic packets. This could contain a positive value or infinity to signify that it is unknown.
- Minimum policed unit (m): This is the minimum packet size in bytes including all the headers such as IP, UDP, RTP, etc. It does not include the link layer header, as it is likely to change with different link layer mediums. The network devices use this element to calculate the overhead needed to carry this packet over the particular link layer medium.
- Maximum packet size (M): This refers to the largest packet that the host will transmit. This includes all the headers as well and is measured in bytes. Any packet that exceeds this size will be policed and not be provided the requested QoS service. Both the *m* and *M* must be positive and m should also be less than or equal to M.

2.1.5 Flow Descriptor

The **FlowDescriptor** is made up of **FilterSpecs** and **FlowSpecs**. The **FlowSpec** describes the desired QoS and is described in 2.3.5 below. The **FilterSpec** describes the sender of the packets. The **FilterSpec** along with the **Session** objects describe the flow to which the QoS is to be provided. The **FlowDescriptor** varies with the reservation style. The different kinds of flow descriptors supported by RSVP are:

- Fixed Filter (FF): The FF flow descriptor contains a list of **FilterSpec** and **FlowSpec** pairs. Each pair consists of a single **FlowSpec** that applies to the particular sender in that pair. Therefore, this style creates a distinct reservation for a specific sender. Example:
  FF flow descriptor = (S1 {Q1}, S2 {Q2})
  where the **FlowSpec** Q1 applies to the sender S1 and the **FlowSpec** Q2 applies to sender S2 etc.
- Shared Explicit (SE): The flow descriptor contains a list of **FilterSpecs** followed by the **FlowSpec**. In this case, the **FlowSpec** applies to all the senders described by the **FilterSpec** list. Therefore, this style creates a shared reservation for an explicit set of senders. Example:
  SE flow descriptor = (S1, S2, …{Q})
  where the **FlowSpec**, Q, applies to all those senders indicated in the list.
- Wildcard (WC): The flow descriptor contains only a **FlowSpec**. The **FilterSpec** is not mentioned as the reservation applies to all senders in that session. Therefore, this style creates a shared reservation that includes all senders. Example:
  WC flow descriptor = (* {Q})
  where the **FlowSpec** Q applies to all the senders.

2.1.6 Flow Specification

This **flowspec** object carries information from the receiver(s) that describes the desired QoS. It consists of a service class and necessary parameters for that class. The two classes of services that are available with Integrated Services are Guaranteed and Controlled Load.

- Controlled Load (CL): This refers to the type of service that the flow will receive when the network device is lightly loaded and ensures with admission control that the same service will be received even during overloaded situations.

  The parameter that is included with the CL service is the Receiver's TSpec. The Receiver's TSpec is of the same format as that of the Sender's TSpec. It contains the description of the traffic for which the QoS is required. The value for *p*, *r*, and *b* reflects the receiver's description of the traffic for which the reservation is required. The values for *m* and *M* depend on the reservation style. For FF reservation, the values of *m* and *M* may be the same as that of the Sender's TSpec. For SE and WC reservation, these values may be calculated from the individual Sender's TSpecs that is received. *m* should be the smallest of the *m*'s that is received otherwise, the reservation is likely to be rejected for the sender that has a smaller packet size. *M* can be set to the largest M's in all the received Sender TSpec. However, if *M* is larger than the Path MTU then the reservation is likely to fail. The path MTU can be learned from the AdSpec parameter in the Path message.

- Guaranteed (GQ): This is a type of service that requests a fixed bandwidth and delay guarantees. This delay specifies the maximum firm end-to-end queuing delay allowed.

  The parameter included with the GQ type of service consists of the receiver's TSpec as well as the Rspec. The receiver's TSpec values are the same as described in CL service. The Rspec consists of 2 elements: rate and slack term. The Rspec rate must be greater or equal to r and the slack term must be nonnegative.

2.2 RSVP Messages

This section briefly discusses the different RSVP messages. There are 7 messages defined by the RSVP specification. RSVP messages are usually sent as "raw" IP datagrams with the protocol number 46. Host systems that do not support "raw" IP I/O can encapsulate the RSVP message as UDP datagrams.

2.2.1 Path

An RSVP-capable host that originates traffic sends a Path message. The Path message contains the description or advertisement of the traffic. The Path message travels from the sender to the receiver along the same path as that of the media traffic. The RSVP-capable network devices along the path will collect the necessary information from the Path message and store it as path state. Session, Sender Template and Sender Tspec are some of the objects carried in the Path message.

2.2.2 Resv

When a receiving host receives a Path message, it can request resources for the traffic described in the Path message by transmitting a Resv message. The Resv message is transmitted along the reverse path as that of the Path message and the traffic flow. Each device along the path receives this message and decides whether to accept or deny the request. If the request is accepted, then the necessary state is stored and the message is forwarded towards the sending host. If the device denies the request, then a ResvError message is sent to the host that made the reservation and the Resv message is not forwarded any further. Some of the objects contained in the Resv message are session, style, **FilterSpec**, and flowdescriptor.

2.2.3 PathTear

The PathTear message originates either from the sender or from any host where the path state has timed out. The PathTear message is sent downstream[2] towards the receiver. When a network device receives this message, it deletes the matching path state. If no matching state is found, then it discards the message without forwarding. The device also makes any necessary adjustment in the related reservation state. The adjustment depends upon the reservation style and is given below:

- SE, FF: The reservation that contains the matching **FilterSpec** is removed.
- WC: The reservation is removed only if the sender is the last sender in that session.

2.2.4 ResvTear

A ResvTear message originates either from the receiver or from any host where the reservation state has timed out. This message is sent upstream[3] towards the sender. When the router receives this message, it deletes the matching reservation state regardless of reservation style. If no matching reservation state is found, then it discards the message and does not forward the message.

2.2.5 PathError

A PathError message is sent when a device encounters an error in processing the Path message. This message travels upstream towards the sender of the Path message. This message does not cause any change in state in any of the intermediate devices. The *ErrorSpec* object in the message specifies the error and also the IP address of the device where the error occurred.

2.2.6 ResvError

A ResvError message is sent to signify an error in the processing of the Resv message. It flows downstream from the point where the error occurred to the endpoint that made the original reservation. It may also be sent when there is a disruption in the reservation. This message causes the reservation to be deleted in the intermediate devices that it passes. The error information is carried in an *ErrorSpec* object. This object also contains the IP address of the device where the error was generated.

2.2.7 ResvConfirm

The ResvConf message is sent by the receiving endpoint when it receives the ResvConf object in the Resv message. It is a confirmation of a successful reservation. The message is sent to the address that is provided in the ResvConf object. The message contains the ResvConf object and also an ErrorSpec object. The ErrorSpec object contains the IP address of the device where the message originated. The error code and value in this object is set to 0 to signify that it is a confirmation and not an error.

2.3 RSVP Operation

This section presents the RSVP operation for both a unicast session as well as for a multicast session.

[2] Downstream refers to the path from the sender to the receiving host.

[3] Upstream refers to the path from the receiver to the sender.

2.3.1 Unicast

Here is a simple unicast example demonstrating the RSVP operation. Please refer to Figure 1 below. Let us assume that endpoint A (EP_A) wishes to transmit a stream to endpoint B (EP_B). Let us also assume that the routers along the path- R_1, R_2 and R_3 are all RSVP-enabled. Figure 1 shows the topology and the flow from EP_A to EP_B through the routers. The RSVP call flow shows the actual RSVP messages that are exchanged.

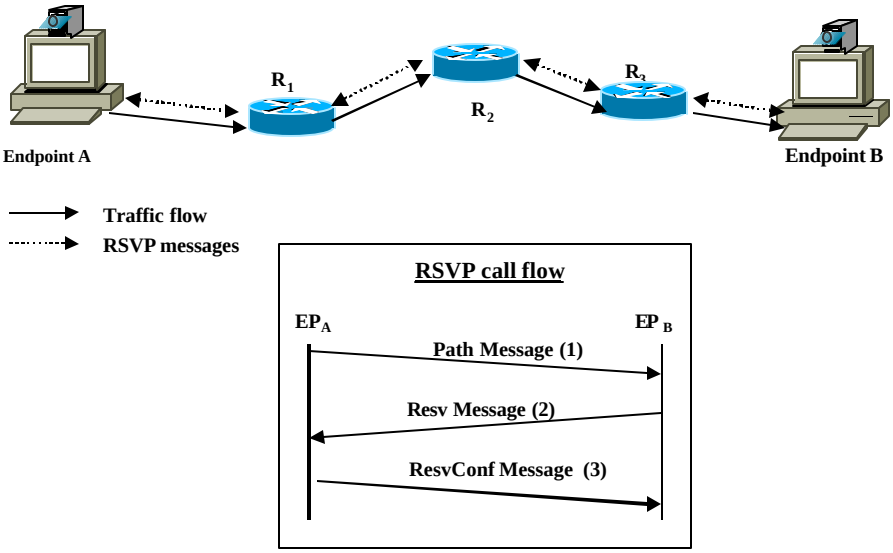


Figure 1: An RSVP Reservation

1. EPA sends a RSVP Path message to EPB with its address as source address and EPB's address as the destination address. The Path message is a description of the traffic flow that EPA is about to originate. R1, R2 and R3 will receive and process the Path message and gather state from it. The Path message will ultimately reach EPB.
2. If EPB decides to make a reservation for the stream described in the Path message, it will respond with a reservation request (Resv message). This message will travel through R3 and R2 and R1 requesting resources from each of them. Each router will independently decide whether to grant the request. If the request is granted then the router will store the necessary state and forward the message to the next device. The Resv message describes its reservation requirements in an object called the FlowSpec. If EPB requires a confirmation message for its request, then it includes a ResvConf object in the Resv message.
3. If all the routers along the path grant the request then the Resv message eventually reaches EP_A. If the Resv message contains the ResvConf object then EP_A will initiate a ResvConf message. This message is sent to the address contained in the ResvConf object. In this example, the ResvConf object contains the address of EP_B. After receiving the Resv message, EP_A can transmit the stream. The stream will be granted the requested QoS as long as it is conformant to the traffic specification provided in the RSVP Resv message.

If any routers decide to deny the reservation request, then it generates a ResvError message and transmits it back to the requesting endpoint. The router does not forward the Resv message any further. The ResvError message travels back the same path as the Resv message, removing the reservations in devices where the reservation has already been made. Let us assume that R_2 does not have the necessary resources that are being requested. Therefore, R_2 rejects the reservation and returns a ResvError message to EP_B. The ResvError message travels back through R_3. When R_3 receives this error message, it removes the reservation it had originally installed. The ResvError message finally reaches EP_B. The reason for the reject as well as the address of the node where the reservation was rejected is contained in the error message.

When the reservation is no longer required, RSVP teardown messages are sent to remove the reservation from the routers. There are two kinds of teardown messages: PathTear and ResvTear. The PathTear message is sent by the transmitting endpoint and removes the path state as well as the related reservation state from the network devices. The ResvTear message is sent by the receiving endpoint when it no longer requires a reservation. These tear messages are optional. If the tear messages are not sent, then the state in the routers will automatically time out and be deleted after a certain period of time. This is referred to as softstate and is discussed in section 2.4.1.

The traffic stream discussed in this example is unidirectional. It flows from EP_A to EP_B. Hence there is only one set of RSVP messages. If there were two streams, one from each endpoint, then there would have been two sets of RSVP messages.

2.3.2 Multicast

Here is a simple example demonstrating resource reservation using RSVP for a multicast session. Please refer to figure 2 below. Endpoints: EP_1, EP_2 and EP_3 join the multicast session by joining the multicast address (*M*). This enables them to receive from and send to this session.

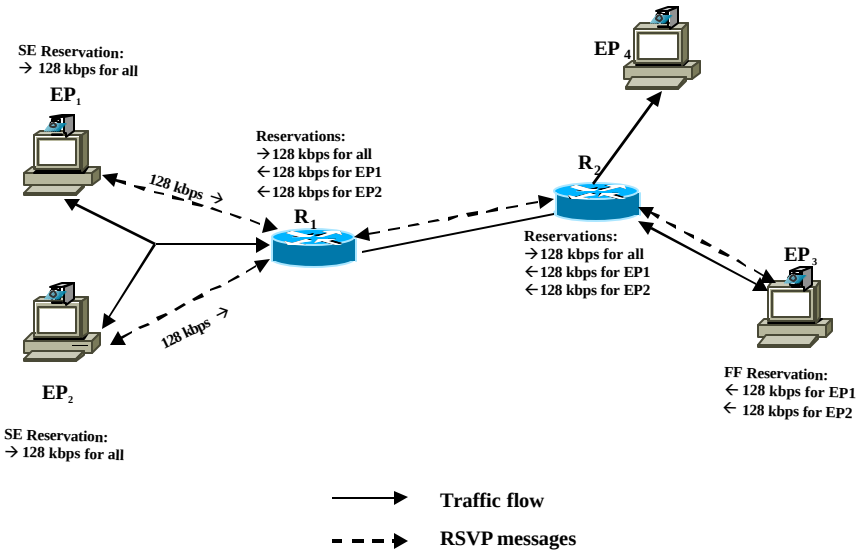


Figure 2: RSVP reservation for a multicast session

Let us assume that all three endpoints wish to transmit traffic averaging 128 kbps to the multicast session. Let us also assume that the routers along the path - R_1 and R_2 - are RSVP-enabled. The endpoints transmit a RSVP Path message using *M* as the destination address. Each endpoint receives the Path message from the other two endpoints and makes a reservation.

A receiving endpoint can make a reservation for streams from multiple senders with a single Resv message. It can specify whether the reservation is to be shared by all the streams or distinct reservations be made for each of the streams by choosing the appropriate style and providing the correct flow descriptor. For instance, a fixed filter style is chosen when distinct reservations are required for each of the streams, whereas shared explicit or wild card provide for shared reservations. The reservations made by different receivers are merged in the intermediate routers based on the rules of merging provided in the RSVP specification. This prevents multiple reservations for the same traffic in the network device.

To illustrate this, let us assume that EP_1 and EP_2 wish to make a shared reservation for all streams for 128kbps whereas EP_3 wish to make a distinct reservation of 128kbps for each stream received. In this case, the reservation of EP_1 and Ep_2 will be merged at R_1 and a single reservation of 128 kbps will be forwarded to R_2. However, in the case of the reservation from EP_3, R_2 will forward a Resv message containing 2 reservations each for 128kbps. These reservations will not be merged, as they are distinct reservations.

2.4 Other RSVP Features

2.4.1 SoftState

The RSVP state maintained by the network devices and hosts is temporary and is valid only for a short period. At the end of this period, the state stored in these devices is deleted. Therefore, RSVP state is considered to be Softstate. In order to maintain the RSVP state and the reservation in a device, RSVP messages have to be sent periodically. This periodic refresh prevents the state from being "timed out". The period between refresh is called the Refresh Interval (R) and the timeout period for the RSVP state is referred to as the "cleanup timeout interval" (C). The suggested value of *R* is 30 seconds. However, the hosts generating the RSVP message may choose a different refresh Interval. Refreshes may be sent more often than R during state changes. The network devices use R to decide on *C*. RSVP specification indicates that to avoid premature loss of state, *C* must satisfy:

$$C >= (K + 0.5) * 1.5 * R$$

The suggested value of *K* is 3. *R* and *C* are likely to vary across hops. To simplify, *C* is usually calculated as 3 times *R* in some implementations.

Some of the advantages of this softstate mechanism are:

- It helps RSVP adjust to the changing routes in the network. If a route changes, RSVP messages initialize the state along the new route. The state in the old route gets automatically deleted as there will be no refreshes to maintain them.
- It enables RSVP for use with multicast connections where participants join and leave at different times. As participants join, RSVP messages are forwarded to the new partipants and RSVP state is created along the new routes. As participants leave, RSVP state is automatically deleted along the routes which are no longer required.
- RSVP is not built with any reliability mechanism. Since RSVP messages are periodically transmitted, an occasional loss of a message does not cause the reservation to be deleted. It matters only if the number of successive RSVP messages lost exceeds the timeout interval.

2.4.2 Non-RSVP clouds

All the routers along the path might not be RSVP-capable. If two RSVP routers have a non-RSVP cloud between them, then RSVP messages will be routed correctly through the cloud as these messages are routed using the local routing table. However, no resource reservation will be performed in the devices in the cloud since these devices are not RSVP-aware and therefore cannot capture and process the RSVP messages. If there is any resource constraint in the non-RSVP cloud, then the end-to-end QoS provided will be affected.

One cannot expect all devices in the network to be RSVP-enabled. For example, routers designed for the Internet core have a fixed number of hardware queues that makes them incapable of creating dynamic queues for individual flows. Fortunately, individual flow guarantees are not necessary in core networks and there is a mechanism that obviates them. The technique is documented in RFC 2998, A Framework for Integrated Services Operation over Diffserv Networks [19]. Under this architecture, RSVP is enabled only at the edges of the networks. This typically implies enabling RSVP on WAN links. QoS in the core or on non-RSVP capable LAN switches is provided by DSCP marking, scheduling and policing.

2.4.3 Policy Control

Using RSVP, applications can obtain preferential treatment from the network for their traffic. There must be some mechanisms in the network to ensure that the network's resources are not misused with RSVP. One such mechanism is policy control. Policy control is a check that ensures that the RSVP request is permissible based on the configured policies before the request is granted. The network administrator creates policies to dictate how the network's resources will be utilized. These policies are maintained in the network. In addition to the general information in the RSVP message such as session, source, destination information, RSVP may also carry additional information such as identity of the user, account, application, etc for policy control purposes. When a reservation arrives at a network device, the policy information from that request is submitted to the policy decision-making points for approval. These decision points, called the Policy Decision Points (PDP), may either reside locally in the devices itself or externally in a policy server. The PDP compares the information received with the policies installed in the network, and decides whether the reservation request should be granted. The network device receives the decision and enforces them. Hence they are referred to as the policy enforcement points (PEP). This is described in [6]

Policy information is transparent to RSVP and is described in subsequent RFCs [7] [8]. In particular, RFC 3182 [11] describes the use of a policy object to carry application identity so that the network device could identify the application and the user on behalf of which the reservation is made. The encoding and processing rules are described in this RFC. This object can also contain user credentials such as Kerberos tickets, or digital certificates, to help authenticate the identity information it contains.

While RFC 3182 describes a generic form of the policy object, RFC 2872[9] describes the contents for one such object. The identity object consists of a globally unique identifier (GUID) that uniquely identifies the application and possibly the vendor as well. The other elements of that object are the application name, application version and optional sub-application information. This information collectively helps a network device to identify an application and enforce policies that are installed.

3 H.323 Tutorial

This section provides a brief overview of the H.323 call-establishment procedures with special emphasis on the QoS parameters. If you are familiar with these topics, please skip this section. Conversely, full details may be found in the associated specifications.

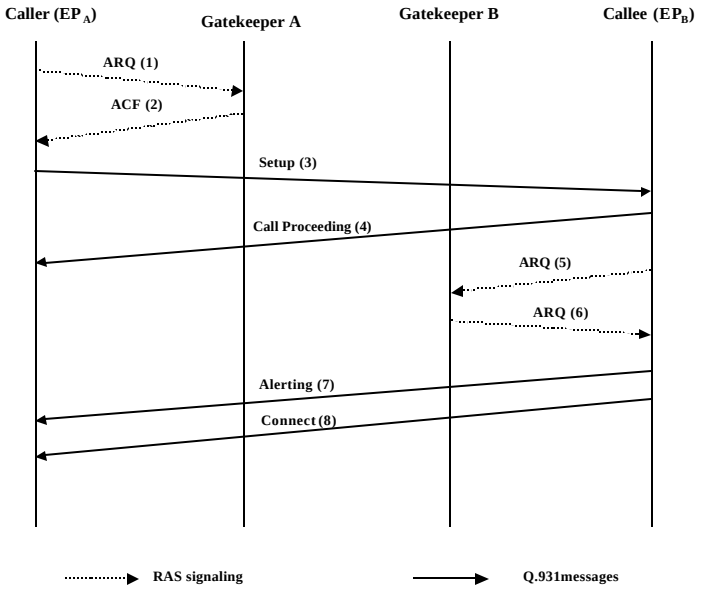
Establishing a call usually requires the use of RAS, Q.931 and H.245 protocols. RAS consists of registrations, admission control, address resolution, bandwidth control and status. The RAS messages are used between endpoints and gatekeepers, and between gatekeepers. RAS signaling occurs in networks where the gatekeeper is present and it uses a dedicated, unreliable channel called the RAS channel. The Q.931 call signaling procedures are used to establish a connection between two endpoints. There are two models of call signaling. One is gatekeeper-routed call signaling where the gatekeeper passes the signals between the endpoints. The other is direct endpoint call signaling where the endpoints directly exchange call-signaling messages between themselves. The gatekeeper decides the call model during the admissions process. In scenarios where the gatekeeper is not present, direct call signaling is used. The recommendations of H.245 are followed to exchange call control messages that decide the operations of the endpoints. These messages are used to negotiate capabilities, set up and tear down media channels, determine master/slave functionality, etc. These messages are usually exchanged directly between the called and the calling endpoints.

In the sections below, initialization, setting up and releasing a call are described in more detail.

3.1 Establishing a Call

Figure 3 shows a simple call being established between 2 endpoints. The caller, EP_A, wishes to call EP_B. EP_A and EP_B are registered with gatekeepers A and B, respectively. This is an example of direct endpoint call signaling.

Figure 3: Direct Endpoint Call Signaling



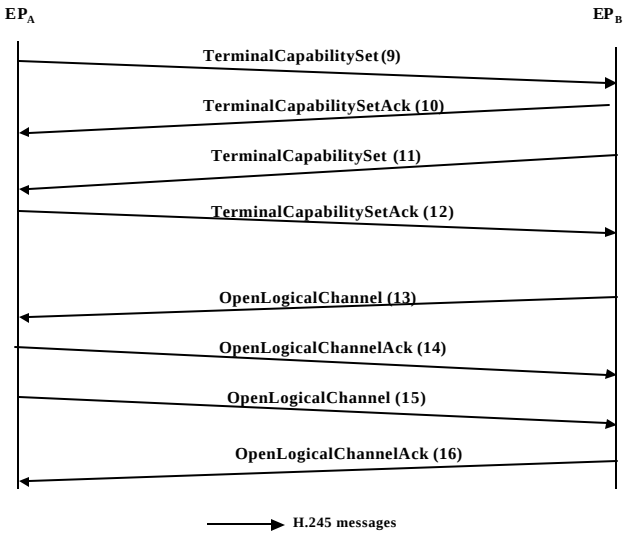
1. The first step in establishing a call is admission control. The calling endpoint (EP_A) must first send an admission request to the gatekeeper for the call it is about to initialize. This request is sent in an Admission Request message (ARQ). The ARQ message contains bandwidth information, source and destination address and other necessary information. The **BandWidth** parameter contains the total bandwidth that will be used in both

directions by all the channels in the call. For example: In a 384 kbps audio/video call, the bandwidth reported is 768kbps (384 * 2)

2. The gatekeeper will either confirm the admission request by sending an Admission Confirm message (ACF) or reject the request by sending an Admission Reject message (ARJ). In this example, the gatekeeper admits the call and sends an ACF message to EP_A. The gatekeeper includes the **BandWidth** parameter to specify the maximum bandwidth that has been approved for the call. This bandwidth may be less than the one requested in the ARQ. The gatekeeper also specifies the method of calling in the **CallModel** parameter. Since our example uses direct call signaling, this parameter will specify "direct" and the destination address in the ACF will contain the address of the destination endpoint. EP_A will use this address to exchange the following Q.931 messages.
3. After admission control, EP_A sends a Setup message to EP_B indicating its desire to setup a connection.
4. EP_B receives the Setup message and can respond with any one of these Q.931 messages: Alerting, Call Proceeding, Connect or Release Complete message. In this example, EP_B sends a Call Proceeding message to indicate that it is proceeding with the establishment of the call.
5. EP_B has to obtain admission control for the call with its gatekeeper. So, it sends an ARQ to gatekeeper B.
6. Let us assume that gatekeeper B admits the call and responds with an ACF message.
7. An Alerting message indicates that the endpoint has alerted the user i.e. ringing or other form of notification to the user interface to indicate that an incoming call is being received. If any response such as Call Proceeding, Connect or Release Complete, is sent within 4 seconds of the Setup message, then the Alerting message is not required. In this example, EP_B alerts the user and sends an Alerting message.
8. When EP_B decides to answer the call (i.e. off hook or answered by the user interface) and is ready for the next phase, it sends a Connect message. At any time, if EP_B decides not to continue with the establishment of the connection, it could send a Release Complete message. The release message contains the reasons for release. Admission Control failure is one such reason.

Upon receiving the Connect message, the endpoints proceed to negotiate capabilities. Figure 4 shows the H.245 messages that are exchanged to negotiate capabilities and to establish media channels.

Figure 4: Establishing Media Connections



9. The endpoints have to select and agree on the various parameters - such as audio codec and video codec - to be used in the call. The exchange of the H.245 TerminalCapabilitySet (TCS) messages allows the endpoints to negotiate such capabilities. EP_A sends a TCS message to EP_B describing its capabilities. This message will contain both the endpoint's receive and transmit capabilities.

10. EP_B acknowledges EP_A's capabilities by sending a TerminalCapabilitySetAck (TCSAck) message.

11. EP_B transmits its receive and transmit capabilities by sending its TCS message.

12. EP_A acknowledges EP_B's capabilities by sending a TCSAck message to EP_A.

13. EP_A sends an OpenLogicalChannel (OLC) message for each media channel that it wants to open. This message contains the description of the channel, including media type, algorithms and other information that are necessary for EP_B to interpret the contents of this channel.

14. EP_B acknowledges with an OpenLogicalChannelAck (OLCAck) message. The OLCAck message contains parameters such as EP_B's transport address and port to be used by EP_A.

15. EP_B sends its OLC message providing the parameters for its channels.

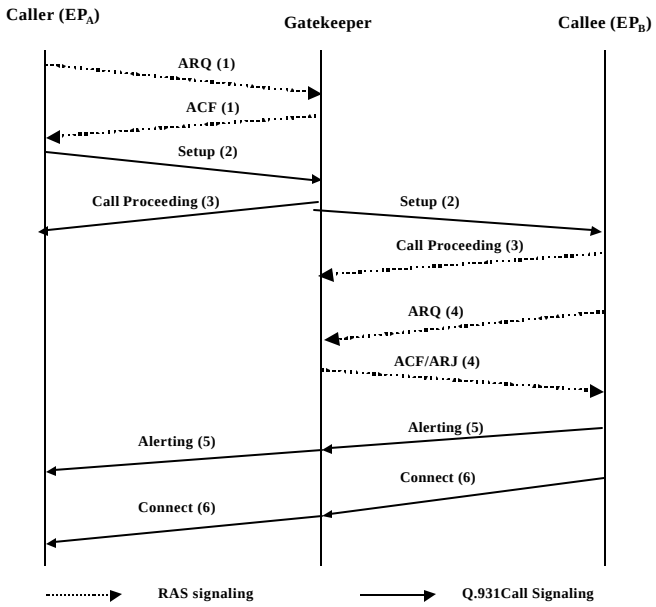
16. EP_A acknowledges the OLC message with its OLCAck.

Once the media channels have been opened, then the endpoints are free to send traffic. The audio and video streams are encapsulated in RTP packets. The H.245 control channel is kept open for the duration of the call. This channel is used for master/slave determination as well for exchanging other control information such as mode, time and counter values etc. Master/Slave determination messages are exchanged to determine which endpoint should perform the role of a Multipoint Controller (MC) when both the endpoints support an MC.

3.2 Establishing a Call with Gatekeeper-Routed Call Signaling

The above was an example of direct endpoint call signaling. In this section, gatekeeper routed call signaling is addressed with examples. The connections and the messages are the same in both these methods. Figure 5 shows a simple call being established between 2 endpoints. In this example, the caller, EP_A, wishes to call EP_B. For the sake of simplicity, we will assume that both EP_A and EP_B are registered with the same gatekeeper.

Figure 5: Gatekeeper routed call signaling



1. The admissions message occurs as before. The endpoint sends the gatekeeper an ARQ requesting admissions and the gatekeeper either responds with an ACF or an ARJ depending on whether the call is accepted or rejected. In this example, the gatekeeper accepts the call by sending an ACF message. In the ACF, the gatekeeper specifies the method of calling in the **CallModel** parameter. Since this example uses gatekeeper-routed method, this parameter will specify, "gatekeeper routed". The **DestCallSignalAddress** parameter will specify the gatekeeper address to which the call signaling is to be sent.
2. EP_A sends the Setup message to the address specified by the gatekeeper in the ACF message. The gatekeeper receives this message and relays it to the destination endpoint, EP_B.
3. The gatekeeper and EP_B send the Call Proceeding message in parallel.
4. EP_B now initiates its admission process with the gatekeeper as before. If it receives a reject then it sends a ReleaseComplete message back to release the process. In this case, the gatekeeper sends an ACF confirming the admissions request.
5. EP_B alerts the user and sends the Alerting message.
6. When EP_B is ready for the call, then it sends a Connect message to the gatekeeper. The gatekeeper then relays this Connect message to the endpoint.
7. Similar to the above, the H.245 control channel messages may also be routed through the gatekeeper. The Connect message sent by EP_B to the gatekeeper contains the

address for the H.245 control channel. Depending on how the gatekeeper is configured, it could include its address or the EP_B's address as the control channel address in the Connect message to EP_A. If its address is included, then the call control will be routed through the gatekeeper.

3.3 Establishing a Call Using H.245 Tunneling

In order to conserve resources and shorten setup time, the endpoints may use the Q.931 call signaling channel to perform the H.245 exchanges instead of opening a separate H.245 channel. This procedure is known as H.245 tunneling or encapsulation. To indicate the use of the H.245 tunneling procedure, the endpoints must set the **H245Tunneling** element to TRUE in every Q.931 message and must encapsulate the H.245 messages within the Q.931 call signaling channel. The call signaling process remains the same. However, at any time the endpoints could decide to open a separate H.245 channel by transmitting the H.245 address. After the H.245 control channel is opened, the tunneling procedure is discontinued.

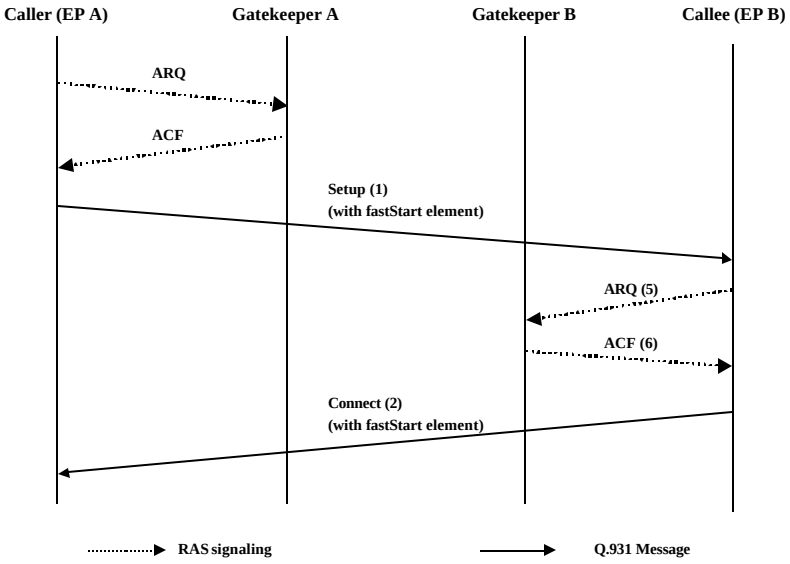
3.4 Establishing a Call using Fast Connect

The calls can be connected as described in the earlier sections or using the "Fast Connect" procedure. The Fast Connect procedure allows the channel information to be exchanged in the Setup message itself instead of using the H.245 procedures. This helps in reducing the setup time. Figure 6 illustrates the fast connect procedure.

1. The fast connect procedure requires the calling endpoint (EP_A) to include the **FastStart** element in the Setup message. The **FastStart** element contains a set of OLC structures describing the media channels that will be used to send and receive.

2. EP_B receives the Setup message with the **FastStart** element. EP_B can respond with any one of these Q.931 messages: Alerting, Call Proceeding, Connect or Release Complete. If EP_B wants to support fast connect then it must include its **FastStart** element in one of its responses. EP_B cannot include a **FastStart** element if it did not receive one from EP_A, nor can it include the element after the Connect message.

Figure 6: Call signaling using the "Fast Connect" procedure

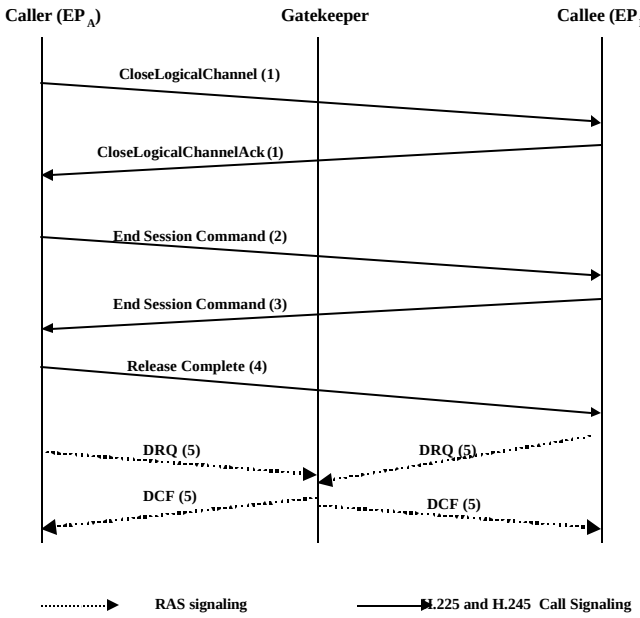


EP_B may start transmitting media any time after sending any Q.931 message containing **FastStart**. As EP_A may receive media even before receiving any response from EP_B, it must be ready to receive data in all the channels it proposed in its Setup message . EP_A may transmit media after receiving the Q.931 message from EP_B containing the **FastStart** element. If EP_A has set the **MediaWaitForConnect** element to true in the Setup message, then EP_B may transmit media only after sending the Connect message.

3.5 Terminating a Call

Figure 7 illustrates call termination.

Figure 7: Call Termination



1. The endpoints (EP_A and EP_B) discontinue the transmission of media and close the logical channels. This involves exchanging the CloseLogicalChannel messages as shown in the above figure. A set of messages is exchanged for each open channel.

2. If the H.245 control channel is opened, then the endpoint should transmit the H.245 EndSessionCommand message on the control channel and stop transmission of H.245 messages.

3. It should then wait for a H.245 EndSessionCommand message. As soon as it receives the message, it can shut down the H.245 control channel.

4. If the call signaling channel is open, then the endpoints must transmit a H.225 ReleaseComplete message and then close the channel.

5. The endpoints must disengage from the gatekeeper by sending H.225 DisengageRequest message (DRQ). The gatekeeper disengages the endpoints and confirms by sending DisengageConfirm (DCF) message back. This process helps the gatekeeper keep an accurate account of the bandwidth utilized.

3.6 QoS parameters

The H.323 specification recommends the use of transport level QoS mechanisms to provide QoS to its real-time voice and video streams. In its appendix, the specification describes the use of RSVP for obtaining QoS. This was first described in version 2 of the specification. To support the use of RSVP, various parameters have been provided. These parameters are described below.

3.6.1 TransportQoS

The **TransportQoS** parameter is an optional parameter in an ARQ message. The endpoint includes this parameter to indicate one of the following:

- Endpoint controlled: This implies that the endpoint can do resource reservation.

- Gatekeeper controlled: This implies that the gatekeeper will perform resource reservation on behalf of the endpoint
- No control: This implies that resource reservation is not required.

The gatekeeper indicates its response by including its **TransportQoS** parameter in the ACF message. If this parameter is present in the ARQ message then the gatekeeper must include it in an ACF message. The gatekeeper may also include this parameter even if it was not part of the ARQ message. If the gatekeeper requires that the resource reservation be controlled by the endpoint and the endpoint either has not included a **TransportQoS** parameter or has specified endpoint-controlled in the **TransportQoS** parameter, then the gatekeeper may reject the call by sending a ARJ message.

3.6.2 QoSCapabilities

This parameter contains the QoS capabilities of the endpoint. One element of this is defined to be the RSVP parameter. QoSCapabilities may be included in the **TransportCapability** in the OLC structures or in the RequestChannelClose message. In the OLC message, this parameter indicates the QoS capabilities available for that channel. In a RequestChannelClose message, the QoS capabiltities specifies the QoS parameters that were in use in that channel.

QoSCapabilities {
…
rsvpParameters
…
}

3.6.3 RSVP parameters

The **RsvpParameters** contain the information that is to be used with the RSVP protocol. There may be a sequence of RSVP parameters prioritized by the QoS mode. The other elements in the **RsvpParameters** are the same as the RSVP's TSpec, which is explained in the RSVP tutorial section (2.1.4) above.

RsvpParameters {

qoSMode
tokenRate (bytes/seconds)
bucketSize (bytes)
peakRate (bytes)
minimumPoliced (bytes)
maxPktSize (bytes)
}

3.6.4 QoSMode

The **QoSMode** parameter is included in the **RsvpParameters**. It may either be empty or may include one of the two modes: **Guaranteed** (GQ) or **ControlledLoad** (CL). If the **QoSMode** is left empty, then it implies **Best Effort** (BE).

qoSMode{
guaranteed,
controlledLoad,
}

Each endpoint has a list of **QoSModes** prioritized by its preference. This list is exchanged between the endpoints. A derived set (D) is created by the endpoints by intersecting the choices

in the two lists. The two endpoints attempt to establish reservation using the **QoSModes** in the derived set in the order of preference. The call handling will depend on the choices in the derived set. Table 1 describes the different choices, the derived set and the call handling information:

Table 1: Derived Set

QoSModes choices	Derived set	Call Handling
EP_A: {GQ} EP_B {GQ, BE}	{GQ}	Endpoints attempt GQ reservation. Call will be released if GQ reservation fails.
EP_A: {GQ} EP_B: {CL, BE}	{}	Called endpoint will release the call, as there is no common QoS mode between the endpoints.
EP_A: {CL, BE} EP_B: {CL, BE}	{CL, BE}	Endpoints will attempt a controlled load reservation. If the reservation fails, the call will continue with no reservation as BE is supported by both endpoints.
EP_A: {GQ, BE} EP_B: {CL, BE}	{BE}	The call will be continued however no reservation will take place.

Note that GQ reservation is a stricter form of reservation than CL and hence should always be prioritized ahead of CL.

4 H.323/RSVP Synchronization

As mentioned in the introduction, it is not sufficient to just add RSVP support; it is necessary to synchronize RSVP with the call signaling. Some of the reasons for that are:

- **RSVP status reporting**: During network congestion, quality of the videoconference depends on the outcome of its RSVP reservations. If a reservation fails, then the network is likely to delay and drop packets, leading to poor quality conferences. If a reservation succeeds, then the videoconferencing traffic is likely to be shielded from network congestion, thus protecting the quality of the call. This difference in quality may be confusing to the end user. Hence, the status of a reservation must be provided to the user, which is possible only if RSVP is synchronized with call signaling.
- **Failure mode:** Some customers may want to disallow poor quality conferences and therefore might require that the call be failed if RSVP fails. RSVP synchronization is necessary to take such failure actions. In addition, it also provides the facility to report to the user the reason for call failure.
- **Pre-ring RSVP**: To support call failure due to reservation failure, it is necessary to complete RSVP reservations before alerting the called party. This enables the call to be failed without involving the called party. Pre-ring RSVP can be possible only with RSVP and call signaling synchronization.

This section provides the details for providing and synchronizing RSVP with the call signaling of an H.323-based videoconference. It is based on the Transport Level Resource reservation procedures, which were first introduced as an appendix in version 2 of the H.323 specifications. This section is divided into multiple sections: Initiating, call establishment and then the teardown. There are also discussions on various other design issues such as DSCP marking, policy control and authentication. Finally, there is a section on some of the interoperability issues that are likely to surface when IP videoconferencing endpoints begin to offer RSVP.

4.1 Who initiates RSVP?

RSVP reservations can be made either by the endpoints or by the gatekeeper on behalf of the endpoints. Endpoints not only include video terminals but also in gateways and MCUs. RSVP messages are routed using the same routing information as the media stream. This causes the RSVP messages and the media streams to travel along the same path towards the destination. Since gatekeepers do not route media and may not be collocated with the endpoints in the same subnet, reservations made by the gatekeeper might not take the same path as that of the media stream. Given that the reservation is effective only if it is made in the network devices that service the media stream, it is more effective for the endpoints to make the RSVP reservation. Endpoint RSVP also enables easier coordination between the RSVP signaling and the call signaling. Hence, endpoint RSVP is recommended.

The endpoint can make its own reservation if permitted by the gatekeeper. An endpoint that desires to make its own reservation must include the **transportQoS** field in its ARQ message to the gatekeeper. This field should be set to "endpoint controlled". If the gatekeeper is to allow the endpoint to control the resource reservation, it does so by indicating "endpoint controlled" in the **transportQoS** field in its ACF message. This allows the endpoint to make the reservation for its streams.

In environments where resource reservations are mandatory, the gatekeeper should reject an ARQ from an endpoint if the **transportQoS** is not included. However, this requires all the endpoints to be RSVP-capable. Section 5.4 of this document contain options for environments that contain endpoints that are not all RSVP-capable.

4.2 Establishing a call with reservation

Usually, RSVP reservations are made only for the media channels such as audio and video. Data channels such as T.120 and the channel that carries the Far End Camera Control (FECC) may not normally require a reservation. However, if the data channel is used for white board functions then reservations may be necessary to keep the data flow in sync with the "voice over". RSVP reservations are required for each direction of the stream. For example, for a two-way audio-only conversation, two RSVP reservations have to be made, one for each direction. Therefore, it is necessary to know how many channels are to be opened before any RSVP messages can be sent. In addition, the following information is required before any RSVP procedures can be initiated for a channel.

- Destination and source information. The destination and source address and port information of the media flow is required.
- Transport layer protocol that is to be used by the media flow. This is usually UDP.
- Traffic specification: Information about the media traffic such as the rate of the flow, the peak bandwidth, the burst size, the maximum and minimum sizes of the packet are required.

The above information is required for creating a RSVP message. Since a RSVP reservation is uni-directional, both sides of the call require this information. However, this information is usually exchanged in the H.245 signaling by which time the called party is usually alerted. To support pre-ring RSVP, the alerting phase must be postponed until the RSVP reservations are complete. In addition, it is also necessary to open the channels prior to sending the RSVP message, since the RSVP messages are addressed to the same port and destination address as that of the media flow. The following sections describe the different types of call establishments that cover the requirements of RSVP synchronization.

4.2.1 A call with RSVP

Figure 8 shows a call between a calling endpoint EP_A and the called endpoint EP_B. Both these endpoints are assumed to be RSVP capable. This call uses direct endpoint call signaling.

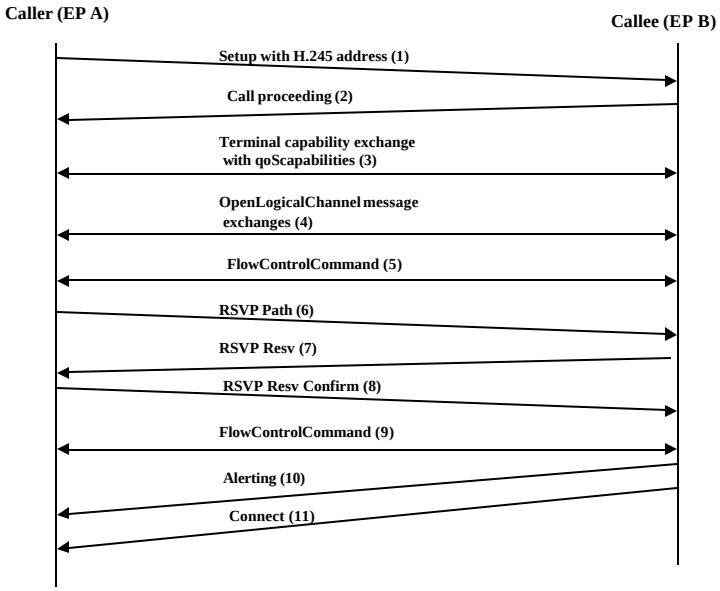


Figure 8: Establishing a call with RSVP

1. EP_A sends a Setup message to EP_B. In this Setup message, EP_A includes the H.245 address to help EP_B initiate the H.245 procedures. Though the H.245 address is optional in a Setup message, it is included to allow H.245 exchanges to occur before alerting the called endpoint. This is required for RSVP synchronization.

2. EP_B retrieves the H.245 address from the incoming Setup message and opens the H.245 control channel to commence the H.245 procedures. To facilitate RSVP synchronization, EP_B does not alert the user and also withholds the Alerting and the Connect message until the reservations are in place. However, it sends a Call Proceeding message to EP_A to indicate that it is continuing with the call setup. This message helps prevent any timer from expiring on the other end.

   EP_A and EP_B exchange the various H.245 messages described in the previous section using the H.245 control channel. The TCS message exchanged between the endpoints must also include a sequence of **qoSCapability** elements to signify their interest in initiating RSVP. Each **qoSCapabilities** contains an **rsvpParameter,** which contains a desired **qosMode**. (Please refer to section 3.6.2 for a description of **qoSCapabilities**). The **rsvpParameters** are arranged based on the preference of the **qosMode.** It is not necessary to complete the remaining values in the **rsvpParameter** at this juncture. The purpose of exchanging the **qosModes** at this time is to indicate that the endpoint is capable of RSVP and also to calculate the derived set.

   Each endpoint can derive the QoS Set for each channel by comparing the **qosModes** received from the other endpoint and its own preferred **qosModes** (Please refer to section 3.6.3 for details regarding the derived QoS set). If the QoS Set contains

guaranteed or controlled load QoS modes, then the endpoint will initiate the RSVP process during its OLC exchange. If the only choice in the QoS set is best effort then the call setup will continue without any RSVP procedures. If the QoS set is empty then it implies that there is no QoS mode that is acceptable to both endpoints. In such as case, the endpoint should initiate release procedures and release the call.

3. Both EP_A and EP_B open their media channels as negotiated in the capability exchange and then exchange the OLC messages. If RSVP is desired (i.e. the QoS set for the channel includes either guaranteed or controlled load) then the OLC message must include a sequence of **qoSCapabilities** in the **transportCapability** field. This sequence is arranged based on the priority of the **qoSMode**. Each element of the **qoSCapability** parameter must contain the channel-specific **rsvpParameters**. The values in the **rsvpParameters** are obtained as follows:
    - **qoSMode**: The QoS modes currently supported includes guaranteed (GQ), controlled load (CL) and best effort (BE). The endpoint will be configured with a list of desired QoS modes in the order of preference. The endpoint must include one QoS Mode in each of the **rsvpParameters** element. If the **qoSMode** in the **rsvpParameters** element is left empty, it is assumed to be best effort. For best effort, the remaining parameters need not be completed.

    - **tokenRate** (bytes/seconds) : This is the rate of the media stream in the channel for which the reservation is requested. Please note that IP/RTP header overhead must be added to the original rate. Here are some examples of calculating the values for tokenRate:

      IP/UDP/RTP Header overhead = 20 bytes (IP header) + 8 bytes (UDP header) + 12 bytes (RTP header) = 40 bytes per packet.

      For audio, the choice of the codec and the length of the sampling period determine the length of the audio sample. For a G.711 codec, with 20 ms sample size, the tokenRate is calculated as follows:

      length of each packet = 8 * 20 = 160 bytes.
      tokenRate = (160 bytes packet size + 40 bytes header) * 50 packets per second
      = 10,000 bytes/second.

      For video streams, the length of the packet is not dependent on codecs and is based on individual implementations. Moreover the sizes of the packets do not remain the same across all packets. So, it might be difficult to estimate the number of packets in a second. However, the number of packets per second is required to calculate the header overhead. If an estimate can be made on the number of packets per second then the calculation proceeds as follows:

      Header overhead can be calculate as:
      Header overhead (bytes/second) = 40 bytes * packets/second

      and the tokenRate can be calculated as:
      tokenRate = header overhead/second + rate of video stream (bytes/second).

      The work around, in the absence of any estimate on the number of packets/second, is to add a certain percentage as allowance for the header. The suggested value is 20%.

    - **bucketSize** (bytes): This value is calculated as the size of the packet times the number of packets in a burst. The burst for audio flows is usually 1 to 2. The bursts for video streams have been observed in current endpoint applications as

4 to 5, which is considered too high. The endpoint must take care to smooth the traffic that it transmits into the network. Bursty traffic is more prone to loss, delay and jitter in the network.

- **peakRate** (bytes): This refers to the maximum bytes/second that the endpoint will transmit at any given time. If the burst is small, as it is in the case of audio streams, the peakRate can be calculated as 1.1 (or 1.2) times the tokenRate.

- **minimumPoliced** (bytes): This should be set to the size of the smallest packet that the endpoint is planning to generate.

- **maxPktSize** (bytes): This should be set to the size of the largest packet that the endpoint is planning to transmit.

  The maximum and minimum packet sizes are easy to calculate for audio streams where the size is a constant and derived from the selected codec and sample size. It is dependent on individual implementations for video streams. The advantage of larger packets is that there are fewer packets and so the total header overhead is less, leading to less bandwidth consumption. The disadvantage is that the packet sizes might be larger than the MTU size of some small links in the network and might cause the reservation to be rejected. A loss of a large packet has worse effects on the quality of the conference than a small one. For these reasons, the sizes of the packets must be chosen carefully.

The values provided in the above parameters must accurately portray the traffic that is to be generated. Any packets that exceed the traffic specification provided will be classified as "non-conformant" by the network device and may get dropped or will not get the QoS it has originally requested.

4. An optional message that can be exchanged at this point is the flowControlCommand message. Once the logical channels are opened, the endpoints are allowed to transmit the media at any time. However, such traffic will not receive any preferential treatment until the reservations are in place. The receiving endpoint may not want to receive any media until the reservations are completed. It may indicate that it is not yet ready to receive any traffic by sending a flowControlCommand message with the maximum bit rate set to 0. Once the transmitting endpoint receives this message, it does not transmit any media until it receives a subsequent flowControlCommand message with a maximum bit rate other than 0.

5. Once the OLC exchanges are completed, both EP_A and EP_B have sufficient information to commence the RSVP procedures. As described above, the RSVP procedures are initiated only for those channels that contain a QoS mode other than best effort. Each of EP_A and EP_B start by listening for an RSVP Path message after it receives the OLC message and is allowed to transmit an RSVP Path message after it sends the OLC Ack message. Since RSVP is unidirectional and also applies to a single stream, the RSVP Path message must be transmitted and received for every channel in both directions. (Even though only one set of such messages in only one direction is shown in the above figure, a set of RSVP messages is required for each direction for each media stream in the call). For example: Let us assume a typical videoconference, with one audio and one video channel, both of which require reserved resources. In such a case, both EP_A and EP_B must send and receive two RSVP Path messages, one for the audio stream and one for the video stream.

   The RSVP Path messages contains the following objects:
   - **Session Id**
     Destination Address: This is the IP address of the other endpoint.

Destination Port: The destination port number is the receiving port of the other endpoint. This can be found in the **portNumber** parameter in the OLC message received from the other endpoint describing the related channel.

Protocol ID: 17. This indicates the transport protocol UDP.

- **Sender Template**
  Sender's address: Endpoint's own IP address

  Sender's port: This is the port on which the media is to be transmitted. This is the same value as that of the **portNumber** parameter in the sender's OLC message.

- **Sender's Tspec:**
  This parameter is filled up from the **rsvpParameters** included in the **qosCapabilities** in the sender's OLC message.

6. On receiving the Path message, EP_B responds with an RSVP Resv message. A Resv message is sent when EP_B wishes to reserve resources for the media traffic it is about to receive. In our example, EP_B will send two Resv message in response to the two Path messages that it receives. Some of the objects carried in the Resv messages are:

   - **Session Id**
     This carries the same information as the session id in the Path message.

   - **Filterspec**
     This is the same as the Sender's template in the Path message.

   - **Style**
     The most appropriate style is Fixed Filter. The reasons for not selecting Shared Explicit style are given in [2].

   - **Flow Descriptor**: As mentioned above, the flow descriptor is made up of flowspec/ **FilterSpec** pairs where each flowspec applies to a particular **FilterSpec**. The filtersp **FilterSpec** ec is collected from the Path message. The flowspec consists of a service class.

     The suggested service class is controlled load. H.323 videoconferencing applications can usually adapt to some minimal jitter and also have some minimal buffering capability and hence can are more suitable to controlled load rather than guaranteed service. Standardization of one service type reduces the chances of interoperability failure and also enables merging of multiple reservations at network devices. With controlled load the only required parameter is the TSpec. It is recommended that the sender's TSpec from the Path message be used.

   - **Resv_Confirm**: This object is included when a reservation confirmation is required. The RSVP confirmation message is the ResvConf message. This object contains the address of the endpoint requesting the confirmation message.

7. If the Resv message received by EP_A contains a ResvConf object then it must send a ResvConf message. This message is directed to EP_B given in the ResvConf object.

8. Once the reservation is in place, EP_B indicates its readiness to receive the media stream by sending a flowControlCommand message with its maximum bit rate set to "no restriction". EP_A can now start sending the media streams.

9. As the reservation is now in place, EP_B continues the call setup procedures. It alerts the user and sends the alerting message.

10. After all the procedures are completed, EP_B sends the connect message to indicate that the call is connected.

4.2.2 A call with gatekeeper routed call signaling

If the gatekeeper routes call signaling as well as the control channel, then the calling endpoint is not aware of the destination endpoint's address information before the alerting phase. Therefore, the endpoint cannot make an end-to-end RSVP reservation with synchronization. In such a case, the gatekeeper may be configured to make the RSVP reservation. The gatekeeper may direct the reservation to the called endpoint or to the called endpoint's gatekeeper depending upon whether the endpoints use the same gatekeeper or not.

Since scalability is an issue with RSVP, network administrators may choose not to enable RSVP in every device in the network. They are usually required in edge devices and in devices where there is a resource constraint such as at WAN edge. If the RSVP reservation made by the gatekeeper passes through different devices than the media, then the reservation made is likely to be inefficient is providing QoS to the call traffic. These issues should be addressed while designing the network in a gatekeeper routed signaling environment.

If the gatekeeper routed just the call signaling and did not route the control channel, then the presence of the H.245 control channel address included in the Setup message could signal to the called endpoint to withhold the alert to the user to facilitate the RSVP process. The endpoints can then initiate and complete the RSVP process as outlined in the previous section. This allows for RSVP synchronization. Since the endpoints themselves initiate the reservation, the issue of reservation made in the wrong device does not arise.

4.2.3 A call with RSVP using H.245 tunneling procedures

In order to conserve resources and shorten setup time, the endpoints may use the Q.931 call signaling channel to perform the H.245 exchanges instead of opening a separate H.245 channel. This procedure is known as H.245 tunneling or encapsulation. This section describes a scenario where the call is established with such tunneling procedures.

In this example (Fig 9), we will assume that both these endpoints are RSVP-capable and desire to use the H.245 tunneling procedures. To indicate the use of the H.245 tunneling procedure, the endpoints must set the **h245Tunneling** element to TRUE in every Q.931 message and must encapsulate the H.245 messages within the Q.931 call signaling channel.

1. EP_A sends a setup message to EP_B. In this setup message, EP_A sets the **h245Tunneling** element to TRUE. With H.245 tunneling, the endpoints do not create a separate H.245 channel. It uses the Q.931 call signaling channel to proceed with the H.245 procedures.
2. EP_B sees the **h245Tunneling** element and decides whether to use the H.245 tunneling procedures. If it does, **h245Tunneling** element is set to TRUE in all its Q.931 messages. If EP_B intents to do RSVP synchronization, it does not send the Alerting or the Connect message until the reservations are in place. However, it sends a call proceeding message to EP_A to indicate that it is continuing with the call setup. In this message, the **h245Tunneling** element is set to TRUE as well.

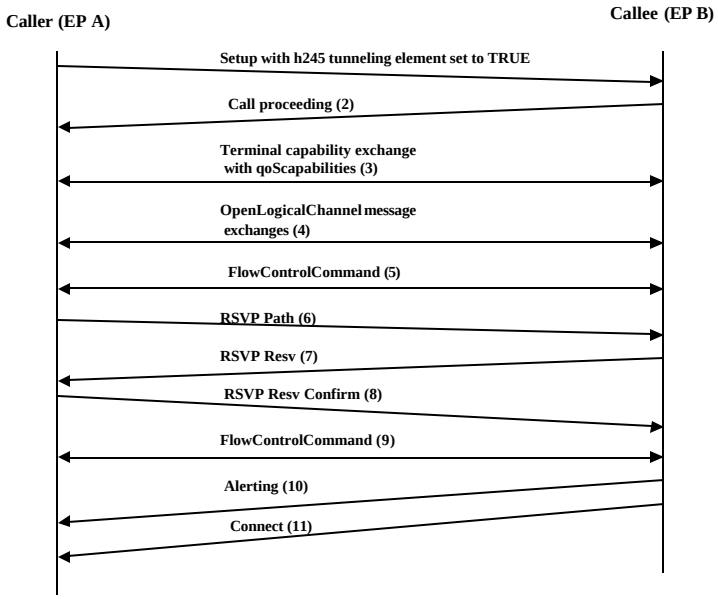


Figure 9: Establishing a call with H.245 tunneling

The rest of the setup process is similar to the steps outlined in the previous section. The only differences are that as long as the H.245 tunneling procedures are required, the Q.931 messages exchanged should have the **h245 tunneling** element set to TRUE and all the H.245 messages transmitted using the Q.931 call-signaling channel. After the H.245 exchanges are completed, the RSVP reservation processes are initiated as detailed above. Once the reservation processes are completed, the user is alerted and the call setup is completed.

4.2.4 A call with RSVP using Fast Connect procedures

Fast connect is another procedure that helps reduce the number of exchanges between endpoints and therefore reduces the call setup time. In this example (Fig 10), we will assume that both these endpoints are RSVP-capable and desire to use the Fast Connect procedures. To indicate the use of the Fast Connect procedure, the endpoints must include the fastStart element in the setup message and in all the Q.931 messages (Call Proceeding, Progress, Alerting and Connect).

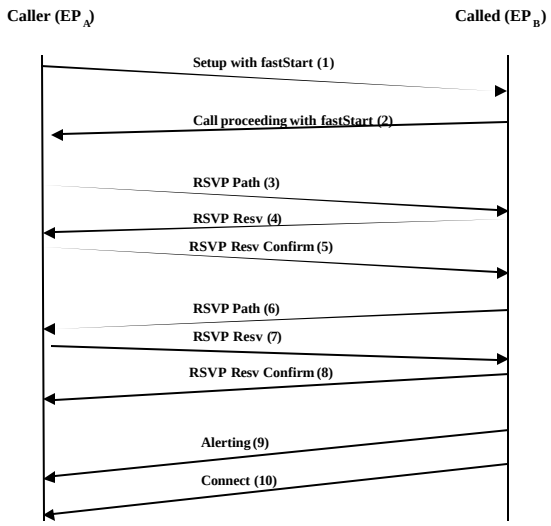


Figure 10: Establishing a call with Fast Connect and RSVP

1. EP_A sends a setup message to EP_B. In this setup message, EP_A includes the fastStart element. The fast start element includes a sequence of OLC structures that indicates the channels on which the endpoint is going to send and receive media. It provides sufficient information to the other endpoint to immediately transmit open the channels and transmit data.
   The OLC structures also contain a sequence of **qoSCapabilities** in the **transportCapability** field. This sequence is arranged based on the priority of the **qoSMode**. Each of the **qoSCapability** elements must contain the channel specific **rsvpParameters**. The values in the **rsvpParameters** are set up as given in section 4.2.1.

   Once EP_A has received the **qosModes** from EP_B, it can now derive the QoS Set for each channel. (Please refer to section 3.6.3 for details regarding the derived QoS set). If the QoS Set contains guaranteed or controlled load QoS modes, then EP_A will initiate the RSVP process. The same will be done by EP_B as well. If the only choice in the QoS Set is best effort, then the call setup will continue without any RSVP procedures. If the QoS Set is empty, then it implies that there is no QoS mode that is acceptable to both endpoints. EP_A should then initiate the release procedures and release the call.

   EP_A should set the **mediaWaitForConnect** element to true to indicate that the media should be sent only after the connect message is transmitted. This indicates to EP_B that to transmit media only after connect, by which time the reservation process is also completed. The same result can also be achieved by sending a flowControlCommand message.

2. Since EP_B also desires to use the fastConnect procedures, it responds to the setup message by issuing the call proceeding message and includes the fastStart element. It also sets the **mediaWaitForConnect** element to true to prevent any media traffic before reservations are established. The OLC structures included in the call proceeding message will include a sequence of **qoSCapability** elements. This sequence will contain only the QoS modes in the derived set.

All ports mentioned in the OLC structures are opened and are ready to receive traffic. The endpoint that has either GQ or CL in the QoS derived set can now commence the RSVP process. EP_A and EP_B should start listening for an RSVP Path message after it receives the setup message with the fastStart element and can transmit an RSVP Path message after it sends the fastStart element in its response. The contents of the RSVP messages are the same as the earlier section (4.2.1).

3. On receiving the Path message, EP_B responds with an RSVP Resv message. A Resv message is sent when EP_B wishes to reserve resources for the media traffic it is about to receive. sd

4. If the Resv message received by EP_A contains a ResvConf object then it must send a ResvConf message. This message is directed to EP_B given in the ResvConf object.

5. As the reservation is now in place, EP_B continues the call setup procedures. It alerts the user, sends the alerting message.

6. Since RSVP reservations are unidirectional, the steps for making reservation have to be repeated by EP_B as well. This is shown in steps 6, 7 and 8. Note that a set of RSVP messages has to be exchanged for each media stream in the call. Hence, even though one set of messages are shown in the above table, two sets of such messages must be exchanged if the call consists of two channels such as audio and video.

9. After all the procedures are completed, EP_B sends the connect message to indicate that the call is connected.

10. Once the connect message is sent, EP_A and EP_B are free to transmit media.

4.3 Releasing a call

If RSVP is involved, the call release procedures must include releasing the reservations as well. If no RSVP refreshes are received for a certain time period then the RSVP state will be automatically deleted. However, exchange of explicit teardown messages is recommended so that the network resources are freed up and made available as quickly as possible. The following steps for releasing a call:

- Close all the open channels after exchanging the CloseLogicalChannel messages.
- Close the H.245 control channel if one had been opened.
- Finally exchange the ReleaseComplete messages and release the call.

These are also explained in Section 3. In addition to the above, the RSVP process in the endpoint must be discontinued. This involves the following steps:

- If receiver, then stop transmitting the Resv refreshes and transmit a ResvTear message for the reservation originally made.
- If sender, then stop transmitting the Path message and transmit a PathTear message to delete the path state in the network.

4.4 Reservation failure

So far, we have assumed that the reservations have been successful. Now let us look at reservation failure. There are 2 levels of decisions that are to be made. They are:

- The first decision pertains to each individual channel. The decision required here is what to do with a media channel for which the reservation has failed. This decision can be derived from the QoS modes in the derived set. The table below gives some examples of the derived

sets and the actions that are required in each case. This decision could cause the media channel to remain un-established.

Table 2: Failure actions

Derived Set	Actions to be taken
{GQ} {CL}	In both these derived sets, best effort is not an option. Hence, if reservation fails, then the media channel is not established.
{GQ, CL}	In this set two QoS modes are provided. Each of the QoS modes refers to a different service type in an RSVP reservation. The RSVP reservation for the first QoS mode (Service Type) is attempted. If reservation fails for the first option, then reservation for the second option is attempted. If reservation requests for both the options are rejected, then the setup of that media channel fails, since best effort is not an option.
{GQ, BE} {CL, BE}	In these sets, best effort is an allowed option. Hence, if reservation fails, then the media channel is established but the RSVP process is discontinued. It is suggested that the endpoint indicate to the user that the call is proceeding without any reserved resource.
{GQ, CL, BE}	This derived set requires that the RSVP reservation be attempted for the first two QoS modes. Since best effort is available as an option, the media channel will be established even if the reservation failed for both the modes. The RSVP process is discontinued and the user is informed that no reservation is available for the call.

- The second decision involves what to do when some of channels in a call have been closed due to reservation failures. A simple decision would be to fail and release the call. However, to allow greater flexibility, an endpoint could decide to allow the call even if some of the channels failed. For example: A videoconferencing call may be allowed to continue as an audio-only call if the video channel was not established due to reservation failure while the audio stream had a successful reservation. The decision to allow the call even if some of the channels are closed is implementation dependent. However, it is suggested that these decisions are made configurable to allow the administrators to choose whatever is best suited for their environment and usage. It is also suggested that user messages are provided to explain the change in the call. Given below are some examples of call handling when some channels are closed.

4.4.1 Example 1: Call Release with reservation failure

In this example, the call consists of 2 channels: one audio and one video channel. Let us assume that the endpoints are configured to fail the call if any media channel is closed.

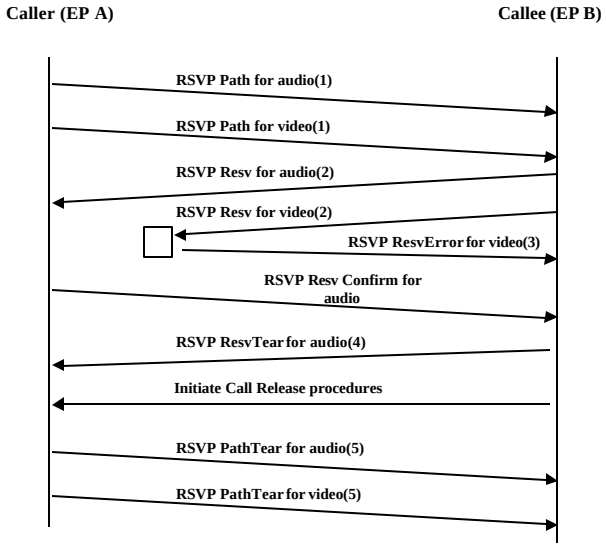


Figure 11: Call failure due to reservation failure

1. EP_A sends a Path message for its audio and video channel.

2. EP_B responds with a Resv Message for both the channels.

3. The reservation for video is denied in one of the network devices along the path and the device sends a ResvError message back to EP_B.

4. Since the reservation for the video channel fails, EP_B decides to close the video channel (Best Effort is not an option in the derived set). On the failure of the video channel, EP_B initiates the release process to release the call. It also sends a ResvTear message for the audio channel to teardown the reservation that has already been made.

5. As part of the release process, EP_A receives a ReleaseComplete message from EP_B. This message indicates the reason for call failure, which in this case is “nobandwidth”. EP_A informs the user of the call failure along with the reason for it. It also sends PathTear messages to tear down the path state that has been established.

4.4.2 Example 2: Modified call with reservation failure

Given below is an example where the reservation for the audio channel succeeds whereas the reservation for the video channel fails. In this case, the call is modified and continues as an audio-only call.

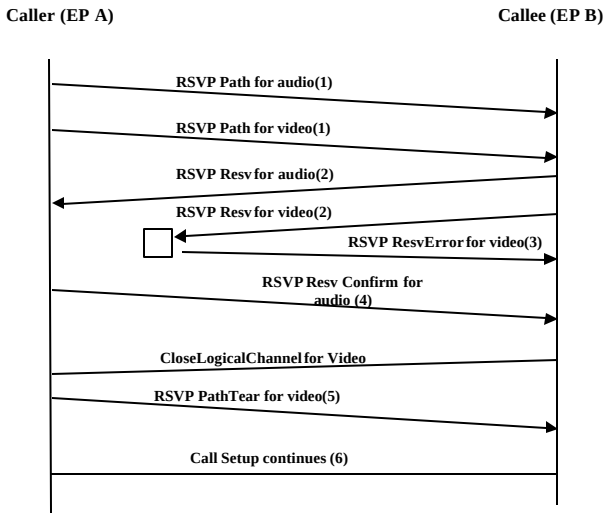


Figure 12: Call continues with reservation failure

1. EP_A sends a Path message for its audio and video channel.

2. EP_B responds with a Resv Message for both the channels.

3. The reservation for video is denied in one of the network devices along the path and the device sends a ResvError message back to EP_B.

4. EP_B receives a ResvConf message for the reservation for the audio channel. Since, the reservation for the video channel fails, EP_B closes the video channel. In this case, EP_B decides to continue the call as an audio-only call as the audio channel has a successful reservation.

5. EP_A receives the reason for the failure of the video channel from the CloseLogicalChannel message and communicates it to the end user. The video channel is closed and EP_A sends a PathTear message to delete the path state for that channel in the network.

6. The call setup continues and establishes an audio-only call.

4.4.3 Retrying a reservation

As explained in the above sections, a call can continue without reservation in two cases:

- If best effort is an allowed option for the channel for which reservation has failed.
- If the call is configured to continue even after the failure of one or more channels.

In the first case, the endpoint implementation could allow for reservation retries. The reservation for the channel can be retried at regular intervals. If successful, the endpoint must provide an indicator to the end user that the channel now has reserved resources. The retry interval is implementation dependent. The endpoint could allow this value to be configurable to allow the administrator to choose an appropriate value.

The Resv message for retrying the reservation cannot be sent in the absence of a path state, which is created by a Path message. Hence, it is necessary that the Path message be continuously refreshed even after a reservation failure. At the end of each retry interval, the receiver endpoint will respond to the Path message by sending a Resv messages to retry the reservation. Once the endpoint receives a ResvConf message to confirm that the reservation has been made, it can inform the user that the channel now has reserved resources.

In the second case, the transmitting endpoint can try to restart the closed channel at certain time intervals. It could open the logical channel by exchanging the OLC messages containing the QoS parameters. By exchanging flow control command the endpoints can disallow media to be transmitted in that channel until the reservation is successful. The RSVP message is exchanged requesting reservations for the newly created channel. If the reservation is successful, then the user is notified and the media is allowed to flow through the newly opened channel.

Thus, this feature enables the call to acquire reserved resources and improved quality during the call even if the reservation had failed originally.

4.5 Modifying a call

The following are some of the ways in which a call may be modified causing the bandwidth used in the call to change. If the bandwidth used is changed, then the RSVP reservation must also be changed:

- **Change in the codec used**: The endpoints may decide to use a different codec from the one selected at call establishment time. This may require the existing logical channel to be closed and a new channel for the newly selected codec to be opened. Hence, the RSVP reservation must also be stopped for the old channel and re-reserved for the new channel.
- **Change in the bit rate used**: The endpoint may decide to either increase or decrease the bit rate used in the call. When an endpoint changes the bandwidth used, it should make a bandwidth request (BRQ) to the gatekeeper. The gatekeeper can either accept or reject the request (BCF/BRJ). This causes the endpoints to close the existing logical channel and open a new one with the new bit rates. If the media channel is closed and re-opened, then the RSVP reservation must also be torn down and requested again. If the physical channel remains the same with no changes to ports or addresses, then the reservation can just be modified. The advantage here is that if the modified RSVP is rejected then the older reservation still remains. In such a case the endpoint may decide to not change to the new bit rate.
- **Opening and closing of media channels:** At any time, the endpoints may decide to either add new channels or release some existing channels without closing the call. If a new channel is added during the call, the RSVP reservation process must be followed. If the reservation is rejected, then the derived set should be used to make a decision about the channel/call. This is similar to the process followed during the call establishment. If a channel is closed then the reservation release process should be followed.
- **Reservation removed due to policy:** At any time of the call, the PDP might revoke any or all of the RSVP sessions. The derived set should be used as described above to make a decision about the call.

4.6 Timers and timeouts

These are some of the timers and timeouts that may be necessary while implementing H.323 RSVP synchronization:

Call Setup timer: This timer is used to ensure that the sending endpoint does not wait indefinitely for a call to be setup. This timer is set as soon as the Setup message is sent. If the timer expires before the completion of the setup, then the call setup is aborted and assumed to have failed.

Resv Timer: The purpose of this timer is to prevent a receiving endpoint from waiting endlessly for a ResvConf or a ResvError message after sending a Resv message. This timer is set as soon as the Resv message is sent and when it expires the endpoint automatically assumes that the reservation has been rejected. If a ResvError or a ResvConf messages arrives before the timer expires, then the timer is cancelled. The period for this timer is implementation dependent. However, It is suggested that the time be the same as the cleanout timer interval (see below).

Refresh Interval: This value indicates how often a Resv or a Path message will be refreshed by the endpoints. This value is inserted in the RSVP messages and is used by the network devices to calculate the timeout of these Resv or Path state. This value may be decided by the endpoints, however, the RSVP specification suggests 30 sec. Bandwidth consumption, memory and processing power are all overheads that increase with more frequent refreshes. On the other hand, changes in routes and reacting to route and other failures happen more quickly with more frequent refreshes. These factors have to be considered when deciding on this refresh interval.

Cleanout timer Interval: This timer interval is discussed in section sd. However, in many implementations, this value is calculated as 3 times the refresh interval.

5 Other QoS Elements

In addition to the synchronization with H.323, there are other QoS elements that can be used in conjunction with RSVP for a more complete QoS solution. These elements are discussed in this section.

5.1 DSCP Marking

Most networks are not likely to strictly adopt just Integrated Services architecture or Differentiated Services architecture. Instead these networks are likely to contain a mixture of the two. Even though the RSVP messages were designed to flow end to end, they will flow transparently within the DiffServ portion of the network [3] [10]. In the DiffServ portion of the network, there will be no explicit admission control or flow-based processing. The traffic will be classified, queued and scheduled using the DiffServ traffic control mechanisms to provide differentiation among the traffic. A common way to classify traffic is to use the DiffServ Code point (DSCP) [2] value in the packet.

The DSCP value may be marked by the endpoints that originate the packet or by the network device through which they traverse. Since the endpoints can do flow-based differentiation, they are the best candidates to mark the packets. However, since endpoints are usually outside the control of the administrators and can knowingly or unknowingly misuse the network's resources, the network administrators are usually reluctant to trust the DSCP markings that originate at these endpoints. The network device can also mark the packet. However, since the network devices cannot make a flow-based differentiation, these markings may not be as granular or appropriate as the marking by the endpoints. It may be possible to recognize the audio packets based on its traffic characteristics, but unless there are some intelligent mechanisms in the network device, it is much harder to differentiate video or control packets from all other packets in the network.

In spite of the still existing dilemma of who should mark the packets, it is suggested that the endpoints mark their packets so that they may be used if there are no trust-related issue. It is important to allow configurations of these values so that they may be modified to suit the individual deployments. The implementations should also allow different values to be set based on whether the related RSVP reservation is accepted or not. The suggested values for the different streams are given in the appendix 1.

RSVP in the network devices could also be used for flow-based differentiation. That is, the RSVP in the network devices may be configured with the values that will be automatically used by the device to mark the packets for which the reservation has been successful. However, this functionality should be synchronized with application-oriented policy control, otherwise the differentiation might not be sufficient to be very useful for IP videoconferencing.

5.2 Policy Control

One question that network administrators often ask is how to allow reservations from only a select number of applications or endpoints and to disallow all other reservations. The network administrators want to be able to control the applications that make reservations of network resource. Additionally, they want to control the amount of bandwidth reserved by each of these allowed applications. To implement such restrictions, it is necessary for the network devices to securely identify the application or the user of the application for which the reservation is requested. While some of this identification can be done using access lists on the router using the application's IP address, ports etc., it cannot be used in all circumstances and also does not offer a scalable solution. Take, for instance, applications that reside in desktop PCs that host multiple applications and dynamically select the ports to use. In such cases, identification of these applications' traffic becomes difficult. A better solution is to use the policy management aspect of RSVP. By implementing policies and checking the reservations for policy information, the network administrator can prevent reservations from some applications while granting others. They can also restrict the amount of bandwidth used for individual reservations.

Section 2.4.3 mentions the inclusion of a policy object in a RSVP message to support a policy control system. To ensure a policy control system, the endpoints and the network devices must operate in tandem. The policy information inserted by the endpoint application in the RSVP message must coincide with the policies in the network. Devices in the network must also be configured to perform policy control.

An endpoint application can insert a policy object describing the application's identity in both the Path and the Resv message. To identify an application, RFC 3182 & 2750 recommend the inclusion of the following attributes:

- Application identity
  - GUID: This optional parameter refers to a globally unique identifier. The GUID must be chosen carefully so that it is unique throughout the network. At present, there is no central authority handing out these GUIDs, nor is there is a procedure to register chosen GUIDs. So, the RFC recommends publishing this information.
  - APP: This refers to the name of the application. Example: "H.323 videoconferencing".
  - VER: This refers to the version of the application. Example: 4 (to imply version 4 of the H.323 specification).
  - SAPP: This optional parameter refers to a sub-application and can be used to better identify the application. Example: video stream (in a H.323 videoconferencing)
  - All the above information is packaged together in a X.500 DN format as an ASCII string.

- Credential
  This attribute contains the credential information. Some of the choices for this are Kerberos ticket, digital certificate or the application name in plain ASCII. One must ensure that the choices used in the endpoint are supported in the network.

This policy information is carried in a RSVP message to the network device that makes the reservation decision. The network device consults the PDP either available externally in a policy server or available in the device itself. The PDP looks up the policy database and decides whether the application identified by the policy information is entitled to the requested reservation and communicates the decision to the network device. This mechanism enables the network administrator to control the reservation system.

5.3 Resource Reservations and MCUs

A Multipoint Control Unit (MCU) is just an endpoint in the network that provides support for multipoint conferences. An MCU consists of: a multipoint controller (MC) and the zero or more multipoint processors (MP). The MC is a mandatory component, which is responsible for negotiating the capabilities necessary to set up a conference between the different endpoints. The MP is an optional component that is responsible for processing of the media streams received from different participants in a conference.

Multipoint conferences may either be centralized or decentralized. In a centralized conference all endpoints exchange the call signaling and the media with the MCU in a point-to-point manner. The MC receives all the H.245 signaling and performs the multipoint control functions. The MP receives and processes the media and then transmits it back to the other endpoints. In a decentralized conference, the endpoints exchange the H.245 signaling with the MC in an MCU but they directly multicast the media to the other endpoints. The H.323 specifications also list combinations of the two methods.

5.3.1 RSVP for a centralized multipoint conference

The MCU performs the RSVP reservation process the same as any endpoint. Each call that is in the conference is treated by the MCU as an independent call for RSVP reservation. Hence, for a conference with 3 endpoints, the MCU makes 3 independent RSVP reservations. However, there are the following issues:

- **End-to-end reservations**: The endpoint that is in a conference makes a reservation for the incoming stream. This stream is sourced at the MCU and not at the other endpoint. If the reservation for some endpoints succeeds while others fail, then the endpoints with the successful reservations assume that they have reservations end to end. However, they only have reservations up to the MCU and not to the other endpoint.
- **Continuous presence**: In a continuous presence mode, the MCU mixes media from different sources to form a single stream. The different streams coming into the MCU might not all have successful reservations. This may cause some packets in some streams to be delayed or lost, causing difficulty in mixing and ultimately affecting the conference quality.

One solution to the above is to configure the MCU to fail the call if any reservation fails. While this may appear to solve the above problems, it may be confusing to the user to have his call not complete due to reservation failure especially if his endpoint is set to allow best effort traffic.

Another solution is to only allow endpoints into the conference for which an RSVP reservation succeeds, disconnecting those endpoints that fail to obtain an RSVP reservation. Combinations defined in section 4.4 apply, such as failing the video channel only, or failing the entire call with that participant. This solution works fine only if the endpoint in question has no best effort in the derived set.

The recommended solution is to treat the conference as one entity and not as N calls for the RSVP process. This is done by defining a **qosMode** for an entire conference. Any participant to the conference is governed by the conference's **qosMode** instead of the individual qosMode set in its application. If best effort is not one of the options then only the endpoints having a reservation to and from the MCU are allowed. If best effort is one of the **qosModes** configured, then the endpoints are allowed in the conference irrespective of their reservation status. The call is continued with best effort status if even one of the reservations fail.

5.3.2 RSVP for a multicast conference

Multicast is used in a decentralized conference, where the media streams are directly transmitted to the various endpoints using a multicast address. Multicast is also used in hybrid conferences where the endpoints multicast either the audio or the video channel while using centralized multipoint for the other channel. The MCU may also allow asymmetric conferences where some of the endpoints receive their media streams from the MCU directly while others participate in a multicast distribution.

The MC is responsible for negotiating and controlling all the call control capabilities using the H.245 signaling. In the H.245 message, the MC indicates the mode of the media such as centralized audio, decentralized audio etc. The MC assigns and distributes the multicast address to the endpoints. When an endpoint receives an OLC message from the MC, it collects the multicast address and joins the multicast group. This is done prior to sending the OLC Ack message. The endpoint also transmits the flowControlCommand to indicate that it is not yet ready for receiving media. However, if some endpoints are already transmitting media, they might not stop their transmission and disrupt the call to the other endpoints. In such a case the endpoint will receive media with no reserved resources until the reservations are in place. The endpoints can make its reservation after it has joined the multicast group and received the RSVP Path message. It transmits its RSVP Path message to the multicast address. The call flow is given below:

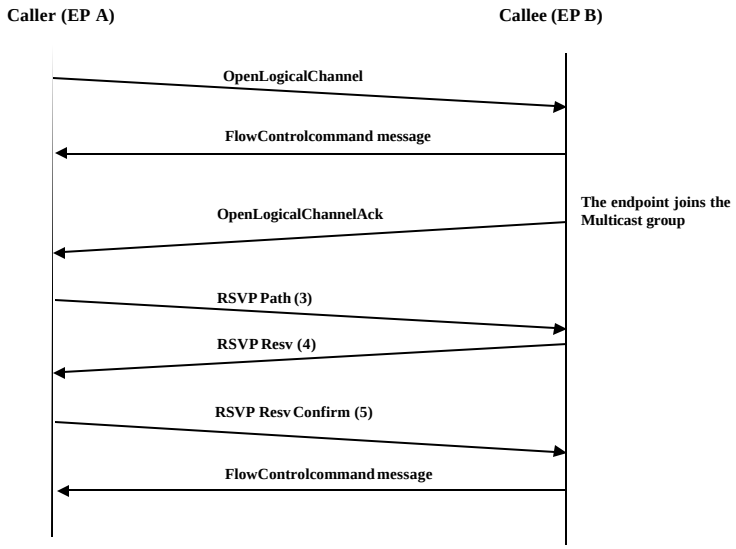


Figure 13: Resource Reservation with Multicast

In a multicast conference, an endpoint may receive multiple Path messages from different endpoints. The endpoint can transmit one Resv message to make a reservation for traffic originating from different endpoints. The suggested style is FixedFilter. This implies distinct reservation for each sender. The flow descriptor that applies to the FF style is given in Section 2.1.5. For the FF flow descriptor, the endpoint provides the flowspec pertaining to each endpoint and has no need to merge the different tspecs received from the different endpoints. Once the endpoint receives the ResvConf message, it can send the flowControlCommand message with the maximum bit rate specified to indicate that it is ready for traffic.

5.4 Interoperability Issues

In most medium and large deployments, endpoints from multiple vendors can be found. Some endpoints are likely to support RSVP sooner than others and the features supported may vary from vendor to vendor. In addition, the network administrators are not likely to enable RSVP in the entire network at one single time. All these factors could cause interoperability problems that will need to be addressed. This section summarizes some of the issues and lists some steps that can be taken minimize their impact.

5.4.1 All endpoints are not RSVP capable

If an RSVP reservation is made mandatory for all calls, then all calls between RSVP-enabled endpoints and endpoints that do not do RSVP are likely to fail. Any one of the following solutions could be adopted to prevent such failures:

- Make RSVP reservations optional: While configuring the QoS modes for an endpoint, best effort can be added as an option. In such a case, the RSVP reservation will be attempted. If one of the endpoints in not RSVP-capable, then the reservation cannot continue and will fail. However, the call will be allowed to proceed as best effort.

- Gatekeeper reservations: If the gatekeeper is capable of making RSVP reservation, then it should attempt to do so in the absence of RSVP-enabled endpoints. If the ARQ message to the gatekeeper does not contain a **transportQoS** field or contains the transport QoS field with any option other than endpoint-controlled, then it implies that the endpoint is either incapable or disinterested in making its own reservation. In such cases, the gatekeeper can perform the reservation process on behalf of the endpoints. However, as mentioned earlier, the reservation made by the gatekeeper might not be as effective as the endpoint RSVP unless they reside in the same LAN.

- Use of proxy: If available, an RSVP proxy device may be used to provide the RSVP capability on behalf of the endpoint. Note that proxy devices might need to insert itself in the middle of the call signaling so that the results of the reservation can be synchronized with the call progress. An example of such a proxy device is the Cisco MultiMedia Conference Manager (MCM).

Even though the gatekeeper and RSVP proxy serve the same function of securing RSVP reservation for streams originating from endpoints that do not have RSVP capability, there are some significant differences. An RSVP proxy can be situated in the path of the media stream and in some cases can receive and forward as well. The advantages are (1) the reservation will be made in the same devices that the media will flow through and (2) other QoS features such as marking the packets with a TOS value can be enabled as well. The disadvantage is that it may lead to an increase in end-to-end delay and may have limited support for non-standard features employed between endpoints from the same vendor. Since gatekeepers do not reside in the middle of the data path, it is not reasonable for them to proxy the media.

Another major issue with using the gatekeeper or the proxy to make the reservation is to figure out the parameters for making the reservation. The number of flows and the parameters of the flows are not known to the gatekeeper and the proxy at the time of making a reservation. Hence it might have to guess the "parameters" from the total bandwidth exchanged in the ARQ or the Setup message. They may have to make a bigger reservation in the beginning and then adjust it when the parameters are known. For example, if the bandwidth indicates 384 kbps in each direction, then the proxy can assume that the bandwidth is for both audio and video. However, not knowing the exact bandwidth used by audio it reserves 64K for audio. 64 kbps is assumed for audio as it covers most of the popular audio codecs used today. The reservation for the video stream is made at 384 kbps. The proxy learns the actual parameters of the flow during the H.245 control signaling after which it adjusts the reservations for the video stream to the correct bandwidth. If the gatekeeper routes the H.245 control channel then it too learns the parameters during the H.245 control signaling.

5.4.2 All endpoints do not support RSVP synchronization

To support RSVP with synchronization, significant changes may have to be made to the call depending on the call model that is supported in the endpoints. So, initially, endpoint vendors may choose to support RSVP without synchronization. Even if one endpoint does not support synchronization, the call will not have any synchronization. This will result in the call suffering from all the effects of not synchronizing such as not having pre-ring capability and not being able to fail the call. The problems surface mainly if the reservation fails.

5.4.3 Endpoints supporting different call models

Endpoints can support different call models such as the one requiring the opening of a separate H.245 channel, Fast Connect, and as well as H.245 tunneling. Since environments are likely to have multiple endpoints, there is likely to be a mismatch in the call models that support RSVP in the individual endpoints. For example, one endpoint might support RSVP with fast connect while another might support it with H.245 tunneling. This could cause interoperability issues. It is suggested that the endpoint vendors support as many call models as they can with RSVP to prevent such interoperability issues. At the least, it is suggested that the ability to send the H.245 address in the setup message is supported.

5.4.4 Gatekeeper is not QoS aware

An RSVP-enabled endpoint will first ensure that it is allowed to take charge of its own reservation by including the **transportQoS** parameter in its ARQ message to the gatekeeper. Since this parameter is optional, some gatekeepers may fail to recognize it. These gatekeepers must just ignore the parameter and not cause any adverse action to this parameter's presence. The resulting ACF message from the gatekeeper to the endpoint is therefore not likely to contain the **transportQoS** back. The endpoint should then assume that the gatekeeper is unaware of QoS and must make its own RSVP reservations.

6 Summary

It is well known that resource reservation in the network is essential for IP-based videoconferencing. This is primarily because conferencing streams are easily affected by delay, jitter and loss in the network. The document "QoS for IP videoconferencing" discussed why current gatekeepers are insufficient to solve this problem and why RSVP is the best choice. This document provided the details of how one could implement RSVP in the endpoints.

We all are aware that high quality of service is one of the most important factors that might influence the success of IP videoconferencing and may lead to more wide spread deployments. Hence, more vendors must support means of providing for this quality of service. Towards that

end, this document hopes to assist a developer in enhancing their endpoint with QoS features in a H.323 IP Videoconferencing environment.

7 References

1. Resource Reservation Protocol (RSVP): [RFC2205]
   http://www.ietf.org/rfc/rfc2205
2. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474),
   http://www.ietf.org/rfc/rfc2474.txt
3. An Architecture for Differentiated Services (RFC 2475)
   http://www.ietf.org/rfc/rfc2475.txt
4. An Assured Forwarding PHB [RFC 2597]
   http://www.ietf.org/rfc/rfc2597.txt
5. An Expedited Forwarding PHB [RFC 2598]
   http://www.ietf.org/rfc/rfc2598.txt
6. The COPS (Common Open Policy Service) Protocol (RFC 2748)
   http://www.ietf.org/rfc/rfc2748.txt
7. COPS usage for RSVP (RFC 2749)
   http://www.ietf.org/rfc/rfc2749.txt
8. RSVP Extensions for Policy Control
   http://www.ietf.org/rfc/rfc2750.txt
9. Application and Sub Application Identity Policy Element for Use with RSVP (RFC 2872)
   http://www.ietf.org/rfc/rfc2872.txt
10. A Framework for Integrated Services Operation over Diffserv Networks (RFC 2998)
    http://www.ietf.org/rfc/rfc2998.txt
11. Identity Representation for RSVP (RFC 3182)
    http://www.ietf.org/rfc/rfc3182.txt

8 Appendix 1: Suggested Configurations

This appendix provides the collection of all the configurations that have been suggested in this document.

8.1 Gatekeeper

The following configurations may be offered in the gatekeeper.

Resource Reservation

- Not supported
- Endpoint Controlled (This is the recommended setting for most cases.)
- Gatekeeper Controlled

8.2 Endpoints

The following configurations may be offered to the endpoints as per the discussion above:

QoS method
Any one of the following may be selected.

- RSVP: RSVP process is initiated
- DSCP: The packets will be marked with the configured DSCP value.
- Both RSVP and DSCP: Will initiate RSVP as well as mark the packets as configured

8.2.1 RSVP Parameters

The following parameters are applicable if either “RSVP” or “both” options are selected above:

8.2.1.1 RSVP reservation type/QoS mode:

- Guaranteed
- Controlled Load
- Best Effort

These selections form the qosMode set for the particular stream for this endpoint and will be used to create the derived set as described in section 3.6.3.The above reservation types are applicable per stream. Allow selecting more than one reservation type. Since multiple values are to be selected, allow a preference value for each selection or require putting them in an ordered list. If no selection is made then best effort is assumed.

The QoS modes selections define the failure action for the conference. For example, if best effort is not selected for any of the streams then it implies that the call is to fail if reservation files. However, if best effort is allowed for audio stream and not for the other stream, then it implies that even if reservation failed, the call will proceed as a audio only call.

8.2.1.2 Retry interval

If the endpoints support retries a reservation, then the retry interval may be made configurable.

8.2.2 DSCP Parameters

The DSCP parameters configured may depend on whether the RSVP is also enabled and if so, the results of that reservation.

8.2.2.1 DSCP values

The following parameters are applicable if either “DSCP” option is selected in the QoS method option described above. These parameters are also used when the option selected is “both” and the RSVP reservation is successful. Please refer to appropriate RFCs for explanation of the different DSCP values [4] [5].

Table 3: DSCP Values

Stream	DSCP	Binary
Audio	AF41	100010
Video	AF41	100010
Control	Class 3	011000
Audio only	EF	101110

If “both” option is selected and the reservation fails, then the traffic may be marked with 0 to represent best effort.