PacketCable™ Dynamic Quality-of-Service Specification

PKT-SP-DQOS-I07-030815

Issued

Document Status Sheet

Document Control Number:	PKT-SP-DQOS-I07-030815			
Document Title:	PacketCable™ Dynamic Quality-of-Service Specification			
Revision History:	I01 – First issued release 12/01/99			
	I02 – Second issued release 8/18/00			
	I03 – Third issued release 1/16/02			
	I04 – Fourth issued release 10/18/02			
	I05 – Fifth issued release 11/27/02			
	I06 – Sixth issued release 4/15/03			
	I07 – Seventh issued release 8/15/03			
Date:	August 15, 2003			
Status:	~~Work in Progress~~	~~Draft~~	Issued	~~Closed~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ PacketCable/Vendor~~	Public

Key to Document Status Codes:

Work in Progress	An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
Draft	A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
Issued	A stable document, reviewed, tested and validated, suitable to enable cross-vendor interoperability, and for certification testing.
Closed	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Contents

Table of Figures

This page intentionally left blank.

1 INTRODUCTION

1.1 Purpose

This specification describes a dynamic Quality-of-Service (QoS) mechanism for the PacketCable™ project. CableLabs® has issued this specification to facilitate design and field-testing leading to the manufacture and interoperability of conforming hardware and software by multiple vendors.

PacketCable is a set of protocols developed to deliver Quality of Service enhanced communications services using packetized data transmission technology to a consumer's home over the cable network. PacketCable utilizes a network superstructure that overlays the two-way data-ready cable television network. While the initial service offerings in the PacketCable product line are anticipated to be Packet Voice and Packet Video, the long term project vision encompasses a large family of packet-based services.

1.2 Scope

This document addresses requirements for a client device to obtain access to PacketCable network resources. In particular, it specifies a comprehensive mechanism for a client device to request a specific Quality of Service from the DOCSIS® network. Extensive examples illustrate the use of the specification.

The scope of this specification is to define the QoS Architecture for the "Access" portion of the PacketCable network, provided to requesting applications on a per-flow basis. The access portion of the network is defined to be between the Multi-media Terminal Adapter (MTA) and the Cable Modem Termination System (CMTS), including the DOCSIS network. The method of QoS allocation over the backbone is unspecified in this document. Introduced is the concept of an RSVP boundary, which may be located at the CMTS. Between the MTA and the RSVP boundary per-flow QoS is performed. Interface to the managed IP backbone and issues related to IP multicast are not within the scope of this document. This specification also recognizes that per-flow reservations may be required within the customer premises, and the protocol developed addresses this potential need.

To summarize, the scope of this document is:

- Allocation of QoS between the MTA and the CMTS.
- Specification of the interfaces which are available for control and delivery of QoS in PacketCable Networks.
- Support of multiple PacketCable subscriber configurations – both embedded and standalone MTA configurations.
- Support of both Network-based Call Signaling (NCS) [11] and Distributed Call Signaling (DCS) [10] models.

This specification assumes that DOCSIS QoS (specifically RFI version 1.1 [9]) is used to control and deliver QoS across the DOCSIS networks.

From time to time this document refers to the voice communications capabilities of a PacketCable network in terms of "IP Telephony." The legal/regulatory classification of IP-based voice communications provided over cable networks and otherwise, and the legal/regulatory obligations, if any, borne by providers of such voice communications, are not yet fully defined by appropriate legal and regulatory authorities. Nothing in this specification is addressed to, or intended to affect, those issues. In particular, while this document uses standard terms such as "call," "call signaling," "telephony," etc., it should be recalled that while a PacketCable network performs activities analogous to these PSTN functions, the manner by which it does so differs considerably from the manner in which they are performed in the PSTN by telecommunications carriers, and that these differences may be significant for legal/regulatory purposes. Moreover, while reference is made here to "IP Telephony," it should be recognized that this term embraces a number of different technologies and network architecture, each with different potential associated legal/regulatory obligations. No particular legal/regulatory consequences are assumed or implied by the use of this term.

1.3 Specification Language

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or event useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to included the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

Other text is descriptive or explanatory.

1.4 Phasing of Requirements

The requirements contained in this specification cover two separate mechanisms for a client device to obtain access to PacketCable network resources. The mechanism of RSVP (contained in Section 3) are applicable to either embedded or standalone MTAs, while the mechanisms of the DOCSIS interface (contained in Section 4) are applicable only to embedded MTAs.

In the initial phase of PacketCable, only the embedded MTA interfaces are REQUIRED, and the CMTS MUST support the DOCSIS MAC interface of Section 4. MTAs therefore MUST utilize the mechanisms of Section 4, even though these mechanisms will become optional with future releases of PacketCable.

In later phases of PacketCable, the full specification is REQUIRED. The CMTS MUST support both mechanisms, standalone MTAs MUST support the RSVP mechanism of Section 3, and embedded MTAs MAY support either the RSVP mechanism of Section 3 or the DOCSIS MAC interface of Section 4.

2 TECHNICAL OVERVIEW

Enhanced Quality of Service is required for supporting interactive multimedia applications. Resources may be constrained in segments of the network, requiring allocation of resources in the network. The scope of this specification is to define the Quality of Service Architecture for the “Access” portion of the PacketCable network. The access portion of the network is defined to be between the Multimedia Terminal Adapter (MTA) and the Cable Modem Termination System (CMTS), including the DOCSIS network. This specification also recognizes that per-flow reservations may be required within the customer premises, and the protocols developed herein address this potential need. Although some segments of the backbone network may require resource reservation to provide adequate quality of service, the protocols for backbone resource management are considered to be outside the scope of this specification.

Resources are allocated on the DOCSIS network for individual flows associated with each session of an application, per subscriber, on an authorized and authenticated basis. A DQoS session, or simply a session, is defined by this specification to be a single bi-directional data flow between two clients. When a multi-media application needs multiple bi-directional data flows (e.g., one for voice and a separate for video), separate DQoS sessions are established for each. Applications may use only half of the session’s bi-directional data flow, thereby providing send-only or receive-only services. For example, in a typical voice communications application, a simple communication between two parties is implemented by a single session, while complex, multiparty communications (e.g., “conference calls”) are implemented by multiple simultaneous sessions.

Two PacketCable Call Signaling protocols are being defined – Network-based Call Signaling (NCS) and Distributed Call Signaling (DCS). This Dynamic QoS specification is the underlying QoS framework for both of these call signaling protocols. QoS is allocated for flows associated with a session in concert with the signaling protocol.

This specification introduces the concept of a segment-by-segment QoS framework. It exploits the information available from signaling protocols to perform the QoS on both the “local” segment (on the DOCSIS network close to the originating party) and the “remote” segment (the DOCSIS network close to the terminating party). Thus, this specification allows different providers to use the most appropriate mechanisms for the segment that they are managing. Using a concatenation of the segments with QoS, end-to-end QoS assurance for the session is provided.

The Dynamic QoS specification incorporates protocols to enable providers of packet-based voice communications using the PacketCable framework to use different charging models, including both flat-rate charging as well as usage-based charging. It is the intent of this specification to ensure that enhanced QoS is provided only to authorized and authenticated users. The specific techniques used for authorizing and authenticating a user are beyond the scope of this specification.

This Dynamic QoS specification recognizes the requirements of a commercially viable voice communications service analogous to that offered by means of the public switched telephone network. It is important to ensure that resources are available before the two parties involved in the session are invited to communicate. Thus, resources are reserved before the recipient of the communication is notified that someone is trying to initiate a communication. If there are insufficient resources for a session, then the session is blocked.

The protocols developed in this specification explicitly recognize the need to ensure that there is no potential for fraud or theft of service by end-points that do not wish to cooperate with the call signaling and QoS signaling protocols with the intent of avoiding being charged for usage. This specification introduces the concept of a two-phase for resource reservation (reserve and commit). The two-phases allow a provider to both allocate resources only when they are required (when the voice path is cut-through) which may be used for billing. Further, because the second phase to commit resources requires an explicit request from the MTA, it enables the provider to prevent fraud and theft of service.

2.1 PacketCable QoS Architecture Requirements

The following list presents the QoS requirements for supporting multimedia applications over PacketCable Networks.

1. *Provide PacketCable accounting for the QoS resources on a per-session basis.*

It is anticipated that, from a billing perspective, one of the resources that will need to be accounted for is the use of QoS in the DOCSIS network. Thus, information needs to be identified and tracked that allows reconciliation of the use of the DOCSIS QoS resource with PacketCable session activity.

2. *Both two-phase (reserve-commit) and single phase (commit) QoS activation models.*

Under application control it should be possible to utilize either a two-phase or single-phase QoS activation model. In the two-phase model the application reserves the resource, and later commits it. In the single phase model both reservation and commitment occur as a single autonomous operation. As in the DOCSIS model, resources that are reserved but not yet committed are available for temporary assignment to other (e.g., best effort) service flows. The current specification provides mechanisms for both two-phase and single-phase activation for embedded MTAs, and for two-phase activation for standalone MTAs. Single phase activation for standalone MTAs is deferred to later releases of this specification.

3. *Provide Packet Cable defined policies to control QoS in both the DOCSIS network and the IP backbone.*

It should be possible for different types of sessions to have different QoS characteristics. For example, sessions within a single MSO provider's domain may receive different QoS than sessions outside the domain (e.g., international sessions including links to the PSTN). This dynamic QoS specification may allow an MSO to provide different QoS for different types of customers (e.g., higher QoS for subscribers of a business service at certain times of the day compared to residential customers), or different types of applications for a single customer.

4. *Prevent (minimize) abusive QoS usage.*

Two types of abusive QoS usage are identified: that which is accurately billed but leads to denying service to others, and that which is not accurately billed and leads to theft of service. Subscriber applications and PacketCable applications (either embedded or PC-based) may inadvertently or intentionally abuse their QoS privileges (e.g., use of enhanced QoS, which the provider wants limited to voice applications, by an FTP application). Even though the DOCSIS network is expected to enforce a subscriber's access to QoS, rich packet classification and signaling control mechanisms should exist to keep the subscriber (and the subscriber devices) from fraudulent use of QoS. Admission control procedures should be employed to reduce denial-of-service attacks.

5. *Provide admission control mechanisms for both upstream and downstream directions in the DOCSIS network.*

Both upstream and downstream QoS should be subject to per-session admission control.

6. *DOCSIS QoS.*

It should be possible to police (defined as marking, dropping, or delaying packets) all aspects of QoS defined in the service at the CMTS using the DOCSIS QoS mechanisms. Furthermore, it should be possible to support multiple flow mapping models – single PacketCable session to a single Service Flow and multiple PacketCable sessions to a single Service Flow.

7. *Policy is enforced by the CMTS.*

Ultimate policy control is entrusted to the CMTS. The philosophy is that any client can make any QoS request, but the CMTS (or an entity behind the CMTS) is the only entity entrusted to grant or deny QoS requests.

8. *PacketCable entities must be as unaware as possible of specific DOCSIS QoS primitives and parameters.*

For PacketCable, like any other application that uses the IP-network, the design objective is to minimize the amount of access-link-specific knowledge contained within the application layer. The less access-link knowledge in the application layer, the more applications will be available for development and deployment, and the fewer testing and support-problems will be encountered.

9. *Reclamation of QoS resources for dead/stale sessions.*

It is necessary to re-claim and re-allocate precious QoS resources for sessions that are no longer active, but have not been properly torn down. There should be no resource ‘leaks’ in the DOCSIS link. For example, if a PacketCable client module malfunctions in the midst of a PacketCable session, all DOCSIS QoS resources used by the session should be released within a reasonable period of time.

10. *Dynamic QoS policy changes.*

It is desirable to dynamically change QoS policies for subscribers. For example, this requirement addresses the ability to change a customer’s service level (e.g., upgraded from a “bronze” service to a “gold” service) on-the-fly without resetting the cable modem.

11. *Absolute minimum session setup latency time and post pickup delay.*

The PacketCable Network should allow for emulation and enhancement of the PSTN experience to the user, and should be equally good, if not better, in session setup and post pickup delay metrics.

12. *Multiple concurrent sessions.*

It is desirable to allocate QoS resources (e.g., bandwidth) for not only individual point-to-point sessions, but also for multiple point-to-point sessions (e.g., conference calls, combined audio/video calls).

13. *Dynamic adjustment of QoS parameters in the middle of PacketCable sessions.*

It should be possible for the Packet Cable service to change QoS mid-session, e.g., network-wide resource adjustments or creation of compatible CODEC parameters (necessitating QoS changes), or user defined feature to vary QoS levels, or detection of fax or modem streams (necessitating change from compressing CODEC to G.711).

14. *Support multiple QoS control models.*

Strong cases can be made for both subscriber-side and network-side initiation of QoS signaling. In subscriber side signaling, an application can initiate its request for QoS immediately when the application believes it needs QoS. Also, subscriber side signaling supports application models that are peer-to-peer. In network-side signaling, implementation of the end-point application can be completely unaware of QoS (especially in the DOCSIS network). Network-side signaling supports application models that are client-server (with the server being trusted). It is expected that both models will be present in PacketCable (and other DOCSIS application) networks. The current specification is for subscriber-side signaling only.

15. *Support both embedded-MTA and standalone-MTA QoS signaling*

It should be possible to signal QoS from both an embedded-MTA and standalone-MTA. In a standalone MTA the only signaling path supported is that specified herein using RSVP. In an embedded MTA, both RSVP and direct access to the DOCSIS MAC signaling is possible (via an internal interface as suggested in the RFI [9] specification).

2.2 IP QoS Access Network Elements

The following network elements are employed to support QoS for PacketCable Networks.

2.2.1 Multimedia Terminal Adapter (MTA)

The PacketCable network client device (i.e., the MTA) can be one of the following devices. These devices reside at the customer site and are connected through the DOCSIS 1.1 channel to the network. All MTAs are assumed to implement some multimedia signaling protocol, such as DCS [10] or NCS [11]. An MTA may be either a device with a standard two-wire telephone set in the MTA-1 configuration, or may add video input/output capabilities in the MTA-2 configuration. It may have minimal capabilities, or may implement this functionality on a multimedia personal computer, and have all of the capabilities of the PC at its disposal.

From the point of view of QoS, there are two types of MTAs.

1. *Embedded/Integrated MTA.*

This is a client multimedia terminal which incorporates a DOCSIS MAC-layer interface to the DOCSIS network.

2. *Standalone MTA.*

This is a Client that implements the multimedia functionality without incorporating a DOCSIS MAC-layer interface. The standalone MTA will typically use Ethernet, USB, or IEEE 1394 as the physical interconnect to a DOCSIS Cable Modem. The standalone MTA may be connected to a customer network, and use transport facilities of the customer network (possibly including intermediate IP routers) to establish sessions over the DOCSIS network.

2.2.2 Cable Modem (CM)

This is a PacketCable network element as defined by DOCSIS 1.1 specification. The CM is responsible for classifying, policing and marking packets once the traffic flows are established by the signaling protocols described herein.

2.2.3 Cable Modem Termination System (CMTS)

The CMTS is responsible for allocating and scheduling upstream and downstream bandwidth in accordance with MTA requests and QoS authorizations established by the network administrator. The CMTS acts as a Policy Enforcement Point (PEP) per the IETF Resource Allocation Protocol (RAP) Framework [6].

The CMTS implements a “PacketCable Dynamic QoS Gate” (hereafter called just “Gate”) between the DOCSIS cable network and an IP Backbone. The Gate is implemented using the packet classification and filtering functions defined for DOCSIS 1.1.

The CMTS may or may not also be configured as an “IS-DS Boundary” entity. An IS-DS Boundary interfaces to an inter-network using the Integrated Services (IntServ) model of QoS control and some other model, e.g., Differentiated Services (DiffServ).

2.2.4 Call Management Server (CMS) and Gate Controller (GC)

The PacketCable Call Management Server (CMS) entity performs services that permit MTAs to establish Multimedia sessions (including voice communications applications such as “IP telephony” or “VoIP”). A CMS using the Network-Controlled call signaling model implements a Call Agent that directly controls the session, and maintains per-call state. A CMS using the Distributed Call signaling model may serve as a “DCS Proxy” and perform services only during initial session setup. The term Gate Controller (GC) is used to refer to the portion of either type of CMS that performs the Quality of Service related functions.

In the PacketCable Dynamic QoS Model, the Gate Controller controls the operation of the Gates implemented on a CMTS. The GC acts as a Policy Decision Point (PDP) per the IETF Resource Allocation Protocol (RAP) Framework [6].

2.2.5 Record Keeping Server (RKS)

The Record Keeping Server is a PacketCable network element that only receives information from PacketCable elements described in this document. The RKS can be used as a billing server, diagnostic tool, etc.

2.3 PacketCable Dynamic QoS Architecture

The PacketCable QoS architecture is based upon DOCSIS 1.1, RSVP, and Integrated Services Guaranteed QoS.

Specifically, the PacketCable QoS architecture uses the protocol as defined in the DOCSIS 1.1 specification within the cable network. These messages support static and dynamic installation of packet classifiers (i.e., Filter-Specs) and flow scheduling (i.e., FlowSpecs) mechanisms to deliver enhanced IP quality of service. DOCSIS QoS is based upon the objects which describe traffic and flow specifications, similar to the TSpec and RSpec objects as defined in the IETF Resource reSerVation Protocol (RSVP). This allows QoS resource reservations to be defined on a per flow basis.

In the DOCSIS QoS architecture, traffic flows are considered as unidirectional – thus, an interactive session comprises two flows, each subject to the operations shown below. For each (unidirectional) flow:

The CM, where traffic enters the IP QoS enabled cable network, is responsible for:

- Classification of IP traffic into IP QoS flows based on defined filter specifications.
- Performing traffic shaping and policing as required by the flow specification.
- Maintaining state for active flows.
- Altering the TOS field in the upstream IP headers based on the network operator’s policy.
- Obtaining the required IP QoS from the CMTS (DOCSIS QoS).
- Applying DOCSIS QoS mechanisms appropriately.

The CMTS is responsible for:

- Providing the required QoS to the CM based upon policy configuration.
- Allocating upstream bandwidth in accordance to CM requests and network QoS policies.
- Classifying each arriving packet from the network side interface and assigning it to a QoS level based on defined filter specifications.
- Policing the TOS field in received packets from the cable network to enforce TOS field settings per network operator policy.
- Altering the TOS field in the downstream IP headers based on the network operator’s policy.
- Performing traffic shaping and policing as required by the flow specification.
- Forwarding downstream packets to the DOCSIS network using the assigned QoS.
- Forwarding upstream packets to the backbone network devices using the assigned QoS.
- Maintaining state for active flows.

The backbone network may either utilize Integrated Services based mechanisms or use Differentiated Services mechanisms. In a DiffServ backbone, network routers forward a packet, providing the appropriate IP QoS, based on the setting of the TOS field. In a DiffServ backbone, no per-flow state is required in the core network devices.

2.4 QoS Interfaces

Quality of service signaling interfaces are defined between many of the components of the PacketCable network as shown in Figure 1. Signaling involves communication of QoS requirements at the application layer (e.g., SDP parameters), network layer (e.g., RSVP), and at the data-link layer (e.g., DOCSIS 1.1 QoS). Also, the requirement for policy enforcement and system linkages between the OSS subscriber provisioning, admission control within the managed IP backbone, and admission control within the DOCSIS network creates the need for additional interfaces between components in the PacketCable network.

An expanded explanation of QoS architecture framework is contained in the PacketCable Architecture Framework Technical Report – First Level Decomposition Specification [18], and is shown in Figure 1.

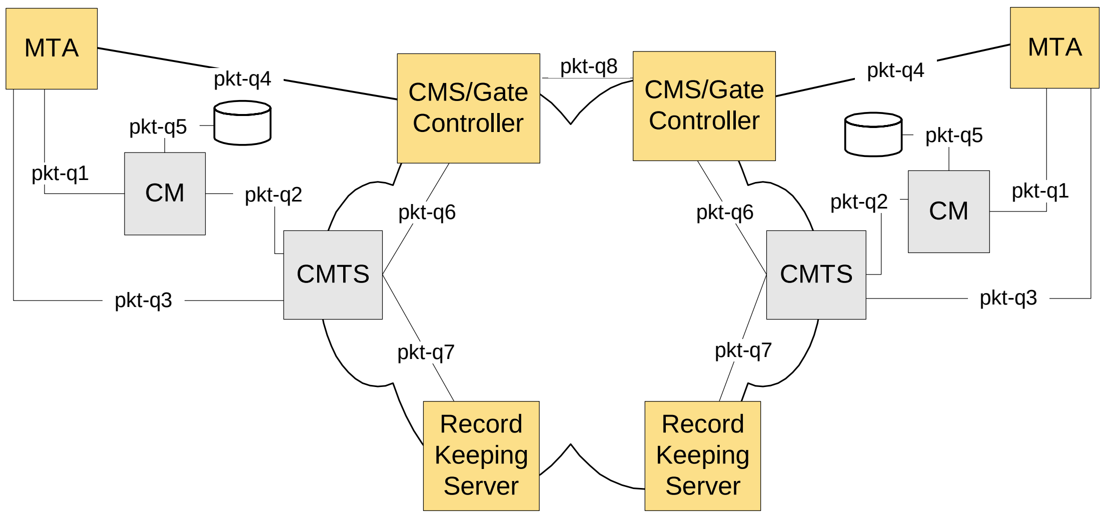


Figure 1. QoS Signaling Interfaces in PacketCable Network

Interfaces Pkt-q1 through Pkt-q8 are available for controlling and processing QoS. Not all interfaces are used in all configurations and protocol variations. All but the Pkt-q5 interface are utilized by DQoS. The following matrix briefly identifies each interface and how each interface is used in this Dynamic QoS Specification (DQoS). Two alternatives are shown for this specification: first a general interface that is applicable to either embedded or standalone MTAs; and second, an optional interface that is available only to embedded MTAs.

Interface	Description	DQoS Embedded/ Standalone MTA	DQoS Embedded MTA (optional)
Pkt-q1	MTA – CM	N/A	E-MTA, MAC Control Service Interface
Pkt-q2	CM – CMTS (DOCSIS)	DOCSIS, CMTS-initiated	DOCSIS, CM-initiated
Pkt-q3	MTA – CMTS	RSVP+	N/A
Pkt-q4	MTA – GC/CMS	NCS/DCS	NCS/DCS
Pkt-q5	CM – Provisioning Server	N/A	N/A
Pkt-q6	GC – CMTS	Gate Management	Gate Management
Pkt-q7	CMTS – RKS	Billing	Billing
Pkt-q8	CMS to CMS	CMS to CMS Signaling	CMS to CMS Signaling

Pkt-q1: Interface between the MTA and CM

This interface is only defined for the embedded MTA. The interface decomposes into three sub-interfaces:

- Control: used to manage DOCSIS service-flows and their associated QoS traffic parameters and classification rules.
- Synchronization: used to synchronize packetization and scheduling for minimizing latency and jitter.
- Transport: used to process packets in the media stream and perform appropriate per-packet QoS processing.

This interface is conceptually defined in Appendix E of the DOCSIS RFI specification [9]. For standalone MTAs no instance of this interface is defined.

Pkt-q2: DOCSIS QoS Interface between CM and CMTS

This is the DOCSIS RFI QoS interface (control, scheduling, and transport). Control functions can be initiated from either the CM or the CMTS. However the CMTS is the final policy arbiter and granter of resources by performing admission control for the DOCSIS network. This interface is defined in the DOCSIS RFI specification [9].

Pkt-q3: Application Layer Interface between the MTA and CMTS

The interface is used to request bandwidth and QoS in terms of delay using standard RSVP and extensions specified herein. As a result of message exchanges between the MTA and CMTS, service flows are activated using CMTS-originated signaling on interface Pkt-q2.

Pkt-q4: Application Layer signaling between GC/CMS and MTA

Many parameters are signaled across this interface such as the media stream, IP addresses, port numbers, and the selection of Codec and packetization characteristics. DCS and NCS are two examples of application layer signaling.

Pkt-q5: Signaling from the DOCSIS/PacketCable Provisioning to the DOCSIS CM.

This interface is not utilized for QoS signaling in DQoS.

Pkt-q6: Interface between the GC/CMS and CMTS

This interface is used to manage the dynamic Gates for media stream sessions. This interface enables the PacketCable network to request and authorize QoS.

Pkt-q7: CMTS to Record Keeping Server

This interface is used by the CMTS to signal to the RKS all changes in session authorization and usage.

Pkt-q8: CMS to CMS interface

This interface is used for session management and resource coordination between a pair of CMSs.

2.5 Framework for PacketCable QoS

In order to justify its costs to the end user, a commercial multimedia service (e.g., voice communications capability) may require a high level of transport and signaling performance, including:

- Low delay – end-to-end packet delay needs to be small enough that it does not interfere with normal multimedia interactions. For normal telephony service using the PSTN, the ITU recommends no greater than 300 ms roundtrip delay.[1] Given that the end-to-end backbone propagation delay may absorb a significant amount of this delay budget, it is important to control delay on the access channel, at least for long-distance calls.
- Low packet loss – packet loss needs to be small enough so that voice quality or performance of fax and voice-band modems is not perceptibly impaired. While loss concealment algorithms can be used to reproduce intelligible speech even with high loss rates, the resulting performance cannot be considered to be adequate as a replacement for existing circuit-switched telephone service. Loss requirements for acceptable voice-band modem performance are even more stringent than those for voice.
- Short post-dial delay – the delay between the user signaling a connection request and receiving positive confirmation from the network needs to be short enough that users do not perceive a difference from the post-dial delay they are accustomed to in the circuit switched network, or believe that the network has failed. This is of the order of one second.
- Short post-pickup delay – the delay between a user picking up a ringing phone and the voice path being cut through needs to be short enough so that the "hello" is not clipped. This should be less than a few hundred milliseconds (ideally less than 100 ms).

A key contribution of the Dynamic QoS framework is a recognition of the need for coordination between signaling, which controls access to application specific services, and resource management, which controls access to network-layer resources. This coordination provides a number of critical functions. It ensures that users are authenticated and authorized before receiving access to the enhanced QoS associated with the service. It ensures that network resources are available end-to-end before alerting the destination MTA. Finally, it ensures that the use of resources is properly accounted for, consistent with the conventions of traditional voice-grade telephone service (to which some PacketCable services are similar from a customer perspective) in which charging occurs only after the party receiving a communication picks up.

In order to support the above requirements, the QoS protocols assure that all resources are committed to all transport segments before the signaling protocols cause alerting of the destination. Likewise, during tear down of a session, the QoS protocols include measures to assure that all resources dedicated exclusively to the session are released. Without this coordination between the two directions of data flows, it would be possible for users to thwart the QoS controls and obtain free service. For example, if the paying client terminates the session, but the non-paying does not, a "half channel" remains that can be used to fraudulently transfer data in one direction. The QoS protocols approximate the "all or nothing" transaction semantics for session creation and destruction.

It is desired that the mechanisms used to implement the session be based on existing standards and practices, and also that the results of this work be usable to support alternative call models. These desires have led to the use of the IETF Real Time Protocol (RTP) to carry multimedia data, carried over the IETF User Datagram Protocol (UDP). In-band signaling to set up Quality of Service is carried out using a superset of the IETF Resource reSerVation Protocol (RSVP).

[1] ITU-T Recommendation G.114 states that a one-way delay of 150 ms is acceptable for most user applications. However, highly interactive voice and data applications may experience degradation even when delays are below 150 ms. Therefore any increase in processing delay (even on connections with transmission times well below 150 ms) should be discouraged unless there are clear service and application benefits.

The QoS architecture should provide support for new emerging applications that are dependent on multicast data delivery. Although this is not a strict requirement in the QoS architecture, providing support for multicast will enable the future development of a rich set of multimedia applications. It has not yet been examined whether the resource management enhancements introduced here will support multicast seamlessly or not.

For purposes of managing Quality of Service, the bearer channel for a session is managed as three distinct segments: the access network for the originating side of the session, a backbone network, and the access network for the terminating side of the session. DOCSIS network resources are managed as a pair of dynamic service flows, using the mechanisms defined in the DOCSIS 1.1 specification. Backbone resources may be managed either per-flow or, more likely, through an aggregated quality of service mechanism. Management of backbone resources is outside the scope of this specification.

Figure 2 graphically shows this model. This specification accommodates a customer environment where a stand-alone MTA may be connected to the CM via a network of links and standard RSVP-capable routers.

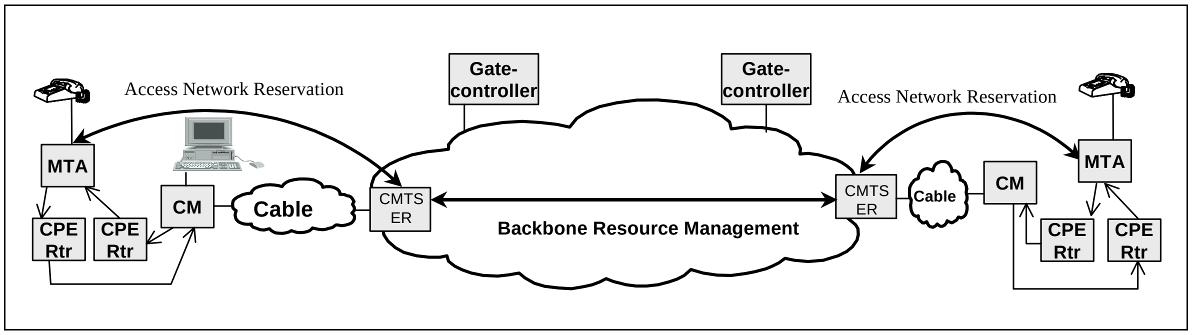


Figure 2. Session Framework

A QoS-defined construct called a gate provides a control point for the connection of access networks to high quality backbone service. A gate is implemented by a CMTS and consists of a packet classifier, a traffic policer, and an interface to an entity that gathers statistics and events (all of these components exist in DOCSIS 1.1). A gate can ensure that only those sessions that have been authorized by the service provider receive high quality service. Gates are managed selectively for a flow. For PacketCable-based voice communications service, they are opened for individual calls. Opening a gate involves an admission control check that is performed when a resource management request is received from the client for an individual session, and it may involve resource reservation in the network for the session if necessary. The upstream packet filter in the gate allows a flow of packets to receive enhanced QoS for a session from a specific IP source address and port number to a specific IP destination address and port number. The downstream packet filter in the gate allows a flow of packets to receive enhanced QoS for a session from a specific IP source address to a specific IP destination address and port number.

A Gate is a logical entity that resides in a CMTS. A GateID is associated with an individual session and is meaningful at the Gate; the GateID is an identifier that is locally unique at the CMTS, and is assigned by that CMTS. A Gate is uni-directional in nature. If a Gate is “Closed”, then data going upstream/downstream on the DOCSIS 1.1 access network may either be dropped or provided best-effort service. The choice of dropping packets or serving them on a best-effort basis is a policy choice of the provider.

The gate controller is responsible for the policy decision of when and whether the gate should be opened. A gate is established in advance of a resource management request. This allows the policy function, which is at the gate controller, to be “stateless” in that it does not need to know the state of sessions that are already in progress.

While the gate controls the QoS-guaranteed stream, other flows, such as RTCP or signaling messages, are not policed by the gate. The support of enhanced QoS for signaling messages may play a very important role if the cable system is utilizing high best effort data traffic. In order to meet the signaling performance targets given at the beginning of this section it may be crucial to use a dedicated signaling flow with proper

QoS constructs. The provisioning specification defines how it is possible to provision the dedicated signaling flow[2]. It should further be noted that the exact nature of the QoS that should be given to the dedicated signal flow depends on traffic and the CMTS design and is left as a vendor differentiation point.

2.6 Requirements of Access Network Resource Management

Providing voice communications service over IP networks with the same level of quality as is available over the PSTN imposes bounds on loss and delay metrics for voice packets and requires active resource management in both the access and backbone networks. The service provider needs to be able to control access to network resources, in order to ensure that adequate capacity is available on an end-to-end basis, even under unusual or overload conditions. The service provider may seek additional revenue for providing a voice communications service with these enhanced quality characteristics (i.e., quality beyond that obtained with a "best-effort" service). The mechanisms provided herein for managed access to enhanced QoS enable the service provider to ensure that access is provided only to authorized and authenticated users on a session-by-session basis and there is no theft of that service.

Clients of the service signal their traffic and performance parameters to the "gate" at the network edge, where the network makes an admission control decision based on both resource availability as well as policy information associated with the gate.

In DOCSIS networks capacity is limited and it is necessary to do resource management on a per-flow basis. In the backbone there may be several alternatives, ranging from per-flow per-hop admission control to coarse-grained resource provisioning. This specification deals only with access network QoS, and is agnostic about backbone network QoS schemes.

This architecture aims to provide a high degree of generality with the intention of enabling new services and future evolution of network architectures. This goal leads to several requirements for a viable QoS architecture, described in the following paragraphs.

2.6.1 Preventing Theft of Service

The network resources dedicated to the session are protected from misuse, including:

- Authorization and Security - ensuring that users are authenticated and authorized before receiving access to the enhanced QoS associated with the voice communications service. The CMS/Gate Controller involved in call signaling is trusted to perform these checks and is the only entity which is trusted to create a new gate in a CMTS. The CMS/GC acts as a policy decision point from the perspective of QoS management.
- Resource control - ensuring that the use of resources is properly accounted for, consistent with the conventions of providers that are part of the PSTN in which charging occurs only after the called party picks up. This includes prevention of utilization of reserved resources for purposes other than the session to which they are assigned. This is achieved through the use of gates and coordination between gates, which bind together address filtering mechanisms with resource reservations.

Since this service may be billed on a per-use basis, there is a significant risk of fraud and service theft. The architecture enables the provider to charge for quality of service. Thus, it prevents theft of service scenarios, several of which are described in Appendix A.

Theft of service scenarios are addressed in this and the Distributed Call Signaling document [10]. They motivate some of the components of the QoS and Call Signaling architectures and protocols.

[2] If defined in the configuration file of the MTA the attributes "Call Signaling SCN Up", "Call Signaling SCN Down", and "Call Signaling Network Mask" define the dedicated signaling flow for embedded-MTA.

2.6.2 Two-phase Resource Commitment

A two-phase protocol for resource commitment is essential to a commercial-grade voice communications service, for two reasons unique to the requirements associated with such a service. First, it ensures that resources are available before signaling the party at the far end that a communication is incoming. Secondly, it ensures that usage recording and billing are not started until the far end picks up, which is also the point at which voice may be cut-through. These properties are provided by conventional telephony signaling protocols; the same semantics will be emulated here. Also, if bandwidth is allocated before the far end picks up, a theft of service becomes possible. Requiring the end-points to explicitly send a commitment message ensures that usage recording is based on knowledge of the end-point and its explicit action.

This framework also supports entities, such as announcement servers and PSTN gateways that need the voice to be cut through after the first phase of the resource management protocol.

2.6.3 Segmented Resource Assignment

The Dynamic QoS Architecture partitions resource management into distinct access and backbone segments. Segmented resource assignment is beneficial for two reasons:

- It allows for different bandwidth provisioning and signaling mechanisms for originator’s network, far end network, and backbone network.
- It allows for resource-poor segments to maintain per-flow reservations and carefully manage resource usage. At the same time, when backbone segments have sufficient resources to manage resources more coarsely, it allows the backbone to avoid keeping per-flow state, and thus enhance scalability.

When the backbone does not require explicit per-flow signaling (such as with a DiffServ backbone), it reduces the time taken to set up a session (minimize post-dial delay) and avoids impacting the voice cut-through time (minimize post-pickup delay).

It potentially reduces the amount of reservation state that is be stored if the remote client is a PSTN gateway.

After the first phase of call signaling, both clients have completed capability negotiation and know what resources are needed end-to-end. Clients send resource management messages using RSVP that may be interpreted hop-by-hop over the local (i.e., user) and access networks (or, optionally for embedded clients, the MAC Control Services Interface). The CMTS maps the resource management messages to the resource management protocol used over the backbone (e.g., IETF DiffServ). It also maps the resource management message to the resource management protocol used over the access link (i.e., DOCSIS 1.1).

2.6.4 Resource Changes during a Session

It is possible to change the resources allocated for a session during the life of the session. This facilitates mid-session changes such as switching from a low-rate voice codec to G.711 when modem tones are detected, and the addition of video data to a session that starts as voice only.

2.6.5 Dynamic Binding of Resources

Dynamic binding of resources (re-reserve) is a requirement to enable efficient use of resources when services such as call waiting are invoked. Abstractly, re-reserving takes bandwidth allocated for a session between a VoIP host and a client and reallocates that same bandwidth to a session with a different client.

It is important to understand the potential danger in de-allocating the session bandwidth, then making a new request for allocation of the new bandwidth. There is a risk of another client using the last remaining bandwidth between the two steps, leaving the original session without an assured quality path. The one-step re-reserve mechanism avoids this, as the bandwidth is not made available to other clients.

2.6.6 Dynamic QoS Performance

QoS messaging takes place in real time while callers wait for services to be activated or changed. Thus, the protocol needs to be fast. The number of messages is minimized, especially the number of messages which transits the backbone, and the number of upstream DOCSIS messages. On the cable network, where there is no possibility that forward and reverse paths will be different, this protocol adds several new objects to RSVP, which enables the CMTS to reduce latency by acting as a proxy for the far-end client.

RSVP messages, DOCSIS management messages, and call signaling messages (collectively referred to as signaling messages) are all transported over the DOCSIS network on a best effort basis. If the CM is also supporting data services, best effort service may be unable to provide the low latency needed for signaling messages. In this situation, the CM MAY be provisioned with a separate service flow, with enhanced QoS, to carry signaling traffic. For example, the signaling service flow could use real-time polling, or non-real-time polling service. This separate service flow is provisioned in the same manner as other DOCSIS media streams, and MAY include classifiers such that its presence is transparent to the MTA.

2.6.7 Session Class

Resources may be reserved for different types of services and each service may in turn define different classes of services for its sessions. QoS reservations for sessions designated by the service provider to be of higher priority (e.g., for E911 calls) suffer a lower likelihood of blocking than normal sessions. The determination of what session class to assign to a session is performed by the service provider, and is a policy that is exercised by the originating Call Agent/Gate Controller complex at the time the initial session request (e.g., first stage INVITE in the case of DCS) is made.

2.6.8 Intermediate Network Support

The architecture should not prohibit intermediate networks between the MTA or Multimedia host and cable modem (e.g., customer network). Although the intermediate network may not fall under the MSO's administrative domain or responsibility, allocation of bandwidth in the MSO DOCSIS network is possible when an intermediate network exists. It is also desirable to present a solution that transparently allows for the reservation of resources on the intermediate network.

2.6.9 Backbone QoS Support

It is possible that some mechanism for explicitly managing backbone resources will be necessary. The scope of this specification is QoS over the DOCSIS network, but the architecture provides open, sufficiently general interfaces that are compatible with many of the known backbone QoS mechanisms.

2.6.10 Handling Multiple Codecs

The NCS signaling used within PacketCable allows for connections to be established with multiple codecs. In the case where a connection is successfully negotiated with multiple codecs in the list, it is important that the proper resources are allocated to make subsequent codec changes within the list work as expected. Here are the resource components that need to be allocated:

Authorized Bandwidth: The CMS/GC MUST authorize the Least-Upper-Bound of the codec bandwidth that can be used on the connection during gate allocation.

Reserved Bandwidth: The MTA MUST reserve the Least-Upper-Bound of the codec bandwidth that can be used during the call (possible codecs are determined from codec negotiation procedure defined in the NCS specification [11]). Note: If the Reserved bandwidth is greater than the committed bandwidth, then the reserved bandwidth needs to be refreshed using DSC's to the CMTS.

Committed Bandwidth: The MTA MUST only commit the current codec in use in the upstream direction. This allows the extra unused bandwidth (difference between the Reserved and Committed) to be used for best-effort traffic. In the downstream direction, the MTA MUST commit the Least-Upper-Bound of the codec bandwidth that can be used during the call (possible codecs are determined from codec negotiation procedure defined in the NCS specification [11]).

This procedure ensures that a CMS request to switch to any one of the vocoders in the negotiated list will be successful. This is especially important in supporting features such as fax/modem that require a switch to G.711 for successful transmission.

If a system provider feels that the above allocation of resources places too much of a constraint on the number of voice channels that can be supported (since resources may be over reserved in many cases), then the CMS only needs to state a single codec in the LocalConnectionOptions of the connection request. This will ensure that the reserved and committed resources are equal (using the same mechanism as defined in the multiple codec case). Then, if the CMS wants to switch codecs it will need to place the new codec in the LocalConnectionOptions of a subsequent modify connection. However, there are certain risks with this approach. For example, when a modem call is detected and reported to the CMS, it may be possible that the resulting modify connection to use G.711 fails due to insufficient resources on the CMTS. This would not be the case if multiple codecs where defined since the LUB would already be reserved and guaranteed accessible for a subsequent commit.

2.6.11 MTA Port-to-Port Calls

When voice calls are established between different ports (endpoints) on the same MTA, DOCSIS forwarding rules specify that the CM must not forward packets over the DOCSIS network. As a result, the actions taken by the CMS and MTA in this special circumstance are different from a typical MTA to MTA call flow. A port-to-port call is defined by both endpoints using the same IP address.

If an MTA receives a connection request without a GateID, it MUST NOT initiate any DSx messaging to the CMTS. If an MTA is instructed to make a port-to-port call, the MTA MUST NOT initiate any DSx messaging to setup a service flow for this connection and MUST NOT send any of the voice packets over the network. In addition, if the MTA has previously created a service flow for a call where the far-end SDP was not available (but a GateID was specified in a CRCX or MDCX), then it MUST subsequently tear down the service flow if a port-to-port call is recognized once the remote SDP is received.

The CMS SHOULD recognize port-to-port calls and SHOULD omit Gate Control to the CMTS and SHOULD omit the GateID in the connection command to the MTA. Similar to the MTA case above, if the CMS has already established a gate for a call where the remote SDP was not available, it SHOULD expect a Gate-Close message from the CMTS once the MTA tears down the service flow upon detection of the port-to-port call. The CMS MUST NOT tear down a call between endpoints with the same IP address on receipt of a GATE-CLOSE message .

2.7 Theory of Operation

2.7.1 Basic Session Setup

Resource reservation is partitioned into separate Reserve and Commit phases. At the end of the first phase, resources are reserved but are not yet available to the MTA. (On the DOCSIS links, the service flows in each direction are admitted.) At the end of the second phase, resources are made available to the MTA and usage recording is started so that the user can be billed for usage. (On the DOCSIS links, the service flows are active.)

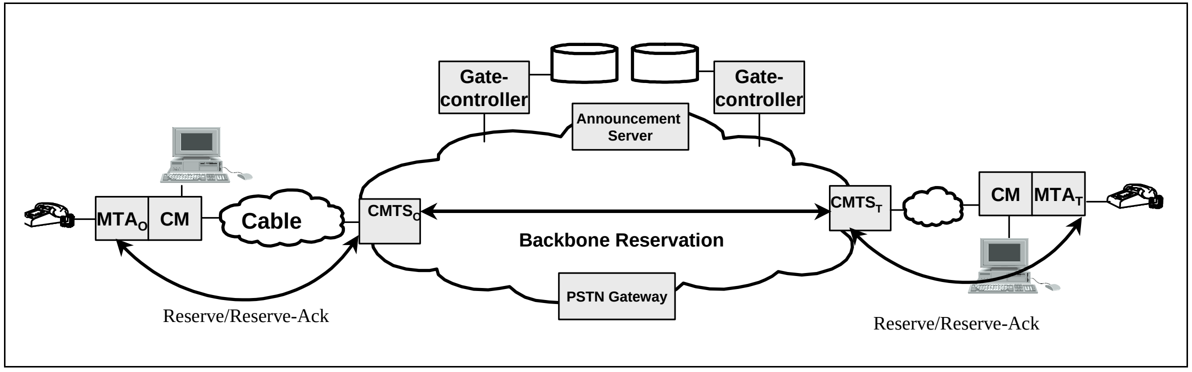


Figure 3. Resource Management Phase 1

Figure 3 shows the first phase of the resource management protocol for a Multimedia application. In this description, subscripts "O" and "T" designate the originating and terminating points of the call. The MTA can be either a standalone VoIP host or an embedded MTA; the latter is shown in Figure 3. MTA_O and MTA_T request resource reservation (PATH message in RSVP, or DOCSIS message in the optional interface for embedded clients) to $CMTS_O$ and $CMTS_T$ respectively. $CMTS_O$ and $CMTS_T$ perform an admission control check for resource availability (initiating signaling for resource reservation in the backbone if necessary) and send a reply to the respective MTAs. In the RSVP framework, the RESV message from the CMTS (where the gate resides) is the acknowledgment to the MTA.

Figure 4 shows the second phase. After determining that resources are available, MTA_O sends a RING message to MTA_T instructing it to start ringing the phone. MTA_T sends a RINGING indication to MTA_O indicating both that resources are available and that the RING message was received. When the called party picks up the phone, MTA_T sends an ANSWERED message to MTA_O and a COMMIT message to $CMTS_T$. When MTA_O receives the ANSWERED message, MTA_O sends a COMMIT message to $CMTS_O$. The COMMIT messages cause resources to be allocated for the call in the DOCSIS networks. The arrival of the COMMIT messages at $CMTS_T$ and $CMTS_O$ causes them to open their gates, and also starts accounting for resource usage. To prevent some theft of service scenarios, each CMTS informs the respective CMS of the state change by sending a Gate-Open message.

The RING, RINGING, and ANSWERED messages shown in this figure and in the above description are logical equivalents to the call signaling messages exchanged by NCS [11] and DCS [10].

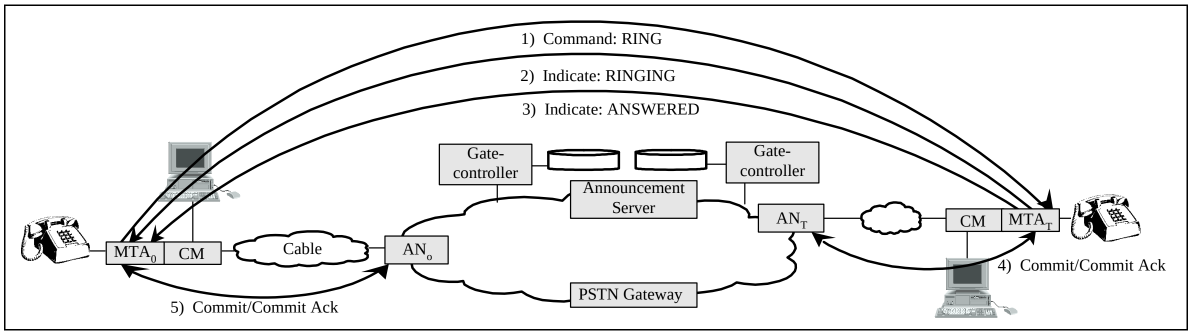


Figure 4. Resource Management Phase 2

2.7.2 Gate Coordination

QoS signaling leads to the creation of a gate at each CMTS associated with a client involved in the session. Each gate maintains usage data for the session and controls whether the packets generated by the associated client receive access to enhanced QoS. Gate coordination is needed to prevent fraud and theft of service in situations where a malfunctioning or modified client does not issue the expected signaling messages. It is essential that protocol mechanisms are robust against abuse.[3] A gate coordination protocol ensures that:

- A potential for one-way session establishment without billing is avoided. Because the clients may have adequate intelligence and are not trusted, one can envisage the clients establishing two one-way sessions to provide the users with an adequate interactive voice communication channel. Gate coordination prevents such sessions being established without the provider being able to charge for them.
- The opening and closing of each gate is closely synchronized with the corresponding state changes on the CMS.

2.7.3 Changing the Packet Classifiers Associated With a Gate

Once a pair of gates is set up, clients can communicate over the network with enhanced QoS. Several features needed for a commercial voice communications service involve changing the clients involved in a session, for example when a session is transferred or redirected, or during three-way calling. This requires the packet classifiers associated with a gate to be modified to reflect the address of the new client. In addition, changing the end-points involved in a session may affect how the session is billed. As a result, gates include addressing information for origination and termination points. A protocol that achieves this change is described in [10].

2.7.4 Session Resources

The relationship between different categories of resources, authorized, reserved, and committed, is shown in Figure 5. A set of resources is represented by an *n*-dimensional space (shown here as 2-dimensional) where *n* is the number of parameters (e.g., bandwidth, burst size, jitter, classifiers) needed to describe the resources. The exact procedures for comparing *n*-dimensional resource vectors are given in [9].

When a session is first established, DQoS protocols authorize the use of some maximum amount of resources, indicated by the outer oval, specifying the authorized resources. When a client makes a reservation for a session, it reserves a certain amount of resources, which are not greater than those for which it has been authorized. When the session is ready to proceed, the client commits to some amount of resources, which are not more than the reserved resources. In many common cases, the committed and reserved resources will be equal. The committed resources represent resources that are currently in use by the active session, whereas reserved resources represent those that are tied up by the client and have been removed from the pool for admission control purposes, but which are not necessarily being used by the client.

[3] Several theft of service scenarios are described in Appendix A.

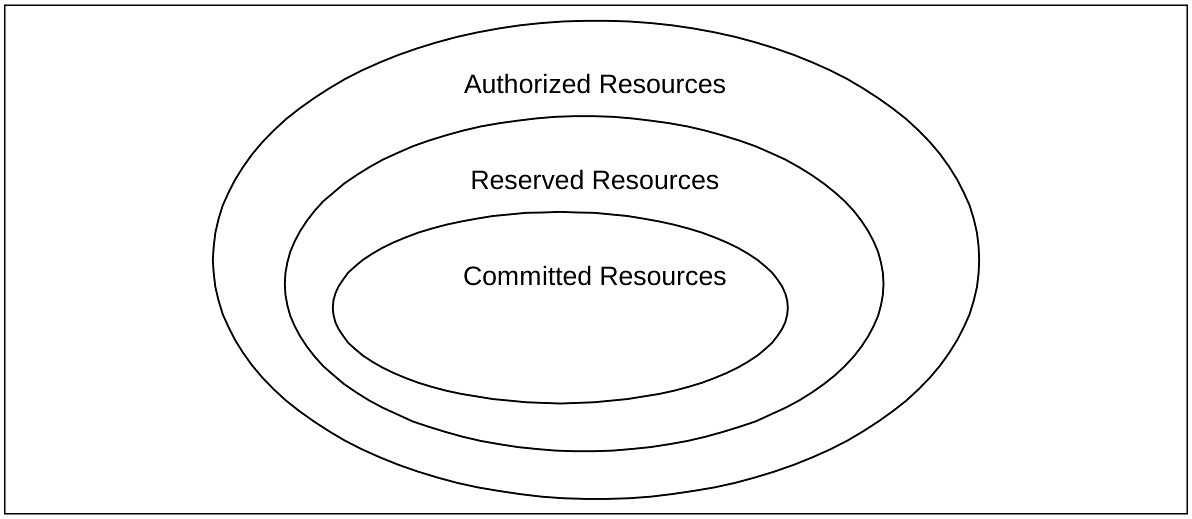


Figure 5. Authorized, Reserved, and Committed Resources

Authorizations only affect future resource reservation requests. Resources that have been reserved prior to an authorization change are not affected.

Resources that have been reserved but not committed are available to the system for short-term uses only, such as handling of best-effort data. These resources are not available for other reservations (i.e., overbooking is not allowed). The maximum portion of the available resources that can be reserved at once is a policy decision by the CMTS, and outside the scope of DQoS.

Excess resources reserved above those committed are released unless the client explicitly requests they be kept through periodic reservation refresh operations. Maintaining such a condition for long periods of time is discouraged, as it reduces the overall capacity of the system. However, there are situations (e.g., call waiting service, where the call on hold requires resources beyond those needed for the active call) where excess reservations are necessary.

2.7.5 Admission Control and Session Classes

It is envisaged that the Gate at the CMTS may use one or more session classes for resources reserved from an MTA. Session classes define provisioned admission control policies, or their parameters. It is expected that the provider would provision the necessary parameters and/or the alternative admission control policies in the CMTS and in the Gate Controller. For instance, a session class for normal voice communications, and an overlapping session class for emergency calls could be defined to allow the allocation of up to, respectively, 50% and 70% of the total resources to these classes of calls, and leaving the remainder 30-50% of the total bandwidth available to other, possibly lower priority, services. Session classes may furthermore enable pre-emption of already reserved resources, in which case the policy for such pre-emption would be provisioned by the service provider. When the Authorized Envelope is communicated to the Gate at the CMTS by the Gate Controller in the Gate-Set message, the Gate Controller includes adequate information to indicate which session class should apply when the corresponding RESERVE request is processed.

2.7.6 Resource Renegotiations

Several of the supported session features require renegotiations of the QoS parameters associated with a session during the lifetime of the session. For example, clients might start communicating using a low-bit-rate audio codec. They can subsequently switch to a higher bit-rate codec or add a video stream, as long as the requested QoS is within the authorized envelope and there is available bandwidth on the network. The use of an authorized QoS envelope that is pre-authorized by the Gate Controller acting as the policy decision point gives clients the flexibility to renegotiate QoS with the network without requiring subsequent Gate Controller involvement. This essentially means that use of resources up to the limits of the envelope is pre-authorized but NOT pre-reserved. Successful allocation of resources within the authorized envelope requires an admission control decision, and is not guaranteed. Subsequent to admission control, the resources are reserved for the flow, although the actual usage of the resources is permitted only after the Commit phase of the Resource Reservation protocol completes. However, no admission control decision is required at the time of commitment of resources. Each change in commitment of resources within the limits of the admission control decision does not require a further reservation. All reservation requests that pass admission control MUST fit within the authorization envelope.

2.7.7 Dynamic Binding of Resources (Re-reserve)

The Dynamic QoS Architecture recognizes that there may be a need to share resources across multiple sessions, especially when resources are in short supply. In particular, when using the call-waiting feature in telephony-like applications, the client may be involved in two simultaneous sessions, but will be active in only one conversation at a time. It is feasible in this case to share the network-layer resources (in particular, on the access link) between the two conversations. Therefore, this architecture allows a set of network layer resources (such as a bandwidth reservation) to be explicitly identified, and allows one or more gates to be associated with those resources. Signaling primitives allow the resources associated with a gate to be shared with another gate at the same CMTS. This improves the efficiency with which resources in the DOCSIS network are utilized.

When switching back and forth between two sessions in a call-waiting scenario, a client needs to keep enough resources reserved to accommodate either of the sessions, which in general may not need the same amount of resources. Thus, the re-commit operation may change the committed resources. However, the reserved resources do not change in this case, as the client should not have to go through admission control when switching back to the other session.

Whereas the committed resources are always associated with the current active session (and its corresponding IP flow), the reserved resources may be bound to different flows and different gates at different times. A handle, called a resource ID, is used to identify a set of reserved resources for the purpose of binding a flow to those resources.

2.7.8 Support for Billing

QoS signaling can be used to support a broad range of billing models, based on only a stream of event records from the CMTS. Since the gate is in the data path, and since it participates in resource management interactions with an client, resource usage accounting is done by the gate. The gate in the CMTS is the appropriate place to do resource accounting, since the CMTS is directly involved in managing resources provided to an client. It is also important to do usage accounting in the CMTS to cope with client failures. If a client that is involved in an active session crashes, the CMTS MUST detect this and stop usage accounting for the session.   This can be accomplished using soft state through a resource management refresh message (by having RSVP PATH messages periodically transmitted for an active session), by monitoring the flow of packets along the data path for continuous-media applications, or by other mechanisms (such as station maintenance) performed by the CMTS. In addition, since the gate retains state for flows that have been authorized by a service-specific Gate Controller, it is used to hold service-specific information related to charging, such as the account number of the subscriber that will pay for the session. The policy function in the Gate Controller thus becomes stateless.

The support required in the CMTS is to generate and transmit an event message to a record keeping server on every change to the QoS, as authorized and specified by a gate. Opaque data provided by the Gate Controller that may be relevant to the record keeping server may also be included in the message. Requirements for handling of event records are contained in other Operations Support specifications [20].

2.7.9 Backbone Resource Management

When a CMTS receives a resource reservation message from an MTA, it first verifies that adequate upstream and downstream bandwidth is available over the access channel using locally available scheduling information. If this check is successful, the CMTS can either generate a new backbone resource reservation message, or forward towards the backbone a modified version of the resource reservation message received from the MTA. The CMTS performs any backbone-technology-specific mapping of the resource reservation that is needed. This enables the architecture to accommodate different backbone technologies, at the service provider's choosing. The specific mechanisms for reserving backbone QoS are outside the scope of this specification.

A bi-directional model is used for resource reservation in the DOCSIS network where the routing is symmetric. A unidirectional model is used for resource reservation in the backbone, which allows routing asymmetries. Thus, when MTA_O makes a reservation with the CMTS, it knows two things: that it has adequate bandwidth in both directions over the DOCSIS network, and that it has adequate bandwidth over the backbone networks for the MTA_O to MTA_T flow. Thus, MTA_O knows that resources are available end-to-end in both directions once it gets a reply from MTA_T.

2.7.10 Setting the DiffServ Code Point

This architecture also allows for the use of a Differentiated Services backbone, where there is adequate bandwidth to carry voice conversations, but access to this bandwidth is on a controlled basis. Access to the bandwidth and differentiated treatment is provided to packets with the appropriate encoding of bits in the field of the IP header specified for Differentiated Service. This is called the DiffServ code point (DSCP). The DS field maintains backward compatibility with the present uses of the IP Precedence bits of the IPv4 TOS byte [32]. It is desirable to be able to set the DiffServ code point of packets that are about to enter the provider backbone from the CMTS. Since resources consumed by these packets in the backbone may depend heavily on this marking, this architecture provides control of the marking to network entities. This allows the network and service provider the control on use of the enhanced QoS rather than trusting the MTA. The provider can configure policies in the CMTS that determine how to set the DSCP for flows that pass through the CMTS. Such policies are sent to the CMTS in the gate setup protocol from the CMS/GC.

For implementation efficiency, the information is passed to the MTA about the appropriate DSCP for it to use on a given session. This is done with the IETF-proposed DCLASS object in RSVP [19]. The CMTS still needs to police received packets to ensure that correct DSCP is being used and that the volume of packets in a given class is within authorized bounds.

2.8 Sample Mapping Of Sdp Descriptions Into Rsvp Flowspecs

Session descriptor protocol messages are used to describe multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation [17], [10]. This section describes a mechanism for mapping the SDP description into RSVP FlowSpecs.

A typical SDP description contains many fields that contain information regarding the session description (protocol version, session name, session attribute lines, etc.), the time description (time the session is active, etc.), and media description (media name and transport, media title, connection information, media attribute lines, etc.). The two critical components for mapping an SDP description into an RSVP FlowSpec message are the media name and transport address (m) and the media attribute lines (a).

The media name and transport address (m) are of the form:

m = <media> <port> <transport> <fmt list>

The media attribute line(s) (a) are of the form:

a = <token>:<value>

A typical IP voice communication would be of the form:

m = audio 3456 RTP/AVP 0

a = ptime:10

On the transport address line (m), the first term defines the media type, which in the case of an IP voice session is audio. The second term defines the UDP port to which the media is sent (port 3456). The third term indicates that this stream is an RTP Audio/Video profile. Finally, the last term is the media payload type as defined in the RTP Audio/Video Profile (reference RFC 1890 [8]). In this case, the 0 represents a static payload type of u-law PCM coded single channel audio sampled at 8KHz. On the media attribute line (a), the first term defines the packet formation time (10ms).

Payload types other than those defined in RFC 1890 [8] are dynamically bound by using a dynamic payload type from the range 96-127, as defined in [17], and a media attribute line. For example, a typical SDP message for G.726 would be composed as follows:

m = audio 3456 RTP/AVP 96

a = rtpmap:96 G726-32/8000

The payload type 96 indicates that the payload type is locally defined for the duration of this session, and the following line indicates that payload type 96 is bound to the encoding "G726-32" with a clock rate of 8000 samples/sec. For every defined CODEC (whether it is represented in SDP as a static or dynamic payload type) there needs to be a table mapping from either the payload type or ASCII string representation to the bandwidth requirements for that CODEC.

For non-well-known codecs, the bandwidth requirements cannot be determined by the media name and transport address (m) and the media attribute (a) lines alone. In this situation, the SDP MUST use the bandwidth parameter (b) line to specify its bandwidth requirements for the unknown codec. The bandwidth parameter line (b) is of the form:

b= <modifier> : <bandwidth-value>

For example:

b= AS:99

This bandwidth parameter along with the media attributes MUST be used to map the SDP into a FlowSpec, which will be used in the policy authorization decision and subsequent gate allocation. Note: It is a policy decision on the CMS/CMTS whether the requested bandwidth in the SDP is accepted or rejected. The bandwidth parameter (b) will include the necessary bandwidth overhead for the IP/UDP/RTP headers. In addition, any PHS used in the DOCSIS link will not be reflected in requested bandwidth. In the specific case where multiple codecs are specified in the SDP, the bandwidth parameter should contain the maximum of the desired codec bandwidths.

The mapping of RTP/AVP code to RSVP FlowSpec is according to the Table 4 of the PacketCable CODEC specification [21].

3 MTA TO CMTS QUALITY-OF-SERVICE PROTOCOL (PKT-Q3)

To meet the requirements described previously, RSVP and the IETF's Integrated Services architecture [2] is used as a basis for the signaling mechanism for providing local QoS. RSVP, as currently specified, needs some additional enhancements to meet the requirements of the Dynamic QoS architecture. The combination of RSVP and these extensions is sometimes called RSVP+ within PacketCable.

RSVP and the Integrated Services architecture specify QoS parameters in generic terms that are independent of the underlying layer 2 technology. It is necessary to specify a means of mapping those general traffic specifications into specific DOCSIS traffic flow specifications. Such mappings exist for other layer 2 protocols (e.g., ATM, 802 LANs); this section describes mappings for DOCSIS networks.

The Dynamic QoS Architecture uses a superset of RSVP with the following differences:

- Since resource reservations are independently initiated for each DOCSIS network (segmented resource allocation model), this specification does not depend on resource management messages propagating end-to-end.
- The resource management exchange between the MTA and CMTS reserves resources in both directions over the local area (i.e., customer-operated) and DOCSIS networks. This allows the CMTS to act as a proxy for the far endpoint, with the benefit of minimizing the number of messages required for resource management in bandwidth-constrained DOCSIS networks, and reduces the post-dial and post-pickup delay.
- In the local area (i.e., customer-operated) portion of the network, existing RSVP capable routers may be present. In this environment, uni-directional reservations are required. To enable these two functions (bi-directional reservations on the DOCSIS network and uni-directional reservations inside the customer-premises), an enhanced PATH message is issued by the MTA to the Gate.
- Ability to bind a single set of resources to a group of multiple reservations, based on information from the MTA that only one reservation in the group will be active at any given time.
- Support for the two-phase admit/activate facility available in DOCSIS, giving the ability to guarantee resources are available before ringing of the far-end phone. The RSVP exchange with the CMTS performs the first stage, the admission control, and the MTA sends a separate message to the CMTS to perform the activation.

The Dynamic Quality of Service operation does not address standard RSVP, which may or may not be supported. Regardless, standard RSVP messages will not trigger the DQoS operations specified in this document.

3.1 RSVP Extensions Overview

3.1.1 Segmented Operation

As defined in [1], RSVP is intended to run between a pair of hosts. However, the PacketCable QoS model requires the signaling to be done in a segmented manner, where one segment is between a client device (MTA or host) and a CMTS. This section illustrates how RSVP can support a segmented model.

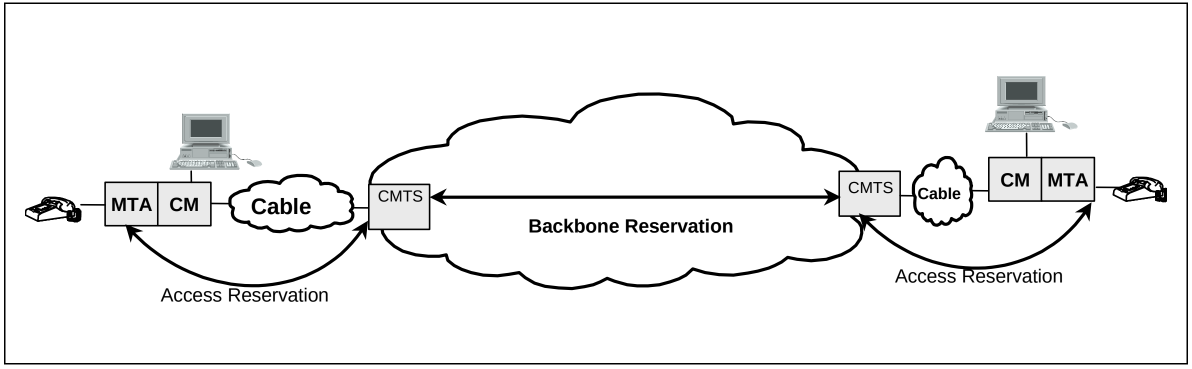


Figure 6. Segmented Signaling Model

In the segmented model, a client (either a PC, IP phone, or an embedded device in the cable modem) communicates with the CMTS. In addition to the simple scenario pictured in Figure 6, this specification allows for more complex scenarios, such as when there is a customer network between the client and the CM, which may include a variety of network elements, including RSVP-capable switches or routers. The presence of a customer network means that the solution works even if the client and the CMTS are not immediately adjacent at the IP layer. The customer network may provide multiple paths between the client and the CM, leading to the possibility of asymmetric routes in this network.

The CMTS intercepts RSVP messages sent from the host to the true client of the session to implement the segmented model. This minimizes changes to RSVP, by keeping the destination address of the PATH messages the same as the destination address of the data.

3.1.2 Bi-directional Reservations

Traditional RSVP makes unidirectional reservations. PATH messages flow in the same direction as data, and RESV messages flow in the opposite direction. To make a bi-directional reservation, it is necessary to add new RSVP objects to define both directions. The CMTS responds to the request by establishing a bi-directional reservation on the DOCSIS link (really two simultaneous uni-directional reservations). If there are RSVP-capable routers between the client and CM, then the CMTS initiates a PATH message that appears to have come from the remote client.

3.1.3 Header Compression, Suppression and VAD

If the CMTS and CM are configured to perform header suppression, then the bandwidth that is needed on a Service Flow may be reduced. It is necessary for the client to convey to the CMTS the fact that compression or suppression that may be applied prior to the installation of a reservation to ensure that appropriate bandwidth is reserved. The general solution to this problem is described in [4].

The sender (client) adds a parameter (Compression_Hint), described in [4], to the Sender-TSpec that identifies the type of compression or header suppression that might be applied to the data. The Compression_Hint parameter contains a Hint field that advertises the type(s) of compression that is possible, for example the DOCSIS Payload Header Suppression, as well as whether the sender is using UDP checksums or IP-Ident; if these are not used, these fields may also be compressed or suppressed. If any field in the IP header is not being suppressed, then the IP checksum MUST NOT be suppressed.

To signal payload header suppression, the CMTS uses an assigned number (TBD) in the Hint field of the Compression_Hint parameter to indicate the amount of DOCSIS payload header suppression that will be performed on this session. This information is used to reduce the effective rate and depth of the token bucket supplied by the sender. If header suppression is not supported on a link, the Compression_Hint parameter is ignored and the full TSpec is used.

When performing header suppression on a DOCSIS link, it is also necessary to communicate the contents of the header that will be suppressed to the CMTS in advance of the first data packet's transmission so that the suppression context can be established at the CM and the CMTS. All this information is present in the

RSVP message that is used to establish the reservation, including source and destination IP addresses and ports. Since PATH messages are processed by any intermediate hops between the client and the CMTS, an arriving PATH message will contain the same TTL value as data packets, provided PATH messages and data packets have the same initial TTL when sent by the client. The CMTS thus uses the contents of the PATH to learn the values of the fields that will be suppressed. The CMTS uses DOCSIS messaging to convey to the CM the fact that suppression should be used for a particular flow, and instructs it to suppress appropriate fields given the presence or absence of UDP checksums.

The CMTS also may suppress the IP Identification field. This field is used only when fragmentation occurs. Since this field changes with every packet, its value cannot be conveyed using RSVP. The question of whether to suppress it or not depends on whether the packet might be fragmented later. There is no need for the host to convey any information to the CMTS regarding the suppressibility of this field; the CMTS may decide to suppress it or not based on a local policy.

The same basic approach enables support of Voice Activity Detection (VAD). A CMTS may use different scheduling algorithms for flows that are using VAD, and thus needs to know which flows may be treated with VAD. The compressibility object carried in the TSpec MUST contain a value which indicates that the data flow for which this reservation is being requested may be treated with VAD (i.e., it has not already undergone silence detection at the client, and it is voice, not fax or data).

3.1.4 Dynamic Binding of Resources

The Dynamic QoS model requires the ability to dynamically modify the binding of resources to flows. For example, to provide Call Waiting, it may be desirable to hold enough resources for only one session in place over the DOCSIS network, and to switch the allocation of those resources from one caller to another. While this capability has been suggested for RSVP in the past, it was not included in RSVP version 1.

In RSVP, the “handle” on a set of reserved resources is the Session-Object. Since the Session contains the destination address of the flow, reallocation of resources to a flow with a different destination address would require a change in the Session-Object. Changing the source address of the flow could be accomplished using a new FilterSpec in the RESV message.

To accommodate this functionality, a Resource-ID object is added to RSVP messages. Routers, which understand this object, will attempt to use the resources associated with that ID. The Resource-ID object is an opaque identifier generated by the node that has control of the resources, i.e., the CMTS in this case.

This is illustrated in Figure 7. When a client issues a reservation request for a new flow, it indicates to the CMTS that this session is willing to share resources for this new gate (Gate 2) with a previously created gate (Gate 1) by including the ResourceID in the request. As long as the QoS requested for the new gate can be satisfied with a bandwidth allocation equal to or less than the existing gate, no new bandwidth is reserved in the DOCSIS network. However, bandwidth may need to be reserved in the backbone network depending on the end-to-end path taken by the new session. Access to the shared reservation occurs in a mutually exclusive manner: a client has to issue a commit message to indicate to the CMTS which flow is currently active, and that commit explicitly removes the committed resources for the other. In the call waiting example, the client sends a commit message to the CMTS to identify the currently active flow when the user switches between sessions.

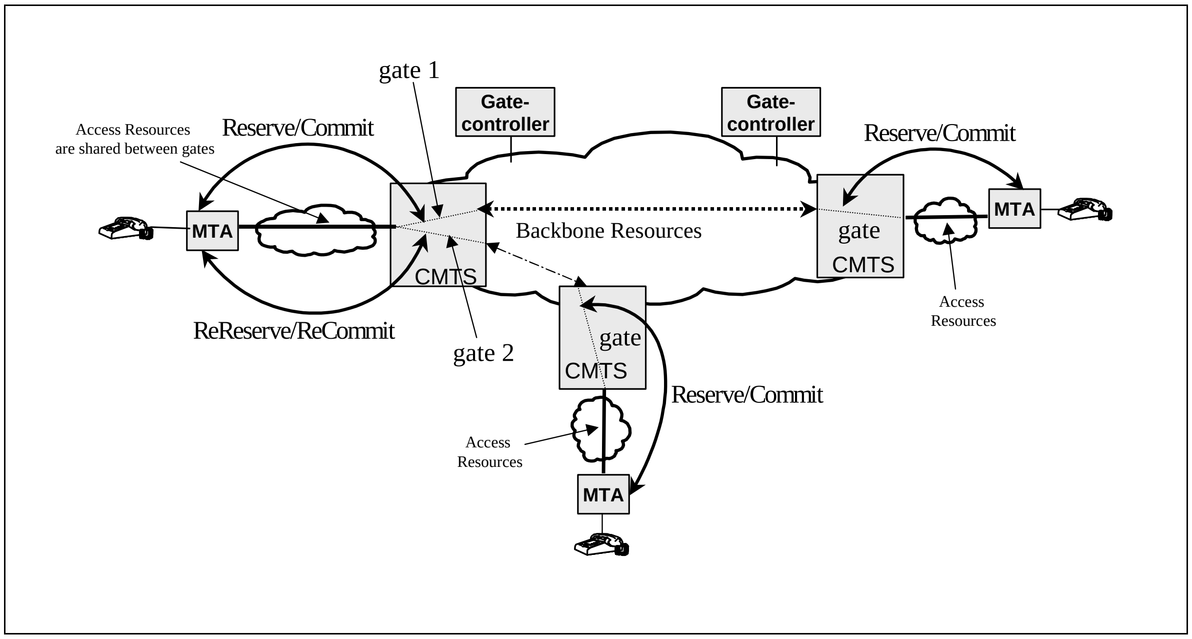


Figure 7. Sharing of Resource Reservations across Gates

In the segmented model, the CMTS includes the Resource-ID in the first RESV message that it sends to the client. The client may include the Resource-ID in subsequent messages that apply to the resources in question. Most importantly, if the client wishes to establish a new session, and reuse the resources of an existing session, it includes the Resource-ID associated with the old session in the PATH message it sends to the CMTS. A PATH message that contains the Resource-ID of a currently allocated set of resources adds a new binding between a flow (as identified in the Session and Sender-Template objects) and those resources. It may optionally change the amount of resources allocated by the inclusion of TSpecs and RSpecs which differ from those previously received by the CMTS for this set of resources. This may include the addition of a new set of TSpecs and RSpecs to accommodate multiple codecs as described in section 3.2.1.

RSVP allows reservations to vary in size over time. A reservation that is no greater than the one currently installed (i.e., does not require an increased level of resources in any dimension for either direction of the session) MUST NOT fail admission control. The same rule applies when using the Resource-ID object. If the amount of resource requested in the new reservation is no greater than previously installed, the reservation MUST NOT fail admission control.

A router that does not understand this new object (e.g., in the customer network) will simply try to install what appears to be a new reservation without re-using previously allocated resources. Since it is unlikely that there is less bandwidth in the home than on the DOCSIS network, this is unlikely to be a problem. The old reservation will timeout if it is not refreshed. In the event that resource scarcity is a problem in the customer network, it would be necessary to upgrade the routers in the home to support this new object. Note that attempting to install reservations on the customer network is worthwhile even if bandwidth is relatively abundant there, as a reservation provides devices in the customer network with the necessary information to isolate reserved flows from excessive delay and jitter that they would otherwise experience if simply mixed with best effort traffic (or reserved flows of widely different traffic characteristics) in a common queue.

3.1.5 Two-Stage Reserve/Commit Process

A significant aspect of the PacketCable Dynamic QoS model is that reservation is a two-phase process, with a Commit phase following the Reserve phase. RSVP is used to cover the Reserve phase, so the CMTS does not actually provide the resources (e.g., unsolicited grants on the upstream channel) until the second stage of the process.

Because the commit phase only involves a client and a local gate, it is a unicast message from the client to the CMTS. The client learns the GateID from the call signaling protocol.

3.1.6 Authentication

The provider is able to ensure that parties do not reserve unauthorized network resources. RSVP provides a number of mechanisms to do this, such as RSVP Integrity-Objects and policy data contained in other RSVP messages. The Dynamic QoS specification includes a GateID as policy data, which MUST be included in the RSVP PATH messages.

3.2 RSVP FlowSpecs

The Integrated Services architecture uses general purpose (layer 2 independent) descriptions of the traffic characteristics and resource requirements of a flow. The description of the traffic is known as a TSpec, the resource requirements are contained in an RSpec, and the combination of these is known as a FlowSpec. In order to reserve resources on a specific layer 2 medium such as a DOCSIS cable network, it is necessary to define a mapping from the layer two independent FlowSpec to specific layer two parameters. Mappings for a variety of other technologies (ATM, 802.3 LANs, etc.) have already been defined.

Other specifications (e.g., the PacketCable CODEC specification [21]) contain the mapping requirements of higher-layer service descriptions (e.g., SDP as used in VoIP applications) into FlowSpecs. This section specifies how the CMTS and MTA MUST map FlowSpecs to DOCSIS layer 2 parameters.

Integrated Services currently defines two types of service, controlled load and guaranteed, the latter being the more suitable for latency sensitive applications. When making a reservation for guaranteed service, the FlowSpec contains:

TSpec
- bucket depth (b) – bytes
- bucket rate (r) – bytes/second
- peak rate (p) – bytes/second
- min policed unit (m) – bytes
- maximum datagram size (M) – bytes

RSpec
- reserved rate (R) – bytes/second
- slack term (S) – microseconds

The TSpec terms are mostly self-explanatory. (r,b) specifies a token bucket that the traffic conforms to, p is the peak rate at which the source will send, and M is the maximum packet size (including IP and higher layer headers) that will be generated by the source. The minimum policed unit, m, is usually the smallest packet size that the source will generate; if the source sends a smaller packet, it will count as a packet of size m for the purposes of policing.

To understand the RSpec, it is helpful to understand how delay is calculated in an Integrated Services environment. The maximum end-to-end delay experienced by a packet receiving guaranteed service is

$$\text{Delay} = b/R + C_{tot}/R + D_{tot}$$

where b and R are as defined above, and C_{tot} and D_{tot} are accumulated 'error terms' provided by the network elements along the path, which describe their deviation from 'ideal' behavior.

The rate R provided in the RSpec is the amount of bandwidth allocated to the flow. It MUST be greater than or equal to r from the TSpec for the above delay bound to hold. Thus, a flow's delay bound is completely determined by the choice of R; the reason to use a value of R greater than r would be to reduce the delay experienced by the flow.

Since it is not permissible to set $R < r$, a node making a reservation may perform the above calculation and determine that the delay bound is tighter than needed. In such a case, the node may set $R = r$ and set S to a non-zero value. The value of S would be chosen such that

$$\text{Desired delay bound} = S + b/R + C_{tot} / R + D_{tot}$$

Guaranteed Service does not attempt to bound jitter any more than is implied by the delay bound. In general, minimum delay that a packet might experience is the speed of light delay, and the maximum is the delay bound given above; the maximum jitter is the difference between these two. Thus jitter may be controlled by suitable choice of R and S.

3.2.1 Complex SDP Descriptions with Multiple Codecs

There are various situations in which a reservation needs to cover a range of possible FlowSpecs. For example, for some applications it is desirable to create a reservation, which can handle a switch from one codec to another mid-session without having to pass through admission control at each switch-over time.

The Sender TSpec MUST contain the Least Upper Bound (LUB) of the necessary flow parameters for the component flows.

The Least Upper Bound (LUB) of two flows A and B, LUB(A, B), is the “smallest” envelope that can carry both of the flows A, B non-simultaneously. LUB(A, B) is calculated on a parameter-by-parameter basis as follows:

Define the TSpec values for a flow α as in section 3.2. Also define the period P_α as M_α / r_α. Then LUB(A, B) is given by:

$$\begin{aligned}\mathrm{LUB}(A, B) \equiv \{\ & b_{\mathrm{LUB}(A, B)} \equiv \mathrm{MAX}(b_A, b_B),\\ & r_{\mathrm{LUB}(A, B)} \equiv (M_{\mathrm{LUB}(A, B)} / P_{\mathrm{LUB}(A, B)}),\\ & p_{\mathrm{LUB}(A, B)} \equiv \mathrm{MAX}(p_A, p_B, r_{\mathrm{LUB}(A, B)}),\\ & m_{\mathrm{LUB}(A, B)} \equiv \mathrm{MAX}(m_A, m_B),\\ & M_{\mathrm{LUB}(A, B)} \equiv \mathrm{MAX}(M_A, M_B)\\ \}\end{aligned}$$

where:

$P_{\mathrm{LUB}(A, B)} \equiv \mathrm{GCF}(P_A, P_B)$;

the function MAX(x, y) means “take the higher of the pair (x, y)”;

the function $\mathrm{MAX}(x, y, z) \equiv \mathrm{MAX}(\mathrm{MAX}(x, y), z)$;

the function GCF(x, y) means “take the Greatest Common Factor of the pair (x, y)”.

The LUB of *n* flows ($n \neq 2$), LUB(n_1, n_2, ...), is defined recursively as:

$$\mathrm{LUB}(n_1, n_2, ..., N) \equiv \mathrm{LUB}(n_1, \mathrm{LUB}(n_2, ..., N))$$

In addition, the slack term in the corresponding RSpec must allow any component flow to use the resources. In order to ensure that this criterion is met, the RSpec for the flow is set to the minimum value of the RSpec values in the component flows. That is:

$$S_{\mathrm{LUB}(A, B)} \equiv \mathrm{MIN}(S_A, S_B)$$

Where the function MIN(x, y) means take the lower of the pair (x, y).

The following example shows how TSpec parameters are determined using LUB algorithm specified above:

1) As the result of codec negotiation, the following codecs are selected for a call:
G711(20ms) and G728(10ms)

2) The LUB bucket depth for the selected codecs is:

G711(20ms) = (8000 / 50) + 40 = 200 bytes
G728(10ms) = (2000 / 100) + 40 = 60 bytes
b[LUB] = m[LUB] = M[LUB] = MAX(200, 60) = 200 bytes

3) The LUB bucket rate for selected codecs is:

P [LUB] = GCF(10ms, 20ms) = 10ms = 0.01 second
r[LUB] = M * 1/P = 200 * 1/0.01 = 20,000 bytes per second
r[G711(20ms)] = 200 * 1/0.02 = 10,000 bytes per second
r[G728(10ms)] = 60 * 1/0.01 = 6,000 bytes per second
p[LUB] = MAX(10000, 6000, 20000) = 20,000 bytes per second

3.2.2 Mapping PacketCable Codecs into DQoS RSVP Requests

The following DQoS profile of the DOCSIS 1.1 QoS mechanisms must be used in support of voice services. Due to the fact that the PacketCable defined codecs [21] are transported using a constant bit-rate data stream, the following rules MUST be applied in constituting DQoS messages:

The parameters *RSVP Bucket Depth* (b), *RSVP Maximum Datagram Size* (M), and *RSVP Minimum Policed Unit* (m) MUST be equal to each other.

The parameters *RSVP Bucket Rate* (r), *RSVP Peak Rate* (p) and *RSVP Reserved Rate* (R) MUST be equal to each other.

The *RSVP Slack Term* MUST be set to a value supplied by the CMS, if the value is not supplied by the CMS then the default values of 800microseconds for upstream and the value zero for downstream MUST be used.

The *RSVP Protocol* MUST be set to UDP.

The *RSVP Destination Address* MUST be set to IP address to which the data stream packets will be sent if known for the direction that the parameter is being used. If the parameter is not known then the value zero MUST be used in this field.

The *RSVP Destination Port* MUST be set to UDP port to which the data stream packets will be send. if known for the direction that the parameter is being used. If the parameter is not known then the value zero MUST be used in this field.

The *RSVP Source Address* MUST be set to IP address from which the data stream packets will be send if known for the direction that the parameter is being used. If the parameter is not known then the value zero MUST be used in this field.

The *RSVP Source Port* MUST be set to UDP port from which the data stream packets will be send if known for the direction that the parameter is being used. If the parameter is not known then the value zero MUST be used in this field.

If a entity receives a DQoS message that does not conform to the requirements outlined in this specification, then the entity MUST permanently reject the request.

3.2.3 Mapping RSVP FlowSpecs into DOCSIS 1.1 QoS Parameters

The CMTS, on receiving a reservation request, decides:

- What type of DOCSIS service to use (e.g., unsolicited grant, real-time polled, etc.)
- What QoS parameters to associate with the corresponding Service Flow

The RSVP objects for both directions must be set as follows:

Sender TSpec and *Reverse Sender TSpec*:

Bucket Depth (b), bytes = VoIP datagram size, including IP/UDP/RTP header overhead

Bucket Rate (r), bytes/second = actual data rate, including IP/UDP/RTP header overhead

Maximum Datagram Size (M), bytes = Bucket Depth (b)

Minimum Policed Unit (m), bytes = Bucket Depth (b)

Peak Rate (p), bytes/second = Bucket Rate (r)

Sender RSpec and *Reverse Sender RSpec*:

Reserved Rate (R), bytes/second = Bucket Rate (r)

Slack Term (s), microseconds = tolerated grant jitter for upstream flows, tolerated latency for downstream flows[4] This value is in the range of 0 <= s <= 2*packetization interval with a default value of 800 microseconds for upstream, and in the range 50 <= s <= 2*packetization interval or a (default) value of zero for downstream direction if these values are not specified by the CMS to the MTA. In the downstream direction, zero indicates no restrictions on latency.

Session and *Reverse Session*:

Protocol = UDP

Destination Address = IP address to which the data stream packets will be sent

Destination Port = UDP port to which the data stream packets will be sent

Sender Template and *Reverse Sender Template*:

Source Address = IP address from which the data stream packets will be sent

Source Port = UDP port from which the data stream packets will be sent

The same values MUST be used when committing the resources.

3.2.3.1 Upstream Quality of Service Encodings

The DOCSIS upstream objects must be set as stated below. All the other service flow quality-of-service TLV encodings MUST NOT be defined, thereby allowing the default values to be used. If the MTA provides one of these TLVs then the CMTS MUST reject the request with a “reject permanent/reject admin” error code.

The *DOCSIS Active Timeout* timer value is used to detect inactivity and initiate resource recovery for committed service flows. MTA/CMTS synchronization may be coordinated by the CMTS by providing an appropriate value in the DSA/DSC REQ/RSP message. This field MUST NOT be populated by the MTA.

The *DOCSIS Admitted Timeout* timer value is used to detect inactivity and initiate resource recovery for reserved service flows. MTA/CMTS synchronization may be coordinated by the CMTS by providing an appropriate value in the DSA/DSC REQ/RSP message. This field MUST NOT be populated by the MTA.

The *DOCSIS Assumed Minimum* Reserved Rate Packet Size parameter MUST NOT be set for upstream flows.

The *DOCSIS Grants per Interval* parameter MUST be set to 1.

The *DOCSIS Nominal Grant Interval* parameter MUST be set to the codec packetization interval.

DOCSIS Nominal Grant Interval = 10000 or 20000 or 30000

The *DOCSIS Tolerated Grant Jitter* parameter MUST be set to a CMS-specified a value which is based on routing cost information. Allowed range for this parameter is between 0 and 2*packetization interval. If the value is not specified by the CMS, a default value of 800 microseconds MUST be used.

[4] The actual value specified depends upon the end-to-end latency budget for a particular call, and the CMS may specify a value based on routing information (e.g., propagation delay to destination).

The *DOCSIS Nominal Polling Interval* parameter MUST NOT be specified for UGS service flows, and SHOULD be set to a value that is integer multiple of the codec packetization interval for UGS/AD service flows.

The *DOCSIS Tolerated Polling Jitter* parameter MUST NOT be specified for UGS service flows, and SHOULD be set to a value that is integer multiple of the codec packetization interval for UGS/AD service flows.

The *DOCSIS Request/Transmission Policy* parameter is a bitmask and bits 0-6 and 8 MUST be set for UGS and UGS/AD service flows.

The *DOCSIS TOS Overwrite* parameter MUST NOT be used. Even though this parameter is defined by DOCSIS, the use of the field is prohibited by PacketCable.

The *DOCSIS Unsolicited Grant Size* parameter MUST be calculated from the DOCSIS MAC header FC to end of CRC. The value includes Ethernet header overhead of 18 bytes (6 bytes for source address, 6 bytes for destination address, 2 bytes for length, and 4 bytes for CRC). The value also incorporates DOCSIS MAC layer overhead, including the DOCSIS base header (6 bytes), the UGS extended header (3 bytes), and the BPI+ extended header (5 bytes). If payload header suppression (PHS) is active then the number of suppressed bytes MUST NOT be included. Note that the PHS extended header (2 bytes) MUST NOT be included for UGS or UGS/AD service flows, since the appropriate information is embedded in the UGS extended header.

DOCSIS Unsolicited Grant Size^8,9 = M + 32-PHS[5][6]

The *DOCSIS Upstream Scheduling Type parameter* MUST be set to either to UGS or UGS/AD, depending on whether silence suppression is supported on the call.

If the MTA is making a reservation or commit for a codec that does not perform Voice Activity Detection then the MTA MUST use the UGS as scheduling type , otherwise it MUST use UGS/AD.

If the MTA is making a reservation for a Service Flow for multiple codecs of which one of them it will perform Voice Activity Detection, then the MTA MUST request the UGS/AD for reservation and commit for only the active codec's properties as described above paragraph.

3.2.3.2 Upstream Packet Classification Encodings

DOCSIS Upstream Packet Classification Requests

The DOCSIS upstream objects must be set as stated below. All the other classification TLV encodings MUST NOT be defined, thereby allowing the default values to be used. If the MTA provides one of TLVs that are to be omitted, then the CMTS MUST reject the request with a "reject permanent/reject admin" error code.

If defined by the CMTS, the *DOCSIS Classifier Identifier* parameter MUST be used. Otherwise, the *DOCSIS Classifier Reference* parameter MUST be set to a unique value per Dynamic Service Message.

The *DOCSIS Service Flow* Reference parameter MUST be set to an E-MTA unique value for existing calls for DSA_REQ messages, and MUST be omitted in all other messages. Instead, the CMTS-issued DOCSIS Service Flow Identifier parameter MUST be used.

The *DOCSIS Rule Priority* parameter MUST be set to 128.

The *DOCSIS Classification Activation* State parameter MUST be set to active (1) when the call utilizing the service flow is committed, and for all the other cases it MUST be set to inactive (0).

The *DOCSIS Dynamic Service Change Action* MAY use the DSC Add Classifier (0), DSC Replace Classifier (1) and DSC Delete Classifier (2) operations per the DOCSIS RFI specification.

[5] This example assumes that BPI+ is being used as mandated by the PacketCable Security specification.

[6] The PHS used in this example is defined in the DOCSIS RFI specification [9]

The *DOCSIS IP TOS* and mask fields MAY be omitted, since PacketCable does not incorporate TOS parameters as part of its classifier. Alternatively, if this parameter is included it MUST correspond with the TOS value specified by the CMS or a provisioned value for voice service flows.

The *DOCSIS IP Protocol* parameter MUST be set to UDP (17).

The *DOCSIS IP Source Address* parameter MUST be set to the same address as that in the Sender Template, so long as a non-zero value is provided. If the address specified in the Sender Template object is zero, this parameter MUST be omitted.

The *DOCSIS IP Source Mask* parameter MUST be omitted.

The *DOCSIS IP Source Port Start* and *DOCSIS IP Source Port End* parameters MUST be set to the same transport port value as the Sender Template.

The *DOCSIS IP Destination Address* parameter MUST be set to the same address as that in the Session Object, so long as a non-zero value is provided. If the address specified in the Session Object is zero, this parameter MUST be omitted.

The *DOCSIS IP Destination Mask* parameter MUST be omitted.

The *DOCSIS IP Destination Port Start* and *DOCSIS IP Destination Port End* parameters MUST be set to the same transport port as the Session Object, so long as a non-zero value is provided. If the Destination IP Port is specified as a value of zero in the Session Object, then the DOCSIS IP Destination Port Start and End TLVs MUST be omitted.

The *DOCSIS Ethernet LLC Packet Classification Encodings* parameters MUST be omitted.

The *DOCSIS 802.1P/Q Packet Classification Encodings* parameters MUST be omitted.

CMTS Behavior for DOCSIS Upstream Packet Classification Requests

Upon Reception of the Classifier Addition request (e.g., via DOCSIS DSx messaging) the CMTS MUST compare the Gate settings referenced by the GateID to the TLVs. If the TLVs do not match, the CMTS MUST return the DOCSIS Classifier Error Encoding with following information:

- The Error Code parameter MUST contain a "reject-authorization-failure" value.
- The Errored Parameter parameter MUST reference the first TLV that failed authorization. Since different implementations MAY authenticate the TLVs in different order, the TLV returned in this field MAY be different under identical conditions.
- The *Error Message* parameter MAY be populated.

3.2.3.3 Payload Header Suppression Encodings

DOCSIS Payload Header Suppression Requests

Payload Header Suppression is optional; however, if used, the requirements below must be followed. These rules apply to PHS on both upstream and downstream flows.

The *DOCSIS Payload Header Suppression Field* parameter references the bytes of the headers which MUST be suppressed by the sending entity, and MUST be restored by the receiving entity.

The *DOCSIS Payload Header Suppression Size* parameter MUST be equal to the total number of bytes in the Payload Header Suppression Field (PHSF).

The *DOCSIS Payload Header Suppression Mask* parameter MUST indicate the bytes to be suppressed.

The *DOCSIS Payload Header Suppression Verification* parameter SHOULD be set to 0 (verify).

The *DOCSIS Classifier Identifier* parameter MUST be used if defined by the CMTS. Otherwise, the *DOCSIS Classifier Reference* parameter that was in used in the classifier definition MUST be used.

The *DOCSIS Classifier Reference* parameter MUST be used if DOCSIS Classifier Identifier is not defined by the CMTS. Otherwise, the DOCSIS Classifier Identifier parameter that was in used in the classifier definition MUST be used.

The *DOCSIS Service Flow* Identifier parameter MUST be used if defined by the CMTS. Otherwise, the DOCSIS Service Flow Reference parameter that was in used in the classifier definition MUST be used.

The *DOCSIS Dynamic Service Change* Action MAY use the Add PHS Rule (0), Set PHS Rule (1) Delete PHS Rule (2), and Delete All PHS Rules operations per the DOCSIS RFI specification.

CMTS Behavior for DOCSIS Payload Header Suppression Requests

The PHS error handling described here provides a fairly sophisticated feedback mechanism between the CMTS which rejects an initial PHS request and the requesting MTA with the intent that the information provided in the error response may be used to facilitate a successful alternative approach (i.e., the successful admission of the UGS flow without suppression or with a simpler PHS rule).

Upon reception of DSx request with DOCSIS Payload Header Suppression, if a CMTS decides that it cannot support the requested suppression (perhaps due to a lack of local processing or memory resources) but can support the Unsolicited Grant Service without suppression, it MUST return the confirmation code “reject-header-suppression” in the DOCSIS Payload Header Suppression Error Encodings along with the DOCSIS Errored Parameter as described below. The DOCSIS Error Message MAY be used.

If the CMTS cannot support a requested complex DOCSIS Payload Header Suppression, but can support a simpler one then the CMTS MUST provide the DOCSIS Payload Header Suppression Mask in the DOCSIS Errored Parameter field.

DOCSIS Errored Parameter = DOCSIS Payload Header Suppression Mask

If the CMTS cannot support a requested size for the DOCSIS Payload Header Suppression but can support a smaller DOCSIS Payload Header Suppression Size, then the CMTS MUST provide the DOCSIS Payload Header Suppression Size in DOCSIS Errored Parameter field.

DOCSIS Errored Parameter = DOCSIS Payload Header Suppression Size

E-MTA Behavior for DOCSIS Payload Header Suppression Requests

Upon reception of a “reject-header-suppression” confirmation code in which the DOCSIS Errored Parameter includes the DOCSIS Payload Header Suppression Mask, the E-MTA MAY re-request the bandwidth without DOCSIS Payload Header Suppression or MAY redefine the DOCSIS Payload Header Suppression Mask such that the mask would contain a simpler suppression rule (e.g., indicating a contiguous block of suppressed bytes).

Upon reception of a “reject-header-suppression” confirmation code in which the DOCSIS Errored Parameter includes the DOCSIS Payload Header Suppression Size, the E-MTA MAY re-request the bandwidth without DOCSIS Payload Header Suppression.

E-MTA use of the DOCSIS UGS Extended Header

The DOCSIS Payload Header Suppression Index parameter MUST contain the pre-established PHS index value or zero when there are no Payload Header Suppression defined for the Service Flow.

The DOCSIS Queue Indicator parameter MUST be set by the E-MTA whenever more than one packet has been queued for transmission. Otherwise, this value SHOULD be cleared to zero.

The DOCSIS Active Grants parameter MUST be set to one whenever the E-MTA is not in the Silence Suppression, and MUST be set to zero whenever the E-MTA is in Silence Suppression for the codec that is being used for the data stream associated with this Service Flow.

3.2.3.4 Downstream Quality of Service Encodings

The DOCSIS downstream service flow quality-of-service TLV encodings MUST be set as stated below. All other TLVs MUST NOT be defined, thereby allowing the default values to be used. If the MTA uses one of these TLVs, then the CMTS MUST reject the request with a “reject permanent/reject admin” error code.

The downstream DOCSIS parameters are calculated from DOCSIS MAC header the byte following the HCS to end of CRC. The MAC layer (i.e., Ethernet) overhead is 18 bytes (6 bytes for source address, 6 bytes for destination address, 2 bytes for length, and 4 bytes for CRC).

Based on this overhead, *the DOCSIS Assumed Minimum Reserved Rate Packet Size* parameter MUST be calculated as:

DOCSIS Assumed Minimum Reserved Rate Packet Size = m + 18 – PHS

The *DOCSIS Maximum Sustained Traffic Rate*[7] parameter is given in bits per second, including Ethernet (not DOCSIS) MAC layer overhead. The conversion from IP-specific parameters involves first determining the packetization rate by dividing the Peak Rate by the Minimum Policed Unit. This value is then multiplied by the packet size, amended to include MAC layer overhead, and the entire product is scaled from bytes to bit. The DOCSIS maximum sustained traffic rate MUST be calculated as:

DOCSIS Maximum Sustained Traffic Rate = p / m * (m + 18 – PHS) * 8

The DOCSIS Minimum Reserved Traffic Rate[8] parameter is calculated in a manner similar to the DOCSIS Maximum Sustained Traffic Rate, except that instead of using the Peak Rate Parameter (p), the Reserved Rate (R) is used.

DOCSIS Minimum Reserved Traffic Rate = R / m * (m + 18 – PHS) * 8

The *DOCSIS Maximum Traffic Burst* parameter MUST be set to the greater of: (1) an integer multiple of Assumed Minimum Reserved Rate Packet Size or (2) the DOCSIS specified minimum value of 1522.

DOCSIS Maximum Traffic Burst = max((M + 18 – PHS) * 3, 1522)

The *DOCSIS Traffic Priority* parameter MUST be set to five.

The *DOCSIS Downstream Latency* parameter MUST NOT be used.

The *DOCSIS Active Timeout* timer value is used to detect inactivity and initiate resource recovery for committed service flows. Because both upstream and downstream service flows and Gates are managed under a single GateID and are deleted in pairs, in the PacketCable model it is not necessary to monitor both upstream and downstream flows for activity. For this reason, only upstream service flows are monitored through the use of the DOCSIS Active Timeout value. This field MUST NOT be populated by the MTA or CMTS for downstream service flows.

The *DOCSIS Admitted Timeout* timer value is used to detect inactivity and initiate resource recovery for reserved service flows. However, by the same logic as described above for the DOCSIS Active Timeout parameter, monitoring of downstream service flows through the use of the DOCSIS Admitted Timeout parameter is not defined in the PacketCable model. This field MUST NOT be populated by the MTA or CMTS for downstream service flows.

3.2.3.5 Downstream Packet Classification Encodings

DOCSIS Downstream Packet Classification Requests

The DOCSIS downstream classification objects MUST be set as stated below. All the other classification TLV encodings MUST NOT be defined, thereby allowing the default values to be used. If the MTA includes one of TLVs that are to be omitted, then the CMTS MUST reject the request with a “reject permanent/reject admin” error code.

If defined by the CMTS, the *DOCSIS Classifier Identifier* parameter MUST be used. Otherwise, the *DOCSIS Classifier Reference* parameter MUST be set to a unique value per Dynamic Service Message

[7] It should be noted that if a value has a fractional value then it has to be rounded up.

[8] It should be noted that if a value has a fractional value then it has to be rounded up.

The *DOCSIS Service Flow Reference* parameter MUST be set to a E-MTA unique value for the existing calls for DSA_REQ messages, and MUST be omitted in all other messages. Instead, the CMTS-issued *DOCSIS Service Flow Identifier* parameter MUST be used.

The *DOCSIS Rule Priority* parameter MUST be set to 128.

The *DOCSIS Classification Activation State* parameter MUST be set to active (1) when the call utilizing the service flow is committed, and for all the other cases it MUST be set to inactive (0).

The *DOCSIS Dynamic Service Change Action* MAY use the DSC Add Classifier (0), DSC Replace Classifier (1) and DSC Delete Classifier (2) operations per the DOCSIS RFI specification.

The *DOCSIS IP TOS* and mask fields MUST NOT be used.

The *DOCSIS IP Protocol* parameter MUST be set to UDP (17).

The *DOCSIS IP Source Address* parameter MUST be set to the same address as that in the Reverse Sender Template, so long as a non-zero value is provided. If the address specified in the Reverse Sender Template object is zero, this parameter MUST be omitted.

The *DOCSIS IP Source Mask* parameter MUST be omitted.

The *DOCSIS IP Source Port Start* and *DOCSIS IP Source Port End* parameters MUST be set to the same transport port value as indicated in the Reverse Sender Template, so long as a non-zero value is provided. If the Source IP Port is specified as a value of zero in the Reverse Sender Template, then the DOCSIS IP Source Port Start and End TLVs MUST be omitted.

The *DOCSIS IP Destination Address* parameter MUST be set to the same address as indicated in the Reverse Session object.

The *DOCSIS IP Destination Mask* parameter MUST be omitted.

The *DOCSIS IP Destination Port Start* and *DOCSIS IP Destination Port End* parameters MUST be set to the same port as indicated in the Reverse Session object.

The *DOCSIS Ethernet LLC Packet Classification Encodings* MUST be omitted.

The *DOCSIS 802.1P/Q Packet Classification Encodings* MUST be omitted.

CMTS Behavior for DOCSIS Downstream Packet Classification Requests

Upon Reception of the Classifier Addition request (e.g., via DOCSIS DSx messaging) the CMTS MUST compare Gate settings referenced by the GateID to the TLVs of the request. If the TLVs do not match, the CMTS MUST return a DOCSIS Classifier Error Encoding with following information:

The *Error Code* parameter MUST contain “reject-authorization-failure”.

The *Errored Parameter* parameter MUST point to the first TLV that failed authorization. Since different implementations may authenticate TLVs in different order, the TLV returned in this field may be different under identical conditions.

The *Error Message* parameter MAY be populated.

3.2.3.6 Example of Mapping

Consider the following example. A voice codec produces a CBR output data stream of 64 kbps which is packetized at 10ms intervals, thus producing an 80 byte payload each 10 ms. The payload is encapsulated using RTP/UDP/IP, an extra 40 bytes, yielding a 120 byte packet each 10 ms. The TSpec in this case is

bucket depth (b) = 120 bytes
bucket rate (r) = 12,000 bytes/second
peak rate (p) = 12,000 bytes/second
min policed unit (m) = 120 bytes
maximum datagram size (M) = 120 bytes

Suppose a client requests a reservation using this TSpec and an RSpec with R = r. A CMTS receiving this request will establish a Service Flow that uses Unsolicited Grant Service because p = r and M = b, indicating a CBR flow. It may use a grant size of M bytes at an interval of M/R = 10 ms.

For the calculation of jitter, the MTA does not know how much the CMTS deviates from ideal in its scheduling behavior. The client should assume that the CMTS is ideal, which means that the delay it will experience with the above TSpec and its reserved rate R = r is simply

b/r + propagation delays

Ignoring the propagation delay, this results in a delay of 10 ms. Suppose that the client is willing to tolerate a 15 ms delay for this session (on the client-CMTS path only). It would then set its slack term (S) to 15 - 10 = 5 ms. On receiving the reservation, the CMTS interprets this as an indication that a 5ms grant jitter is acceptable to the client.

Suppose that the client is willing to tolerate a 25 ms delay, and sets its slack term to 25 - 10 = 15 ms. The CMTS may use this information to determine that it can use a longer grant interval, e.g., 20 ms, since this potentially increases delay up to 20ms for a packet that arrives at the CM right after a grant. There is still 5 ms of slack left, which the CMTS may use to set the grant jitter.

Note that this approach leaves considerable flexibility in the CMTS to meet the requirements of the client with regard to delay in whatever way best matches the capabilities of the CMTS.

3.2.3.7 Payload Header Suppression and VAD

If the CMTS and CM perform header suppression, then the bandwidth that is needed on a Service Flow can be reduced. The client MUST convey to the CMTS the fact that suppression may be applied prior to the installation of a reservation to ensure that appropriate bandwidth is reserved. The general solution to this problem is described in [4]. The sender (client) adds a parameter (Compression_Hint), described in [4], to the Sender-TSpec that identifies the type of compression or header suppression that might be applied to the data. The Compression_Hint parameter contains a Hint field that advertises the type(s) of compression that is possible.

An MTA that desires the CM to perform header suppression MUST include the Compression_Hint parameter [4] in the TSpec. The Compression factor field, a percentage in the range 1 to 100 inclusive, MUST be set to an amount that yields the bandwidth savings when DOCSIS PHS (42 bytes) is used. The value for Compression factor varies relative to the traffic profile of the CODEC. The Hint MUST be set to one of the following values depending on the type(s) of compression/suppression the MTA desires:

0x40090001	Do not suppress UDP checksum AND do not suppress IP-Ident field nor IPChecksum field
0x40090002	Do not suppress UDP checksum AND suppress IP-Ident field and IP-Checksum field
0x40090003	Suppress UDP checksum AND do not suppress IP-Ident field nor IP-Checksum field
0x40090004	Suppress UDP checksum AND suppress IP-Ident field and IP-Checksum field Note that suppression of the IP-Ident field will create problems if the packet is subsequently fragmented within the IP network. For packets less than 576 bytes in length (Internet default value of MAX-MTU), it is reasonable to assume no fragmentation will occur. The MTA SHOULD NOT request the IP-Ident field be suppressed if it will be sending packets longer than 576 bytes.

A CMTS connected to a CM that is capable of performing header suppression uses the Compression_Hint parameter [4] to reduce the effective rate and depth of the token bucket supplied by the sender. If header suppression is not supported on a link, the Compression_Hint parameter is ignored and the full TSpec is used.

When performing header suppression on a DOCSIS link, it is also necessary to communicate the contents of the header that will be suppressed to the CMTS in advance of the first data packet's transmission so that the suppression context can be established at the CM and the CMTS. All this information is present in the RSVP message that is used to establish the reservation, including source and destination IP addresses and ports. Since PATH messages are processed by any intermediate hops between the client and the CMTS, an arriving PATH message will contain the same TTL value as data packets, provided PATH messages and data packets have the same initial TTL when sent by the client. The CMTS MUST use the contents of the PATH to learn the values of the fields that will be suppressed.   The CMTS MUST use DOCSIS messaging to convey to the CM the fact that suppression should be used for a particular flow, and instructs it to suppress appropriate fields given the presence or absence of UDP checksums and IP Sequence numbers.

If the MTA initiates a PATH message specifying a wildcard sender, then no contents of the PHS field can be accurately determined. The CMTS MUST specify the PHS Size so the CM can accurately assess the resource needs of the service flow.

The same basic approach enables support of Voice Activity Detection (VAD). A CMTS may use different scheduling algorithms for flows that are using VAD, and thus needs to know which flows may be treated with VAD. The Compression_Hint parameter carried in the TSpec MUST contain the flag bit to indicate that the data flow for which this reservation is being requested may be treated with VAD.

3.2.4 UGS and UGS/AD Authorization and Behavior

The GC does not specify the DOCSIS service flow scheduling type for DQOS flows. As such, guidelines are established regarding their use.

For a normal session:

- The CMTS should select UGS or UGS/AD service flow scheduling type for a standalone MTA using RSVP based upon the VAD compression hint parameter. It may provide provisioned parameters to control the decision.

For cases where a reservation covers multiple FlowSpecs (Least Upper Bound)

- In RSVP messaging if VAD is indicated, the CMTS should create a UGS/AD service flow scheduling type. The CMTS may also provide provisioned parameters to control the decision.

For cases where resources are to be shared among flows

- In RSVP messaging the CMTS must use the same service flow scheduling type for all shared resources. Therefore, the MTA must request identical settings for VAD. The CMTS will reject any request to share resources if the VAD setting does not match the existing flow.

3.2.5 CMTS Authorization and Behavior

The CMTS upon receiving bandwidth reservation or commitment requests containing a GateID must perform admission control on the bandwidth request using the gate objects associated with the GateID.

Each DSA or DSC request originating from an E-MTA in support of a particular call session MUST contain a GateID in the Authorization Block, otherwise the CMTS MUST reject the request with confirmation code 24 (Authorization Failure).   If a DSC request message is received which contains a GateID different from the GateID provided in the DSA request used to create the service flow, then the CMTS MUST perform normal authorization and admission control procedures using the Gate associated with the new GateID.

If authorization and admission control succeed, the CMTS MUST associate the new GateID with the modified service flow, replace the DOCSIS Admitted Flow Timeout and Active Flow Timeout values of the associated service flow with the T7 and T8 timers of the new upstream Gate, and include those timer values in the DSC response to the MTA . In this case the CMTS MUST remove the original Gate immediately and notify the CMS via a Gate-Close with Reason-Sub-Code 0 (Normal).

The CMTS and CMS elements MUST NOT reuse a Gate previously associated with a service flow in authorizing a separate service flow.   A CMTS MUST reject a reservation or commit request for a new service flow against a Gate authorizing a separate service flow with DOCSIS confirmation code 24 (Authorization Failure).

If the PacketCable authorization module receives a bandwidth reservation request without an authorization block, the CMTS MUST reject the request with confirmation code “24: authorization failure”.

If the CMTS cannot find a gate associated with the GateID it MUST return a confirmation code “24: authorization failure”, indicating that this request has failed authorization and will be rejected.

If the CMTS finds a gate associated with the GateID then the CMTS must conduct the following authorization procedure. In order to perform admission control on DOCSIS DSx messages and to compare these messages on a parameter basis with those authorized via the GateSpec object, the CMTS must normalize QoS parameters either to layer-two or layer-three by adding or subtracting link-layer overhead. The examples provided in this specification assume that normalization results in layer-three parameters by converting DOCSIS parameters to their RSVP equivalents using the methods described in Section 3.2.

Sender TSpec and *Reverse Sender TSpec*:

The GateSpec Bucket Depth (b), MUST be greater than or equal to the MTA requested value.

The GateSpec Bucket Rate (r), MUST be greater than or equal to the MTA requested value.

The GateSpec Maximum Datagram Size (M), MUST be greater than or equal to the MTA requested value.

The GateSpec Minimum Datagram Size (m), MUST be greater than or equal to the MTA requested value.

The GateSpec Peak Rate (p), MUST be greater than or equal to the MTA requested value.

Sender RSpec and *Reverse Sender RSpec*:

The GateSpec Reserved Rate (R), MUST be greater than or equal to the MTA requested value.

The GateSpec Slack Term (s), MUST be less than or equal to the MTA requested value.

Session and *Reverse Session*:

The GateSpec Protocol MUST be equivalent to the MTA requested protocol.

The GateSpec Destination Address MUST be the same as the MTA requested address, if the GateSpec contains a non-zero value. If the GateSpec contains a zero value then this comparison MUST be omitted.

The GateSpec Destination Port MUST be the same as the MTA requested port if the GateSpec contains a non-zero value. If the GateSpec contains a zero value then this comparison MUST be omitted.

Sender Template and *Reverse Sender Template*:

The GateSpec Source Address MUST be the same as the MTA requested address, if the GateSpec contains a non-zero value. If the GateSpec contains a zero value then this comparison MUST be omitted.

The GateSpec Source Port MUST be the same as MTA requested port if the GateSpec contains a non-zero value If the GateSpec contains a zero value then this comparison MUST be omitted.

If one of the above authorization comparisons fails for a message requesting a new service flow or altering the reservation parameters of an existing flow, then the CMTS MUST NOT honor the request by creating a new service flow or altering the parameters of the existing service flow. If the MTA requests a commit operation for a reserved flow then the authorization MUST be carried out using the DOCSIS parameters and the method defined in DOCSIS [9].

3.3 Definition of Additional RSVP objects

Several new RSVP objects MUST be added to the original PATH message sent by the MTA. All new objects have a class-number with the high order two bits set, which means that RSVP nodes that do not recognize these objects should forward them without modification. This section defines the formats of the various new objects that are to be carried in RSVP messages. All objects use the encoding scheme of RSVP [1].

3.3.1 Reverse-RSpec

Reverse-RSpec object:

Length (= 24)		Class (= 226)	C-Type (= 1)
0 (a)	Reserved	4 (b)	
2 (c)	0 Reserved	3 (d)	
130 (e)	0 (f)	2 (g)	
Rate [R] (32-bit IEEE floating point number)			
Slack Term [S] (32-bit integer)			

(a) - Message format version number (0)

(b) - Overall length (4 words not including header)

(c) - Service header, service number 2 (Guaranteed)

(d) - Length of service 1 data, 3 words not including header

(e) - Parameter ID, parameter 130 (Guaranteed Service RSpec)

(f) - Parameter 130 flags (none set)

(g) - Parameter 130 length, 2 words not including parameter header

See [2] for explanation of fields.

The Reverse-RSpec applies to data sent by the client, i.e., upstream in the DOCSIS network. It is included in the PATH message sent by the client, and is turned into the Forward-RSpec object in the RESV message generated by the CMTS in its role as proxy for the remote endpoint.

3.3.2 Reverse-Session

IPv4 Reverse Session object:

Length (= 12)		Class (= 226)	C-Type (= 2)
IPv4 Destination Address (4 bytes)			
Protocol ID	Flags	Destination Port	

The Reverse-Session object describes the destination information of the data stream to be received by the client, i.e., downstream in the DOCSIS network. It becomes the Session Object in the PATH message generated by the CMTS in its role as proxy for the remote endpoint.

3.3.3 Reverse-Sender-Template

IPv4 Reverse-Sender-Template object:

Length(= 12)		Class(= 226)	C-Type(= 3)
IPv4 Destination Address (4 bytes)			
Reserved	Reserved	Source Port	

The Reverse-Sender-Template describes the source information of the data stream to be received by the client, i.e., downstream in the DOCSIS network. It becomes the Sender-Template object in the PATH message generated by the CMTS in its role as proxy for the remote endpoint.

3.3.4 Reverse-Sender-TSpec

Reverse-Sender-TSpec object; has the same fields as Sender-TSpec described in [4].

Length(= 48)		Class(= 226)	C-Type(= 4)
0 (a) ¦ Reserved		10 (b)	
1 (c)	0¦ Reserved	9 (d)	
127 (e)	0 (f)	5 (g)	
Token Bucket Rate [r] (32-bit IEEE floating point number)			
Token Bucket Size [b] (32-bit IEEE floating point number)			
Peak Data Rate [p] (32-bit IEEE floating point number)			
Minimum Policed Unit [m] (32-bit integer)			
Maximum Packet Size [M] (32-bit integer)			
126(h)	flags (i)	2 (j)	
Hint (assigned number) (k)			
Compression factor (32-bit integer) (l)			

(a) - Message format version number (0)

(b) - Overall length (10 words not including header)

(c) - Service header, service number 1 (default/global information)

(d) - Length of service 1 data, 9 words not including header

(e) - Parameter ID, parameter 127 (Token_Bucket_TSpec)

(f) - Parameter 127 flags (none set)

(g) - Parameter 127 length, 5 words not including header

(h) - Parameter ID, parameter 126 (Compression_Hint)

(i) - Parameter 126 flags (none set)

(j) - Parameter 126 length, 2 words not including header

(k) - Hint value defined for DOCSIS PHS

0x????0001	Do not suppress UDP checksum AND do not suppress IP-Ident field nor IP-Checksum field
0x????0002	Do not suppress UDP checksum AND suppress IP-Ident and IP-Checksum field
0x????0003	Suppress UDP checksum AND do not suppress IP-Ident nor IP-Checksum field

0x????0004 Suppress UDP checksum AND suppress IP-Ident field and IP-Checksum field

(Note: ???? = to be assigned by IANA)

(l) - Compression factor value – the percentage reduction in packet size as a result of using DOCSIS PHS, which saves 42 bytes per packet. Note this varies depending on the CODEC used.

See [2] for explanation of fields.

The Reverse-Sender-TSpec describes the data flow to be sent by the client, i.e., upstream in the DOCSIS network. It becomes the Sender-TSpec object in the PATH message generated by the CMTS in its role as proxy for the remote endpoint.

3.3.5 Forward-RSpec

Forward-RSpec object:

Length(= 24)		Class(= 226)	C-Type(= 5)
0 (a) ¦ Reserved		4 (b)	
2 (c)	0 ¦ Reserved	3 (d)	
130 (e)	0 (f)	2 (g)	
Rate [R] (32-bit IEEE floating point number)			
Slack Term [S] (32-bit integer)			

For definitions of these fields, see section 3.3.1 Reverse-RSpec.

The Forward-RSpec applies to data flowing toward the client, i.e., downstream in the DOCSIS network. This object appears in a PATH message sent by the client, and the contents are incorporated into the FlowSpec object in the returned RESV message.

3.3.6 Resource-ID

Resource-ID object:

Length(= 8)	Class(= 226)	C-Type(= 7)
Resource ID (32-bit integer)		

The Resource-ID object is returned in a RESV message to the client, and contains the identifier used for future resource changes. It is also included in PATH messages sent by the MTA in requests to share resources across multiple sessions.

3.3.7 GateID

GateID object:

Length(= 8)	Class(= 226)	C-Type(= 8)
Gate ID (32-bit integer)		

The GateID object is included in PATH messages from the MTA to identify the proper resource authorization at the CMTS.

3.3.8 Commit-Entity

IPv4 Commit-Entity object:

Length(= 12)	Class(= 226)	C-Type(= 9)
IPv4 Destination Address (4 bytes)		
Reserved	Destination Port	

The Commit-Entity object is returned in a RESV message from the CMTS, and indicates the destination address and port number to which the MTA is to send the COMMIT message.

3.3.9 DCLASS

DCLASS object:

Length(= 8)		Class(= 225)	C-Type(= 1)
Unused	Unused	Unused	DSCP

The DCLASS object is returned in a RESV message from the CMTS, and indicates the DSCP that SHOULD be used by the MTA when sending data packets over this reservation to the CMTS. The use of the DCLASS object is described in [19].

3.4 Definition of RSVP Messages

This section defines the enhanced RSVP messages that MUST be generated by the MTA and MUST be supported by the CMTS.

RSVP messages MUST be sent as raw IP datagrams with protocol number 46. The RSVP PATH message MUST be sent with the RouterAlert option [14] in the IP header. Each RSVP message MUST occupy exactly one IP datagram.

All RSVP messages MUST consist of a Common Header, followed by a variable number of variable-length objects. The Common Header MUST be as follows:

Version	Flags	Message Type	RSVP Checksum
Sent-TTL		(Reserved)	RSVP Message Length

Values of each field MUST be as specified in [1].

Each object MUST consist of one or more 32-bit words with a one-word header, of the following format:

Length in bytes	Class-Number	C-Type
Object Contents …		

Values of each field MUST be as specified in [1].

The format of the RSVP PATH message and RSVP RESV message compliant with this specification MUST contain the following objects (items in italics are defined in this specification, all others in [1] and/or [2]).. For objects not defined in this specification, the object ordering rules MUST be followed according to [1]. There is no ordering requirement for the <Resource-ID>, <GateID>, and <Commit-Entity> objects. <Reverse-RSpec> and <Downstream-FlowSpec> MUST follow the <Sender-TSpec> object. Objects defined in <Downstream-FlowSpec> MUST follow the order shown in their BNF below:

```
<PATH-Message> :: = <Common-Header> [<Integrity-Object>]
                    <Session-Object> <RSVP-Hop> <Time-Values>
                    [<Policy-Data> …] <Sender-Template>
                    <Sender-TSpec> <Reverse-RSpec>
```

```
                        <Downstream-FlowSpec> [<Resource-ID>]
                        <GateID>

<Downstream-FlowSpec> :: = <Reverse-Session> <Reverse-Sender-Template>
                             <Reverse-Sender-TSpec><Forward-RSpec>

<RESV-Message> :: = <Common-Header> [<Integrity-Object>]
                        <Session-Object> <RSVP-Hop> [<DCLASS>]
                        <Time-Values> [<RESV-Confirm>] [<Scope>]
                        [<Policy-Data> …] <Resource-ID>
                        <Commit-Entity> <Style> <FlowSpec>
                        <Filter-Spec>
```

The various components of these messages are described in the following sections.

3.4.1 Message Objects for Upstream Reservation

A standard RSVP PATH message contains, at a minimum, the following objects:

<Session> <RSVP-Hop> <Time-Values> <Sender-Template> <Sender-TSpec>

However, in the segmented model, it is necessary to get all the information to the CMTS that would enable it to make a bi-directional reservation on the DOCSIS link. It is also necessary to enable it to send an RSVP RESV towards the client. A standard RSVP RESV message contains, at a minimum, the following objects:

<Session> <RSVP-Hop> <Time-Values> <Style> <FlowSpec> <Filter-Spec>

The CMTS MUST generate such a message towards the client after receiving an RSVP PATH message from the MTA. The only object here that is not derivable from the RSVP PATH or local information is the FlowSpec. The Filter-Spec, which consists of the IP address and source port to be used by the client, is derived from the Sender-Template in the PATH. Almost everything in the FlowSpec can be derived from the Sender-TSpec in the PATH message. The exceptions to this are the values of R (reserved rate) and S (slack), which together constitute the RSpec. Thus, the client provides a suitable RSpec, containing R and S for guaranteed service, which is encoded as in RFC2210 [2] This is enclosed in a Reverse-RSpec object, which is described in section 3.3.1.

3.4.2 Message Objects for Downstream Reservation

The client MUST provide enough information to allow the CMTS to construct an RSVP PATH message for the downstream data flow given that it has just received an RSVP PATH message for the upstream data flow. This means the client MUST provide the following objects that relate to the downstream (CMTS->MTA) data flow.

<Session> <Sender-Template> <Sender-TSpec>

These objects have their normal RSVP definitions, and apply to the simplex data stream that will flow from the far endpoint to the client. In the RSVP PATH message sent by the client, they are assigned new object codes (as noted above) and new names (Reverse-session, Reverse-sender-template, Reverse-Sender-TSpec). The Reverse-Session-Object MUST contain the IP address of the client, the protocol type, and the port (if applicable) on which it will receive data for this flow. The Reverse-Sender-Template MUST contain the IP address of the far endpoint, or zeroes if the source is intended as a wildcard. The Reverse-Sender-Template MUST contain the port number, if applicable and known, otherwise zero. The Reverse-Sender-TSpec MUST contain the TSpec information that describes the data flow from the far endpoint. The CMTS MUST use its own address as the RSVP-Hop and choose a value for Time-Values that indicates how often it will refresh the RSVP PATH message. Even if the CMTS does not need to generate the RSVP PATH message to send it to the MTA, this information is necessary to enable it to establish a reservation and create packet classifiers in the downstream direction.

Given the information described above, the one additional piece of information that the CMTS requires to make a reservation in the downstream direction is an RSpec. Again, it is assigned a new object number and name, Forward-RSpec. It contains the same information elements and is encoded in the same way as a conventional RSpec.

Note that a Forward-RSpec applies to data that is flowing towards the client, which means that it is sent by the client in the same direction as the RSVP RESV that would normally carry this information. It is provided in the RSVP PATH message simply as an optimization to reduce setup latency. A Reverse-RSpec is sent by the client in the opposite direction to the RSVP RESV that would normally carry this information.

3.5 Reservation Operation

This section describes the required behavior of the MTA and CMTS to cooperatively perform resource reservations.

For the purposes of this discussion, the endpoint that is in direct communication with the CMTS is referred to as the client, and the other endpoint of the session is referred to as the far endpoint. No assumptions are made about what types of devices these might be (gateways, PCs, embedded clients). It is assumed that the client uses RSVP to communicate QoS requests to the CMTS, and no assumptions are made about the capabilities of the far endpoint. The data flow from client to CMTS is referred to as upstream, and the flow from CMTS to client is referred to as downstream.

3.5.1 Reservation Establishment

RSVP operation under the segmented model is as follows:

The client MUST send a RSVP PATH message towards the far endpoint of the session, which MUST be intercepted by the CMTS. This initiates the process of reserving both upstream and downstream bandwidth. The RSVP PATH MUST carry information about both upstream (i.e., Reverse-RSpec) and downstream (i.e., Reverse-Sender-TSpec, Forward-RSpec) resource requirements in the case where reservations are required in both directions.

The CMTS MUST verify that the amount of resources requested is within the authorized amount for this session and that it has sufficient local resources to accommodate the reservation. It then reserves upstream and downstream resources and MUST perform the DOCSIS level messaging to allocate appropriate resources on the DOCSIS link.

The CMTS MUST establish classifiers for the upstream and downstream flows. The upstream classifier MUST contain the client's source IP address and port number from the Sender Template object. The upstream classifier MUST contain the protocol type, destination IP address and port number from the Session Object. If the Reverse-Sender-Template Object is present, and contains an address other than 0.0.0.0, then the downstream classifier MUST contain this address as the source IP address. If the Reverse-Sender-Template is present, and contains a port number other than zero, then the downstream classifier MUST contain this value as the source port. The downstream classifier MUST contain the protocol type, destination IP address, and port number from the Reverse Session Object.

The CMTS MUST perform any backbone resource reservation necessary, based on the provisioned algorithm defined for the particular backbone configuration.

If the access and backbone reservations succeed, the CMTS MUST send a RSVP RESV to the client. The contents of the RSVP RESV MUST be derived from the RSVP PATH: the Session-Object is copied from the RSVP PATH, Style is set to Fixed-Filter, FlowSpec is formed from the Sender-TSpec and Forward-RSpec, Filter-Spec is set from the Sender-Template, and the Resource-ID is generated, containing the Resource-ID assigned to the allocated resources. The Commit-Entity Object MUST be included, and contain the address of the CMTS and port number on which the CMTS will accept the COMMIT message (as described in Section 3.6). The DCLASS object SHOULD be included and value set based on the DiffServ Code Point field of the gate.

If the address of the previous hop differs from the Source Address of the RSVP PATH message, then the CMTS MUST generate a RSVP PATH for downstream reservations. The contents of the RSVP PATH MUST be derived from the RSVP PATH received from the client. The Session-Object MUST be obtained from the Reverse-Session-Object in the RSVP PATH message. If the address contained in the Reverse-Sender-Template is 0.0.0.0, or the port number is zero, then the Sender-TSpec and Sender-Template are not sent in the RSVP PATH. Otherwise, the Sender-TSpec is obtained from the Reverse-Sender-TSpec, the Forward-RSpec is obtained from the Reverse-RSpec, and the Sender-Template is obtained from the Reverse-Sender-Template. The Resource-ID object is generated, and contains the Resource-ID assigned to the allocated resources. The MTA MAY use the Reverse-Sender-TSpec it sent in the RSVP PATH message in calculating the Filter-Spec returned in its RSVP RESV reply, or MAY generate a Wildcard-Filter reply.

On receipt of the RSVP RESV message, the client knows that necessary resources have been reserved. At this point, in the case of a successful reservation, the client knows that it has a reservation in both directions, and can proceed with the call signaling to ring the phone at the far end.

If the reservation does not succeed, the CMTS MUST send a RSVP PATH-ERR message to the client, indicating why the reservation failed (e.g., lack of authorization, insufficient resources, etc.) If the reservation failed for policy reasons, the RSVP PATH-ERR message MUST contain a RSVP-ERROR-SPEC object with the following Error Codes and Error Values.

- Error Code = 2 (Policy Control Failure), Error Value = 3 (Generic Policy Rejection) is returned if the RSVP PATH did not contain a GateID object or the GateID object did not match any gates known to the CMTS.
- Error Code = 1 (Admission Control Failure), Error Value = 2 (Requested bandwidth unavailable) is returned if the RSVP PATH was rejected because there was no more resources that are available for the priority level of the gate. In these cases, the MTA MAY take special action indicating the specific error to the user. If the RSVP PATH failed for non-policy reasons, it MUST contain a RSVP-ERROR-SPEC object with an Error Code and Error Value as defined in [1].

The sender of a RSVP PATH (MTA or CMTS) is responsible for reliably installing the reservation. When the sender transmits a RSVP PATH, it MUST receive an RSVP RESV or RSVP PATH-ERR message within a configured timeout interval of Timer T3 (see Section 6).

Whenever an MTA or CMTS transmits an RSVP message that requires an acknowledgement, the sender MUST include an RSVP-MESSAGE-ID object in that message, and the ACK_Desired flag of the RSVP-MESSAGE-ID object MUST be set. The MTA and CMTS MUST set the Refresh-Reduction-Capable flag in the common header of every RSVP message. When the MTA or CMTS receives an RSVP message with an RSVP-MESSAGE-ID object, it MUST respond with an RSVP message that contains an RSVP-MESSAGE-ACK or RSVP-MESSAGE-NACK object. The RSVP-MESSAGE-(N)ACK object MAY be piggy-backed onto standard RSVP messages, but MAY be transmitted in an RSVP-ACK message if the receiver of the RSVP-MESSAGE-ID object has no other RSVP message to send at the time. For example, the CMTS SHOULD NOT delay processing of a received RSVP PATH message, but if it chooses to delay, it MUST reply immediately with an RSVP-ACK message, to be followed by a RSVP RESV message later.

RSVP-ACK messages carry one or more RSVP-MESSAGE-(N)ACK objects. They MUST NOT contain any other RSVP objects except an optional RSVP-INTEGRITY object. When included, an RSVP-MESSAGE-(N)ACK object MUST be the first object in the message, unless an RSVP-INTEGRITY object is present (in which case, the RSVP-MESSAGE-(N)ACK object MUST immediately follow the RSVP-INTEGRITY object). The MTA or CMTS MAY use RSVP-INTEGRITY objects.

The use of RSVP-MESSAGE-ID and RSVP-MESSAGE-(N)ACK objects can be used to ensure reliable RSVP message delivery in the face of network loss. Since the MTA or CMTS sets the ACK_Desired flag, it MUST retransmit unacknowledged messages at a more rapid interval than the standard RSVP refresh interval until the message is acknowledged or until an interval of Timer T3 (see Section 6) elapses. A rapid retransmit rate based on well-known exponential back-off functions MUST be used. An initial retransmit timeout of Timer T6 (see Section 6) MUST be used, with a power of 2 back-off. The rapid retransmit process ends when either an RSVP-MESSAGE-(N)ACK object is received or Timer T3 expires. If RSVP PATH sender does not receive a RSVP RESV, RSVP PATH-ERROR, or RSVP-MESSAGE-(N)ACK before the next retransmit, it MUST assume either its original RSVP PATH or the response from the other end was lost and re-sends the RSVP PATH. Since all RSVP messages are idempotent, no duplications of reservations will occur.

In PacketCable, only RSVP PATH messages MUST include RSVP-MESSAGE-ID objects with the ACK_Desired flag set. RSVP-MESSAGE-ID objects MAY be used in other RSVP messages.

RSVP-MESSAGE-IDs are used on a per-RSVP-hop basis. Each RSVP-capable hop in the path that supports refresh reduction does its own fast retransmit until it sees an acknowledgement from the next upstream node. So if a standalone MTA behind an RSVP-capable CM receives an RSVP-MESSAGE-ACK object from the CM for an RSVP PATH and the CM is waiting for an RSVP-MESSAGE-ACK from the CMTS for the RSVP PATH, the CM will do the fast retransmit while the standalone MTA waits for its normal (30 sec) RSVP PATH refresh timer to expire. (The MTA no longer does fast retransmit because it got an acknowledgement). If an RSVP-capable CM gives up its fast retransmit, it will send back an RSVP PATH-ERROR to the standalone MTA. This way, retransmits do not affect the entire path, just the loss-prone hops.

Reliable message delivery for RSVP messages is defined and described more completely in [23]. [23] also includes mechanisms to reduce the number of RSVP signaling messages required to refresh RSVP state. The MTA and CMTS implementations of [23] MUST:

- enable the Refresh-Reduction-Capable bit in the RSVP header
- support the MESSAGE_ID extension
- support the Summary Refresh extension for unicast sessions
- support reception of Bundle messages.

Additionally, the MTA and CMTS implementations of [23] MAY:

- support the Summary Refresh extension for multicast sessions
- support transmission of Bundle messages

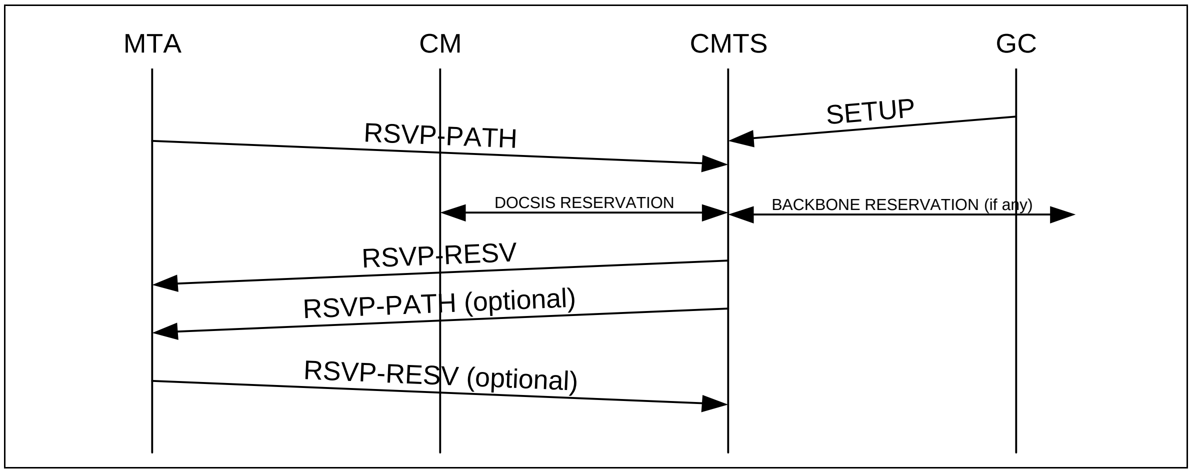


Figure 8. Reservation Establishment

The CMTS MUST enforce upstream packet classification filters for PacketCable Service Flows. That is, the CMTS MUST discard upstream packets, which do not match the set of upstream packet classifiers for the Service Flow. Upstream packet classification filtering is an optional CMTS requirement in DOCSIS 1.1. This specification requires its implementation for Service Flows used to carry PacketCable media streams. If a CMTS chooses to enforce upstream classification filters only on the PacketCable Service Flows, and not on other Service Flows, it is a CMTS vendor-specific decision as to how the particular PacketCable Service Flows are determined. An example CMTS policy would be to enforce upstream packet classification only on non-Primary Upstream Service Flows.

3.5.2 Reservation Change

In addition to establishing a reservation for some amount of resources, it may be necessary to change the resources allocated. Resource usage may need to be increased or decreased. RSVP handles changes in resource usage by changes in the FLOWSPEC object of a RSVP RESV message and/or a change in the Sender-TSpec in a RSVP PATH message. A reservation change MUST follow the same series of steps as the establishment of a new reservation. Admission control SHOULD always succeed for a session, which is changing its resource requirements in a way that does not cause an increase in any dimension relative to the resources previously reserved. Because resources are described by multi-dimensional vectors, a change in reservation that increased resources in one direction and decreased them in another MUST pass through admission control. Note that in order to pass admission control, the resources MUST be within the amount of authorized resources for the session and also within the amount of resources available to the CMTS.

In the event that an existing reservation is preempted because a session with a higher priority gate must be established in the presence of insufficient bandwidth, then the CMTS MUST send a RSVP PATH-ERR and/or PATH-RESV-ERR message for the session that is being preempted. This message SHOULD be sent as soon as possible. In response, the MTA SHOULD tear down the reservation and MAY notify the user of the preemption (e.g. e.g., play a special tone to phone user) The RSVP PATH-ERR (or RSVP RESV-ERR) message in this case MUST contain a RSVP-ERROR-SPEC object with an Error Code of 2 (Policy Control Failure) and an Error Value of 5 (Flow was preempted).

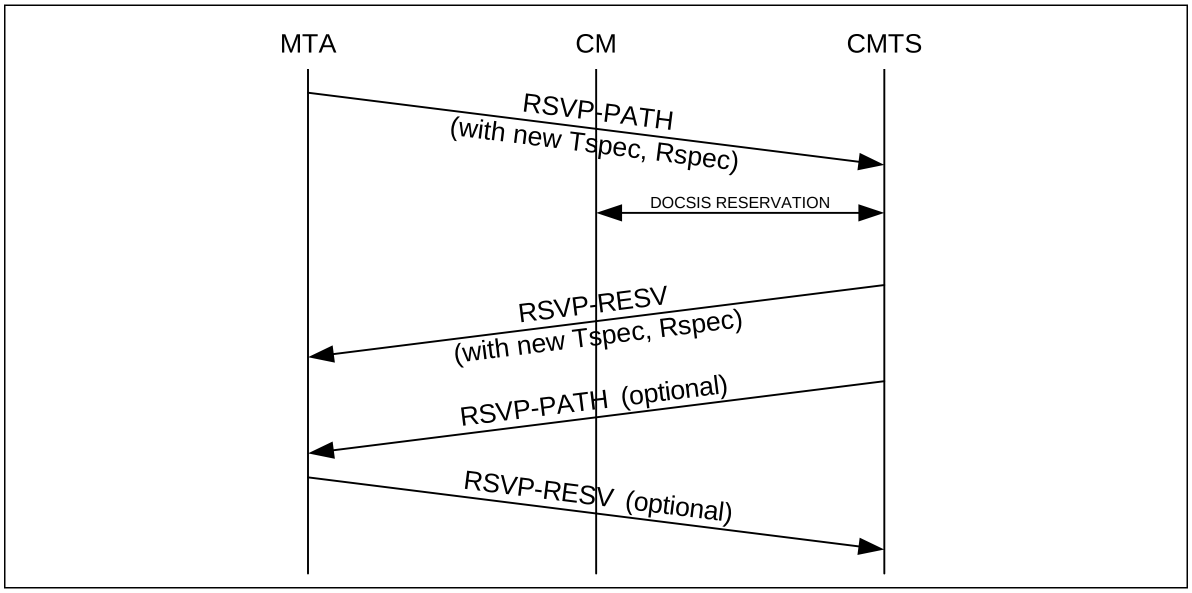


Figure 9. Reservation Change

3.5.3 Reservation Deletion

RSVP provides two messages for the explicit removal of Path and Reservation state, the RSVP PATH-TEAR and RSVP RESV-TEAR messages. To delete a reservation at the CMTS, the MTA SHOULD send a RSVP PATH-TEAR message. To delete a reservation from RSVP-capable devices between the client and the CMTS, the client MAY send a RSVP RESV-TEAR message. The format of these messages MUST be in accordance with [1], and MUST include both the Session-Object and Sender-Template to enable the CMTS to identify the proper gate.

If Path state and Reservation state are not periodically refreshed, they MUST timeout. This is appropriate when a client crashes, for example. More details of refresh mechanisms appear in Section 3.5.4.

The CMTS MUST respond to a received RSVP PATH-TEAR by sending a RSVP RESV-TEAR to the MTA. The format of these messages MUST be as given in [1].

RSVP version 1 provides no means to ensure the reliable delivery of RSVP PATH-TEAR and RSVP RESV-TEAR messages, on the assumption that the state will timeout anyway. However, to avoid any delay in teardown (which causes short-term resource wastage and may cause over billing), the message reliability extension to RSVP described in [5] MAY be used.

3.5.4 Reservation Maintenance

RSVP has a soft-state model, in that reservation state times out if not periodically refreshed. This characteristic is retained in the segmented model described here. Since the entire reservation process in this model is initiated by the MTA, the MTA MUST periodically refresh all RSVP state information. The MTA MUST send RSVP PATH messages as described in Section 3.5.1 within the time interval given by the CMTS in the RSVP RESV Time-Values Object. The CMTS MUST generate RSVP RESV messages towards the MTA on receipt of the RSVP PATH (and a RSVP PATH message as well if RSVP capable nodes have been detected as described in Section 3.5.1). This retains the soft state nature of RSVP, which retains its resiliency in the face of routing changes and node failures.

The MTA (or CMTS) MAY also implement RSVP Summary Refresh as another way to conserve upstream bandwidth when refreshing reservation state. It allows RSVP-capable nodes to "compress" their Path (or Resv) states for multiple reservations into single message. [23] describes summary refresh as follows:

> "The summary refresh extension enables the refreshing of RSVP state without the transmission of standard Path or Resv messages. The benefits of the described extension are that it reduces the amount of information that must be transmitted and processed in order to maintain RSVP state synchronization. Importantly, the described extension preserves RSVP's ability to handle non-RSVP next hops and to adjust to changes in routing. This extension cannot be used with Path or Resv messages that contain any change from previously transmitted messages, i.e., are trigger messages.
>
> The summary refresh extension builds on the previously defined MESSAGE_ID extension. Only state that was previously advertised in Path and Resv messages containing MESSAGE_ID objects can be refreshed via the summary refresh extension.
>
> The summary refresh extension uses the objects and the ACK message previously defined as part of the MESSAGE_ID extension, and a new Srefresh message. The new message carries a list of Message_Identifier fields corresponding to the Path and Resv trigger messages that established the state. The Message_Identifier fields are carried in one of three Srefresh related objects. The three objects are the MESSAGE_ID LIST object, the MESSAGE_ID SRC_LIST object, and the MESSAGE_ID MCAST_LIST object.
>
> The MESSAGE_ID LIST object is used to refresh all Resv state, and Path state of unicast sessions. It is made up of a list of Message_Identifier fields that were originally advertised in MESSAGE_ID objects. The other two objects are used to refresh Path state of multicast sessions. A node receiving a summary refresh for multicast path state will at times need source and group information. These two objects provide this information. The objects differ in the information they contain and how they are sent. Both carry Message_Identifier fields and corresponding source IP addresses. The MESSAGE_ID SRC_LIST is sent in messages addressed to the session's multicast IP address. The MESSAGE_ID MCAST_LIST object adds the group address and is sent in messages addressed to the RSVP next hop.
>
> The MESSAGE_ID MCAST_LIST is normally used on point-to-point links.
>
> An RSVP node receiving an Srefresh message, matches each listed Message_Identifier field with installed Path or Resv state. All matching state is updated as if a normal RSVP refresh message has been received. If matching state cannot be found, then the Srefresh message sender is notified via a refresh NACK.
>
> A refresh NACK is sent via the MESSAGE_ID_NACK object. As described in the previous section, the rules for sending a MESSAGE_ID_NACK object are the same as for sending a MESSAGE_ID_ACK object. This includes sending MESSAGE_ID_NACK object both piggy-backed in unrelated RSVP messages or in RSVP ACK messages. "

Complete details on how summary refresh works can be found in [23].

3.6 Definition of Commit Messages

This section defines the Commit messages that MUST be generated by the MTA and MUST be supported by the CMTS.

Commit messages MUST be sent as UDP/IP datagrams with protocol number 17 (UDP). Each Commit message MUST occupy exactly one UDP/IP datagram. The destination IP address and port number in the UDP header MUST be as specified from the Commit-Entity Object returned in the RSVP RESV message. The source port number MUST be the port on which the MTA will accept the acknowledgement message.

The Commit messages MUST consist of a Common-Header, followed by a variable number of variable-length objects. The Common Header MUST be as follows:

Version	Flags	Message Type	Message Checksum
Sent-TTL		(Reserved)	Message Length

Values of each field MUST be as specified in [1]. Message types MUST be as follows:

COMMIT	240
COMMIT-ACK	241
COMMIT-ERR	242

Each object MUST consist of one or more 32-bit words, with a one-word header of the following format:

Length in bytes	Class-Number	C-Type
Object Contents …		

Values of each field MUST be as specified in [1].

The format of the COMMIT message and COMMIT-ACK message compliant with this specification MUST be as follows (items in italics are defined in this specification in section 3.3, all others in [1] and/or [2]).

<COMMIT-Message> :: = <Common-Header> <Session>
<Sender-Template> <*GateID*>
[<FlowSpec>][<*Downstream-FlowSpec*>]

<COMMIT-ACK-Message>:: = <Common-Header> <Session>
<Sender-Template><*GateID*>

<COMMIT-ERR-Message>:: = <Common-Header> <Session>
<Sender-Template><*GateID*><Error-Spec>

See Section 3.4, Definition of RSVP Messages, for the BNF definition of <Downstream-FlowSpec>.

The Session and Sender-Template objects identify the sender and destination IP addresses and ports, and MUST be present. The Committed resources MAY be less than the total reserved resources (especially in a call-waiting or codec-change scenario), so that a Commit message MAY also contain a <FlowSpec> object for each direction of the session. This provides a mechanism by which the size of the committed resources can be modified up or down as long as the amount of resources committed does not exceed the reserved resources. Note that a set of resources MAY be put on hold (frozen) by lowering the committed resources to zero while leaving the reserved resources in place. If either FlowSpec is omitted, the CMTS MUST set the amount of committed resource in that direction to equal to the amount of reserved resources.

3.7 Commit Operations

A significant aspect of the Dynamic QoS model is that reservation is a two-phase process, with a Commit phase following the Reserve phase. Section 3.5 above described the Reserve phase, while this section describes the Commit Phase and its relationship to the Reserve phase.

A conformant CMTS MUST perform all admission control and resource allocation functions on receipt of the original RSVP PATH message, but MUST NOT allow access to those resources by the MTA until a COMMIT message is received, unless told otherwise in the GATE-SET parameters.

To perform a COMMIT the client MUST send a unicast message from the MTA to the CMTS. This is desirable because the Commit phase only involves a client and a gate. The client learns the CMTS address and port number from the Commit-Entity object in the RSVP RESV message.

Note that a COMMIT message differs in an important way from a standard RSVP message. It is sent directly from the client to the CMTS rather than hop-by-hop as an RSVP message would be. However, it contains objects that are syntactically the same as RSVP objects.

The CMTS MUST verify the value of GateID, and verify the contents of the Session and Sender-Template objects match the existing reservation with the same value of GateID, and that the Reverse-Session and Reverse-Sender-Template, if present, match the existing reservation with the same value of GateID. The CMTS MUST acknowledge the receipt of a COMMIT with a COMMIT-ACK message or a COMMIT-ERR message.

A COMMIT-ACK is sent after the successful completion of DOCSIS DSC message exchange between the CMTS and the CM. If the DOCSIS messaging fails, a COMMIT-ERR is sent instead. When an MTA does not receive the acknowledgement within a timeout interval of Timer T4 (see Section 6, the MTA MUST resend the COMMIT, up to a limit of seven attempts.

If the MTA desires to change the amount of committed resources within the reserved envelope, another COMMIT / COMMIT-ACK sequence is REQUIRED.

If the MTA desires to change the amount of reserved resources, then the RSVP PATH/RSVP RESV exchange MUST be repeated.

4 EMBEDDED MTA TO CM QOS PROTOCOL (PKT-Q1)

Rather than using the pkt-q3 interface as described in Section 3, an embedded MTA MAY dynamically reserve local QoS resources using only mechanisms defined in DOCSIS. Using this alternate approach, an embedded MTA directly signals for the local access QoS using the MAC Control Service interface defined in the DOCSIS 1.1 RFI specification [9]. As opposed to Section 3, the QoS signaling across the DOCSIS RFI (pkt-q2 interface) is initiated by the CM instead of the CMTS. All other interfaces remain unchanged. An embedded MTA signals its session level QoS requirements in signaling protocols (DCS and NCS). Once the embedded MTA determines that QoS resources need to be reserved or committed, the MTA MUST initiate DOCSIS Dynamic Service Flow signaling to cause the creation, change, and/or deletion of Service Flow(s) and the allocation of DOCSIS resources. Whether the session is originated by the embedded MTA or by a peer or network node, the MTA passes the QoS requirements to the DOCSIS MAC via the MAC Control Service Interface. This results in the creation or modification of the necessary Service Flow(s) for the session using the Dynamic Service Flow messaging mechanisms of DOCSIS 1.1. The sections that follow discuss the MTA's mapping of session level QoS requirements into DOCSIS, the DOCSIS support for two phase reserve/commit, and use of the DOCSIS 1.1 MAC Control Service Interface.

4.1 Mapping FlowSpecs into DOCSIS 1.1 QoS Parameters

For a detailed description of the DOCSIS parameter mapping process to be used in establishing and maintaining both upstream and downstream service flows, please refer to Section 3.2.3 of this specification. The MTA MUST use the requirements defined in that section for mapping session level QoS requirements into DOCSIS QoS parameters.

As a supplement to these requirements, embedded MTAs MUST include their own send (i.e., upstream source) and receive (i.e., downstream destination) addresses and ports in all classifier TLVs provided via DSx messaging. Far-end addresses and receive ports MAY be wildcarded if the far-end SDP has not been provided and values have not been provided via LCO. If these values are provided in either format, they MUST be included in the classifier TLVs. Far-end source ports MUST in all cases be wildcarded since this parameter is not communicated via SDP.

It should be noted that the examples included in Section 3 do include the overhead associated with the DOCSIS BPI+ extended header, as mandated in the PacketCable Security specification. If BPI+ is disabled (e.g., for testing purposes) the values provided in these examples should be updated appropriately by subtracting five bytes of link-layer overhead from the upstream Grant Size calculation.

4.2 DOCSIS 1.1 Support for Resource Reservation

In DOCSIS 1.1 there is no defined way of passing authorization information from the CM to the Authorization Module within the CMTS. The Authorization Module is a logical function of the CMTS defined in DOCSIS 1.1. This specification utilizes a new DOCSIS TLV which passes an Authorization Block consisting of an arbitrary string of length *n* to the CMTS to be interpreted and processed only by the Authorization Module.

The DQoS model is one in which each session is authorized. The authorization of each session uses a handle given to both the CMTS and to the MTA, which is used to match requests with authorizations. This handle is the GateID. Upon receiving call signaling information, the MTA passes the GateID to the CMTS using the AuthBlock TLV contained in a DSA/DSC message.

4.2.1 Two Phase QoS Reservation/Commit

A DOCSIS 1.1 Service Flow has three associates sets of Quality of Service Parameters, referred to as the Provisioned, Admitted, or Active QoS Parameter Set. The relationship between these is identical to the description of Authorized, Reserved, and Committed resources given in Section 2.7.4.

The Reserve and Commit operations are both performed by the use of DOCSIS Dynamic Service messages, by changing the values of the AdmittedQoSParameterSet and ActiveQoSParameterSet of the Service Flow. In a Dynamic Service Addition (DSA) or Dynamic Service Change (DSC) message, Reserve is accomplished by including, in the Upstream Service Flow Encodings or Downstream Service Flow Encodings, the QoSParameterSetType TLV with value set to Admitted (value 2). Similarly, Commit is accomplished by setting the QoSParameterSetType TLV to Active (value 4) or Admitted+Active (value 6).

DSA and DSC exchanges between the CM and CMTS are three-way handshakes, consisting of a request message followed by a response followed by an acknowledgement. This is illustrated in Figure 10.

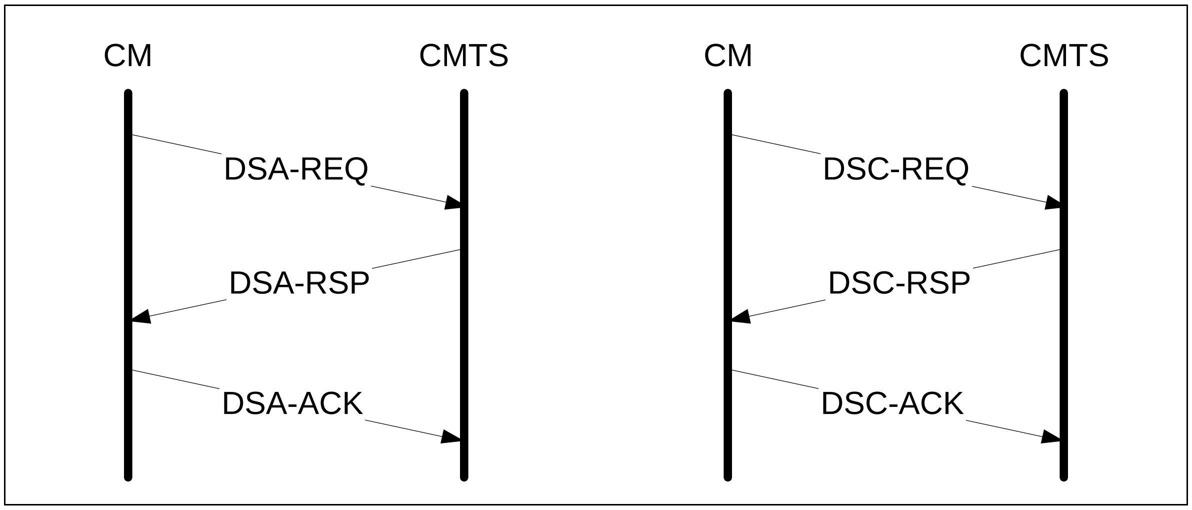


Figure 10. DSA and DSC exchanges between CM and CMTS

For example, the following DSA-REQ message causes the Upstream and Downstream Service Flows to be admitted, meaning the QoS resources to be used in the DOCSIS network are reserved.

DSA-REQ

TransactionID		1
UpstreamServiceFlow	ServiceFlowReference	1
	QoSParameterSetType	Admitted (2)
	ServiceFlowScheduling	UGS (6)
	NominalGrantInterval	10ms
	ToleratedGrantJitter	2ms
	GrantsPerInterval	1
	UnsolicitedGrantSize	222
DownstreamServiceFlow	ServiceFlowReference	2
	QoSParameterSetType	Admitted (2)
	TrafficPriority	3
	MaximumSustainedRate	12,000

As a further example, the following DSC-REQ message causes the Service Flow to be activated, meaning the QoS resources used in the DOCSIS network are committed.

DSC-REQ

TransactionID		1
UpstreamServiceFlow	ServiceFlowID	10288
	QoSParameterSetType	Admitted + Active(6)
	ServiceFlowScheduling	UGS (6)
	NominalGrantInterval	10ms
	ToleratedGrantJitter	2ms
	GrantsPerInterval	1
	UnsolicitedGrantSize	222
DownstreamServiceFlow	ServiceFlowID	10289
	QoSParameterSetType	Admitted + Active(6)
	TrafficPriority	3
	MaximumSustainedRate	12,000

Specification of Admitted and Activated QoS parameter sets by the MTA is via the MAC_CREATE_SERVICE_FLOW.request and MAC_CHANGE_SERVICE_FLOW.request. By the time a Service Flow is admitted, it typically has associated classifier(s).

4.2.2 Reservation with Multiple Service Flow Specifications

There are various situations in which a reservation needs to cover a range of possible specifications. For example, some applications desire to create a reservation which can handle a switch from one flow specification to another mid-session without having to pass through admission control at each switch-over time. In order for the ActiveQoSParameterSet of a Service Flow to vary during a session, a suitable AuthorizedQoSParameterSet needs to be specified through policies at the Gate Controller. This is accomplished through the use of the least-upper-bound approach (see section 3.2.1). In accordance with section 3.2.4, the Least Upper Bound of flows with two different DOCSIS Scheduling Types is null. Refer to the DOCSIS RFI [9] for information on calculating the least-upper-bound for DSx message flows.

4.2.3 Reservation Maintenance

Whereas RSVP has a soft-state model as described in Section 3.5.4, DOCSIS provides only a timeout mechanism across the DOCSIS interface. The DOCSIS 1.1 Service Flow QoS parameters “Timeout for Active QoS Parameters” and “Timeout for Admitted QoS Parameters” allow a session to be terminated and its resources released due to inactivity.

The TimeoutForActiveQoSParameters is intended to recover resources allocated to CMs that die, crash, or otherwise lose their connectivity to the cable network. Normal transmission of data packets on the service flow is sufficient to prevent this recovery action.

If the DOCSIS Active Timeout expires at the CMTS for a service flow that is authorized via a gate (i.e., a PacketCable service flow) then the CMTS will delete all service flows that are associated with the gate using a DOCSIS DSD request. The CMTS when informing the GC about the gate closure will specify “Timer T8 expiration; Service Flow inactivity at the upstream direction”.

If the MTA is performing Voice Activity Detection, with a service flow scheduling type of UGS/AD, and the CMTS is actively monitoring the upstream flow for activity, then during extended silence periods the MTA MUST either send periodic data packets on the service flow or refresh the active timer via DSC messaging.

The TimeoutForAdmittedQoSParameters is intended to recover resources that are reserved by a CM but not committed. In typical cases, the committed parameters will be identical to the reserved parameters, and this will not be a problem. When the commitment is for less than the reservation, it is necessary to periodically reset the CMTS timer. This is accomplished by performing a DSC-REQ operation that reserves the same resources as previous.

4.2.4 Support for Dynamic Binding of Resources

Dynamic binding of resources, as required in Section 2.7.7 and described in Section 3.1.4, is accomplished in DOCSIS through the use of the authorization block TLV.

The CMTS MUST include the Resource-ID in the authorization block TLV for the DSA-RSP message that it sends to the client. The client MAY include the Resource-ID in subsequent DOCSIS messages that apply to the resources in question. Most importantly, if the client wishes to establish a new session, and reuse the resources of an existing session, it MUST include the Resource-ID associated with the old session in the DSA-REQ message it sends to the CMTS.

4.2.5 QoS Parameter Mapping for Authorization

The Gate identified by the GateID is parameterized by RSVP objects (both FlowSpec and TSpec) for each direction. The authorization module in the CMTS must convert the DOCSIS QoS Parameters into RSVP Objects using the rules defined below:

The parameters *RSVP Bucket Depth* (b), *RSVP Maximum Datagram Size* (M), and *RSVP Minimum Policed Unit* (m) MUST be set to *DOCSIS Unsolicited Grant Size* minus the DOCSIS upstream UGS overhead [9] for upstream direction and *DOCSIS Assumed Minimum Reserved Rate Packet Size* minus the DOCSIS downstream overhead[10] for the downstream direction.

For the downstream, the parameters *RSVP Bucket Rate* (r), and *RSVP Peak Rate* (p) MUST be calculated by converting the *DOCSIS Maximum Sustained Rate* to layer 3 terms by dividing it by the *DOCSIS Assumed Minimum Reserved Rate Packet Size* and then multiplying the result with the previously calculated *RSVP Maximum Datagram Size. For the upstream*, the parameters *RSVP Bucket Rate* (r), and *RSVP Peak Rate* (p) MUST be set equal to the *DOCSIS Nominal Grant Interval* multiplied by the *Unsolicited Grant Size*.

For the downstream, the parameter *RSVP Reserved Rate* (R) MUST be calculated by converting the *DOCSIS Maximum Reserved Traffic Rate* to layer 3 terms by dividing it by the *DOCSIS Assumed Minimum Reserved Rate Packet Size* and then multiplying the result with the previously calculated *RSVP Minimum Policed Unit.* For the upstream, parameter *RSVP Reserved Rate* (R) MUST be set equal to the *DOCSIS Nominal Grant Interval* multiplied by the *Unsolicited Grant Size*.

The *RSVP Slack Term* MUST be set to *DOCSIS Tolerated Grant Jitter* for the upstream. upstream. The RSVP Slack Term MUST be set to zero for the downstream flow, indicating that this parameter will not be specified by the MTA.

The *RSVP Protocol* MUST be set to the *DOCSIS IP Protocol*.

The *RSVP Destination Address* MUST be set to *DOCSIS IP Destination Address.* If this parameter is omitted the value MUST be set to zero.

The *RSVP Destination Port* MUST be set to *DOCSIS IP Destination Port Start.* If this parameter is omitted then the value MUST be set to zero.

[9] The overhead should include Ethernet header overhead of 18 bytes (6 bytes for source address, 6 bytes for destination address, 2 bytes for length, and 4 bytes for CRC). The value also incorporates DOCSIS MAC layer overhead, including the DOCSIS base header (6 bytes), the UGS extended header (3 bytes), and the BPI+ extended header (5 bytes). If payload header suppression (PHS) is active then the number of suppressed bytes must be added to the DOCSIS Unsolicited Grant Size.

[10] The DOCSIS MAC layer overhead is 18 bytes (6 bytes for source address, 6 bytes for destination address, 2 bytes for length, and 4 bytes for CRC). If PHS is used on the downstream, then the number of suppressed bytes must be subtracted from the *DOCSIS Assumed Minimum Reserved Rate Packet Size.*

The *RSVP Source Address* MUST be set to *DOCSIS IP Source Address.* If this parameter is omitted the value MUST be set to zero.

The *RSVP Source Port* MUST be set to *DOCSIS IP Source Port Start* If this parameter is omitted then the value MUST be set to zero.

The resulting converted RSVP objects must then be verified against the corresponding Gate using following rules:

All the requested parameters of *RSVP FlowSpec* and *RSVP Slack Term* MUST be less than or equal to Gate specified values.

All the requested parameters of *RSVP TSpec* MUST be equal to the Gate-specified values, except for the case that the Gate has a value of zero, in which case the corresponding requested parameter MUST NOT be verified.

If verification is successful, then the CMTS MUST continue to process the request. If verification is not successful, then the CMTS MUST reject the request permanently due to authorization failure.

For example, assuming a G.711 codec, framing at 20 ms, with a 2-byte RTP-S MAC, and BPI+ enabled:

G.711 @ 20 ms
64 kbps nominal bit rate
8 kBps nominal byte rate

20 ms framing rate = 50 packets/second
8kBps / 50= 160 bytes per packet of payload

42 bytes of IP/UDP/RTP header
160 + 42 = 202 bytes per packet total
202 * 50 = 10.1 kBps actual byte rate
10.1 * 8 = 80.8 kbps actual bit rate

The resulting GateSpec Parameters as set by CMS would be:

Bucket Depth (b) = datagram size including IP/UDP/RTP-S header overhead = 202 bytes
Minimum Policed Unit (m) = Bucket Depth (b) = 202 bytes
Maximum Datagram Size (M) = Bucket Depth (b) = 202 bytes
Bucket Rate (r) = actual data rate, including IP/UDP/RTP-S header overhead = 10100 bytes per second
Peak Rate (p) = Bucket Rate (r) = 10100 bytes per second
Reserved Rate (R) = Bucket Rate (r) = 10100 bytes per second

Upstream DOCSIS parameters include overhead from the FC byte through the CRC.

DOCSIS base header (FC to HCS, no extended headers): 6 bytes
UGS Extended Header: 3 bytes
BPI+ Extended header: 5 bytes
Ethernet header: 14 bytes
CRC: 4 bytes

Total Upstream Overhead: 32 bytes per packet

DOCSIS Upstream Service Flow Parameters

Upstream Scheduling Type: UGS
Request/Transmission Policy (bitmask): bits 0-6, 8 set (101111111 binary)
Grant Size: 234 bytes
Grants per Interval (integer): 1
Grant Interval: 20000 microseconds
Tolerated Grant Jitter: 800 microseconds

The CMTS Authorization control procedure for the upstream parameters is conducted as follows:

To compare with GateSpec parameters, MAC-layer overhead must be subtracted from DOCSIS parameters.

> GateSpec Bucket Depth (b) >= DOCSIS Unsolicited Grant Size - 32 bytes
> 202 bytes >= 234 bytes - 32 bytes = 202 bytes

> GateSpec Bucket Rate (r) >= 1/DOCSIS Grant Interval * (DOCSIS Unsolicited Grant Size – 32)
> 10.1 kBps >= 1/20 ms * (234 bytes – 32 bytes) = 50 packets per second * 202 bytes per packet = 10.1 kBps

Downstream DOCSIS parameters include overhead from the byte following the HCS through the CRC.

> Ethernet header: 14 bytes
> CRC: 4 bytes

> Total Downstream Overhead: 18 bytes per packet

DOCSIS Downstream Service Flow Parameters

> Maximum Traffic Burst (minimum value of 1522): 1522 bytes
> Maximum Sustained Rate: 88000 bits per second
> Assumed Minimum Reserved Rate Packet Size: 220 bytes
> Minimum Reserved Rate: 88000 bits per second
> Traffic priority: 5

The CMTS Authorization control procedure for the downstream parameters is conducted as follows:

Again, this overhead must be subtracted from the DOCSIS parameters in order to perform the GateSpec comparison. The procedure is straightforward (subtraction) or the DOCSIS Assumed Minimum Reserved Rate Packet Size parameter. However, adjusting the Minimum Reserved Rate parameter is a bit more involved.

> GateSpec Minimum Policed Unit (m) >= DOCSIS Assumed Minimum Reserved Rate Packet Size - 18 bytes
> 202 bytes >= 220 bytes - 18 bytes = 202 bytes

> GateSpec Bucket Rate (r) >= (DOCSIS Minimum Reserved Rate / (8 * DOCSIS Assumed Minimum Reserved Rate Packet Size)) * (DOCSIS Assumed Minimum Reserved Rate Packet Size - 18 bytes)
> 10.1 kBps >= (88 kbps / (8 * 220 bytes)) * (220 bytes - 18 bytes) = 10.1 kBps

4.2.6 Authorization Block Encoding

The authorization block consists of a string of bytes. To allow for flexibility, the authorization block MUST be encoded using Type-Length-Value (TLV) fields. The TLV-tuple fields are unordered, and may be nested. The size of the value field (bytes) must be greater than zero; the sizes of the type and length field are each one byte. Note that the length only includes the value field and not the entire TLV-tuple.

The format of the authorization block is as follows:

- PacketCable Authorization Block Encoding

  This field defines the parameters associated with the PacketCable authorization block. Note that this field consists of nested sub-fields.

Type	Length	Value
1	n	see sub-fields below

- GateID Encoding

  The value of this field specifies the GateID handle used for authorization.

Type	Length	Value
[1].1	4	GateID

- Resource-id Encoding

  The value of this field specifies the resource-id handle used to uniquely identify the set of resources associated with a service flow.

Type	Length	Value
[1].2	4	resource-id[11]

4.3 Use of DOCSIS 1.1 MAC Control Service Interface

The DOCSIS 1.1 QoS parameters for the Service Flow derived from the SDP are signaled to establish the Service Flow(s). This section describes how this can be done using the DOCSIS 1.1 MAC control service interfaces [9].

At the level of DOCSIS MAC Control Service Interface primitives, the Embedded MTA signals for QoS resources as follows:

1. MAC_CREATE_SERVICE_FLOW.request

   As described in [9], the Embedded MTA can request that a Service Flow be added via this primitive. This primitive may also be used to define classifiers for the new Service Flow, as well as supply the Admitted and Active QoS Parameter Sets of the Service Flow. The success or failure of the primitive is indicated via the MAC_CREATE_SERVICE_FLOW response primitive.

2. MAC_CHANGE_SERVICE_FLOW.request

   The Embedded MTA can initiate a change in the Admitted and Active QoS Parameter Sets via this primitive. One possible scenario is the case of putting a callee on hold. The success or failure of the primitive is indicated via the MAC_CHANGE_SERVICE_FLOW response primitive.

3. MAC_DELETE_SERVICE_FLOW.request

   When the Embedded MTA no longer needs the Service Flow, it issues a MAC_DELETE_SERVICE_FLOW.request to the Embedded CM to zero the Active and Admitted QoS Parameter Sets of the Service Flow.

The parameters of these primitives match the parameters associated with the DSA, DSC, and DSD messages as given in [9].

4.3.1 Reservation Establishment

The MTA initiates the reservation of QoS resources by use of the MAC_CREATE_SERVICE_FLOW.request primitive. The MTA MUST include the GateID in the Authorization Block TLV. Upon reception of this message, the MAC layer of the CM invokes DSA signaling by sending a DSA_REQ to the CMTS. The CMTS MUST check the authorization based on the GateID (contained in the Authorization Block TLV), and reject the request if the gate is invalid or the authorized resources are insufficient for the request. Upon receiving the DSA_RSP from the CMTS, the MAC service notifies the upper layer using the MAC_CREATE_SERVICE_FLOW.response message. This is illustrated in the following figure.

[11] dqos-n-01192, po 12/11/01

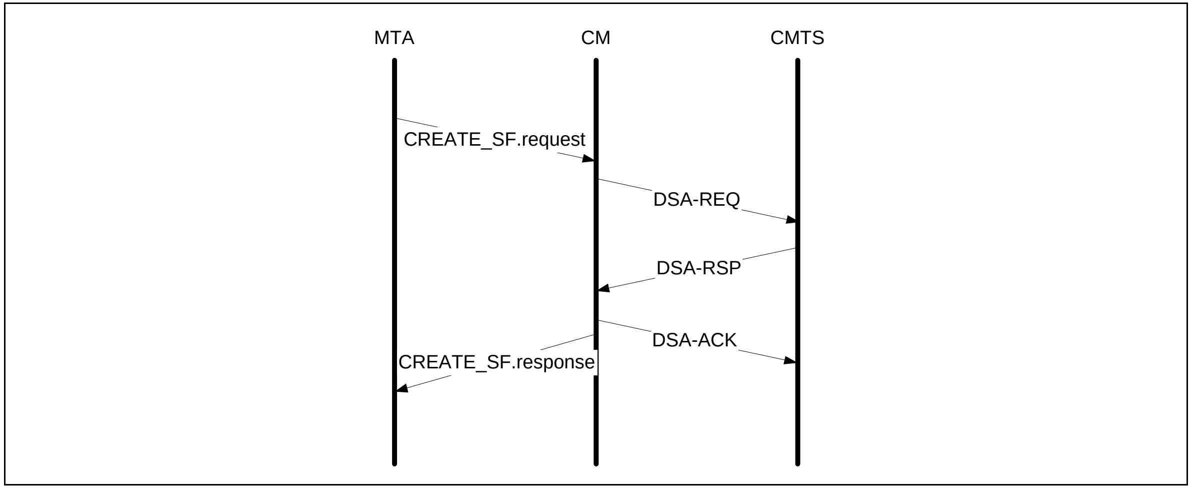


Figure 11. Reservation Establishment

4.3.2 Reservation Change

The MTA initiates changes in QoS resources by use of the MAC_CHANGE_SERVICE_FLOW.request primitive. This is illustrated in the following figure.

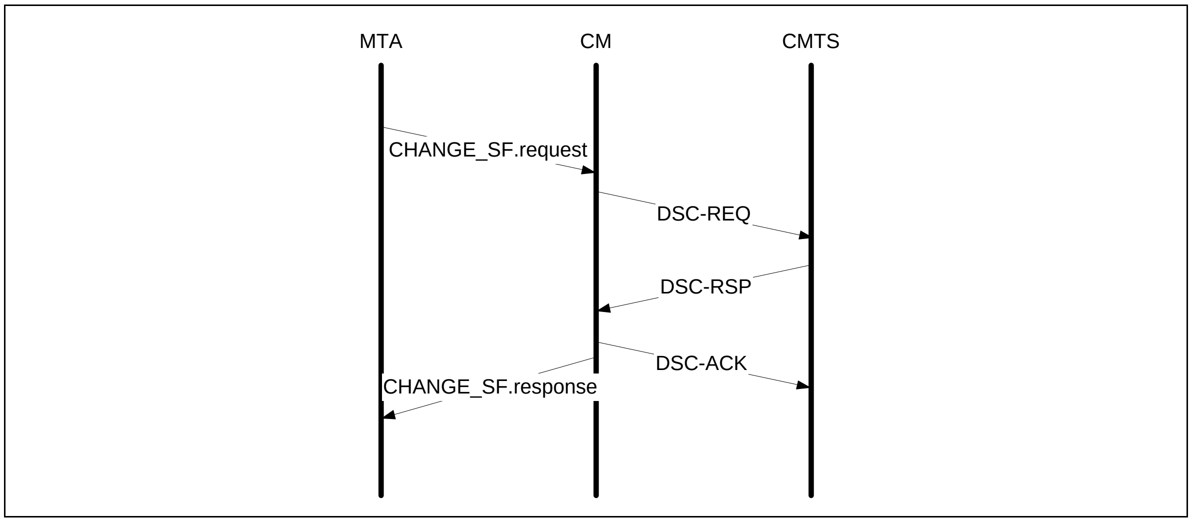


Figure 12. Reservation Change

Upon reception of this message, the MAC layer of the CM invokes DSC signaling. Upon receiving the DSC_RSP from the CMTS, the MAC service notifies the upper layer using the MAC_CHANGE_SERVICE_FLOW.response message.

4.3.3 Reservation Deletion

The MTA initiates the deallocation of QoS reservation by use of the MAC_DELETE_SERVICE_FLOW.request primitive. Upon reception of this message, the MAC layer invokes DSD signaling. Upon receiving the DSD_RSP from the CMTS, the MAC service notifies the upper layer using the MAC_DELETE_SERVICE_FLOW.response message. This is illustrated in the following figure.

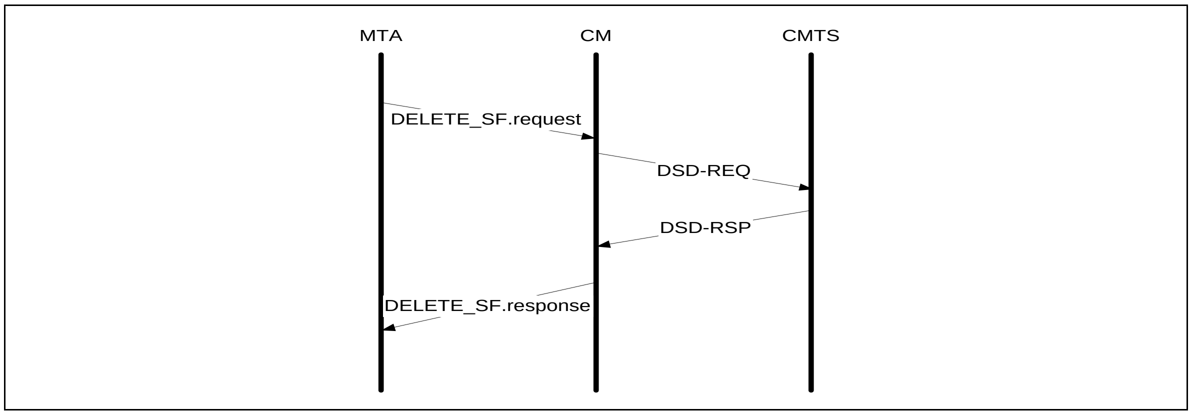


Figure 13. Reservation Deletion

5 AUTHORIZATION INTERFACE DESCRIPTION (PKT-Q6)

This section describes the interfaces between the CMTS and Gate Controller for purposes of authorizing the MTA to receive high Quality of Service. Signaling is required between the Gate Controller and CMTS to support gate management and IP QoS Admission Control Service. In addition, accurate subscriber billing requires the CMTS to indicate actual “committed” QoS resource usage on a per session basis. This section describes the use of the COPS protocol to transport PacketCable QoS defined messages between the Gate Controller and CMTS.

5.1 Gates: the Framework for QoS Control

A PacketCable Dynamic QoS “Gate” is a policy control entity implemented at the CMTS to control access to enhanced QoS Services of a DOCSIS 1.1 cable network by a single IP flow. Gates are unidirectional, in that a single gate controls access to a flow in either the upstream or downstream direction. Gates enable the creation of DOCSIS 1.1 Service Flow Classifiers, which in turn control the routing of packets to DOCSIS Service Flows.

While a Gate also has a N-tuple just like a Classifier, it is not identical to a Classifier. The CMTS MUST setup the Gate when a flow is authorized, until explicitly disabled to terminate the authorization for a flow. A DOCSIS 1.1 Classifier MAY be set up and associated with a Gate. A Gate MAY exist before and after the Classifier it authorizes exists. A Gate MAY be considered to be associated with exactly zero, one, or two Classifiers.

A CMTS conforming to this specification MUST NOT dynamically create a Classifier with a DOCSIS Dynamic Service Addition (DSA) request or response unless it is authorized to do so by the existence of a Gate for that Classifier. An identifier, called the GateID is associated with Gates. The GateID, locally administered by the CMTS where the Gates exists, MAY be associated with one or more unidirectional Gates. For a point-to-point session, typically two unidirectional Gates exist, associated with a single GateID. In addition, DOCSIS 1.1 Classifiers exist for each unidirectional flow that is established.

5.1.1 Classifier

A classifier is a six-tuple:

- Direction (Upstream/Downstream)
- Protocol
- Source IP
- Destination IP
- Destination Port
- Source Port

If there is a upstream and an associated (part of the same session) downstream flow, then there MUST exist separate classifiers for the upstream flow and the downstream flow. The Classifier is updated by the RSVP message for the reservation performed for the upstream and downstream flows. The session data flow MUST match the classifier to receive the Quality of Service associated with the RSVP reservation.

5.1.2 Gate

A Gate is associated with a unidirectional flow, and comprises the following:

- GateID
- Prototype Classifier
- Various flag bits described below
- Authorized Envelope (FlowSpec)
- Reserved Envelope (FlowSpec)
- Resource-ID

The GateID (described below) is a local 32 bit identifier that is allocated from the local space at the CMTS where the Gate resides. Up to two gates MAY share the same GateID. Typically, a GateID will identify a single upstream flow and a single downstream flow, and correspond to a single Multi-media session.

The Prototype Classifier consists of the same six elements as a Classifier, as described above. The Source IP is the IP address (as seen at the CMTS) of the originator of the flow. In the case of an upstream Gate on the DOCSIS 1.1 channel, the Source IP is the IP address of the local MTA. For the downstream flow, the Source IP address is the IP address of the remote MTA. For selected parameters of a Gate's prototype classifier, a wild card is allowed. In Multimedia call signaling, the source UDP Port is not signaled, so its value is not considered to be part of a Gate's information.

The Source Port MAY be wild-carded, to support both PacketCable Call Signaling Protocols (DCS and NCS). If the Source Port is wild-carded, its value in the Gate parameters will be zero.

The Source IP address MAY be wild-carded, to support the NCS Call Signaling Protocol. If the Source IP address is wild-carded, its value in the Gate parameters will be zero.

The Authorized and Reserved Envelopes are RSVP FlowSpecs (both TSpec and RSpec) as described in the earlier sections.

A reservation request for resources (as specified in the PATH message or Dynamic Service Flow Add/Change message) MUST be checked against what has been authorized for the GateID associated with the direction for the resource request. The resources authorized are specified in the Authorized envelope. Also checked is the wild-card in the Gate for particular entries.

The Resource-ID is a local 32-bit identifier that is allocated from the local space at the CMTS where the Gate resides. Any number of gates MAY share a resource-ID, and therefore share a common set of resources, with the restriction that only one of these gates in each direction have resources committed.

5.1.3 Gate Identification

A GateID is a unique identifier that is locally allocated by the CMTS where the Gate resides. The GateID is a 32 bit identifier. A GateID MAY be associated with one or more Gates. In both the NCS and DCS call signaling protocols, a GateID is associated with each call leg, and consists of a single upstream gate and a single downstream gate.

A GateID MUST be associated with the following information:

- One or two Gates, which MUST be one of the following combinations:
    - Single upstream gate
    - Single downstream gate
    - Single upstream gate and a single downstream gate
- Accounting and Billing information
    - Address:Port of the Primary Record-Keeping-Server that should receive event records
    - Address:Port of the Secondary Record-Keeping-Server, for use as specified in [20] if the primary is unavailable.
    - Flag indicating whether the Event Messages are to be sent to the Record Keeping Server in real-time, or whether they are to be batched and sent at periodic intervals
    - Billing-Correlation-ID, which will be passed to the Record-Keeping-Server with each event record.
    - Additional billing information, if supplied, which will be used to generate Call-Answer and Call-Disconnect event messages.
    - Omission of Event Generation information (i.e., the Event-Generation-Info object) implies that Event Message generation MUST NOT be performed for a Gate.

The GateID MUST be unique among all current gates allocated by the CMTS. The value of the 32-bit quantity SHOULD NOT be chosen from a set of small integers, since possession of the GateID value is a key element in the authentication of the COMMIT messages from the MTA. An algorithm that MAY be used to assign values of GateIDs is as follows: partition the 32-bit word into two parts, an index part, and a random part. The index part identifies the gate by indexing into a small table, while the random part provides some level of obscurity to the value. Regardless of the algorithm chosen, the CMTS SHOULD attempt to minimize the possibility of GateID ambiguities by ensuring that no GateID gets reused within three minutes of its prior closure or deletion. For the algorithm suggested previously this could be accomplished by simply incrementing the index part for each consecutively assigned GateID, wrapping around to zero when the maximum integer value of the index part is reached.

5.1.4 Gate Transition Diagram

Gates are considered to have the following states:

- Allocated – the initial state of a gate created at the request of the GC
- Authorized – GC has authorized the flow with resource limits defined
- Reserved – resources have been reserved for the flow
- Committed – resources are being used

The CMTS MUST support gate states and transitions as shown in Figure 14 and described in this section. All gates assigned the same GateID by the CMTS MUST transition together through the states shown in Figure 14. This is true even when only one of the upstream/downstream flows is permitted to pass traffic. In the interest of clarity, the gate transition diagram of Figure 14 does not completely describe all transitions that must be implemented, although all included transitions must be implemented as shown.

A gate is created in the CMTS by either a Gate-Alloc command or a Gate-Set command from the GC. In both cases, the CMTS allocates a locally unique identifier called a GateID, which is returned to the GC. If the gate was created by a Gate-Set message, then the CMTS MUST mark the gate in state “Authorized” and MUST start Timer T1. If the gate was created by a Gate-Alloc message, then the CMTS MUST mark the gate in state “Allocated,” start Timer T0, and MUST wait for a Gate-Set command, at which point the gate MUST be marked in state “Authorized.” If the Timer T0 expires with the gate in state “Allocated” or Timer T1 expires with the gate in state “Authorized,” then the CMTS MUST delete the gate. Timer T0 limits the amount of time the GateID will remain valid without any specified gate parameters. Timer T1 limits the amount of time the authorization will remain valid.

A gate in the “Allocated” state MUST be deleted upon receipt of a Gate-Delete message. When this happens, the CMTS MUST respond with a Gate-Delete-Ack message. and MUST stop Timer T0. Similarly, a gate in the “Authorized” state MUST be deleted upon receipt of a Gate-Delete message. When this happens, the CMTS MUST respond with a Gate-Delete-Ack message and MUST stop Timer T1.

A gate in the “Authorized” state is expecting the client to attempt to reserve resources. The client does this with either a RSVP PATH message or via the MAC Control Services Interface. On receipt of this reserve request, the CMTS MUST verify the request is within the limits established for the gate, and perform admission control procedures.

The CMTS MUST implement at least two admission control policies, one for normal voice communications and one for emergency communications. These two policies MUST have provisionable parameters that specify, at a minimum, (1) a maximum amount of resources that may be allocated non-exclusively to sessions of this type (which may be 100% of the capacity), (2) the amount of resources that may be allocated exclusively to sessions of this type (which may be 0% of the capacity), and (3) the maximum amount of resources that may be allocated to sessions of the two types. The admission control policy MAY also specify whether a new session of that type may “borrow” from lower priority classes or should pre-empt an existing session of some other type to satisfy the admission control policy settings.

If the admission control procedures are successful and only resource reservation was requested, the gate MUST be marked in the “Reserved” state. If admission control procedures are successful and single stage resource reservation and committal was requested, the gate MUST be marked in the “Committed” state and the CMTS MUST send a Gate-Open Message to the GC and stop timer T1.

If admission control procedures are not successful, the gate MUST remain in the “Authorized” state.

Note that the actual reservation made by the client may be for less than that authorized, e.g., reservation for upstream only when a pair of gates were established authorizing upstream and downstream flows.

In the "Reserved" state the gate is expecting the client to Commit to the resources, and thereby activate them. The Commit command from the client is either a unicast UDP message, or a request to activate a service flow via the MAC Control Service Interface. If the gate is still in the "Reserved" state and Timer T1 expires (i.e., the client does not issue the Commit command), the CMTS MUST release any resources reserved, and delete the gate. If a Gate-Delete message is received in the "Reserved" state, the CMTS MUST release all resources associated with this gate, MUST respond with a Gate-Delete-Ack message and MUST stop Timer T1.

For the purpose of this state transition diagram, a "Commit" from the client is a message that commits the upstream flow. If the CMTS receives an asymmetric request such that traffic may pass on the downstream flow but not on the upstream flow, the CMTS MUST NOT move out of the "Reserved" state. If, on the other hand, the CMTS receives an asymmetric request such that traffic may pass on the upstream flow but not on the downstream flow, the CMTS MUST treat the request as a Commit and must transition its state in accordance with the description below.

If the T0 timer expires on the CMTS prior to receiving a Gate-Set command from the CMS, the CMTS MUST initiate a Gate-Close message using the "Timer T0 expiration; no Gate-Set received from CMS" as reason code, and delete the associated gate.

If the T1 timer expires on the CMTS prior to receiving a Commit command from the MTA, the CMTS MUST release any resources reserved with in association with the corresponding GateID, initiate a Gate-Close message using the "Timer T1 expiration; no COMMIT received from MTA" as reason code, and delete the associated gate(s).

If in the "Reserved" state the CMTS receives a Commit command from the client, the CMTS MUST mark the gate in the "Committed" state, stop timer T1, and initiate a Gate-Open message.

If the T7 timer expires and a service flow corresponding to the gate(s) referenced via the associated GateID has not been committed on the CMTS, the CMTS MUST initiate a Gate-Close message using the "Timer T7 expiration; Service Flow reservation timeout" as reason code, and delete the associated gate(s). Otherwise, the CMTS MUST set the reserved envelope equal to the committed envelope for flows corresponding to the gates referenced via the associated GateID.

If the T8 timer expires on the CMTS due to inactivity on the service flow, the CMTS MUST initiate a Gate-Close message using the "Timer T8 expiration; Service Flow inactivity in the upstream direction" as reason code, and delete the associated gate.

Once in the "Committed" state, the gate has reached a stable configuration. Resources have been activated at the local gates. Resources will continue to be activated until either the local client indicates a release command, the DOCSIS Active Timer expires, or the CMS issues a Gate-Delete command.

If, in the "Committed" state, the CMTS receives a Release command from the client, either in the form of a RSVP PATH-TEAR message or via the MAC Control Service interface, or from a failure of the client to refresh a reservation, or from internal DOCSIS mechanisms that detect a client failure, the CMTS MUST deactivate all resources committed for the client, release all resources reserved, initiate a Gate-Close message to the gate-coordination entity, and delete the gate.

If, in the "Committed" state, the CMTS receives a Gate-Delete message, the CMTS MUST deactivate all resources committed for the local client, release all resources reserved, and delete the gate. Additionally, the CMTS must respond with a Gate-Delete-Ack message.

While in the "Committed" state, the CMTS MUST allow the client to initiate changes in the resource reservation or activation, within the limits of the authorization and local admission control.

5.1.5 Gate Coordination

The Gate Coordination messages in the COPS Gate Control interface, Gate-Open and Gate-Close, provide an unsolicited feedback mechanism from the CMTS to the CMS in order to maintain state-synchronization between these elements. This is particularly useful in the case of a pre-mature MTA-initiated reservation or committal request which is not stimulated by the CMS or in the event that an MTA fails, initiating resource recovery at the CMTS. In both of these potential scenarios, the internal state maintained within the CMS will be updated to reflect the state change at the CMTS and the CMS will be able to take appropriate action based on this information.

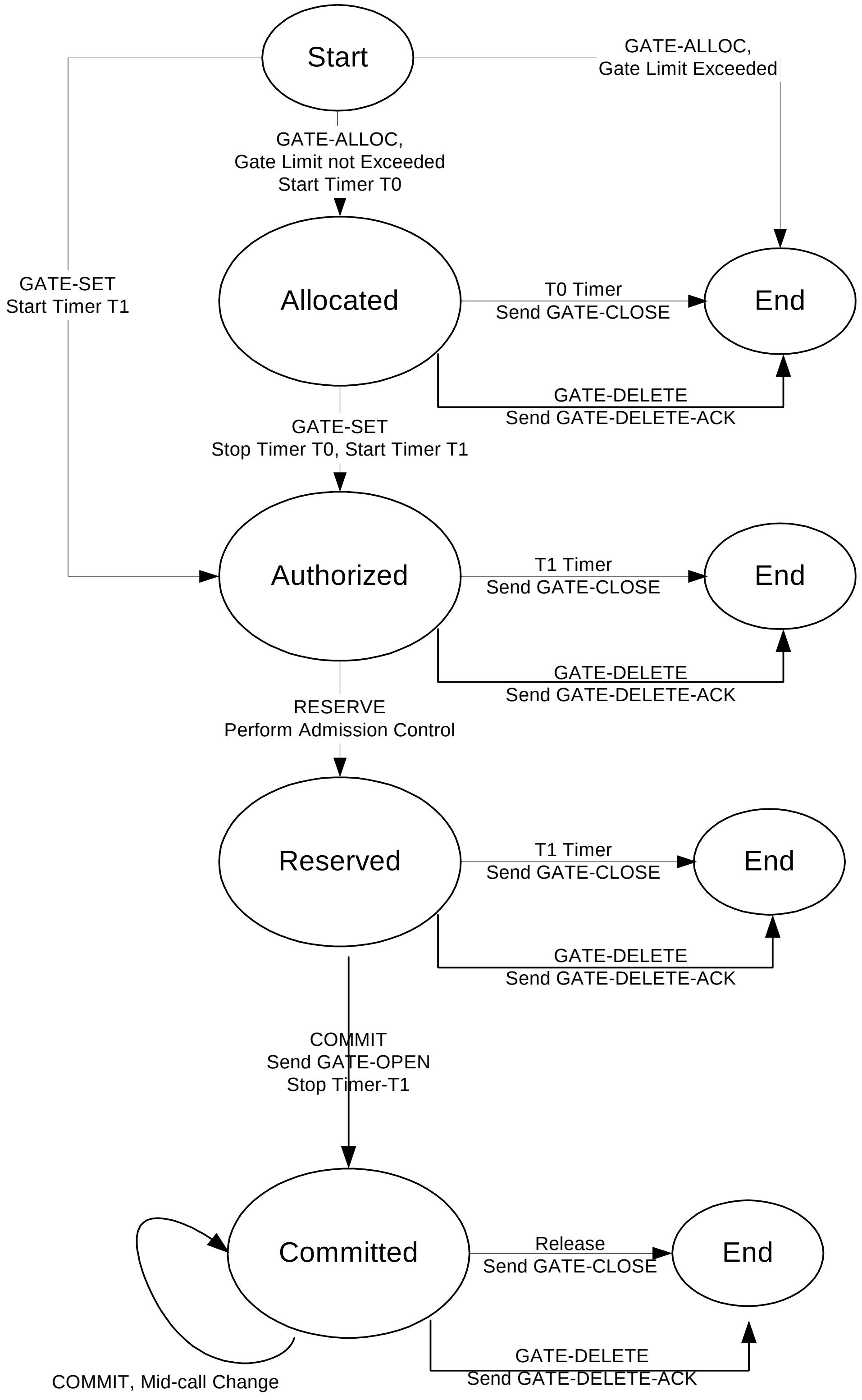

Start
GATE-ALLOC, Gate Limit Exceeded
GATE-ALLOC, Gate Limit not Exceeded Start Timer T0
GATE-SET Start Timer T1
Allocated
T0 Timer Send GATE-CLOSE
End
GATE-DELETE Send GATE-DELETE-ACK
GATE-SET Stop Timer T0, Start Timer T1
Authorized
T1 Timer Send GATE-CLOSE
End
GATE-DELETE Send GATE-DELETE-ACK
RESERVE Perform Admission Control
Reserved
T1 Timer Send GATE-CLOSE
End
GATE-DELETE Send GATE-DELETE-ACK
COMMIT Send GATE-OPEN Stop Timer-T1
Committed
Release Send GATE-CLOSE
End
COMMIT, Mid-call Change
GATE-DELETE Send GATE-DELETE-ACK

Figure 14. Gate State Transition Diagram

5.2 COPS Profile for PacketCable

IP QoS Admission Control is the act of managing QoS resource allocation based on administrative policies and available resource. IP QoS Admission Control Service uses a client/server architecture. The high level operational modules are depicted in Figure 15. The administrative policies are stored as policy database and controlled by the COPS Server. While a typical IntServ implementation of COPS has the server determine available resources, a DiffServ implementation pushes the policy into the client so that the client can make admission control decisions

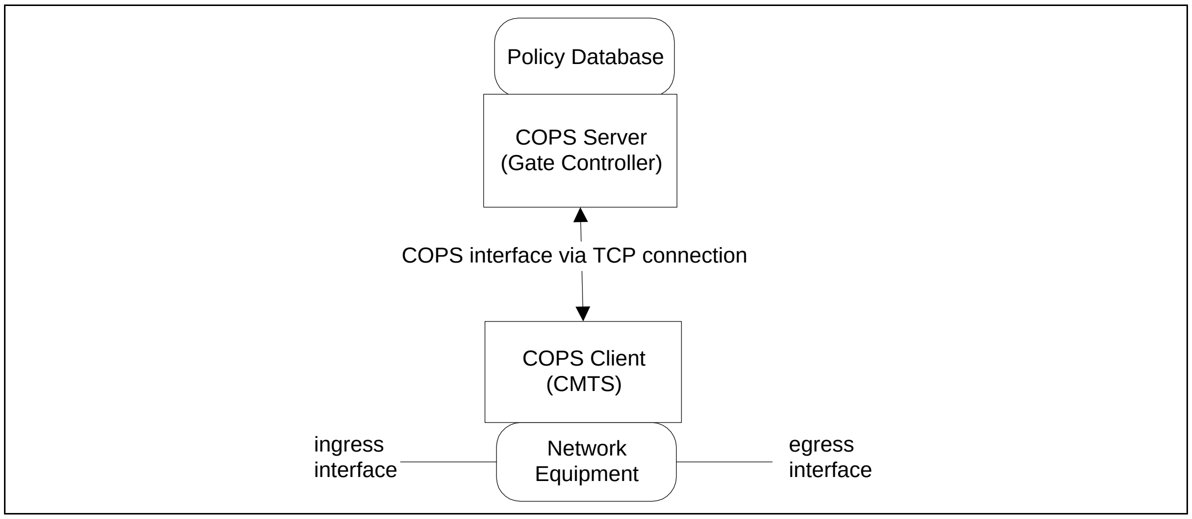


Figure 15. QoS Admission Control Layout

The QoS Admission Control decisions made by the COPS Server MUST be passed to the COPS Client using COPS. The COPS Client MAY make QoS Admission Control requests to the COPS Server based on network events triggered by either the QoS signaling protocol, or via data flow detection mechanisms. The network event can also be the need of QoS bandwidth management, e.g., a new QoS capable interface becomes operational.

QoS policy decisions made by the COPS Server MAY be pushed to the COPS Client based on an external, out of band, QoS service request, e.g., request from the terminating CMTS or a Gate Controller. These policy decisions MAY be stored by the COPS client in a local policy decision point and the CMTS may access that decision information to make admission control decisions on incoming session requests received at the CMTS.

The COPS Client-COPS Server interaction support for QoS Admission Control is provided by IETF's COPS protocol [3]. The COPS protocol includes the following operations:

- Client-Open (OPN)/Client-Accept(CAT)/Client-Close(CC). The COPS Client sends an OPN message to initiated a connection with the COPS Server, and the Server responds with a CAT message to accept the connection. The server sends a CC message to terminate the connection with the Client.
- Request (REQ). The COPS Client sends a REQ message to the server to request admission control decision information or device configuration information. The REQ message may contain client-specific information that the server uses, together with data in the session admission policy database, to make policy-based decisions.
- Decision (DEC). The server responds to REQs by sending a DEC back to the client that initiated the original request. DEC messages may be sent immediately in response to a REQ (i.e., a solicited DEC) or at any time after to change/update a previous decision (i.e., an unsolicited DEC).
- Report State (RPT). The COPS Client sends a RPT message to the COPS Server indicating changes to the request state in the COPS Client. The COPS Client sends this to inform the COPS

Server the actual resource reserved after the COPS Server has granted admission. The COPS Client can also use Report to periodically inform the COPS Server the current state of the COPS Client.

- Delete Request State (DEL). The COPS Client sends a DEL message to the COPS Server for request state cleanup. This can be the result of QoS resource release by the COPS Client.
- Keep Alive (KA). Sent by both the COPS Client and COPS Server for communication failure detection.
- Synchronize State Request (SSR)/Synchronize State Complete (SSC). SSR is sent by the COPS Server requesting current COPS Client state information. The client re-issues request queries to the server to perform the synchronization, and then the client sends a SSC message to indicate synchronization is complete. Because the GC is stateless, the SSR/SSC operations are of no importance in PacketCable and are not used by the CMTS or GC.

Within the PacketCable architecture, the Gate Controller is a COPS Policy Decision Point (PDP) entity and the CMTS is the COPS Policy Enforcement Point (PEP) entity.

The details of the COPS protocol are provided in [3]. This IETF RFC provides a description of the base COPS protocol, independent of client type. Additional drafts provide information for using COPS for Integrated Services with RSVP [22] and for Differentiated Services (i.e., provisioning clients) [7].

5.3 Gate Control Protocol Message Formats

Protocol messages for Gate Control are transported within the COPS protocol messages. COPS uses a TCP connection established between the CMTS and the Gate Controller, and uses the mechanisms specified in [12] to secure the communication path.

5.3.1 COPS Common Message Format

Each COPS message consists of the COPS header followed by a number of typed objects. The GC and CMTS MUST support COPS messaging as defined below.

0		1	2	3
Version	Flags	Op-Code	Client-type	
Message length				

Figure 16. Common COPS Message Header

Version is a 4-bit field giving the current COPS version number. This MUST be set to 1.

Flags is a 4-bit field. 0x1 is the solicited message flag. When a COPS message is sent in response to another message (e.g., a solicited decision sent in response to a request) this flag MUST be set to 1. In other cases (e.g., an unsolicited decision) the flag MUST NOT be set (value = 0). All other flags MUST be set to zero.

Op-code is a 1-byte field that gives the COPS operation to be performed. COPS operations used in this PacketCable specification are:

1 = Request	(REQ)
2 = Decision	(DEC)
3 = Report-State	(RPT)
6 = Client-Open	(OPN)
7 = Client-Accept	(CAT)
9 = Keep-Alive	(KA)

Client type is a 16-bit identifier. For PacketCable use the Client type MUST be set to PacketCable client (0x8008). For Keep-Alive messages (Op-code = 9) the client-type MUST be set to zero, as the KA is used for connection verification rather than per client session verification.

Message length is a 32-bit value giving the size of the message in octets. Messages MUST be aligned on 4-byte boundaries, so the length MUST be a multiple of four.

Following the COPS common header are a variable number of objects. All the objects follow the same object format; each object consists of one or more 32-bit words with a four-octet header, using the following format:

0	1	2	3
Length		C-Num	C-type
(Object contents)			

Figure 17. Common COPS Object Format

The length is a two-octet value that MUST give the number of octets (including the header) that compose the object. If the length in octets is not a multiple of four, padding MUST be added to the end of the object so that it is aligned to the next 32-bit boundary. On the receiving side, a subsequent object boundary MUST be found by rounding up the previous stated object length to the next 32-bit boundary.

C-Num identifies the class of information contained in the object, and the C-Type identifies the subtype or version of the information contained in the object. Standard COPS objects (as defined in [3]) used in this specification, and their values of C-Num, are:

1 = Handle
6 = Decision
8 = Error
9 = Client Specific Info
10 = Keep-Alive-Timer
11 = PEP Identification

5.3.2 Additional COPS Objects for Gate Control

As with the COPS-PR and COPS-RSVP client types, the PacketCable client type defines a number of object formats. These objects MUST be placed inside a Decision object, C-Num = 6, C-Type = 4 (Client specific Decision Data) when carried from GC to CMTS in a decision message. They MUST also be placed in a ClientSI object, C-Num = 9, C-Type = 1 (Signaled Client SI) when carried from CMTS to GC in a Report message. They are encoded similarly to the client-specific objects for COPS-PR; detailed encodings appear below. As in COPS-PR, these objects are numbered using a client-specific number space, which is independent of the top-level COPS object number space. For this reason, the object numbers and types are given as S-Num and S-Type respectively.

Additional COPS objects are defined for use by PacketCable in the following sections.

5.3.2.1 TransactionID

The TransactionID contains a token that is used by the GC to match responses from the CMTS to the previous requests, and the command type that identifies the action to be taken or response.

Length = 8	S-Num = 1	S-Type = 1
Transaction Identifier	Gate Command Type	

Transaction Identifier is a 16-bit quantity that MAY be used by the GC to match responses to commands.

Gate Command Type MUST be one of the following:

GATE-ALLOC	1
GATE-ALLOC-ACK	2
GATE-ALLOC-ERR	3
GATE-SET	4

GATE-SET-ACK	5
GATE-SET-ERR	6
GATE-INFO	7
GATE-INFO-ACK	8
GATE-INFO-ERR	9
GATE-DELETE	10
GATE-DELETE-ACK	11
GATE-DELETE-ERR	12
GATE-OPEN	13
GATE-CLOSE	14

5.3.2.2 Subscriber-ID

The Subscriber-ID identifies the subscriber for this service request. Its main use is to prevent various denial-of-service attacks.

Length = 8	S-Num = 2	S-Type = 1
IP v4 address (32-bits)		

or

Length = 20	S-Num = 2	S-Type = 2
IP v6 address (128-bits)		

5.3.2.3 GateID

This object identifies the gate or set of gates referenced in the command message, or assigned by the CMTS for a response message.

Length = 8	S-Num = 3	S-Type = 1
GateID (32-bits)		

5.3.2.4 Activity-Count

When used in a GATE-ALLOC message, this object specifies the maximum number of gates that can be simultaneously allocated to the indicated subscriber-ID. This object returns, in a GATE-SET-ACK or GATE-ALLOC-ACK message, the number of gates assigned to a single subscriber. It is useful in preventing denial-of-service attacks.

Length = 8	S-Num = 4	S-Type = 1
Count (32-bits)		

5.3.2.5 Gate-Spec

Length = 60		S-Num = 5	S-Type = 1
Direction	ProtocolID	Flags	Session Class
Source IP Address (32-bits)			
Destination IP Address (32-bits)			
Source Port (16-bits)		Destination Port (16-bits)	
DiffServ Code Point			
Timer T1 value		Reserved	
Timer-T7 value		Timer-T8 value	
Token Bucket Rate [r] (32-bit IEEE floating point number)			
Token Bucket Size [b] (32-bit IEEE floating point number)			
Peak Data Rate (p) (32-bit IEEE floating point number)			
Minimum Policed Unit [m] (32-bit integer)			
Maximum Packet Size [M] (32-bit integer)			
Rate [R] (32-bit IEEE floating point number)			
Slack Term [S] (32-bit integer)			

Direction is either 0 for a downstream gate, or 1 for an upstream gate.

ProtocolID is the value to match in the IP header, or zero for no match.

Flags are defined as follows:

0x01	Auto-Commit and Commit-Not-Allowed functionality which was formerly signaled through the flags field has been deprecated. As a result, bits one and two are reserved.

All bits MUST be zero.

Session class identifies the proper admission control policy or parameters to be applied for this gate. Permissible values are:

0x00	Unspecified
0x01	Normal priority VoIP session
0x02	High priority VoIP session (e.g., E911)

All other values are currently reserved.

Source IP address and destination IP address represent a pair of 32-bit IPV4 addresses, or zero for no match (i.e., a wildcard specification that will match any request from the MTA).

Source port and destination port define a pair of 16-bit values, or zero for no match

The values r, b, p, m, M, and R are as described in Section 3.2. Instead of the RSVP RFC defined slack-term the value S would represent in microseconds the minimum allowed grant jitter in the upstream direction that can be admitted, and the minimum allowed delay in the downstream direction that can be admitted.

Other sections of this specification provide normative requirements which represent constraints on the authorization envelope which is defined by these parameters. Specifically, the multiple codec discussion in Section 2.6.10 defines an upper bound on the authorization envelope, while Section 5.5 later in this chapter provides a set of minimal requirements for these parameters. It is strongly recommended that CMS implementations constrain authorization parameters as much as possible as these constructs are fundamental in defining and enforcing service provider bandwidth management policies.

The DS field is defined by the following structure:

0	1	2	3	4	5	6	7
Differentiated Services Code Point (DSCP)						Not Used	Not Used

For backward compatibility with current system implementations and use of the IP Precedence as defined in [31] and [32], the appropriate bits of the IPv4 TOS byte shown below MAY be inserted in the DS field. The IP TOS field (bits 3-6) is not supported in DiffServ networks.

0	1	2	3	4	5	6	7
IP Precedence			IPv4 IP TOS				Not Used

Timer T1 is given in seconds, and used in the Gate Transition Diagram described in Section 5.1.4. If multiple Gate-Spec objects appear in a single COPS message, the value of T1 MUST be identical in all Gate-Spec occurrences. If the T1 values differ in the upstream and downstream GateSpec objects, then the CMTS MUST use the T1 value specified in the upstream GateSpec to manage the pair of Gates.

Timers T7 and T8 are values in seconds and used to control the DOCSIS Timeout for Admitted QoS Parameters and Timeout for Active QoS Parameters respectively.

5.3.2.6 Remote-Gate-Info

Length=36		S-Num = 6	S-Type = 1
CMTS IP Address (32-bits)			
CMTS Port (16-bits)		Flags, defined below	
Remote GateID			
Algorithm	Reserved		
Security Key (16 bytes)			

This object has been deprecated by dqos-n-02148. S-Num 6 is reserved to prevent misinterpretation.

5.3.2.7 Event-Generation-Info

The object contains all the information necessary to support the event messages as specified and required in [20].

Length = 44	S-Num = 7	S-Type = 1
Primary-Record-Keeping-Server-IP-Address (32-bits)		
Primary-Record-Keeping-Server-Port	Flags, see below	Reserved
Secondary-Record-Keeping-Server-IP-Address (32-bits)		
Secondary-Record-Keeping-Server-Port	Reserved	
Billing-Correlation-ID (24 bytes)		

Primary-Record-Keeping-Server-IP-Address is the address of the record keeper to whom event records are sent.

Primary-Record-Keeping-Server-Port is the port number for event records sent.

Flag values are as follows:

0x01 Batch processing indicator. If set, the CMTS MUST accumulate event records as part of a batch file and send to Record Keeping Server at periodic intervals. If clear, the CMTS MUST send the event records to the Record Keeping Server in real-time.

The rest are reserved and MUST be zero.

Secondary-Record-Keeping-Server-IP-Address is the address of the secondary record keeper to whom records are sent if the primary record keeping server is unavailable.

Secondary-Record-Keeping-Server-Port is the port number for event records sent.

Billing-Correlation-ID is the identifier assigned by the CMS for all records related to this session. See [20] for format.

5.3.2.8 Media-Connection-Event-Info

This object has been deprecated by dqos-n-02185 in coordination with em-n-02182. S-Num 8 is reserved to prevent misinterpretation.

5.3.2.9 PacketCable-Reason

This object contains the reason why the gate is being deleted.Length = 8	S-Num = 13	S-Type = 1
Reason-code	Reason Sub-code	

The Reason-code values defined in this specification are:

0: Gate-Delete Operation
1: Gate-Close Operation

The Reason Sub-codes are defined as:

Gate-Delete Operation:

0 = Normal operation
1 = Local gate-coordination not completed
2 = Remote gate-coordination not completed
3 = Authorization revoked
4 = Unexpected Gate-Open
5 = Local Gate-Close failure
127 = Other, unspecified error

Gate-Close Operation:

0 = Client initiated release (normal operation)
1 = Reservation reassignment (e.g., for priority session)
2 = Lack of reservation maintenance (e.g., RSVP refreshes)
3 = Lack of DOCSIS MAC-layer responses (e.g., station maintenance)
4 = Timer T0 expiration; no Gate-Set received from CMS
5 = Timer T1 expiration; no COMMIT received from MTA
6 = Timer T7 expiration; Service Flow reservation timeout
7 = Timer T8 expiration; Service Flow inactivity in the upstream direction
127 = Other, unspecified error

5.3.2.10 *PacketCable-Error*

A client-specific error object is defined as follows.

Length = 8	S-Num = 9	S-Type = 1
Error-code	Error Sub-code	

The Error-code values defined in this specification are:

1 = No gates currently available
2 = Unknown GateID
3 = Illegal Session Class value
4 = Subscriber exceeded gate limit
5 = Gate already set
6 = Missing Required Object
7 = Invalid Object
127 = Other, unspecified error

The error sub code field is used to provide further information about the error. In the case of error codes 6 through 7, this 16-bit field contains as two 8-bit values the S-Num and S-Type of the object that is missing or in error. The order of the S-Num and S-Type values within the Error Sub-code MUST be the same as in the original message. In cases where multiple valid alternatives exist for the S-Type of a missing object, this portion of the Error Sub-code should be set to 0.

5.3.2.11 Electronic-Surveillance-Parameters

Length = 24	S-Num = 10	S-Type = 1
DF-IP-Address-for-CDC (32-bits)		
DF-Port-for-CDC (16-bits)	Flags, defined below	
DF-IP-Address-for-CCC (32-bits)		
DF-Port-for-CCC (16-bits)	Reserved	
CCCID (32-bits)		

DF-IP-Address-for-CDC is the address of the Electronic Surveillance Delivery Function to whom the duplicated event messages are to be sent.

DF-Port-for-CDC is the port number for the duplicated event messages.

Flags are defined as follows:

0x0001	DUP-EVENT. If set, CMTS MUST send a duplicate copy of all event messages related to this gate to the DF-IP-Address-for-CDC.
0x0002	DUP-CONTENT. If set, CMTS MUST send a duplicate copy of all packets matching the classifier(s) for this gate to the DF-IP-Address-for-CCC

The rest are reserved and MUST be zero.

DF-IP-Address-for-CCC is the address of the Electronic Surveillance Delivery Function to whom the duplicated call content packets are to be sent.

DF-Port-for-CCC is the port number for the duplicated call content.

CCCID is the identifier for duplicated call content packets.

5.3.2.12 Session-Description-Parameters

Length	S-Num = 11	S-Type = 1

This object has been deprecated by dqos-n-03026. S-Num 11 is reserved to prevent misinterpretation.

5.3.3 Definition of Gate Control Messages

Messages that perform gate control between the GC and CMTS MUST be defined and formatted as follows. Note that messages from GC to CMTS are COPS Decision messages, and messages from CMTS to GC are COPS Report messages

```
<Gate-Control-Cmd> :: = <COPS-Common-Header> <Handle>
                        <Context> <Decision-Flags>
                        <ClientSI-Data>

<ClientSI-Data> :: = <Gate-Alloc> | <Gate-Set> | <Gate-Info> |
                     <Gate-Delete>

<Gate-Control-Response> :: = <COPS-Common-Header> <Handle>
                             <Report-Type> <ClientSI-Object>
```

```
<ClientSI-Object> :: = <Gate-Alloc-Ack> | <Gate-Alloc-Err> |
                       <Gate-Set-Ack> | <Gate-Set-Err> |
                       <Gate-Info-Ack> | <Gate-Info-Err> |
                       <Gate-Delete-Ack> | <Gate-Delete-Err>

<Gate-Alloc> :: = <Decision-Header> <TransactionID> <Subscriber-ID>
                       [<Activity-Count>]


<Gate-Alloc-Ack> :: = <ClientSI-Header> <Transaction-ID> <Subscriber-ID> <Gate-ID> <Activity-
Count>

<Gate-Alloc-Err> :: = <ClientSI-Header> <Transaction-ID> <Subscriber-ID> <PacketCable-Error>

<Gate-Set> :: = <Decision-Header> <Transaction-ID> <Subscriber-ID>

               [<Activity-Count>] [<Gate-ID> ] [<Event-Generation-Info>]

               [<Electronic-Surveillance-Parameters>]

                <Gate-Spec> [<Gate-Spec>]

<Gate-Set-Ack> :: = <ClientSI-Header> <Transaction-ID> <Subscriber-ID> <Gate-ID> <Activity-
Count>

<Gate-Set-Err> :: = <ClientSI-Header> <TransactionID> <Subscriber-ID><PacketCable-Error>

<Gate-Info> :: = <Decision-Header> <TransactionID> <GateID>

<Gate-Info-Ack> :: = <ClientSI-Header> <TransactionID> <Subscriber-ID> <GateID>

                    [<Event-Generation-Info>] [<Electronic-Surveillance-Parameters>]

                    [<Gate-Spec> ] [<Gate-Spec>]

<Gate-Info-Err> :: = <ClientSI-Header> <TransactionID> <GateID><PacketCable-Err>

<Gate-Delete> :: = <Decision-Header> <TransactionID> <GateID> <PacketCable-Reason>

<Gate-Delete-Ack> :: = <ClientSI-Header> <TransactionID> <GateID>

<Gate-Delete-Err> :: = <ClientSI-Header> <TransactionID> <GateID> <PacketCable-Err>

<Gate-Open> :: = <ClientSI-Header> <TransactionID> <GateID>
<Gate-Close> :: = <ClientSI-Header> <TransactionID> <GateID> <PacketCable-Reason>
```

The Context object (C-NUM = 2, C-TYPE = 1) in the COPS Decision message has the R-Type (Request Type Flag) value set to 0x08 (Configuration Request) and the M-Type set to zero. The Command-Code field in the mandatory Decision-Flags object (C-NUM = 6, C-TYPE = 1) is set to 1 (Install Configuration). Other values should cause the CMTS to generate a Report message indicating failure. The Report-Type object (C-NUM = 12, C-TYPE = 1) included in the COPS Report message has the Report-Type field set to 1 (Success) or 2 (Failure) depending on the outcome of the gate control command. All Report messages carrying the gate control response should have the solicited message flag bit set in the COPS header. All decision (DEC) messages, except the first one, should have the solicited message flag set to false in the COPS header. The first decision message sent from the CMS to CMTS should have the solicited flag set to true. The values of this flag are set to comply with the COPS specification. They should not affect the operation of the gate control protocol.

If an object, that is received in a gate control message, contains an S-Num or S-Type that is not recognized that object MUST be ignored. The presence of such an object within a gate control message MUST not be treated as an error provided that after such parameter is dropped, all required objects are present in the message.

5.4 Gate Control Protocol Operation

5.4.1 Initialization Sequence

When the CMTS (i.e., COPS PEP) boots, it MUST listen for incoming COPS connections on the IANA-assigned TCP port number 2126. Any Gate Controller with a need to contact the CMTS MUST establish a TCP connection to the CMTS on that port. It is expected that multiple Gate Controllers will establish COPS connections with a single CMTS. When the TCP connection between the CMTS and GC is established, the CMTS sends information about itself to the GC in the form of a CLIENT-OPEN message. This information includes the provisioned CMTS-ID in the PEP Identification (PEPID) object. The CMTS SHOULD omit the Last PDP Address (LastPDPAddr) object from the CLIENT-OPEN message.

In response, the Gate Controller sends a CLIENT-ACCEPT message. This message includes the Keep-Alive-Timer object, which tells the CMTS the maximum interval between Keep-Alive messages.

The CMTS then sends a REQUEST message, including the Handle and Context objects. The Context object (C-NUM = 2, C-TYPE = 1) MAY have the R-Type (Request Type Flag) value set to 0x08 (Configuration Request) and M-Type set to zero. The Handle object contains a number that is chosen by the CMTS. The only requirement imposed on this number is that a CMTS MUST NOT use the same number for two different requests on a single COPS connection; in the PacketCable environment the handle has no other protocol significance. This completes the initialization sequence, which is shown in Figure 18.

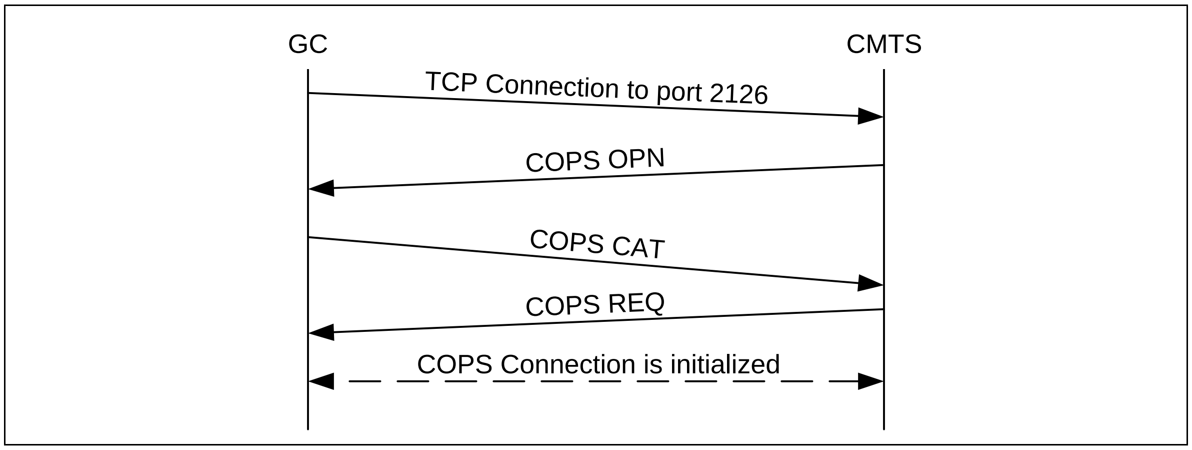


Figure 18. COPS Connection Establishment

Periodically the CMTS MUST send a COPS KEEP-ALIVE (KA) message to the GC. Upon receipt of the COPS KA message, the CMS MUST echo a COPS KA message back to the CMTS. This transaction is shown in the figure below, and is fully documented in [3]. This MUST be done at least as often as specified in the Keep-Alive-Timer object returned in the CLIENT-ACCEPT message. The KEEP-ALIVE message is sent with Client-Type set to zero.

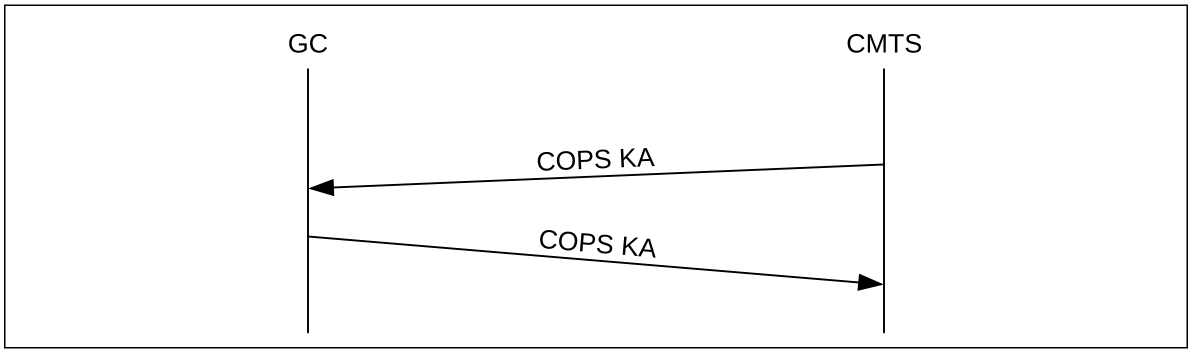


Figure 19. COPS Keep-Alive Exchange

5.4.2 Operation Sequence

The protocol between the Gate Controller and CMTS is for purposes of resource control and resource allocation policy. The Gate Controller implements all the allocation policies, and uses that information to manage the set of gates implemented in the CMTS. The Gate Controller initializes the gates with specific source, destination, and bandwidth restrictions; and once initialized, the MTA is able to request resource allocations within the limits imposed by the Gate Controller.

Messages initiated by the Gate Controller include Gate-Alloc, Gate-Set, Gate-Info, and Gate-Delete, and messages initiated by the CMTS include Gate-Open and Gate-Close. The procedures for these messages are described in the following sections.

The GC-initiated messages are sent using client specific objects within the decision object of COPS DECISION messages. The responses to the GC initiated messages are sent as a REPORT-STATE message with client specific objects in the ClientSI object by the CMTS. For the ACK messages the COPS Report-Type value MUST be 1 and for the ERR messages the Report-Type MUST be 2 . Gate-Open and Gate-Close messages MUST be sent as a unsolicited REPORT-STATE message with transaction ID of zero, with client specific objects in the ClientSI object, using the Report-Type 3, to the CMS through the TCP connection that originally constructed the gate . If that TCP connection is no longer valid, then the CMTS MUST silently drop the GC messages.

The DECISION messages and REPORT-STATE messages MUST contain the same handle as was used in the initial REQUEST sent by the CMTS when the COPS connection was initiated.

GATE-ALLOC validates the number of simultaneous sessions allowed to be setup from the originating MTA, and allocates a GateID to be used for all future messages regarding this gate or set of gates.

GATE-SET initializes and modifies all the policy and traffic parameters for the gate or set of gates, and sets the billing and gate coordination information.

GATE-INFO is a mechanism by which the Gate Controller can find out all the current state and parameter settings of an existing gate or set of gates.

The CMTS MUST periodically send a Keep Alive (KA) message to the GC to facilitate the detection of TCP connection failures. The Gate Controller keeps track of when KAs are received. If the Gate Controller has not received a KA from the CMTS in the time specified by [3] or the Gate Controller has not received an error indication from the TCP connection, then the Gate Controller MUST tear down the TCP connection and attempt to re-establish the TCP connection before the next time it is requested to allocate a gate from that CMTS.

Gate-Delete allows a Gate Controller to delete a recently allocated gate under certain (see below) circumstances.

Gate-Open allows the CMTS to inform the Gate-Controller that the gate resources have been committed. The Gate-Open message, along with the Gate-Close message described below, provide a feedback path from CMTS to CMS in order to allow for accurate call-state management at the CMS element.

Gate-Close allows CMTS to inform the GC that the gate has been deleted due to MTA interaction or inactivity."

5.4.3 Procedures for Allocating a new Gate

A GATE-ALLOC message is sent by the Gate Controller to the CMTS at the time the 'Call_Setup' message is sent from the originating MTA (e.g., INVITE message when using DCS), as shown in Figure 19.

The use of GATE-ALLOC ensures that not too many sessions are being simultaneously requested from a given MTA. This mechanism may be used to control a denial of service attack from the MTA. The CMTS, in its response to the GATE-ALLOC message, compares the number of currently allocated gates for the indicated subscriber-ID against the Count field of the Activity-Count object in the GATE-ALLOC message. If the current number of gates is greater than or equal to the Count field in the GATE-ALLOC, then the CMTS MUST return a GATE-ALLOC-ERR message. If the current number of gates is greater than the Count field in the GATE-ALLOC, then it is likely that the subscriber has been re-provisioned to have a lower gate limit than before. In this case, the subscriber's current sessions are not affected but any new sessions by that subscriber will be rejected by the CMTS until the subscriber's session count goes below the value specified in the Count field.

The determination of the actual value to be contained in the Count field is an operational issue. It should be set sufficiently high (per MTA) that no possible legitimate calling scenario is adversely affected, while being sufficiently low as to prevent a viable denial-of-service attack to be mounted.

If the Activity-Count object is not present, the CMTS does not perform the gate limit check. A GC seeking to reduce call setup time MAY decide to perform the gate limit check upon receipt of the GATE-ALLOC-ACK instead of having the CMTS perform the check so that the GC can do the GATE-ALLOC and subscriber policy lookup operations in parallel. When the results of both operations are available, the GC can do the gate limit check. If the check fails, the GC SHOULD send a GATE-DELETE message to the CMTS to delete the gate that was incorrectly allocated (see section 5.4.8) The GC MAY include the Activity-Count object in subsequent GATE-ALLOCs for that subscriber once the policy has been cached.

The following figure is an example of the GATE-ALLOC signaling.

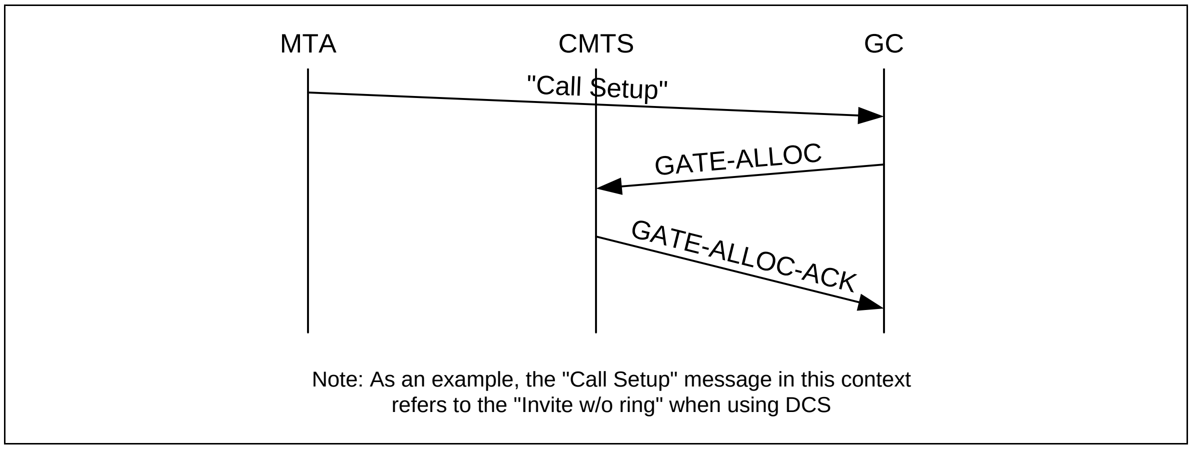

MTA
CMTS
GC
"Call Setup"
GATE-ALLOC
GATE-ALLOC-ACK
Note: As an example, the "Call Setup" message in this context refers to the "Invite w/o ring" when using DCS

Figure 20. Sample Signaling of Gate-Alloc

The CMTS MUST respond to a GATE-ALLOC message with either a GATE-ALLOC-ACK (indicating success) or a GATE-ALLOC-ERR (indicating failure). The TransactionID in the response MUST match the TransactionID in the request.

Errors in allocating gates are reported by a GATE-ALLOC-ERR response. The PacketCable Error object contains one of the following Error-Codes:

1 = No gates currently available
4 = Subscriber exceeded gate limit
6 = Missing Required Object
7 = Invalid Object
127 = Other, unspecified error

5.4.4 Procedures for Authorizing Resources Through a Gate

The GATE-SET message is sent by the Gate Controller to the CMTS to initialize or modify the operational parameters of the gate(s). Figure 21 is an example of the GATE-SET signaling.

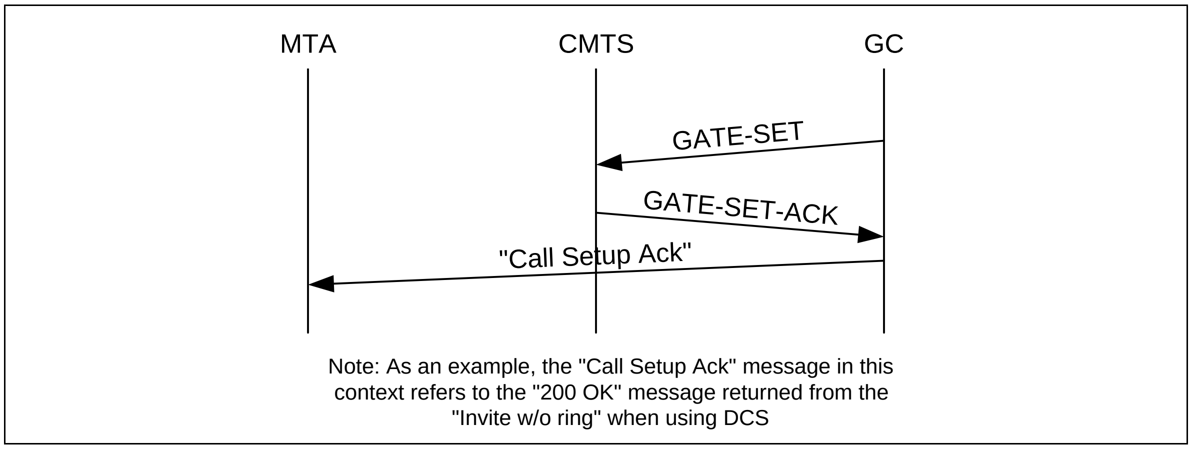

MTA
CMTS
GC
GATE-SET
GATE-SET-ACK
"Call Setup Ack"
Note: As an example, the "Call Setup Ack" message in this context refers to the "200 OK" message returned from the "Invite w/o ring" when using DCS

Figure 21. Sample Signaling of Gate-Set

If a GateID Object is present in the GATE-SET message, then the request is to modify an existing gate. If the GateID Object is missing from the GATE-SET message, then it is a request to allocate a new gate, and the Activity-Count Object MAY be present so that the CMTS can determine if the subscriber has exceeded the maximum number of simultaneous gates (see section 5.4.3).

The GATE-SET message MUST contain exactly one or two Gate-Spec objects, describing zero or one upstream gates, and zero or one downstream gates.

The CMTS MUST respond to a GATE-SET message with either a GATE-SET-ACK (indicating success) or a GATE-SET-ERR (indicating failure). The TransactionID in the response MUST match the TransactionID in the request.

Errors in allocating or authorizing gates are reported by a GATE-SET-ERR response. The PacketCable-Error object contains one of the following Error-Codes:

1 = No gates currently available
2 = Unknown GateID
3 = Illegal Session Class value
4 = Subscriber exceeded gate limit
5 = Gate already set
6 = Missing Required Object
7 = Invalid Object
127 = Other, unspecified error

In handling a reservation request from an MTA, the CMTS MUST determine the proper gate by use of the RSVP GateID object, or by the use of the Authorization Block TLV. The CMTS MUST verify the reservation request is within the authorized limits specified for the gate.

The CMTS then updates the reservation request based on gate parameters. If the QoS-Parameter-Set is Admitted (2), then the CMTS MUST set the Timeout-For-Admitted-QoS-Parameters to Timer T7. The CMTS MUST use the DiffServ Code Point or TOS value from the Gate-Spec object to overwrite the IP Type of Service octet before forwarding packets.

The CMTS MUST perform an admission control function, based on provisioned policy parameters and the Session Class value of the gate.

Note that a GATE-SET message can be used to allocate (and set) a gate instead of the GATE-ALLOC message. In such situations, it is possible that the port number being used by the remote gate for receiving gate coordination messages is not available to the gate controller. If that is the case, the CMTS-port in the Remote-Gate-Info object (carried in the GATE-SET message) is set to zero. This causes the CMTS to ignore the gate coordination port number. However, when the gate controller (later) learns about the port number being used by the remote gate, it must send another GATE-SET message (with the port number in the Remote-Gate-Info object) to inform the CMTS about this port.

The intention of a Gate-Set is that the most recent parameter values would be used for admission control when moving a gate from state Authorized to state Reserved. Once resources have been reserved, the MTA is guaranteed that any commit operation within the reserved envelope would succeed. After this time (i.e., the gate state is Reserved or Committed), the gate MUST remain static. If, due to outside events (codec change, RTP port or IP address change, etc.), the gate parameters become insufficient to carry a forthcoming media stream, the Gate Controller MUST attempt to create a new gate to handle the modified media stream.

5.4.5 Procedures for Querying a Gate

When a Gate Controller wishes to find out the current parameter settings of a gate, it sends to the CMTS a GATE-INFO message. The CMTS MUST respond to a GATE-INFO message with either a GATE-INFO-ACK (indicating success) or a GATE-INFO-ERR (indicating failure). The TransactionID in the response MUST match the TransactionID in the request. GateSpec object(s) MUST be included in the Gate-Info-Ack if they have previously been provided to the CMTS in association with a Gate.

Errors in querying gates are reported by a GATE-INFO-ERR response. The Error object contains one of the following Error-Codes:

2 = Unknown GateID
127 = Other, unspecified error

5.4.6 Procedures for Committing a Gate

When the MTA performs a Commit operation (as described in Section 3.7 for an RSVP-based MTA, or Section 4.2.1 for an embedded MTA), the CMTS MUST send a Gate-Open message .

5.4.7 Procedures for Closing a Gate

The CMTS MUST release all resources associated with the gate, delete the gate, delete associated Service Flow(s) using a DOCSIS DSD message, and send a Gate-Close message when it receives an explicit release message from the MTA client (as described in Section 3.5.3 for an RSVP-based MTA, or in Section 4.3.3 for embedded MTAs), or when it detects that the client is no longer actively generating packets and not generating proper refreshes for the flow associated with a gate.

5.4.8 Procedures for Deleting a Gate

In a normal call flow, a gate is deleted by the CMTS when it receives an RSVP PATH-TEAR message or a DSD-REQ message (from an embedded MTA that does not support RSVP). The CMTS also deletes a gate at the receipt of a GATE-CLOSE message from a remote CMTS (DCS model) or a CMS (NCS model).

A gate controller, typically, does not initiate a gate delete operation. However, there could be certain abnormal situations when a gate controller might want to delete a gate on the CMTS. For instance, if the gate controller learns (at the receipt of a GATE-ALLOC-ACK response) that a subscriber has exceeded its gate limit, it might want to delete the recently allocated gate at the CMTS. In such scenarios, it SHOULD send a GATE-DELETE message to the CMTS (instead of allowing the gate to timeout). There could be other situations in which the delete functionality might be useful.

The CMTS MUST respond to a GATE-DELETE message with a GATE-DELETE-ACK (indicating success) or a GATE-DELETE-ERR (indicating failure). The TransactionID in the response MUST match the TransactionID in the request. Errors in deleting gates are reported by a GATE-DELETE-ERR response. The Error object contains one of the following Error-Codes:

2 = Unknown GateID
127 = Other, unspecified error

5.4.9 Termination Sequence

When the CMTS is shutting down its TCP connection to the GC, it MAY first send a DELETE-REQUEST-STATE message (including the handle object used in the REQUEST message). The CMTS MAY follow that with a CLIENT-CLOSE message. These messages are optional because the GC is stateless and because the COPS protocol requires a COPS server to automatically delete any state associated with the CMTS when the TCP connection is terminated.

When the Gate Controller is going to shutdown, it SHOULD send a COPS Client-Close (CC) message to the CMTS. In the COPS CC message, the Gate Controller SHOULD NOT send the PDP redirect address object <PDPRedirAddr>. If the CMTS receives a COPS CC message from the Gate Controller with a <PDPRedirAddr> object, the CMTS MUST ignore the <PDPRedirAddr> while processing the COPS CC message.

5.4.10 Failure Scenario

When a CMTS detects the loss of the TCP connection to the GC, e.g. if the GC suffers a catastrophic failure, the CMTS MUST keep all established gates in place. Gates that have been committed will remain committed and gates in any other states will remain in that state until their state is actively changed or the appropriate timers expire. Maintaining gates across GC/CMS failures would allow for any critical flows (e.g. a 911 call) to remain in place.

5.5 CMS Use of Gate Protocol

The CMS MUST ensure that all codecs agreed during negotiation fit within the resource envelope requested from the CMTS using gate communication. CMS MUST use LUB algorithm provided in Section 3.2.1 to determine b, r, p, m, and M values.

The CMS SHOULD make sure that the Gate Control message communicated to the CMTS contains proper end-point IP addresses and ports such that the call end-points are referenced and possible theft of service is prevented.

The CMS MUST set the Slack Term to a value that is 800 us for upstream direction if it is not sending a upstream grant jitter parameter to the MTA . Otherwise, the value that is used in the gate should be less than or equal to the value that is send to the MTA for use as the DOCSIS Tolerated Grant Jitter parameter. For the downstream direction the CMS MUST set the value to zero.

5.6 Gate-Coordination

The Gate-Controller keeps the state of each gate. The Gate-Controller creates a gate on the CMTS using the Gate-Alloc or Gate-Set message. The Gate-Controller may delete a gate via the Gate-Delete command or may query the CMTS for information associated with a particular gate using the Gate-Info message. The CMTS informs the GC of state changes that occur due to MTA messages or inactivity using the Gate-Open and Gate-Close messages.

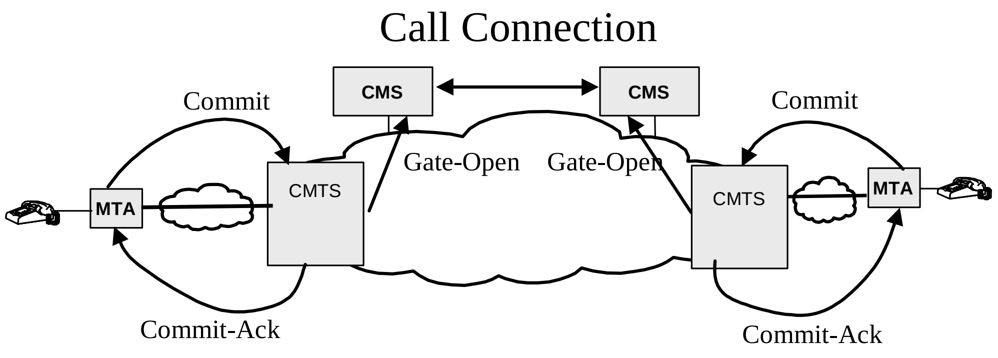
The Gate-Open message is generated by the CMTS when the MTA commits QoS resources, thereby starting the call. The Gate-Close message signals the closure of the Gate at the CMTS and the release of associated QoS resources. Both Gate-Open and Gate-Close are informative messages regarding state changes at the CMTS related to a specific Gate, and do not require feedback from the CMS.

Appendix A Gate-Open and Gate-Close events at the local and remote end-points must be synchronized to prevent possible theft of service scenarios. This synchronization is accomplished using either CMS internal logic or, in the case of multiple CMS', using CMS-to-CMS signaling.

5.6.1 Connecting a Call

The successful connection of a normal call requires three events happening in close succession:

- The CMS requests commitment of resources at the local MTA
- The CMTS indicates that resources have been committed by the local MTA
- Coordination of local and remote resource commitment is coordinated on the signaling plane

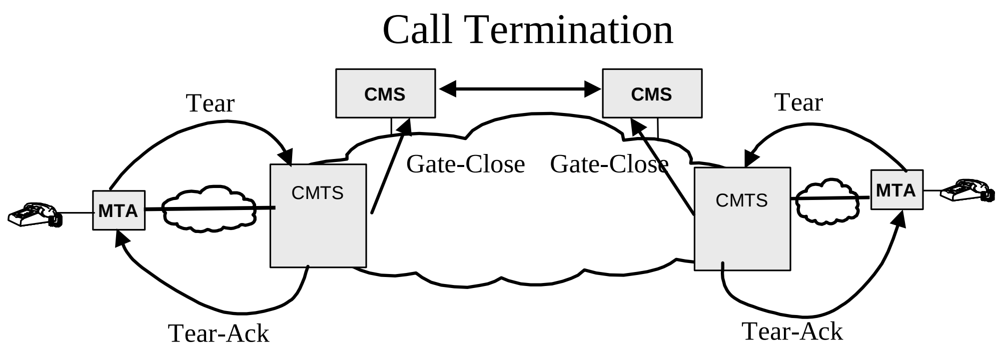


If a CMS receives a Gate-Open message for a gate that it has not communicated the resources are to be committed then the CMS MUST delete the gate with ‘Unexpected Gate-Open’ described in the reason-code.

5.6.2 Terminating a Call

The termination of a call, as in the case of the connection, requires three events within a short time frame:

- The CMS requests release of resources at the local MTA
- The CMTS indicates that resources have been released by the local MTA
- Coordination of local and remote resource release is coordinated on the signaling plane



When the CMS sends the MTA a message to delete the connection, the CMS MUST start a timer for T5 period of time. If by the expiration of the timer the CMTS has not indicated closure of the gate, then the CMS MUST issue a Gate-Delete command to delete the gate at the CMTS with 'Local gate-close failure' described in the reason-code .

When the CMS receives a Gate-Close message it must update its internal state reflecting the gate removal on the CMTS.

6 TIMER DEFINITIONS AND VALUES

Several timers are referenced in this specification. This section contains the list of those timers, and their recommended values.

6.1 Timer T0

This timer is implemented in the CMTS in the Gate state machine, and limits the period of time that a gate may be allocated without the gate parameters being set. This enables the CMTS to recover the GateID resources when the Call Management System fails to complete the signaling sequence for a new session.

This timer is started when a gate is allocated.

This timer is reset when the gate parameters are set.

On expiration of this timer, the CMTS MUST consider the assigned GateID to be invalid.

The RECOMMENDED value of this timer is 30 seconds.

6.2 Timer T1

This timer is implemented in the CMTS in the Gate state machine, and limits the period of time that may elapse between the authorization and a commit is performed.

This timer is started whenever a Gate is established.

This timer is reset when the Gate goes to a COMMITTED state.

On expiration of this timer, the CMTS MUST release all resources reserved in the CMTS for this gate, revoke any reservations made by the MTA that were authorized by this gate by signaling the CM via a DSC or DSD to release resources it had reserved, and initiate a GATE-CLOSE message for the gate.

Timer T1 MUST be set to the value given in the GATE-SET message. If the value given in the GATE-SET message is zero, then Timer T1 MUST be set to a provisionable default value. The RECOMMENDED value of this default is in the range 200-300 seconds.

If the T1 timer value in the Gate-Set is 0, the CMTS MUST return either the provisioned CMTS T1 value or zero for T1 in the GateSpec object of the Gate-Info-Ack message. The provisioned value for T1 is preferred in this case.

6.3 Timer T2

This timer is no longer used.

6.4 Timer T3

This timer is implemented in the MTA or CMTS in the handling of RSVP reservations. It controls the total time that can elapse before the RSVP retransmit process gives up without receiving an acknowledgement in the presence of network loss. It is short enough to recover quickly from lost messages and not significantly impact the post-dial delay, but is long enough to allow the CMTS to acknowledge the request and all intermediate routers in the customer network.

This timer is started when the MTA or CMTS sends an RSVP message that requires an acknowledgement (such as RSVP PATH). This timer is reset when the sender of the message to be acknowledged receives a response to that message. In the case of an RSVP PATH message, such a response MAY be RSVP RESV, RSVP PATH-ERROR, or RSVP-MESSAGE-ACK, or RSVP-MESSAGE-NACK.

This timer is reset when the CMTS has received a COMMIT message from the MTA, a GATE-OPEN and GATE-OPEN-ACK message for the gate.

The RECOMMENDED value of this timer is 4 seconds (4000ms).

6.5 Timer T4

This timer is implemented in the MTA in the handling of COMMIT messages. It controls the retransmission of COMMIT messages that may have been lost by the network. It is short enough to recover quickly from lost commit requests and not significantly impact the post-pickup delay, but is long enough to allow processing of the COMMIT request at the CMTS.

This timer is started when the MTA sends a COMMIT message.

This timer is reset when the MTA receives a COMMIT-ACK or COMMIT-NAK message that is recognized as a response to the COMMIT.

On expiration of this timer, the MTA re-sends the COMMIT message.

The RECOMMENDED value of this timer is 500ms.

6.6 Timer T5

This timer is implemented in the CMS. It controls the synchronization between local MTA resource release and CMTS verification of the closure of the local gate.

When the CMS sends the MTA a message to delete the connection, the CMS MUST ensure that the gate is closed in the CMTS within T5. This timer is reset when the CMS receives a confirmation for the local gate closure via the Gate-Close message.

On expiration of this timer, the CMS deletes the gate at the CMTS using Gate-Delete message with 'Local gate-close failure' described in the reason-code.

The RECOMMENDED value of this timer is 5 seconds.

6.7 Timer T6

This timer is implemented in the MTA or CMTS in the handling of RSVP reservations. It controls the initial delay used by the RSVP retransmit procedure.

The RECOMMENDED value of this timer is 500ms.

6.8 Timer T7

The CMTS MUST set the Timeout for Admitted QoS Parameters for the service flow to the value specified for this timer. The Timeout for Admitted QoS Parameters limits the period of time that the CMTS must hold resources for a service flow's Admitted QoS Parameter Set while they are in excess of its Active QoS Parameter Set. Please see [9] for more details on the use of the Timeout for Admitted QoS Parameters.

In order to allow the EMTA to refresh this timer, the CMTS MUST inform the EMTA of the Timeout for Admitted QoS Parameters value in the response (i.e. in the DSA-RSP) to the EMTA's reservation request.

The recommended default value of this timer is 200 seconds.

6.9 Timer T8

The CMTS MUST set the Timeout for Active QoS Parameters for the service flow to the value specified in for this timer. The Timeout for Active QoS Parameters limits the period of time resources remain unused on an active service flow. Please see [9] for more details on the use of the Timeout for Active QoS Parameters.

In order to allow the EMTA to refresh this timer, the CMTS MUST inform the EMTA of the Timeout for Active QoS Parameters value in the response (i.e. in the DSA-RSP) to the EMTA’s reservation request.

The default value of this timer is 0, which instructs the CMTS not to poll for activity on the service flow.

APPENDIX A THEFT OF SERVICE SCENARIOS

We outline here several possible theft of service scenarios to highlight the need for the dynamic authorization, the need for the two-phase resource reservation protocol, the need for gates, and the need for gate coordination. The system design places much of the session control intelligence at the clients, where it can easily scale with technology and provide new and innovative services. While this “future-proofing” is a goal of the design, we must recognize that it leaves open a wide range of fraud possibilities. This appendix discusses some of those possibilities, and how the QoS signaling architecture prevents them.

The basic assumption is that the MTA is not immune to customer tampering, and that the significant incentive for free service will lead to some very sophisticated attempts to thwart any network controls placed on the MTA. This customer tampering includes, but is not limited to, opening the box and replacing ROMs, replacing integrated circuit chips, probing and reverse engineering of the MTA design, and even total replacement of the MTA with a special black-market version. While technical solutions exist to the physical security of the MTA (e.g., booby trapping the box with lethal gas), they are not considered acceptable.

Since the MTA can be distinguished only by its communication over an DOCSIS network, it is possible, and quite likely, that PC software will be written that will emulate the behavior of a MTA. Such a PC may be indistinguishable from a real MTA. The software behavior in this case is under the total control of the customer.

Further, it is intended that new services will be implemented in the MTA, and that software control of those new services will be provided by a variety of vendors. This updated software will be downloaded into the MTA, leaving open the possibility of customers downloading special hacked versions that provide free service. We do not concern ourselves here with the problem of “trojan horses” in such downloaded software, as this is considered identical to the problem today of customers giving away their credit card numbers and/or PINs. We are concerned with the customer intentionally downloading special software that does only what is in his/her best interest.

Scenario #1: Customers establishing high QoS Connections themselves

The MTA, with sufficient intelligence, can remember past destinations dialed and the destination address, or use some other mechanism to determine the IP address of a destination. It can then signal that destination itself (with some cooperation of the other client), and negotiate a high quality-of-service connection via the RSVP mechanism or via the DOCSIS interface for an embedded client. Since no network agent is used in initiating the session, there will be no billing record produced. Prevention of this scenario is done by requiring dynamic authorization at the CMTS; without the authorization the attempt to obtain the high quality-of-service will fail.

The above scenario required the cooperation of two altered MTAs. Similar theft of service could be accomplished with only the originator being modified. If the originating MTA used the network agent to establish the session, thereby informing the destination in the standard manner of an incoming session, but again negotiated the high quality-of-service itself, there would be no billing record generated and the originator would get a free session. Again, the solution is to require the use of gates in the CMTSes.

Scenario #2: Customers using provisioned QoS for non-voice applications

Statically provisioned QoS can only identify a customer as one who is authorized for high Quality of Service. There is no restriction on the usage of the service. In particular, a customer who has subscribed for a commercial-grade voice communications service, and is therefore authorized to activate high-bandwidth low-latency connections through the DOCSIS network, can use this ability for web surfing or other PC applications. Prevention of this scenario is done by requiring dynamic authorization at the CMTS; without the authorization the attempt to obtain the high quality-of-service will fail.

Scenario #3: MTA non-cooperation for billing

One can easily imagine what would happen if there was a message from the MTA on session establishment that said, “OK, callee has answered, start billing me now,” or a message on hangup that said, “session has completed, stop billing now.” However, there are more subtle ways that a user could have the same affect as tinkering with such messages if they existed.

It is essential in providing a commercial-grade voice communications service using PacketCable to ensure network capacity exists prior to signaling the CPE at the receiving party’s location. This function is done with the RESERVE messages. If the RESERVE message were to actually allocate the bandwidth (i.e., combining the RESERVE and COMMIT mechanisms), then there would be no incentive for the MTA to ever issue the COMMIT. The MTA could merely start transmitting voice packets immediately, and the destination could start transmitting voice packets as soon as the phone is answered. The COMMIT message becomes, in effect, the billing start message above. It is therefore essential that the RESERVE not actually allocate the bandwidth, but rather it must check all current allocations and pending reservations to ensure that the bandwidth will be available at the time of a COMMIT message.

Scenario #4: MTA altering the destination address in voice packets

Another example is when two MTAs, which are far apart, each establish a local session. Once the bandwidth and connection are established, the MTAs then change the IP addresses in the RTP streams to point to each other. The billing system continues to bill each of them for a local session, while the customers are actually engaged in a long distance session. This requires us to have mechanisms at the CMTSes that provide access to higher QoS only based on packet filters previously authorized. Thus, in addition to the 2-phase resource management, this scenario motivates the need for packet filters at the gates.

Scenario #5: Use of half-connections

This is an example of theft of service that could occur in the absence of gate coordination. Suppose one client in a session sends a COMMIT message and the other doesn’t. For example, say the terminating client sends a COMMIT, but fails to send the proper signaling message, so the originator never sends a COMMIT. In this case, only one gate is opened, and the users and network are left with a half-connection. Given that the originator did not send a COMMIT message, the network can't legitimately bill the user for the half-connection. However, it is possible for two colluding clients to set up two half-connections, neither of which is billable, which can be combined to give a full connection between the two parties. This results in a free session. Fraud of this type can only be prevented by synchronizing the operation of the two gates.

Scenario #6: Early termination leaving a half-connection

Gate coordination is also required on completion of the session. Suppose that MTA_O calls MTA_T and pays for the session. Since MTA_O is being charged for the session, it clearly has an incentive to issue a RELEASE message to $CMTS_O$ to close its gate and stop the billing. However, if MTA_T does not issue the RELEASE message to close the gate at $CMTS_T$, a half-connection remains. In this case MTA_T can continue to send voice and/or data to MTA_O without billing for the session. Hence, a GATE-CLOSE message must be issued from the originating side gate at $CMTS_O$ to close the terminating side gate at $CMTS_T$.

Scenario #7: Forged Gate Coordination messages

Each MTA knows the identity of its CMTS, and knows the 5-tuple that its CMTS uses to identify the GateID. MTAs can do various kinds of end-to-end negotiation before asking for resources; in particular they can easily exchange the information about their GateID. Then the MTA can fake the GATE-OPEN message being sent to the non-paying end, and get a non-billed one-way connection. Doing this twice gets a full non-billed connection. One solution to this problem is for the GateController to give the CMTS a key to use for the CMTS-CMTS messages, on a per-session (or per-gate) basis.

Scenario #8: Fraud directed against unwanted callers

Due to details of the call setup sequence, it is possible that the bandwidth authorization at the destination will be more generous than that at the source. Given this, it is then possible for a called party to reserve and allocate bandwidth far in excess of the final negotiated amount, resulting in the calling party being charged for more than expected. If available, this would likely be used against telemarketers, fighting back for unwanted calls during dinner.

Gate coordination, which was used previously to guard against half-connections, also protects from this type of fraud. The GATE-OPEN message tells the bandwidth that was allocated as a result of the COMMIT, and the COMMIT-ACK sent to the originator tells exactly what bandwidth will be charged for the session. If the originator detects anything amiss, he can immediately terminate the session.

APPENDIX B COPS (COMMON OPEN POLICY SERVICE)

COPS Procedures and Principles

This Appendix provides a brief description of COPS procedures and principles, and how COPS relates to other protocols such as LDAP. COPS is currently defined in an Internet Draft [3].

Common Open Policy Service (COPS) protocol is a client/server protocol being defined in the IETF RSVP admission policy (rap) working group for use in admission control in RSVP/IntServ and DiffServ QoS networks. COPS runs over TCP/IP, using a well-known port number 3288. COPS entities would reside at a network edge device and a policy server. Three functional entities are defined for the rap framework:

- Policy Decision Point (PDP) - the server entity in COPS, which makes the final decision on session admission or rejection, based on policy information that it has access to. This is expected to be implemented as an application on a standalone server device.
- Policy Enforcement Point (PEP) - the client entity in COPS, which consults with the PDP to make policy decisions or to obtain policy information that it may itself use to make admission control decisions; the PEP may receive requests for service and initiate a query to the PDP that will result in a go/no-go response, or the PEP may inform the PDP that it wishes to receive decisions and policy related information on an unsolicited basis.
- Local Decision Point (LDP) - a local version of the PDP that can make decisions based on local information or information cached from previous decisions. A PDP decision always takes precedence over the LDP.

A COPS sequence, as used in an RSVP/IntServ environment, is shown below.

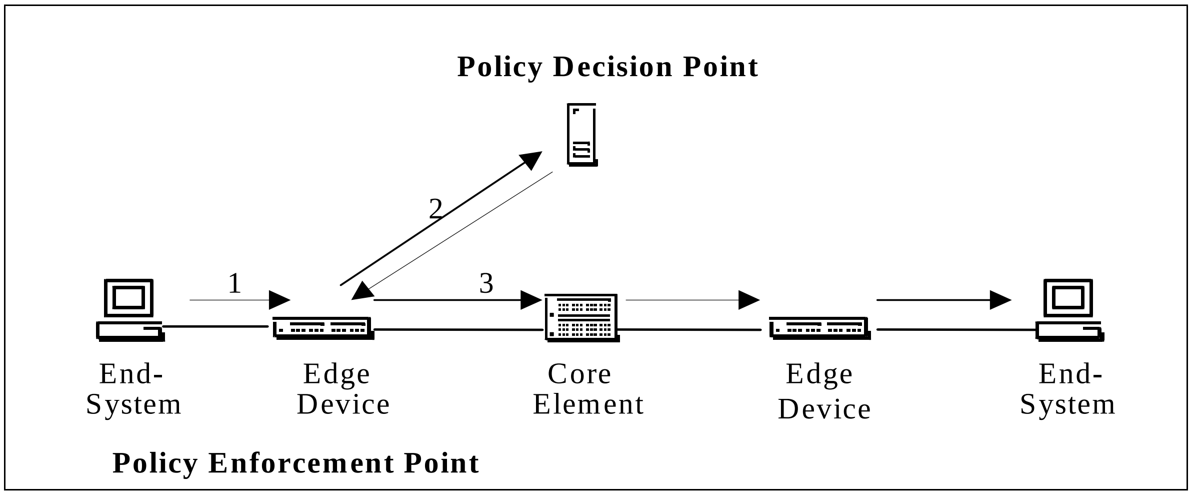


Figure 22. COPS Protocol

In the COPS sequence, the client PEP is responsible for initially establishing a session with the PDP, using information that is either configured in the PEP or determined by some other means. Once the session is established, if the Edge Device receives an RSVP message (1), it generates a request for handling to the PDP (2) that describes the context of the request and carries information about the request. The PDP then responds (3) with a decision to accept or reject the request, and if it is accepted, the Edge Device continues by forwarding the RSVP message out into the network (4).

Each session is maintained by a Keep Alive message that verifies that the session is active in case no message has been received recently. Each RSVP or other request is identified by a Handle, which can be used to associate the response, subsequent unsolicited responses, and clearing.

The protocol messages are extensible to other tasks. They consist of an Op Code identifying if the message is a Request, Response, or other type, followed by self-identifying objects, each containing an object class and version identifier. Each object includes a Class Number that defines what the object is, for example, a Timer object, or a Decision object, plus a Class Type that identifies the subtype or version of the Class that is being used.

Other object classes include Bandwidth allocation Data needed for identifying the resources requested by the user, and Policy objects that can be passed down from the PDP to be included in the RSVP message when it is sent out into the network.

Comparison of COPS and LDAP for Policy

Both COPS and LDAP have been associated with Policy-Based Management, however, they would provide very different functions.

COPS is designed for the client to request a decision from a Policy Decision Point and to interact with the PDP to actively participate in the management of policy and policy-related issues. The PEP that makes the request may have no actual knowledge of policies, and relies on the PDP to make decisions based on its knowledge of policies. The protocol allows the PEP to pass information about the request to the PDP, and the PDP to pass back a decision to allow or reject the request.

LDAP is designed for the client to request a directory record from a directory. The function for using the record is dependent on the client, which must be capable of understanding the retrieved record and deciding how to use the information. The server must be capable of finding the correct record based on information in the request, which may involve a search function, or retrieval of multiple records.

Both COPS and LDAP could be used in the context of RSVP admission control. COPS would be used between the PEP and PDP to forward a request for policy-based analysis. LDAP would be used between the PDP and a Directory Server to retrieve policy records associated with the originating and destination addresses for the RSVP request. The PDP would then make a decision based on the retrieved policy information, and use COPS to pass that decision back to the PEP.

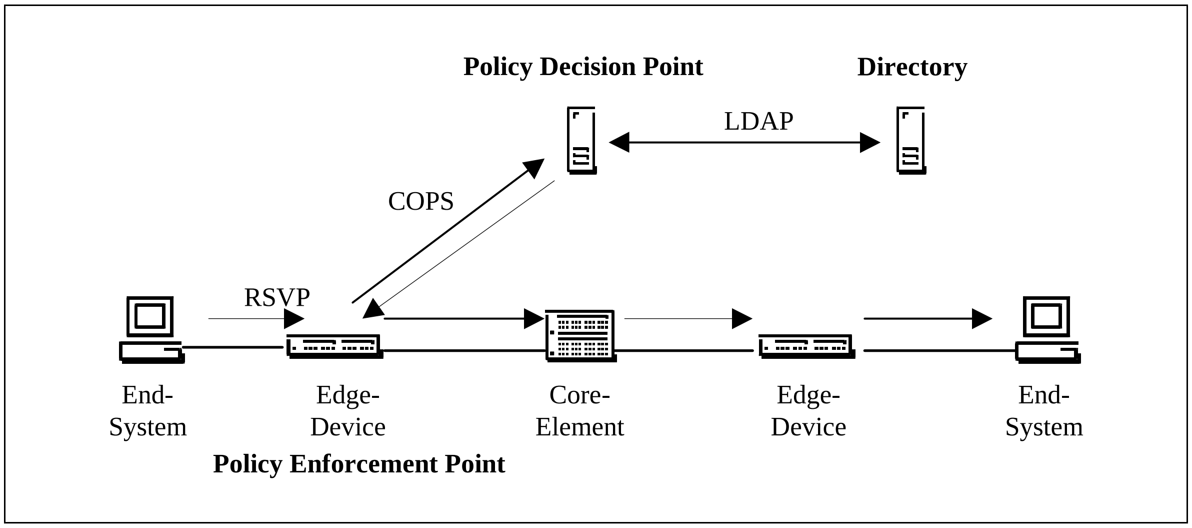


Figure 23. COPS and LDAP model

APPENDIX C RSVP (RESOURCE RESERVATION PROTOCOL)

RSVP Procedures and Principles

This appendix provides a brief description of RSVP procedures and principles. RSVP is currently defined in IETF RFC 2205.

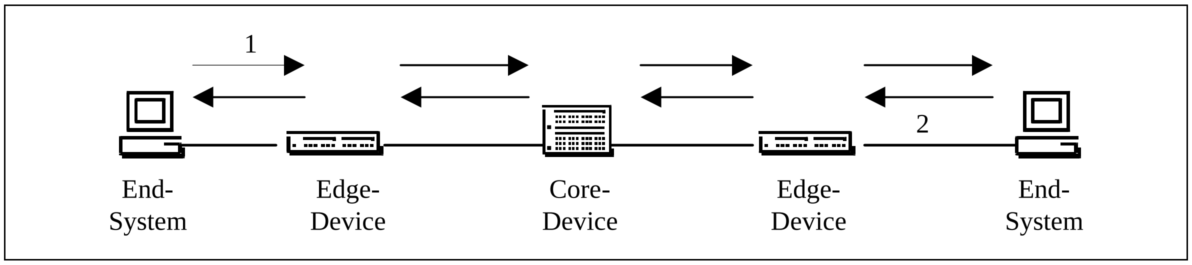


Figure 24. RSVP

RSVP was developed in the IETF for resource reservation to support information flows across the Internet. Some of the main characteristics of RSVP are:

- resource reservation hop by hop to support end-to-end information flows
- state information kept at every participating router
- non-participating routers treat RSVP messages like normal IP packets
- soft state – reservation must be refreshed periodically or it automatically cancels
- request driven – an initial PATH message establishes state in the router. A RESV message from the receiver actually results in reservation of resources.

In RSVP, the source initiates a session by sending out a PATH message (1). This is routed through the network based on its destination address (which may be multicast) and creates a flow state at every RSVP-supporting router that it passes through. The PATH message is routed using the same procedures as other IP packets with that destination address, so that it duplicates the route to be followed by data packets. As it progresses, it records the address of the last RSVP-capable router, and this is added to the state information at the next router.

At the receiving end, the receiver joins the session by sending out a RESV message (2) that identifies a flow or flows that this receiver wishes to receive out of the different flows supported in the session. The RESV message traces back the sequence followed by the PATH message, using the records of the last RSVP-capable router, and causes resources to be reserved at each hop. If multiple RESV messages are received at the same router, they may be merged into a singe RESV message with combined resource reservation request.

The process requires state establishment at multiple internal nodes and resource reservation at the same nodes. It establishes a fixed path for the information flow. However, it ensures that resources have been allocated at all RSVP-supporting points in the path.

RSVP flow spec

An elementary RSVP reservation request consists of a "FlowSpec" together with a "Filter-Spec"; this pair is called a "flow descriptor". The FlowSpec specifies a desired QoS. The filterspec, together with a session specification, defines the set of data packets -- the "flow" -- to receive the QoS defined by the FlowSpec. The FlowSpec is used to set parameters in the node's packet scheduler or other link layer mechanism, while the Filter-Spec is used to set parameters in the packet classifier. Data packets that are addressed to a particular session but do not match any of the Filter-Specs for that session are handled as best effort traffic.

The FlowSpec in a reservation request will generally include a service class and two sets of numeric parameters: (1) an "RSpec" (R for `reserve') that defines the desired QoS, and (2) a "TSpec" (T for `traffic') that describes the data flow.

It is important to note that the formats and contents of TSpecs and RSpecs are determined by the integrated service models [2] defined in the intserv working group of the IETF, and are generally opaque to RSVP itself. RSVP defines the signaling mechanism, and not the traffic model.

APPENDIX D TCP CONSIDERATIONS

This specification defines an interface between a Gate Controller (GC) and a Cable Modem Termination System (CMTS) to be used for gate authorization, which basically supports a transaction based protocol where each transaction is independent. TCP may be used as a transport for this messaging. However, there were concerns raised about the performance implications of using TCP. This appendix examines a few of these concerns and proposes some potential solutions that can provide an acceptable transport through implementation optimizations and tuning of the TCP implementation.

The design of the network should support the desired degree of reliability and real time performance.

Requirements

Let us first consider requirements on the transport protocol for the interaction between the GC and CMTS.

1. Reliable message delivery for messages exchanged between the GC and CMTS is required.
2. The message exchange should have low latency (of the order of milliseconds), in the normal case (without packet loss). We also need it to have reasonably low latency even under packet loss (of the order of tens of milliseconds).
3. We want multiple requests to be outstanding concurrently. This is because multiple call setups are likely to be in progress concurrently.
4. If there is likely to be head-of-the-line (HOL) blocking, this should be avoided.
5. There is likely to be a long-standing association (at least of the order of several minutes) between the GC and the CMTS. However, when there is a failure of a GC, the process of establishing a new connection to the CMTS should not take excessive time. This is especially true when the establishment of a new connection occurs during the time that a call is being setup.

Recommended Changes

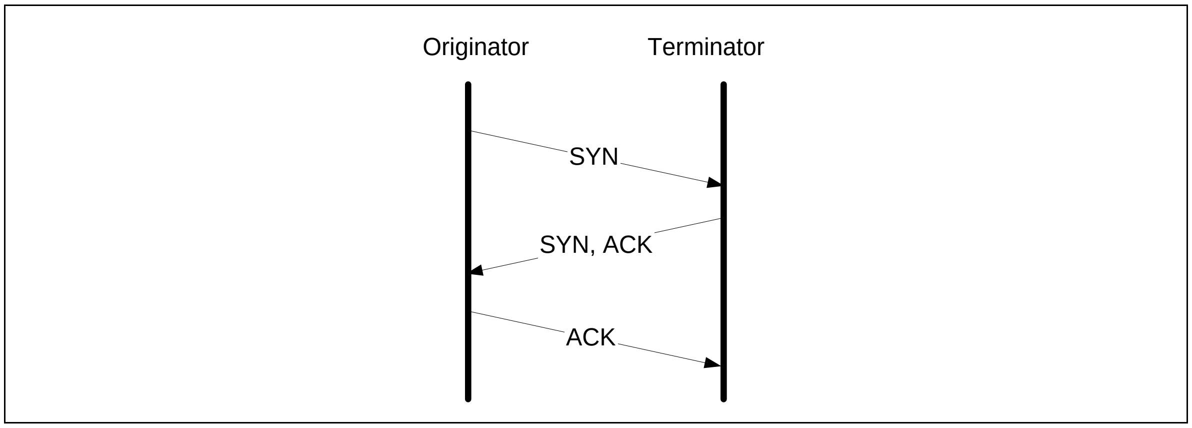
Briefly, the changes we recommend to a vanilla TCP implementation are the following:

1. Modify the time-out mechanism for connection establishment (make it more aggressive)
2. Allow for a larger window after connection establishment.
3. Have multiple TCP connections per GC-CMTS pair to work around potential HOL problems (e.g., use them on a round-robin basis).
4. Lower the 500 ms granularity of the time-out.
5. Disable Nagle's algorithm on the transmit end so as to reduce the latency.
6. Have a non-blocking interface between the application and the TCP stack.

The remainder of this appendix gives details of how these may be implemented.

TCP Connection Establishment impacting Post-dial Delay

TCP connection establishment uses a three-way handshake as follows:



TCP retransmits segments assumed to be lost based on a round-trip time estimate, A, and a mean deviation, D, from A. The retransmission timeout value (RTO) is generally calculated using the formula:

RTO = A + 4D

but the initial RTO is calculated using the formula:

RTO = A + 2D

where A and D are initialized to 0 and 3 seconds respectively. When a retransmission occurs, an exponential backoff using a multiplier of 2 is applied to the current RTO value. Thus, for the first segment, the RTO is calculated as

RTO = 0 + 2 * 3 = 6

Thus, if the initial SYN segment is lost, a retransmission will not occur until 6 seconds later. At that time, RTO will be calculated as:

RTO = 0 + 4 * 3 = 12

and an exponential backoff of 2 applied, leading to a new retransmission timeout value of 24 seconds. Thus, should the retransmission be lost as well, a total of 30 seconds will have elapsed before the third retransmission occurs.

The importance of this problem entirely depends on the frequency with which GC->CMTS connection establishment falls during the post-dial-delay period. In the currently envisioned scenarios, this occurrence should be very much the exception rather than the rule. The connection setup delay impacting the post-dial delay is an important reason to avoid having connection establishment in the post-dial-delay period. Diff-serv marking of the packets to reduce both latency and loss probability, analogous to what is done with routing traffic today, could be used to reduce connection setup delays due to lost packets.

Need Low Latency for packets between the GC and CMTS, even under loss.

Requirement (2), which deals with recovery of packet loss needs a few remedies available for TCP to recover from loss quickly. When there are only a few packets being transmitted, and the receiver is unable to generate a sufficient number of duplicate acks, the recovery from packet loss is from a retransmission time-out. The TCP retransmission algorithm is based on a smoothed average of the observed round-trip time (RTT), A, and a smoothed average of the mean deviation in RTT. As described above, the retransmission time-out value is then set to:

RTO = A + 4D

and if the timer fires, the segment in question is retransmitted, and RTO is backed off exponentially using a multiplier of 2[12] until an upper limit of 64 seconds for RTO. Once a segment has been passed to TCP, the

[12] TCP furthermore uses duplicate ACK's to trigger retransmission of potentially lost segments, however we will ignore that for this part of the discussion.

segment is either transmitted successfully to the destination or the connection is closed after some period of time has passed (generally a large period of time, e.g., 2 to 9 minutes).

While the above retransmission strategy is deemed desirable, we believe it has two (related) problems for the interface considered:

1. If the segment is not delivered successfully within a small period of time, the call that is in the process of being setup will most likely be abandoned and the transaction should therefore be able to be aborted.
2. The 64 second cap on the retransmission timeout is ill-suited for real-time communication and should be set much lower.

A separate, but related issue is that of the granularity of RTO. While the TCP specification itself does not specify the granularity of RTO, it is very common to have a granularity of 500 ms in commercial operating systems. Thus, a lost segment will generally not be detected within less than 500 ms, and two lost segments will not be detected within less than 500 ms + 1000 ms = 1.5 seconds.

To recover rapidly from packet loss in a sequence of packets (without having to depend on multiple duplicate acks to trigger fast retransmit or having to wait for the RTO timer to fire), it may be desirable to implement TCP-SACK, which aids recovery even if the fast-retransmit threshold is not reached. It is also recommended that the TCP implementation use a smaller timer granularity (possibly less than 500 milliseconds).

Head of Line Blocking

Head of line blocking refers to the fact, that TCP provides an in-order data delivery service where a lost segment can block later segments from being delivered to the application. Thus, if segments 1, and 2 are sent from A to B, and segment 1 is lost, segment 2 cannot be delivered to the application until segment 1 has been successfully retransmitted.

For the interface considered, this head of line blocking can probably be overcome reasonably well by having multiple TCP connections established between the GC and CMTS, and then use the set of TCP connections in e.g., a round-robin fashion for the transactions. Thus, if a segment is lost on one connection, it will not affect segments, i.e., transactions sent on other connections.

The downside to this approach is that a lost segment is not likely to be retransmitted until its retransmission timer fires (as opposed to a duplicate ACK being received), since there should not be any additional segments to transmit until then.

TCP Slow Start

TCP's ability to start transmitting a stream of data packets is sometimes limited by the TCP slow start mechanism, especially when the stream is a small number (greater than 1) of data packets. It is desirable to choose an initial window that is larger than 1 (both at the beginning of the life of the connection as well as after congestion recovery from a single packet loss.) Choosing an initial window size of 2 to 4 MSS is considered desirable. It is important however to ensure that this initial window not exceed 4 MSS, because of the potential to cause congestion itself.

Delaying of packets: Nagle's Algorithm

TCP/IP was originally designed for supporting multiple user sessions over a slow network. In order to optimize network utilization, the Nagle algorithm was introduced for keyboard input users. Essentially, this algorithm delays the transmission of a packet until a sufficiently large transmit buffer is accumulated or until a certain period of time (usually around 200 milliseconds) elapses.

Due to the real-time nature of this traffic, it is advisable to disable the Nagle algorithm for GC-CMTS communication. On most Unix based platforms, Nagle's algorithm can be disabled by issuing the following system call on the socket's file descriptor:

Example 1: Setting the TCP_NODELAY Option

```
/* set TCP No-delay flag (disable Nagle algorithm) */
int flag = 1;

setsockopt(fd, IPPROTO_TCP, TCP_NODELAY, &flag,
                  sizeof(flag));
```

Most other languages and platforms have a similar feature to disable the Nagle algorithm, usually known as the TCP_NODELAY option.

Non-Blocking Interface

By default, most operating systems provide a blocking interface for TCP/IP sockets. Although it may allow for an improved error recovery scheme, it has an impact on the performance of the communication channel.

Essentially, a system call such as send() with blocking interface never returns until the operating system confirms that the message was successfully stored in the transmit buffer.

It may be desirable to use a non-blocking interface in order to improve performance and to support asynchronous events using the select() function call on a UNIX based architecture. A non-blocking socket interface can be setup by using the following call on the newly created socket.

Example 2: Setting the O_NONBLOCK Option

```
/* set the socket to non blocking */
fcntl( fd, F_SETFL, O_NONBLOCK );
```

Most other languages and platform have a similar feature.

APPENDIX E TERMS AND DEFINITIONS

PacketCable specifications use the following terms:

Access Control	Limiting the flow of information from the resources of a system only to authorized persons, programs, processes, or other system resources on a network.
Active	A service flow is said to be "active" when it is permitted to forward data packets. A service flow must first be admitted before it is active.
Admitted	A service flow is said to be "admitted" when the CMTS has reserved resources (e.g., bandwidth) for it on the DOCSIS^{TM} network.
A-link	A-Links are SS7 links that interconnect STPs and either SSPs or SCPs. 'A' stands for "Access."
Asymmetric Key	An encryption key or a decryption key used in public key cryptography, where encryption and decryption keys are always distinct.
Audio Server	An Audio Server plays informational announcements in PacketCable network. Media announcements are needed for communications that do not complete and to provide enhanced information services to the user. The component parts of Audio Server services are Media Players and Media Player Controllers.
Authentication	The process of verifying the claimed identity of an entity to another entity.
Authenticity	The ability to ensure that the given information is without modification or forgery and was in fact produced by the entity that claims to have given the information.
Authorization	The act of giving access to a service or device if one has permission to have the access.
Cipher	An algorithm that transforms data between plaintext and ciphertext.
Ciphersuite	A set which must contain both an encryption algorithm and a message authentication algorithm (e.g., a MAC or an HMAC). In general, it may also contain a key-management algorithm, which does not apply in the context of PacketCable.
Ciphertext	The (encrypted) message output from a cryptographic algorithm that is in a format that is unintelligible.
Cleartext	The original (unencrypted) state of a message or data. Also called plaintext.
Confidentiality	A way to ensure that information is not disclosed to anyone other then the intended parties. Information is encrypted to provide confidentiality. Also known as privacy.
Cryptanalysis	The process of recovering the plaintext of a message or the encryption key without access to the key.
Cryptographic algorithm	An algorithm used to transfer text between plaintext and ciphertext.
Decipherment	A procedure applied to ciphertext to translate it into plaintext.
Decryption	A procedure applied to ciphertext to translate it into plaintext.
Decryption key	The key in the cryptographic algorithm to translate the ciphertext to plaintext.
Digital certificate	A binding between an entity's public key and one or more attributes relating to its identity, also known as a public key certificate.

Digital signature	A data value generated by a public-key algorithm based on the contents of a block of data and a private key, yielding an individualized cryptographic checksum.
Downstream	The direction from the headend toward the subscriber location.
Encipherment	A method used to translate plaintext into ciphertext.
Encryption	A method used to translate plaintext into ciphertext.
Encryption Key	The key used in a cryptographic algorithm to translate the plaintext to ciphertext.
Endpoint	A Terminal, Gateway or Multipoint Conference Unit (MCU).
Errored Second	Any 1-second interval containing at least one bit error.
Event Message	A message capturing a single portion of a connection.
F-link	F-Links are SS7 links that directly connect two SS7 end points, such as two SSPs. 'F' stands for "Fully Associated."
Flow [DOCSIS Flow]	(a.k.a. DOCSIS-QoS "service flow") A unidirectional sequence of packets associated with a Service ID (SID) and a QoS. Multiple multimedia streams may be carried in a single DOCSIS Flow.
Flow [IP Flow]	A unidirectional sequence of packets identified by OSI Layer 3 and Layer 4 header information. This information includes source/destination IP addresses, source/destination port numbers, protocol ID. Multiple multimedia streams may be carried in a single IP Flow.
Gateway	Devices bridging between the PacketCable IP Voice Communication world and the PSTN. Examples are the Media Gateway, which provides the bearer circuit interfaces to the PSTN and transcodes the media stream, and the Signaling Gateway, which sends and receives circuit switched network signaling to the edge of the PacketCable network.
H.323	An ITU-T recommendation for transmitting and controlling audio and video information. The H.323 recommendation requires the use of the ITU-T H.225 and ITU-T H.245 protocol for communication control between a "gateway" audio/video endpoint and a "gatekeeper" function.
Header	Protocol control information located at the beginning of a protocol data unit.
Integrity	A way to ensure that information is not modified except by those who are authorized to do so.
IntraLATA	Within a Local Access Transport Area.
Jitter	Variability in the delay of a stream of incoming packets making up a flow such as a voice communication.
Kerberos	A secret-key network authentication protocol that uses a choice of cryptographic algorithms for encryption and a centralized key database for authentication.
Key	A mathematical value input into the selected cryptographic algorithm.
Key Exchange	The swapping of public keys between entities to be used to encrypt communication between the entities.
Key Management	The process of distributing shared symmetric keys needed to run a security protocol.
Key Pair	An associated public and private key where the correspondence between the two are mathematically related, but it is computationally infeasible to derive the private key from the public key.

Keying Material	A set of cryptographic keys and their associated parameters, normally associated with a particular run of a security protocol.
Keyspace	The range of all possible values of the key for a particular cryptographic algorithm.
Latency	The time, expressed in quantity of symbols, taken for a signal element to pass through a device.
Link Encryption	Cryptography applied to data as it travels on data links between the network devices.
Network Layer	Layer 3 in the Open System Interconnection (OSI) architecture that provides network information that is independent from the lower layers.
Network Management	The functions related to the management of data across the network.
Network Management OSS	The functions related to the management of data link layer and physical layer resources and their stations across the data network supported by the hybrid fiber/coax system.
Nonce	A random value used only once that is sent in a communications protocol exchange to prevent replay attacks.
Non-Repudiation	The ability to prevent a sender from denying later that he or she sent a message or performed an action.
Off-Net Call	A communication connecting a PacketCable subscriber out to a user on the PSTN.
On-Net Call	A communication placed by one customer to another customer entirely on the PacketCable Network.
One-way Hash	A hash function that has an insignificant number of collisions upon output.
Plaintext	The original (unencrypted) state of a message or data. Also called cleartext.
Pre-shared Key	A shared secret key passed to both parties in a communication flow, using an unspecified manual or out-of-band mechanism.
Privacy	A way to ensure that information is not disclosed to any one other then the intended parties. Information is usually encrypted to provide confidentiality. Also known as confidentiality.
Private Key	The key used in public key cryptography that belongs to an individual entity and must be kept secret.
Proxy	A facility that indirectly provides some service or acts as a representative in delivering information, thereby eliminating the need for a host to support the service.
Public Key	The key used in public key cryptography that belongs to an individual entity and is distributed publicly. Other entities use this key to encrypt data to be sent to the owner of the key.
Public Key Certificate	A binding between an entity's public key and one or more attributes relating to its identity, also known as a digital certificate.
Public Key Cryptography	A procedure that uses a pair of keys, a public key and a private key, for encryption and decryption, also known as an asymmetric algorithm. A user's public key is publicly available for others to use to send a message to the owner of the key. A user's private key is kept secret and is the only key that can decrypt messages sent encrypted by the user's public key.

Root Private Key	The private signing key of the highest-level Certification Authority. It is normally used to sign public key certificates for lower-level Certification Authorities or other entities.
Root Public Key	The public key of the highest level Certification Authority, normally used to verify digital signatures generated with the corresponding root private key.
Secret Key	The cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a symmetric key.
Session Key	A cryptographic key intended to encrypt data for a limited period of time, typically between a pair of entities.
Signed and Sealed	An “envelope” of information which has been signed with a digital signature and sealed using encryption.
Subflow	A unidirectional flow of IP packets characterized by a single source and destination IP address and single source and destination UDP/TCP port.
Symmetric Key	The cryptographic key used in a symmetric key algorithm, which results in the secrecy of the encrypted data depending solely upon keeping the key a secret, also known as a secret key.
Systems Management	Functions in the application layer related to the management of various Open Systems Interconnection (OSI) resources and their status across all layers of the OSI architecture.
Transit Delays	The time difference between the instant at which the first bit of a Protocol Data Unit (PDU) crosses one designated boundary, and the instant at which the last bit of the same PDU crosses a second designated boundary.
Trunk	An analog or digital connection from a circuit switch that carries user media content and may carry voice signaling (M_F, R_2, etc.).
Tunnel Mode	An IPsec (ESP or AH) mode that is applied to an IP tunnel, where an outer IP packet header (of an intermediate destination) is added on top of the original, inner IP header. In this case, the ESP or AH transform treats the inner IP header as if it were part of the packet payload. When the packet reaches the intermediate destination, the tunnel terminates and both the outer IP packet header and the IPsec ESP or AH transform are taken out.
Upstream	The direction from the subscriber location toward the headend.
X.509 certificate	A public key certificate specification developed as part of the ITU-T X.500 standards directory.

APPENDIX F ABBREVIATIONS

PacketCable specifications use the following abbreviations.

AAA	Authentication, Authorization and Accounting.
AES	Advanced Encryption Standard. A block cipher, used to encrypt the media traffic in PacketCable.
AF	Assured Forwarding. This is a DiffServ Per Hop Behavior.
AH	Authentication header. An IPsec security protocol that provides message integrity for complete IP packets, including the IP header.
AMA	Automated Message Accounting. A standard form of call detail records (CDRs) developed and administered by Bellcore (now Telcordia Technologies).
ASD	Application-Specific Data. A field in some Kerberos key management messages that carries information specific to the security protocol for which the keys are being negotiated.
AT	Access Tandem.
ATM	Asynchronous Transfer Mode. A protocol for the transmission of a variety of digital signals using uniform 53-byte cells.
BAF	Bellcore AMA Format, also known as AMA.
BCID	Billing Correlation ID.
BPI+	Baseline Privacy Plus Interface Specification. The security portion of the DOCSIS 1.1 standard that runs on the MAC layer.
CA	Certification Authority. A trusted organization that accepts certificate applications from entities, authenticates applications, issues certificates and maintains status information about certificates.
CA	Call Agent. The part of the CMS that maintains the communication state, and controls the line side of the communication.
CBC	Cipher Block Chaining mode. An option in block ciphers that combine (XOR) the previous block of ciphertext with the current block of plaintext before encrypting that block of the message.
CBR	Constant Bit Rate.
CDR	Call Detail Record. A single CDR is generated at the end of each billable activity. A single billable activity may also generate multiple CDRs.
CIC	Circuit Identification Code. In ANSI SS7, a two-octet number that uniquely identifies a DSO circuit within the scope of a single SS7 Point Code.
CID	Circuit ID (Pronounced "kid"). This uniquely identifies an ISUP DS0 circuit on a Media Gateway. It is a combination of the circuit's SS7 gateway point code and Circuit Identification Code (CIC). The SS7 DPC is associated with the Signaling Gateway that has domain over the circuit in question.
CIF	Common Intermediate Format.
CIR	Committed Information Rate.
CM	DOCSIS Cable Modem.
CMS	Cryptographic Message Syntax.
CMS	Call Management Server. Controls the audio connections. Also called a Call Agent in MGCP/SGCP terminology. This is one example of an Application Server.

CMTS	Cable Modem Termination System. The device at a cable headend which implements the DOCSIS RFI MAC protocol and connects to CMs over an HFC network.
CMSS	Call Management Server Signaling.
Codec	COder-DECoder.
COPS	Common Open Policy Service protocol. Currently an internet draft, which describes a client/server model for supporting policy control over QoS Signaling Protocols and provisioned QoS resource management.
CoS	Class of Service. The type 4 tuple of a DOCSIS configuration file.
CRCX	Create Connection.
CSR	Customer Service Representative.
DA	Directory Assistance.
DE	Default. This is a DiffServ Per Hop Behavior.
DES	Data Encryption Standard.
DF	Delivery Function.
DHCP	Dynamic Host Configuration Protocol.
DHCP-D	DHCP Default. Network Provider DHCP Server.
DNS	Domain Name Service.
DOCSIS™	Data-Over-Cable Service Interface Specifications.
DPC	Destination Point Code. In ANSI SS7, a 3-octet number which uniquely identifies an SS7 Signaling Point, either an SSP, STP, or SCP.
DQoS	Dynamic Quality-of-Service. Assigned on the fly for each communication depending on the QoS requested.
DSA	Dynamic Service Add.
DSC	Dynamic Service Change.
DSCP	DiffServ Code Point. A field in every IP packet that identifies the DiffServ Per Hop Behavior. In IP version 4, the TOS byte is redefined to be the DSCP. In IP version 6, the Traffic Class octet is used as the DSCP.
DTMF	Dual-tone Multi Frequency (tones).
EF	Expedited Forwarding. A DiffServ Per Hop Behavior.
E-MTA	Embedded MTA. A single node that contains both an MTA and a cable modem.
EO	End Office.
ESP	IPsec Encapsulating Security Payload. Protocol that provides both IP packet encryption and optional message integrity, not covering the IP packet header.
ETSI	European Telecommunications Standards Institute.
F-link	F-Links are SS7 links that directly connect two SS7 end points, such as two SSPs. ‘F’ stands for “Fully Associated.”
FEID	Financial Entity ID.
FGD	Feature Group D signaling.
FQDN	Fully Qualified Domain Name. Refer to IETF RFC 2821 for details.
GC	Gate Controller.
GTT	Global Title Translation.

HFC	Hybrid Fiber/Coaxial. An HFC system is a broadband bi-directional shared media transmission system using fiber trunks between the headend and the fiber nodes, and coaxial distribution from the fiber nodes to the customer locations.
HMAC	Hashed Message Authentication Code. A message authentication algorithm, based on either SHA-1 or MD5 hash and defined in IETF RFC 2104.
HTTP	Hypertext Transfer Protocol. Refer to IETF RFC 1945 and RFC 2068.
IANA	Internet Assigned Numbered Authority. See www.ietf.org for details.
IC	Inter-exchange Carrier.
IETF	Internet Engineering Task Force. A body responsible, among other things, for developing standards used on the Internet. See www.ietf.org for details.
IKE	Internet Key Exchange. A key-management mechanism used to negotiate and derive keys for SAs in IPsec.
IKE–	A notation defined to refer to the use of IKE with pre-shared keys for authentication.
IKE+	A notation defined to refer to the use of IKE with X.509 certificates for authentication.
IP	Internet Protocol. An Internet network-layer protocol.
IPsec	Internet Protocol Security. A collection of Internet standards for protecting IP packets with encryption and authentication.
ISDN	Integrated Services Digital Network.
ISTP	Internet Signaling Transport Protocol.
ISUP	ISDN User Part. A protocol within the SS7 suite of protocols that is used for call signaling within an SS7 network.
ITU	International Telecommunication Union.
ITU-T	International Telecommunication Union–Telecommunication Standardization Sector.
IVR	Interactive Voice Response system.
KDC	Key Distribution Center.
LATA	Local Access and Transport Area.
LD	Long Distance.
LIDB	Line Information Database. Contains customer information required for real-time access such as calling card personal identification numbers (PINs) for real-time validation.
LLC	Logical Link Control. The Ethernet Packet header and optional 802.1P tag which may encapsulate an IP packet. A sublayer of the Data Link Layer.
LNP	Local Number Portability. Allows a customer to retain the same number when switching from one local service provider to another.
LSSGR	LATA Switching Systems Generic Requirements.
MAC	Message Authentication Code. A fixed-length data item that is sent together with a message to ensure integrity, also known as a MIC.
MAC	Media Access Control. It is a sublayer of the Data Link Layer. It normally runs directly over the physical layer.
MC	Multipoint Controller.
MCU	Multipoint Conferencing Unit.
MD5	Message Digest 5. A one-way hash algorithm that maps variable length plaintext into fixed-length (16 byte) ciphertext.

MDCP	Media Device Control Protocol. A media gateway control specification submitted to IETF by Lucent. Now called SCTP.
MDCX	Modify Connection.
MDU	Multi-Dwelling Unit. Multiple units within the same physical building. The term is usually associated with high-rise buildings.
MEGACO	Media Gateway Control IETF working group. See www.ietf.org for details.
MF	Multi-Frequency.
MG	Media Gateway. Provides the bearer circuit interfaces to the PSTN and transcodes the media stream.
MGC	Media Gateway Controller. The overall controller function of the PSTN gateway. Receives, controls and mediates call-signaling information between the PacketCable and PSTN.
MGCP	Media Gateway Control Protocol. Protocol follow-on to SGCP. Refer to IETF 2705.
MIB	Management Information Base.
MIC	Message Integrity Code. A fixed-length data item that is sent together with a message to ensure integrity, also known as a Message Authentication Code (MAC).
MMC	Multi-Point Mixing Controller. A conferencing device for mixing media streams of multiple connections.
MSB	Most Significant Bit.
MSO	Multi-System Operator. A cable company that operates many headend locations in several cities.
MSU	Message Signal Unit.
MTA	Multimedia Terminal Adapter. Contains the interface to a physical voice device, a network interface, CODECs, and all signaling and encapsulation functions required for VoIP transport, class features signaling, and QoS signaling.
MTP	The Message Transfer Part. A set of two protocols (MTP 2, MTP 3) within the SS7 suite of protocols that are used to implement physical, data link, and network-level transport facilities within an SS7 network.
MWD	Maximum Waiting Delay.
NANP	North American Numbering Plan.
NANPNAT	North American Numbering Plan Network Address Translation.
NAT Network Layer	Network Address Translation. Layer 3 in the Open System Interconnection (OSI) architecture. This layer provides services to establish a path between open systems.
NCS	Network Call Signaling.
NPA-NXX	Numbering Plan Area (more commonly known as area code) NXX (sometimes called exchange) represents the next three numbers of a traditional phone number. The N can be any number from 2-9 and the Xs can be any number. The combination of a phone number’s NPA-NXX will usually indicate the physical location of the call device. The exceptions include toll-free numbers and ported number (see LNP).
NTP	Network Time Protocol. An internet standard used for synchronizing clocks of elements distributed on an IP network.
NTSC	National Television Standards Committee. Defines the analog color television broadcast standard used today in North America.
OID	Object Identification.
OSP	Operator Service Provider.

OSS	Operations Systems Support. The back-office software used for configuration, performance, fault, accounting, and security management.
OSS-D	OSS Default. Network Provider Provisioning Server.
PAL	Phase Alternate Line. The European color television format that evolved from the American NTSC standard.
PCES	PacketCable Electronic Surveillance.
PCM	Pulse Code Modulation. A commonly employed algorithm to digitize an analog signal (such as a human voice) into a digital bit stream using simple analog-to-digital conversion techniques.
PDU	Protocol Data Unit.
PHS	Payload Header Suppression. A DOCSIS technique for compressing the Ethernet, IP, and UDP headers of RTP packets.
PKCROSS	Public-Key Cryptography for Cross-Realm Authentication. Utilizes PKINIT for establishing the inter-realm keys and associated inter-realm policies to be applied in issuing cross-realm service tickets between realms and domains in support of Intradomain and Interdomain CMS-to-CMS signaling (CMSS).
PKCS	Public-Key Cryptography Standards. Published by RSA Data Security Inc. These Standards describe how to use public key cryptography in a reliable, secure and interoperable way.
PKI	Public-Key Infrastructure. A process for issuing public key certificates, which includes standards, Certification Authorities, communication between authorities and protocols for managing certification processes.
PKINIT	Public-Key Cryptography for Initial Authentication. The extension to the Kerberos protocol that provides a method for using public-key cryptography during initial authentication.
PSC	Payload Service Class Table, a MIB table that maps RTP payload Type to a Service Class Name.
PSFR	Provisioned Service Flow Reference. An SFR that appears in the DOCSIS configuration file.
PSTN	Public Switched Telephone Network.
QCIF	Quarter Common Intermediate Format.
QoS	Quality of Service. Guarantees network bandwidth and availability for applications.
RADIUS	Remote Authentication Dial-In User Service. An internet protocol (IETF RFC 2865 and RFC 2866) originally designed for allowing users dial-in access to the internet through remote servers. Its flexible design has allowed it to be extended well beyond its original intended use.
RAS	Registration, Admissions and Status. RAS Channel is an unreliable channel used to convey the RAS messages and bandwidth changes between two H.323 entities.
RC4	Rivest Cipher 4. A variable length stream cipher. Optionally used to encrypt the media traffic in PacketCable.
RFC	Request for Comments. Technical policy documents approved by the IETF which are available on the World Wide Web at http://www.ietf.cnri.reston.va.us/rfc.html.
RFI	The DOCSIS Radio Frequency Interface specification.
RJ-11	Registered Jack-11. A standard 4-pin modular connector commonly used in the United States for connecting a phone unit into a wall jack.

RKS	Record Keeping Server. The device, which collects and correlates the various Event Messages.
RSA	A public-key, or asymmetric, cryptographic algorithm used to provide authentication and encryption services. RSA stands for the three inventors of the algorithm; Rivest, Shamir, Adleman.
RSA Key Pair	A public/private key pair created for use with the RSA cryptographic algorithm.
RSVP	Resource Reservation Protocol.
RTCP	Real-Time Control Protocol.
RTO	Retransmission Timeout.
RTP	Real-time Transport Protocol. A protocol for encapsulating encoded voice and video streams. Refer to IETF RFC 1889.
SA	Security Association. A one-way relationship between sender and receiver offering security services on the communication flow.
SAID	Security Association Identifier. Uniquely identifies SAs in the DOCSIS Baseline Privacy Plus Interface (BPI+) security protocol.
SCCP	Signaling Connection Control Part. A protocol within the SS7 suite of protocols that provides two functions in addition to those provided within MTP. The first function is the ability to address applications within a signaling point. The second function is Global Title Translation.
SCP	Service Control Point. A Signaling Point within the SS7 network, identifiable by a Destination Point Code that provides database services to the network.
SCTP	Stream Control Transmission Protocol.
SDP	Session Description Protocol.
SDU	Service Data Unit. Information delivered as a unit between peer service access points.
SF	Service Flow. A unidirectional flow of packets on the RF interface of a DOCSIS system.
SFID	Service Flow ID. A 32-bit integer assigned by the CMTS to each DOCSIS Service Flow defined within a DOCSIS RF MAC domain. SFIDs are considered to be in either the upstream direction (USFID) or downstream direction (DSFID). Upstream Service Flow IDs and Downstream Service Flow IDs are allocated from the same SFID number space.
SFR	Service Flow Reference. A 16-bit message element used within the DOCSIS TLV parameters of Configuration Files and Dynamic Service messages to temporarily identify a defined Service Flow. The CMTS assigns a permanent SFID to each SFR of a message.
SG	Signaling Gateway. An SG is a signaling agent that receives/sends SCN native signaling at the edge of the IP network. In particular, the SS7 SG function translates variants ISUP and TCAP in an SS7-Internet Gateway to a common version of ISUP and TCAP.
SGCP	Simple Gateway Control Protocol. Earlier draft of MGCP.
SHA – 1	Secure Hash Algorithm 1. A one-way hash algorithm.
SID	Service ID. A 14-bit number assigned by a CMTS to identify an upstream virtual circuit. Each SID separately requests and is granted the right to use upstream bandwidth.
SIP	Session Initiation Protocol. An application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants.

SIP+	Session Initiation Protocol Plus. An extension to SIP.
S-MTA	Standalone MTA. A single node that contains an MTA and a non-DOCSIS MAC (e.g., ethernet).
SNMP	Simple Network Management Protocol.
SOHO	Small Office/Home Office.
SS7	Signaling System number 7. An architecture and set of protocols for performing out-of-band call signaling with a telephone network.
SSP	Service Switching Point. SSPs are points within the SS7 network that terminate SS7 signaling links and also originate, terminate, or tandem switch calls.
STP	Signal Transfer Point. A node within an SS7 network that routes signaling messages based on their destination address. This is essentially a packet switch for SS7. It may also perform additional routing services such as Global Title Translation.
TCAP	Transaction Capabilities Application Protocol. A protocol within the SS7 stack that is used for performing remote database transactions with a Signaling Control Point.
TCP	Transmission Control Protocol.
TD	Timeout for Disconnect.
TFTP	Trivial File Transfer Protocol.
TFTP-D	Default – Trivial File Transfer Protocol.
TGS	Ticket Granting Server. A sub-system of the KDC used to grant Kerberos tickets.
TGW	Telephony Gateway.
TIPHON	Telecommunications and Internet Protocol Harmonization Over Network.
TLV	Type-Length-Value. A tuple within a DOCSIS configuration file.
TN	Telephone Number.
ToD	Time-of-Day Server.
TOS	Type of Service. An 8-bit field of every IP version 4 packet. In a DiffServ domain, the TOS byte is treated as the DiffServ Code Point, or DSCP.
TSG	Trunk Subgroup.
UDP	User Datagram Protocol. A connectionless protocol built upon Internet Protocol (IP).
VAD	Voice Activity Detection.
VBR	Variable Bit Rate.
VoIP	Voice-over-IP.

APPENDIX G ACKNOWLEDGEMENTS

This specification was developed and influenced by numerous individuals representing many different vendors and organizations. PacketCable hereby wishes to thank everybody who participated directly or indirectly in this effort. In particular, PacketCable wants to recognize the following individuals for their significant involvement and contributions to this specification:

Burcak Beser (3Com)

Frank Neil, John Ulm, and Fran Reichmeyer (Nortel/Bay/LanCity/Arris Interactive)

K.K. Ramakrishnan and Bill Marshall (primary author) (AT&T)

Bruce Davie and Pratip Banerji (Cisco)

John Pickens and James Yee (Com21)

Clive Holborow and Jon Fellows (General Instrument)

DR Evans (Lucent Cable Communications)

Mike Patrick (Motorola)

Flemming Andreasen (Telcordia)

Glenn Russell and Andrew Sundelin (CableLabs)

APPENDIX H REFERENCES

[1] IETF RFC 2205, *Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*, September 1997.

[2] IETF RFC 2210, *The Use of RSVP with IETF Integrated Services*, September 1997.

[3] IETF RFC 2748, *The COPS (Common Open Policy Service) Protocol*, January 2000.

[4] IEFT RFC 3006, *Integrated Services in the Presence of Compressible Flows*, November 2000.

[5] IETF RFC 3209, *RSVP-TE: "Extensions to RSVP for LSP Tunnels*", December 2001.

[6] IETF RFC 2753, *A Framework for Policy Based Admission Control*, January 2000.

[7] IETF RFC 3084, *COPS Usage for Policy Provisioning*, March, 2001.

[8] IETF RFC 1890, *RTP Profile for Audio and Video Conferences with Minimal control*, January 1996.

[9] *Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification,* SP-RFIv1.1-I10-030730, July 30, 2003, Cable Television Laboratories, Inc.

[10] *PacketCable Distributed Call Signaling Specification*, PKT-SP-DCS-D03-000428, April 28, 2000, Cable Television Laboratories, Inc., ftp://ftp.cablelabs.com/pub/pkt-sp-dcs-d03-000428.pdf.

[11] *PacketCable Network-Based Call Signaling Protocol Specification*, PKT-SP-EC-MGCP-I08-030728, July 28, 2003, Cable Television Laboratories, Inc., http://www.PacketCable.com/

[12] *PacketCable Security Specification*, PKT-SP-SEC- I09-030728, July 28, 2003, Cable Television Laboratories, Inc., http://www.PacketCable.com/

[13] IETF RFC 1321, *MD5 Message-Digest Algorithm*, April 1992.

[14] IETF RFC 2113, *IP Router Alert Option*, February 1997.

[15] IETF RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*, June 2000.

[16] IETF RFC 2866, *RADIUS Accounting*, June 2000.

[17] IETF RFC 2327, *SDP: Session Description Protocol*, April 1998.

[18] *PacketCable Architecture Framework Technical Report*, PKT-TR-ARCH-I01-991201, December 1, 1999, Cable Television Laboratories, Inc., http://www.PacketCable.com/

[19] IETF RFC 2996, *Use and Format of the DCLASS Object with RSVP Signaling*, November 2000.

[20] *PacketCable Event Messages*, PKT-SP-EM-I07-030728, July 28, 2003, Cable Television Laboratories, Inc., http://www.PacketCable.com/

[21] *PacketCable Audio/Video Codecs Specification*, PKT-SP-CODEC-I04-021018, October 18, 2002, Cable Television Laboratories, Inc., http://www.PacketCable.com/

[22] IETF RFC 2749, *COPS Usage for RSVP*, January 2000.

[23] IETF RFC 2961, *RSVP Refresh Overhead Reduction Extensions*, April 2001.

[24] IETF RFC 2750, *RSVP Extensions for Policy Control*, January 2000.

[25] ITU-T, Recommendation G.114 (05/00) - One-way transmission time.

[26] IETF RFC 2543, *SIP: Session Initiation Protocol*, March 1999.

[27] ITU-T, Recommendation G.711 (02/00) - Pulse code modulation (PCM) of voice frequencies.

[28] ITU-T, Recommendation G.729 – Annex E (02/00) - Coding of speech at 8 kbit/s using (CS-ACELP).

[29] ITU-T, Recommendation G.726 Adaptive differential pulse code modulation (ADPCM) (12/90)

[30] ITU-T, Recommendation G.728 (09/92) – Coding of speech at 16 kbit/s using low-delay code-excited linear prediction.

[31] IETF RFC 3168, *The Addition of Explicitly Congestion Notification (ECN) to IPI*, September 2001.

[32] IETF RFC 0791, Internet Protocol, September, 1981.

APPENDIX I REVISION HISTORY

The following Engineering Change Notices (ECNs) were incorporated into PKT-SP-DQOS-I02-000814

ECN	Date Accepted	Author
dqos-n-00002	5/5/2000	Bill Marshall
dqos-n-00007	5/5/2000	Bill Marshall
dqos-n-00012	5/5/2000	Roger Levesque
dqos-n-00013v2	8/17/2000	Roger Levesque
dqos-n-00015v2	5/5/2000	Roger Levesque
dqos-n-00016	5/5/2000	Roger Levesque
dqos-n-00017v2	6/9/2000	Roger Levesque
dqos-n-00037	6/9/2000	Diego Mazzola
dqos-n-00039	6/9/2000	Roger Levesque
dqos-n-00040	6/9/2000	JC Ferguson
dqos-n-00041v2	6/9/2000	JC Ferguson
dqos-n-00042	6/9/2000	JC Ferguson
dqos-n-00046v5	7/5/2000	Madhu Sudan
dqos-n-00047	6/9/2000	Madhu Sudan
dqos-n-00051v3	6/28/2000	Anthony Toubassi
dqos-n-00061v3	6/28/2000	Roger Levesque
dqos-n-00068v2	7/24/2000	Madhu Sudan
dqos-n-00090v2	8/2/2000	Itay Sherman
dqos-n-00091v2	8/2/2000	JC Ferguson
dqos-n-00092	8/2/2000	Roger Levesque
dqos-n-00093	8/2/2000	David Flanagan
dqos-n-00094v2	8/2/2000	Glenn Russell
dqos-n-00095v2	8/2/2000	Bill Hanks
dqos-n-00096	8/2/2000	JC Ferguson
dqos-n-00097v2	8/2/2000	Glenn Russell

The following Engineering Change Notices (ECNs) were incorporated into PKT-SP-DQOS-I03-020116.

ECN	Date Accepted	Author
dqos-n-00121	12/27/01	JC Ferguson
dqos-n-00123	12/27/01	JC Ferguson
dqos-n-00129	12/27/00	David Flanagan
dqos-n-00130	12/27/00	DR Evans
dqos-n-00135	12/27/00	DR Evans
dqos-n-00136	12/27/00	DR Evans
dqos-n-00137-v2	12/27/00	DR Evans
dqos-n-00140	12/27/00	DR Evans
dqos-n-00142	12/27/00	JC Ferguson
dqos-n-00143-v2	2/12/01	R. Ghatta
dqos-n-00144-v2	2/12/01	DR Evans
dqos-n-00147	12/27/00	DR Evans
dqos-n-00148	12/27/00	DR Evans
dqos-n-00149v2	1/22/01	Sophia Scoggins
dqos-n-00152-v2	2/12/01	Madhu Sudan
dqos-n-00153	2/12/01	Madhu Sudan
dqos-n-00154	2/12/01	Madhu Sudan
dqos-n-01003	3/19/01	DR Evans
dqos-n-01048	8/13/01	David Flanagan
dqos-n-01069-v2	7/2/01	Rex Coldren
dqos-n-01070-v2	7/2/01	Rex Coldren
dqos-n-01071-v2	7/2/01	Rex Coldren
dqos-n-01072-v3	7/2/01	Rex Coldren
dqos-n-01073	7/2/01	R. Coleren
dqos-n-01082	7/2/01	Rex Coldren
dqos-n-01102	11/19/01	Bill Hanks
dqos-n-01103-v2	9/17/01	Bill Hanks
dqos-n-01113	9/17/01	DR Evans
dqos-n-01114	9/17/01	DR Evans
dqos-n-01152	12/29/01	Burcak Beser
dqos-n-01153	11/19/01	Burcak Beser

dqos-n-01154	12/3/01	Glenn Russell
dqos-n-01174	11/19/01	DR Evans
dqos-n-01177	11/19/01	David Flanagan
dqos-n-01178	11/19/01	David Flanagan
dqos-n-01181	12/17/01	DR Evans
dqos-n-01183	11/26/01	Burcak Beser
dqos-n-01188	12/10/01	Burcak Beser
dqos-n-01190	11/19/01	Chad Griffiths, Nabil Valji
dqos-n-01191	12/3/01	Chad Griffiths, Nabil Valji
dqos-n-01192	12/3/01	Chad Griffiths, Nabil Valji
dqos-n-01208	11/19/01	David Flanagan
dqos-n-01209	11/19/01	Roger Levesque
dqos-n-01211	11/19/01	Chad Griffiths, Nabil Valji
dqos-n-01212	12/3/01	Chad Griffiths, Nabil Valji
dqos-n-01214	12/3/01	Chad Griffiths, Nabil Valji
dqos-n-01215	11/19/01	Chad Griffiths, Nabil Valji
dqos-n-01217	11/19/01	Tony MacDonald
dqos-n-01218	12/3/01	Tony MacDonald

The following ECNs were incorporated into PKT-SP-DQOS-I04-021018.

ECN	Date Accepted	Problem Description
dqos-n-02003	7/1/02	Include PacketCable CODEC references, move Appendices A and B.
dqos-n-02013	7/1/02	Multiple admitted QoS parameter sets.
dqos-n-02021	7/1/02	Defines term “RSVP+” to match usage within PacketCable project.
dqos-n-02022	07/10/02	Clarifies the gate identity used in gate coordination messages.
dqos-n-02028	07/01/02	Defines how to properly handle the QoS related operations when multiple codecs are specified within a given connection.
dqos-n-02042	07/01/02	Corrects example call flow
dqos-n-02046	07/01/02	Reflects the change in length BCID to match PacketCable event messages specification.
dqos-n-02051	07/01/02	Adds reference to the call Signalling service flow creation as described in the provisioning specification
dqos-n-02053	07/01/02	Describes the CMTS behavior when the COPS connection is lost.
dqos-n-02054	07/01/02	Restoring DSC refreshes method deleted by dqos-n-02141.
dqos-n-02064	07/01/02	The mapping of RVP objects to DOCSIS downstream QoS TLV’s is specified.
dqos-n-02067	07/01/02	Distinguish between DOCSIS behavior and gate state when a gate is auto-committed.
dqos-n-02077	07/01/02	Clarify settings of the solicited flag in COPS header.
dqos-n-02092	07/01/02	The remote-gate-info object format needs clarification.
dqos-n-02095	07/01/02	Changing granularity and size of existing timers T1 and T2 and addition of 2 additional timers.
dqos-n-02152	08/22/02	Defines how to handle QoS related operations when port to port calls are made on the same MTA.

The following ECNs were incorporated into PKT-SP-DQOS-I05-021127.

ECN	Date Accepted	Problem Description
dqos-n-02148	10/21/02	Consolidation of Gate-Coordination into Gate-Control Protocol
dqos-n-02153	10/21/02	Clarifying the vague points in engineering change dqos-n-02064
dqos-n-02185	11/20/02	Remove Media-Connection-Event-Info object for CMTS

The following ECNs were incorporated into PKT-SP-DQOS-I06-030415.

ECN	Date Accepted	Problem Description
dqos-n-02233	2/24/03	Corrects improper parameter normalization procedure for upstream rate authorization.
dqos-n-02234	3/13/03	Allows for authorization against multiple Gates on a single flow.
dqos-n-03011	3/3/03	Removes ambiguity surrounding T1 timer and Gate-Info-Ack protocol elements.
pkt-n-03066	3/3/03	Update the standard definition of SFID.

The following ECNs were incorporated into PKT-SP-DQOS-I07-030815.

ECN	Date Accepted	Problem Description
dqos-n-03026	7/21/03	Remove Session Description Object from the COPS Gate Control interface in keeping with em-r-03030; add CCCID in support of electronic surveillance.