EMIR TOKTAR

CONTROLE DE ADMISSÃO DE RSVP UTILIZANDO XACML

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

CURITIBA

2003

EMIR TOKTAR

CONTROLE DE ADMISSÃO DE RSVP UTILIZANDO XACML

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

Área de Concentração: *Metodologia e Técnicas de Computação*
Orientador: Prof. Edgar Jamhour, PhD.

CURITIBA
2003

Toktar, Emir
Controle de Admissão de RSVP Utilizando XACML. Curitiba, 2003. 160p.

Dissertação(Mestrado) – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Segurança 2. RSVP 3. XACML 4. Política.
I. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada II.

"FAÇO O MELHOR QUE SEI, O MELHOR QUE POSSO, E O FAÇO ATÉ O FINAL. SE AO FIM TUDO DEU CERTO, O QUE DIZEM CONTRA MIM NÃO IMPORTA. SE O FIM RESULTA NUM ERRO TOTAL, DEZ ANJOS DIZENDO QUE EU ESTAVA CERTO NÃO FARÃO A MENOR DIFERENÇA"

Abraham Lincoln

Agradecimentos

Neste momento, humildemente me curvo diante de todos aqueles que me incentivaram, acreditando-me capaz de vencer, em especial ao Prof. Edgar Jamhour.

Igualmente, agradeço com sinceridade a todos que disseram que este trabalho era inviável ou difícil de ser realizado, pois estes semearam em meu espírito a perseverança, que jamais permite que se desista de um verdadeiro sonho.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviações

ACL	*Access Control List*
CIM	*Common Information Model*
COPS	*Common Open Policy Service*
CORBA	*Common Object Request Broker Architecture*
DAC	*Discretionary Access Controls*
DMTF	*Distributed Management Task Force*
DoD	*Department of Defense*
GIOP	*General Inter-ORB Protocol*
HTTP	*HyperText Transport Protocol*
IANA	*Internet Assigned Numbers Authority*
ICPM	*IPsec Configuration Policy Model*
IETF	*Internet Engineering Task Force*
IIOP	*Internet Inter-ORB Protocol*
LDAP	*Lightweight Directory Access Protocol*
MAC	*Mandatory Access Controls*
OASIS	*Organization for the Advancement of Structured Information Standards*
OMA	*Object Management Group*
OMG	*Object Management Architecture*
ORB	*Object Request Broker*
OSI-RM	*Open System Interconnection – Reference Model*
PBNM	*Policy-Based Network Management*
PCIM	*Policy Core Information Model*
PCIMe	*Policy Core Information Model Extensions*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PIB	*Policy Information Base*

QoS	*Quality of Service*
QPIM	*Policy QoS Information Model*
RBAC	*Role-Based Access Control*
RSVP	*Resource ReSerVation setup Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifiers*
XACML	*eXtensible Access Control Markup Language*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*

Resumo

O avanço tecnológico determinou o surgimento de novas aplicações sobre a Internet, sendo necessário a garantia de serviços prestados aos usuários através de um efetivo controle dos recursos disponíveis da rede de computadores.

O acesso à rede com garantia de serviços é comandado por arquiteturas de otimização de desempenho e acesso de alocação de recursos, que podem priorizar, garantir ou negar o acesso à rede ao usuário ou aplicativo, segundo a política de acesso relacionada à qualidade de serviço (QoS).

Este estudo tem como proposta utilizar o protocolo RSVP (*Resource ReSerVation setup Protocol*) para alocação de recursos, submetidos a uma política descritiva centralizada XACML (*eXtensible Access Control Markup Language*), que ainda não possui padronização definida pelo OASIS (*Organization for the Advancement of Structured Information Standards*) a respeito do protocolo de comunicação.

O principal objetivo deste estudo é controlar o acesso aos recursos da rede e determinar a quantidade de recursos que será disponibilizada ao usuário, propondo uma extensão da estrutura XACML para controle de acesso com o protocolo RSVP.

Palavras-Chave: 1. Segurança 2. RSVP 3. XACML 4. Política

Abstract

The technological advance determined the sprouting of new applications on the Internet, being necessary to the guarantee of services given to the users through an effective control of the available resources of the computer network.

The access to the net with guarantee of services is commanded by architectures of performance optimization and resource allocation, that can prioritize, guarantee or deny the access to the net of the users or of the applications, according to politics of access related to the quality of service (QoS).

This study has as proposal to use *Resource ReSerVation setup Protocol* (RSVP) protocol for resource allocation, submitted to one descriptive centered politics (*eXtensible Access Control Markup Language* - XACML), that still does not possess standardization defined for the OASIS (*Organization for the Advancement of Structured Information Standards*) regarding the communication protocol.

The main objective of this study is to control the access to the network resources and to determine the amount of resources that will be available to the user, considering an extension of XACML structure for access control using the RSVP protocol.

Keywords: 1. Security 2. RSVP 3. XACML 4. Policy

Capítulo 1

Introdução

A comunidade científica vem se empenhando durante muitos anos em pesquisas para o aprimoramento da tecnologia Internet, devido ao sucesso e crescimento admirável de redes de computadores, resultando na necessidade de novos aplicativos e aprimoramentos nos sistemas operacionais.

Ao mesmo tempo que a Internet crescia e se popularizava com o correio eletrônico, acesso remoto e transferência de arquivos, ocorria a descentralização de políticas e dificuldade de administrar as redes. Para este problema foram propostos dois modelos para arquitetura de políticas: o PCIM e o XACML.

A *Distributed Management Task Force* (DMTF) juntamente com a *Internet Engineering Task Force* (IETF) projetou o PCIM (*Policy Core Information Model*) na publicação RFC3460 [MOO03] como um modelo orientado a objetos para representação de política de informações.

O OASIS (*Organization for the Advancement of Structured Information Standards*) propôs o XACML (*eXtensible Access Control Markup Language*), que é um modelo voltado para representação de políticas descritivas, o que a *priori*, parece ser mais intelegível do que um modelo orientado a objetos, porém ainda não existe padronização definida pelo OASIS a respeito do protocolo de comunicação e tecnologias de armazenamento de políticas.

Os modelos de segurança para controle de acesso a sistemas de computadores, conforme especificação do *Trusted Computer System Evaluation Criteria* (TCSEC)[1] [TCS85]

[1] TCSEC são critérios que foram previamente usados para classificar o grau de segurança oferecido por um sistema de computador, sendo referido como "*the Orange Book*" por causa da cor alaranjada da capa. URL: http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html.

são de dois tipos: o mandatório ou *Mandatory Access Controls* (MAC) e o discricionário ou *Discretionary Access Controls* (DAC). Os controles MAC são apropriados para aplicações de segurança militares em multiníveis, e os controles DAC são utilizados pelos sistemas operacionais para fins comerciais na indústria e governos civis. Um terceiro modelo, denominado "controle de acesso baseado em papéis" (RBAC-*Role Based Access Control*), é feito considerando-se a função (papel) de cada usuário e não sua identidade. Esta concepção facilita e simplifica o gerenciamento de autorização, e fornece uma maior flexibilidade nas especificações e regras de negócio, o que reflete na política de segurança em sistemas [BAR97], [CHO97], [SAN98], [FER00].

Uma evolução do TCSEC foi desenvolvida com a finalidade de padronizar a avaliação de sistemas comerciais para o mercado internacional e originou o padrão conhecido como CCITSE[2] (*Common Criteria for Information Technology Security Evaluation*), usualmente referido como "Critério Comum" (CC - "*Common Criteria*") [CCI99], e utiliza critérios diferentes para avaliação de sistemas em relação ao modelo TCSEC.

O acesso à rede de comunicações por computadores, usuários ou aplicativos, controlados pelo sistema operacional para prover recursos de rede, é comandado por arquiteturas de otimização de performance e acesso de alocação de recursos, que podem priorizar, garantir ou negar o acesso à rede ao usuário ou aplicativo, segundo a política de acesso relacionada à qualidade de serviço (QoS – *Quality of Service*).

A alocação de recursos com suporte em QoS pode ser feita por: Priorização ou Reserva de Recursos.

Priorização ou Serviços Diferenciados (*DiffServ*) [BLA98] prioriza o envio de pacotes processados na rede de acordo com a classificação dos mesmos. Não utiliza protocolo de sinalização e não oferece garantia de serviço.

Reserva de Recursos ou Serviços Integrados (*IntServ*) [BRA94] também está submetido à política de gerenciamento de largura de banda, assim como o *DiffServ*, porém para enviar qualquer pacote na rede precisa de um protocolo de sinalização como o *Resource Reservation Protocol* (RSVP), mas fornece, entre outros, a garantia da reserva de recursos.

Com o avanço tecnológico surgiram novas aplicações sobre a Internet como vídeo-conferência, telefonia sobre a Internet, salas de discussões, multimídia e outros, que tornaram

---

[2] CCITSE: Em 1996 sete nações desenvolveram um padrão para avaliação de segurança na área de Tecnologia da Informação para sistemas comerciais e amplamente utilizada pela comunidade internacional, apresentando testes e avaliações em diversas tecnologias e produtos. URL: http://www.commoncriteria.org

obrigatório a garantia de serviços prestados aos usuários. Baseado nessa premissa, é necessário um efetivo controle dos recursos disponíveis da rede.

O presente estudo tem como proposta utilizar o protocolo mais complexo de QoS, o RSVP, para alocação de recursos, submetidos a uma política descritiva centralizada (XACML), controlando o acesso aos recursos da rede, porém tendo como principal objetivo determinar a quantidade deste recurso que será disponibilizada ao usuário.

O trabalho apresentado nesta dissertação está estruturado em sete capítulos, sendo este o primeiro capítulo, Introdução, e na seqüência os demais capítulos organizados conforme descrição abaixo:

Capítulo 2, Qualidade de Serviço (QoS): Apresenta de forma sucinta o conceito de Qualidade de Serviço, arquiteturas propostas pela comunidade científica para QoS, tipos e classificação de QoS em relação às aplicações. É discutido a seguir o protocolo RSVP, utilizado neste estudo, suas características e especificações de tráfego para algumas séries de aplicações.

Capítulo 3, Gerência de QoS Baseada em Políticas: Apresenta a Gerência de Redes Baseadas em Políticas (PBNM), modelos de processamento de PBNM, cenário e representação de políticas no modelo para representação de políticas orientada a objetos, com enfoque em QoS, e trabalhos relacionados.

Capítulo 4, XACML: Apresenta o XACML, um modelo descritivo para representação de políticas proposto pela OASIS, as especificações e os modelos XACML, finalizando o capítulo com uma representação de política utilizando este conceito.

Capítulo 5, Proposta: Descreve extensões para o XACML suportar QoS utilizando o protocolo RSVP, e apresenta um cenário envolvendo PBNM com XACML em um processo de reserva de fluxo RSVP.

Capítulo 6, Estudo de Caso e Avaliação: Descreve exemplos de políticas de QoS em um cenário típico com XACML.

Capítulo 7, Conclusões: apresenta as conclusões obtidas neste trabalho e possíveis trabalhos futuros.

Capítulo 2

Qualidade de Serviço (QoS)

Durante muitos anos a Internet foi utilizada em pesquisas de rede e troca de informações entre computadores utilizando o protocolo *Transmission Control Protocol*/*Internet Protocol* (TCP/IP). Acesso remoto, transferência de arquivos e correio eletrônico (*e-mail*) foram as aplicações mais populares. O crescimento da Internet tem proporcionado o surgimento de novas aplicações, sistemas e multimídia[3]. Devido a sua flexibilidade e abrangência, outros tipos de redes como telefone, rádio e televisão também estão convergindo para o ambiente IP. Muitas aplicações exigem a **garantia de performance** e a **diferenciação de serviço**, produzindo novos desafios em interoperabilidade e infra-estrutura da rede IP.

A **Garantia de Performance** envolve o gerenciamento de recursos na rede e a Internet é inexpressiva neste aspecto. As aplicações em tempo real, como exemplo a vídeo conferência, requerem um nível mínimo de recursos para operarem eficazmente. A **garantia de recursos**, que possibilita tanto uma reserva de recursos para fluxos de dados de aplicações e/ou uma combinação de aprovisionamento, priorização e controle de tráfego, é crítica às aplicações que venham a utilizar a Internet ou redes IP [WAN01].

A **Diferenciação de Serviço** deve prover diferentes níveis de serviços para atender os requerimentos de aplicações e as redes IP oferecem um único nível de serviço, o "melhor-esforço" ou *best-effort*.

A capacidade de fornecer **Garantia de Recursos** e **Diferenciação de Serviços** em uma rede, oferecidos através da definição de parâmetros tais como: largura de banda, atrasos

[3] Multimídia: genericamente, aplicações que envolvem a transferência de múltiplos tipos de mídia como: dados, voz, vídeo, gráficos, e outros, com requisitos de tempo e sincronização para operar com qualidade.

de pacotes e perda de pacotes, é referida como **Qualidade de Serviço** (*QoS – Quality of Service*).

Os protocolos de QoS são designados para adicionar um diferencial na rede, propiciando distinguir o tráfego de dados com requerimento de tempo de outros tráfegos que podem tolerar variações de atrasos (*jitter*) e perda de pacotes. Não é possível para uma rede fornecer o que não tem, logo, disponibilidade de largura de banda é o ponto de partida. O QoS não cria largura de banda (*bandwidth*), mas propicia uma melhor utilização por conhecer a disponibilidade e requerimentos de aplicações. A meta do QoS é fornecer **previsibilidade** e **controle** em redes IP.

Por mais de uma década a comunidade científica da Internet tem desenvolvido novas tecnologias para melhorar a Internet ou redes com suporte ao TCP/IP com capacidades em QoS, apontando duas questões chaves: **Otimização de Performance** e **Alocação de Recursos**.

A **Alocação de Recursos** deve prever um controle de fluxo de pacotes para evitar o descarte ou atraso devido à indisponibilidade de banda ou espaço de armazenamento de dados (*buffers*), conhecido como **Controle de Admissão**. Sem este, os usuários ou processos podem enviar pacotes na rede e comprometer os recursos disponíveis da mesma, das aplicações de tempo real e aplicações sensíveis à latência e perda de pacotes.

A comunidade Internet desenvolveu duas arquiteturas para **Alocação de Recursos** que são importantes para o suporte em QoS[4]**: Reserva de Recursos** e **Priorização**.

Para garantir **Reserva de Recursos** a pacotes particulares de uma aplicação, é necessário utilizar um protocolo de sinalização como o RSVP (***R****esource Re****S****er****V****ation* ***P****rotocol*), escolhido pela *Internet Engineering Task Force* (IETF).

O presente estudo enfoca a interação do RSVP com um servidor de políticas atuando com seus próprios mecanismos de controle de admissão e política de acesso, em um ambiente cliente/servidor.

2.1 Tipos de QoS

A maneira de caracterizar o QoS se apresenta na literatura sob diversos aspectos [CIS02c], [STA99a], [BRA94], [WAN01], [BRA97a]. A classificação em níveis das

[4] **Reserva de Recursos** ou (*Integrated Services – IntServ*) descrito por [BRA94] e **Priorização** ou Serviços Diferenciados (*DiffServ*) descrito por[BLA98].

aplicações depende de seus requesitos específicos. O nível de serviço para cada fluxo de dados é configurado no **Controle de Admissão** que em QoS suporta três níveis de serviço: *best-effort*, *controlled-load* e *guaranteed service*. Para determinar o nível apropriado a cada requerimento de qualidade de serviço é necessário entender os padrões de tráfego das aplicações, viabilizando sua utilização. As aplicações são qualificadas em duas classes: aplicações elásticas (***elastic***) que suportam Melhor-esforço (*best-effort*) e aplicações de tempo-real (***real-time***) que suportam Carga controlada (*controlled-load*) e/ou Serviço Garantido (*guaranteed service*).

Aplicações em **Tempo-Real** se referem a um grupo de aplicações que possuem requisitos rigorosos quanto ao atraso. Áudio e vídeo-conferência são exemplos de aplicações em tempo-real. Durante a transmissão os sinais acumulam o atraso nos elementos de rede que são removidos na recepção, são tratados para obtenção do sinal original e com pouca distorção em relação à fidelidade ou atraso. Aplicações deste tipo são chamadas *playback applications* e medidas em duas dimensões para performance: latência e fidelidade. As *playback applications* podem ser classificadas como intolerantes ou torelantes, dependendo do requisito para fidelidade, perda de pacotes ou atrasos. A tabela 2.1 apresenta uma classificação para exemplificação.

Tabela 2.1 Classificação de aplicações *Playback* [BRA94]

<table>
<tr><td>Intolerantes</td><td>guaranteed services (intolerant services)</td><td colspan="2">Pode ser interpretado como “absoluto ou estatístico”, “estrito ou aproximado”. São aplicações que pelas suas características requerem fidelidade ou não toleram perda de pacotes ou distorções no tempo de entrega.</td></tr>
<tr><td rowspan="2">Tolerantes</td><td rowspan="2">predictive service (tolerant services)</td><td>adaptativas</td><td>Aplicações mais flexíveis que podem ainda alterar o ponto de reprodução dos pacotes a partir dos atrasos reais experimentados pelas mesmas na rede.</td></tr>
<tr><td>não adaptativas</td><td>Aplicações que toleram pacotes atrasados, mas não são flexíveis para se adaptarem aos atrasos experimentados na rede.</td></tr>
</table>

2.2 Resource ReSerVation Protocol (RSVP)

Resource Reservation Protocol (RSVP) é um protocolo QoS de sinalização eficiente, robusto e flexível que suporta fluxos de dados *multicast* e *unicast*, adapta-se dinamicamente

para mudar membros do grupo e mudanças de rota. RSVP é o protocolo mais complexo de todas as tecnologias de QoS para aplicações (*host*) e elementos de rede (roteadores e *switches*). Suporta os métodos de comunicação: ponto-a-ponto, ponto-a-multiponto, multiponto-a-multiponto. Um *host* usa o RSVP para requisitar um QoS específico na rede para uma aplicação em particular e, através de uma sinalização do protocolo RSVP, habilita a reserva de recursos da rede. O RSVP também é utilizado pelos roteadores para entregar o QoS requisitado para todos os nós envolvidos no caminho dos fluxos de dados, estabelecendo e mantendo o estado para prover o serviço requisitado. Uma requisição RSVP geralmente resultará em recursos sendo reservados em cada nó ao longo do caminho [BRA97a].

As conexões RSVP são unidirecionais, conhecidas como ***simplex flow***, usadas em fluxos distintos de aplicação ou ***Per flow***, entre duas aplicações e identificadas unicamente por uma *tupla* de cinco elementos: <protocolo de transporte, endereço de origem, número da porta origem, endereço do destino e número da porta destino>. O protocolo é orientado à recepção, ou seja, o receptor de um fluxo de dados inicia e mantém a reserva de recurso usado para aquele fluxo. Para habilitar a comunicação com garantia de serviço para o emissor e receptor, dois fluxos individuais são requeridos. Portanto, o RSVP trata o emissor logicamente distinto do receptor e, uma mesma aplicação pode atuar como um emissor e um receptor ao mesmo tempo, mantendo o estado da conexão com atualizações periódicas chamadas de *"soft state"*.

RSVP não é um protocolo de roteamento, mas designado para operar com protocolos de roteamento *multicast* e *unicast*. As mensagens RSVP são enviadas ponto-a-ponto (*hop-by-hop*) entre roteadores compatíveis com RSVP (*RSVP-Aware*), utilizando datagramas em modo *IP raw*, protocolo IP 46. Dispositivos de rede que não suportam o RSVP sobre IP podem utilizar o protocolo UDP encapsulando RSVP, utilizando as portas 1698 - *"end system"* e a porta 1699 - *"RSVP-enabled Router"*. Em relação a camada que o RSVP atua, segundo o modelo OSI-RM, o protocolo RSVP opera sobre a camada 3 (*Network Layer*) ou em relação ao *DoD model* [POS80a], sobre as camadas IPv4 ou IPv6, ocupando o lugar de um protocolo de transporte na pilha, apresentado na figura 2.1, [BRA97a].

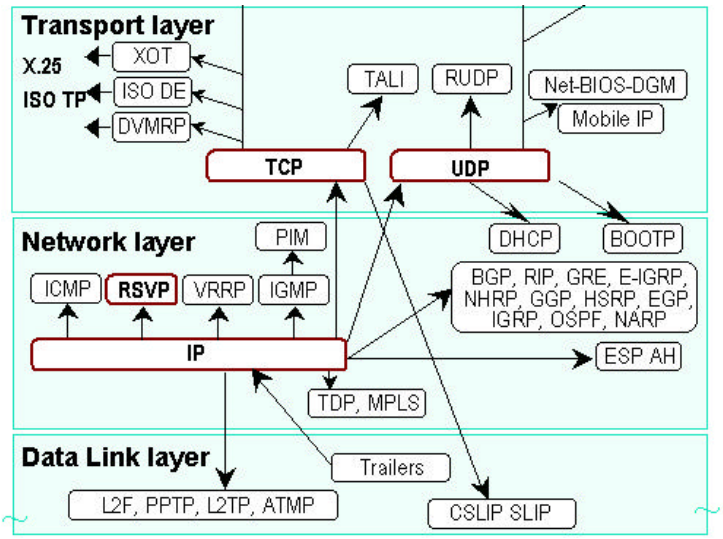

Figura 2.1 RSVP em modo *raw* – OSI-RM

Um processo RSVP consulta a base local de roteamento para obter as rotas. No caso de *multicast*, um *host* envia mensagens RSVP para juntar-se a um grupo *multicast* e, então envia mensagens RSVP para reservar recursos entre este host e o grupo *multicast*. O protocolo de roteamento determina para onde serão enviados os pacotes e o RSVP somente se ocupa com o QoS destes pacotes, que são enviados de acordo com a rota.

Para agregar eficientemente grandes grupos, membros de grupos dinâmicos e requisitos de receptores heterogêneos, o protocolo RSVP estabelece esta responsabilidade ao receptor que requisita a **reserva** de um QoS específico. A requisição da reserva de QoS de uma aplicação no *host* receptor, chamado *RECEIVER*, é passada para o processo local RSVP (*RSVP process* ou *RSVP daemon*) responsável pela sinalização através dos elementos de rede envolvidos (roteadores e *hosts*), ao longo do caminho inverso dos dados recebidos do emissor, chamado *SENDER*. Em uma comunicação *multicast*, o processamento é muito alto para reservas e cresce logaritmicamente de acordo com o número de receptores [ZHA93], [BRA97a].

2.2.1 Controle de Tráfego

A qualidade de serviço é implementada em um fluxo de dados particular através de um mecanismo chamado controle de tráfego (*TC*) que é responsável pelo controle do fluxo de pacotes nos *hosts* e roteadores, baseado na prioridade do fluxo. O mecanismo de controle de tráfego inclui: classificador de pacotes (*packet classifier*), controle de admissão (*admissions control*) e um escalonador de pacotes (*packet scheduler*), apresentado na figura 2.2.

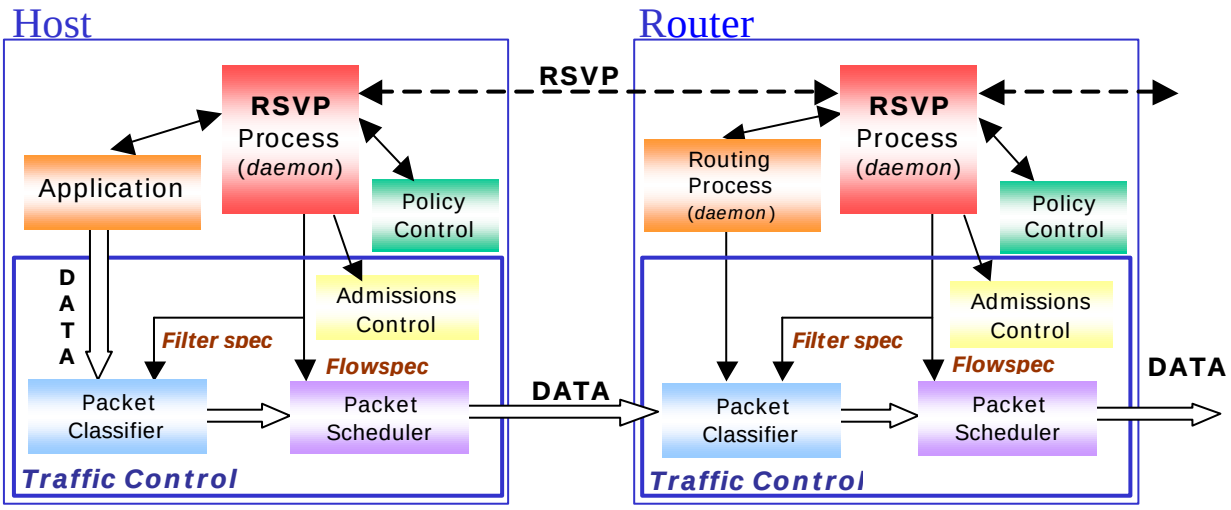

Figura 2.2 Mecanismo de Controle – *Traffic Control* (TC)

O ***packet classifier*** determina a classe de QoS e a rota para cada pacote, enquanto o ***packet scheduler*** aloca recursos para transmissão dos dados saindo da interface, utilizando um meio de enlace e que garante sua execução (*enforcement*). Se a camada de enlace possui gerenciamento próprio, o *packet scheduler* é responsável pela negociação com o enlace para obter o QoS requisitado pelo RSVP, sendo implementado em roteadores e *hosts* para controle de alocação da conexão (*link allocation*). O *packet scheduler* tem como função básica reordenar as filas utilizando técnicas de escalonamento tipo *Round-Robin* ou alguma variante como *Weighted Fair Queueing* (WFQ), utilizadas para o tipo de serviço garantido [BRA94], [CIS01a], [CIS01b], [CIS02a], [CIS02b], [CIS02d], [CIS02e].

O protocolo RSVP oferece mecanismos para criação e manutenção do estado de reservas de recursos em rotas utilizadas para comunicações *multicast* ou *unicast*. Os parâmetros de requisição de QoS e os de controle de políticas não são definidos, e controlados pelo RSVP, que os transporta e repassa aos responsáveis por interpretá-los.

Durante o processo de configuração (*setup*) de reserva de recurso, uma requisição QoS RSVP é passada para dois módulos de decisão local: o controle de política (*Policy Control*) e o controle de admissão[5] (*Admissions Control*). O módulo de **Controle de Admissão** determina quando o nó (*host* ou roteador) tem recursos disponíveis suficientes para fornecer o requerimento de QoS na interface RSVP. O mecanismo de **Controle de Política** (*Policy*

[5] O protocolo *IntServ*, uma extensão para a arquitetura IP desenvolvido pela IETF, não inclui um importante aspecto de **controle de admissão**: a habilidade de controlar, monitorar e forçar o uso de recursos de rede e serviços baseados políticas com critérios semelhantes a identificação de usuários e aplicações, requerimentos de tráfego ou largura de banda, considerações de segurança e horário de acesso.

Control) é responsável para determinar quando um usuário tem permissão administrativa para fazer a reserva, ou seja, está autorizado [BRA97a].

Nesta proposta, o mecanismo de controle de políticas RSVP solicitará autorização, em um ambiente cliente/servidor, a um servidor de políticas utilizando XACML (*eXtensible Access Control Markup Language*), desenvolvido pelo OASIS, um modelo voltado para representação de políticas descritivas. O XACML descreve uma linguagem tanto para representação de políticas, como para decisão de controle de acesso de requisição/resposta usando o XML (*eXtensible Markup Language*). O XML é uma linguagem de marcação de dados que fornece um formato para descrever dados estruturados, desenvolvido pela W3C (*World Wide Web Consortium*).

2.2.2 Fluxos de Dados – *RSVP Data Flows*

O RSVP define uma **sessão** como sendo um fluxo de dados com um destino particular e um protocolo de transporte, tratando cada sessão independentemente. A sessão é definida por endereço destino, protocolo-Id e porta destino (*DestAddress*, *ProtocoloID*, [*DstPort*]). O parâmetro opcional da porta destino (*DstPort*) é considerado uma "porta generalizada", sendo definido como uma porta em TCP/UDP, ou outro campo equivalente em outra aplicação ou protocolo de transporte. Não é estritamente necessário incluir uma porta destino na definição da sessão quando for *multicast*, desde que diferentes sessões possuam diferentes endereços *multicast*, como exemplo a figura 2.3, contudo, em *unicast* a porta destino é necessária para permitir uma ou mais sessões endereçadas para o mesmo *host* receptor, ou seja, possibilidade de comunicação de vários *SENDERS* (emissores) para um *RECEIVER* (receptor) em várias sessões. Se um *host* em *multicast* possui vários emissores, este *host* deverá utilizar portas destino generalizadas para distinguir o tráfego das sessões [BRA97a].

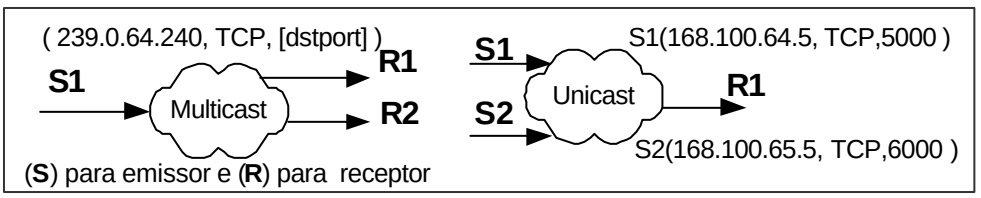

Figura 2.3 Distribuição *Multicast* e *Unicast*

2.2.3 Níveis de Serviços RSVP

A especificação do fluxo orienta os roteadores ou *hosts* como o tráfego deve ser manuseado e propicia a qualidade de serviço desejada. Os tipos de tráfego suportados pelo

RSVP são três e dependem do QoS implementado:

1. ***best-effort***: mesmo nível de serviço oferecido pelo IP tradicional.
2. ***controlled-load***: similar ao *best-effort* em uma situação com elemento de rede sem congestionamento de dados, mas usa o controle de admissão para assegurar o serviço mesmo quando o elemento de rede está em situações de congestionamento [WRO97b]. É designado para aplicações tolerantes de tempo real que requerem uma quantidade suficiente de largura de banda e podem tolerar atrasos ocasionais e perdas de pacotes.
3. ***guaranteed service***: serviço que garante a entrega de pacotes com atrasos variáveis dentro do limite máximo chamado de *maximal queuing delay* [6] devido, principalmente, em função de dois parâmetros, do mecanismo de controle de tráfego *Token Bucket* e a requisição da taxa de dados da aplicação que serão abordados nas mensagens de serviços RSVP no tópico 2.2.7.

2.2.4 Solicitação de Serviços RSVP

De acordo com a RFC2205 descrita por Braden [BRA97a], as aplicações *host RSVP*, denominadas *SENDER*, devem escolher qual o tipo de serviço mais apropriado para seus requisitos de transmissão, baseados nas características de tráfego, requisitos de performance, preferências do usuário e qualquer outro critério que defina como o dado deva ser transmitido, especificando a descrição do fluxo de dados para o processo local RSVP ou *RSVP daemon*. O *RSVP daemon* após ser invocado, é responsável pela troca de mensagens RSVP e estabelecimento da reserva de recursos nos elementos de rede envolvidos (*hosts* e *routers*) compreendidos entre o caminho do emissor e receptor.

Nesta proposta de dissertação é importante notar que a aplicação *host RSVP* não executa chamadas diretas ao *RSVP daemon*. É utilizado o mecanismo de Controle de Políticas do *host RSVP* que estende sua responsabilidade para efetivação das políticas (PEP-*Policy Enforcement Point*), e um servidor de políticas para consultas de autorização, obtenção de parâmetros de QoS e tipos de serviço (PDP-*Policy Decision Point*). A comunicação é realizada em um ambiente cliente/servidor entre o mecanismo de Controle de Políticas e um Servidor de Políticas, que descreve as classes de serviço de QoS de acordo com a aplicação

---

[6] *Queuing delay*: é proveniente de atrasos de filas na saída de **área de armazenamento** temporária de dados à espera de processamento, chamado de *buffers*, nas portas seriais. Os *buffers* criam atrasos variáveis, chamados de *jitter*, através da rede [CISCO].

utilizando XACML. É necessário, ainda, usar um protocolo transacional de políticas para troca de mensagens entre o *host RSVP* e o servidor de políticas, como COPS (*Commom Open Policy Service Protocol*).

O COPS é um protocolo de consultas e respostas que suporta de forma eficiente o controle de políticas. O COPS, PEP e PDP serão esmiuçados no capítulo 3.

2.2.5 Mensagens RSVP

O protocolo RSVP utiliza mensagens de sinalização para configuração e manutenção do fluxo de dados. Para especificar a direção do fluxo de dados (*DATA*), são definidos termos para o *RECEIVER* e *SENDER*, respectivamente: "*upstream*" e "*downstream*", "*previous hop*" (PHOP) e "*next hop*" (NHOP), "*incomming interface*" e "*outgoing interface*". Exemplificando na figura 2.4:

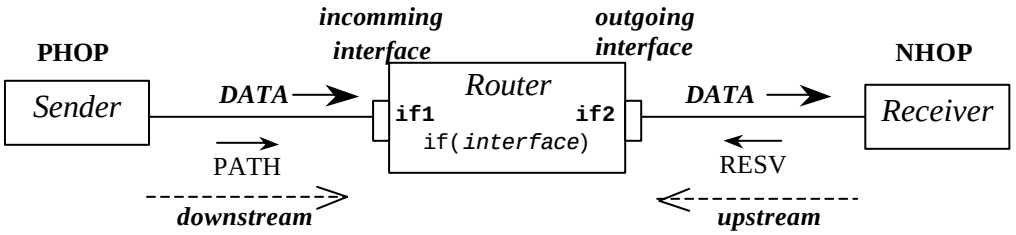

Figura 2.4 Termos para direção do fluxo de dados

As duas principais mensagens do RSVP são PATH e RESV. A comunicação sempre é iniciada pelo *SENDER* enviando a mensagem PATH ao *RECEIVER* com parâmetros de QoS. O *RECEIVER* ao receber a mensagem PATH, inicia o processo de reserva de fluxo enviando a mensagem RESV no caminho inverso do PATH até o *SENDER*. Portanto, o PATH é sempre enviado pelo *SENDER* e RESV pelo *RECEIVER*. O resumo de um processo de reserva de fluxo *simplex* é apresentado na figura 2.5 e nos passos descritos a seguir.

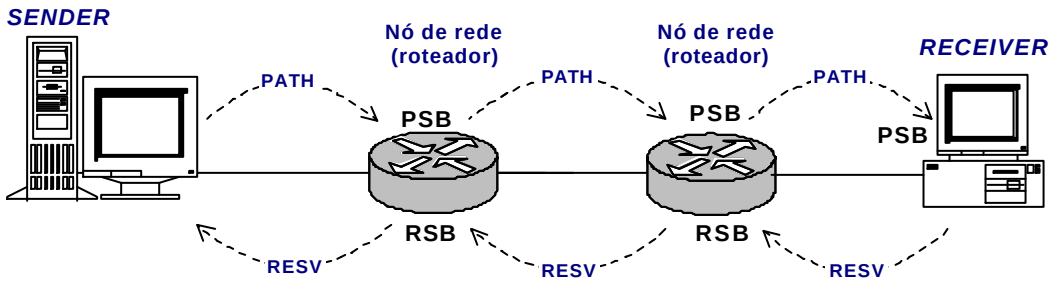

Figura 2.5 Reserva de um fluxo *simplex*

1. Quando uma conexão entre um *SENDER* e um *RECEIVER* é estabelecida, a sinalização RSVP é disparada (*triggered*) invocando o *RSVP daemon* do *SENDER*, que transmite a mensagem RSVP PATH *downstream* para comunicação *unicast* ou *multicast* através de rotas fornecidas por protocolo(s) de roteamento.
2. As mensagens PATH armazenam o "*path state block*" (PSB) em cada nó determinando a rota utilizada. O *path state block* armazena informações da sessão e do *SENDER*, especificados na RFC 2209 [BRA97b].
3. O *RECEIVER* ao receber a mensagem PATH invoca seu RSVP *daemon*, que estabelece a sessão RSVP associando o *path state block* e informações do tráfego QoS contidas na mensagem PATH, enviando a requisição da reserva de recursos através da mensagem RSVP RESV *upstream*, seguindo o caminho inverso na rede através das informações *path state block* armazenadas nos nós ao longo da rota.
4. A mensagem RESV solicita a instalação da reserva de fluxo a cada roteador que suporta RSVP. Se rejeitada pelo Controle de Tráfego, é enviada uma mensagem *RESV error* ao *RECEIVER* (*hop-by-hop*), caso contrário, instala e armazena a reserva "*reservation state block*" (RSB) seguindo ao próximo roteador, e assim sucessivamente até o *SENDER*. Como neste início de comunicação a reserva QoS ainda não está estabelecida, o controle de tráfego é configurado para transmitir o tráfego em modo "*best-effort*".
5. O *SENDER* quando recebe a mensagem RESV estabelece as comunicações com o controle de política e controle de admissão, e se for autorizado, atualiza as informações junto ao controle de tráfego (TC) que altera o fluxo atual *best-effort* para o fluxo indicado na reserva da mensagem RESV.
6. Se houver comunicações *full-duplex*, reservas separadas devem ser feitas em cada direção.
7. A manutenção da reserva do fluxo é realizada periodicamente pelo *SENDER* com mensagens PATH, com intervalos de 30 segundos, sugerida na especificação RSVP e que são correspondidas pelo *RECEIVER* RSVP RESV.
8. Caso não seja recebida a sinalização de manutenção (*Refresh*) por um erro qualquer de mensagens PATH ou RESV, o tempo expira gerando uma

interrupção (*timeout*) cancelando as reservas, PSB e RSB armazenados ao longo da rota.

9. Mensagens *PATH tear* e *RESV tear* são geradas pelo *SENDER* e *RECEIVER* para remoção de PSB e RSB, respectivamente, ou por *"timeout"* experimentado por algum roteador na manutenção de estado de uma das mensagens, PSB ou RSB.

O modelo básico de reserva do RSVP é *"one pass"* em que o *RECEIVER* envia a reserva *upstream* e cada nó entre o emissor e receptor, aceita ou rejeita a requisição. Este esquema dificulta o *RECEIVER* a descobrir o resultado de um serviço "fim-a-fim". A otimização deste modelo básico é conhecido como OPWA (*One Pass With Avertising*).

O RSVP envia a mensagem PATH *downstream* incluindo o OPWA que é atualizado a cada nó da rota com informações sobre disponibilidade de recursos. O resultado armazenado pelo OPWA é entregue ao *RECEIVER* que podem ser utilizadas para construir ou ajustar dinamicamente uma reserva apropriada.

Um resumo dos tipos, propósitos e direção das mensagens RSVP, é apresentada na tabela 2.2:

Tabela 2.2 Mensagens RSVP

Tipo da mensagem	Propósito	Direção
PATH	Instala o PSB e especificação de tráfego	*Downstream*
RESV	Requisita recursos QoS	*Upstream*
RESV *confirmation*	Envia confirmação de reserva para o RECEIVER	*Direto ao Receiver*
PATH *error*	Reporta erro no processamento do PATH	*Upstream*
RESV *error*	Reporta erro na instalação de Reserva	*Downstream*
PATH *tear*	Remove explicitamente o PSB PATH	*Downstream*
RESV *tear*	Remove explicitamente o RSB RESV	*Upstream*

2.2.6 Fluxos RSVP

Um fluxo de dados em RSVP é uma seqüência de mensagens que possuem a mesma origem, um ou mais destinos e qualidade de serviço desejada. Uma requisição de reserva de RSVP (**RESV**) consiste de um ***Flow Descriptor***, formado por um objeto ***flowspec*** e um objeto ***filter spec***, que atua no nó RSVP, *host* ou *router*. O ***filter spec*** junto com a especificação da sessão, definem os pacotes de dados (fluxo RSVP) para receberem o QoS. A especificação de QoS desejado é definida pelo ***flowspec***.

O ***flowspec*** é usado para configurar parâmetros no *Packet Scheduler* ou outro mecanismo de camada de enlace. O ***filter spec*** é utilizado para configurar o *Packet Classifier*. Os Pacotes de dados que são enviados a uma determinada sessão e que não se equiparam ao ***filter spec*** correspondente, são tratados como tráfego *best-effort*. A figura 2.6 apresenta o ***Flow Descriptor*** e seus objetos atuando sobre o controle de tráfego em um nó RSVP.

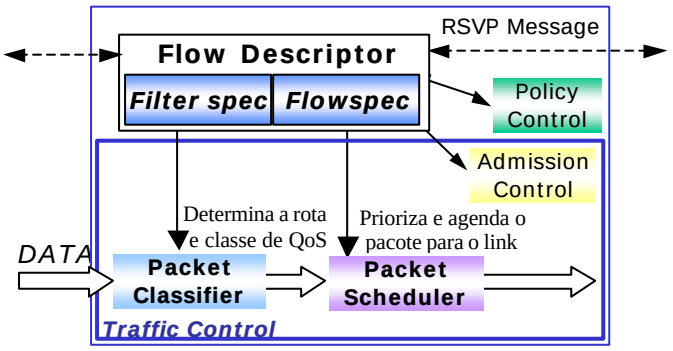

Figura 2.6 Processo de reserva de recursos – RSVP

O ***flowspec*** geralmente inclui uma classe de serviço e dois conjuntos de parâmetros numéricos, um é o ***Rspec*** (*Request Specification*) que descreve o nível de QoS desejado para um fluxo requisitado a um *router* ou *host* e, o outro é ***Tspec*** (*Traffic Specification*) que descreve o tráfego de dados para qual o serviço está sendo requisitado e que será reservado pelo *router* ou *host*, ambos definidos na **RFC 2210** [WRO97a] e não são interpretados pelo RSVP.

A estrutura e as informações contidas no objeto ***Tspec*** são utilizadas em ambos serviços *Guaranteed* e *Controlled-Load*. Para reserva RSVP RESV *Controlled-load* o ***flowspec*** define somente a classe de serviço ***Tspec***, visto que o nível de serviço não especifica as características de atraso para o fluxo. Quando solicitado um nível de serviço *Guaranteed*, este deve definir características de atraso do fluxo no objeto ***Rspec*** em adição ao objeto ***Tspec*** no ***flowspec*** da mensagem RESV, representadas na figura 2.7.

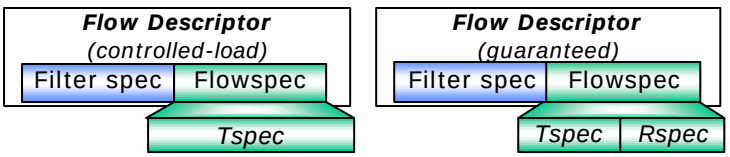

Figura 2.7 Descrição do fluxo de dados para Serviço Controlado e Garantido.

2.2.7 Especificação funcional das mensagens RSVP

Uma mensagem RSVP consiste de um cabeçalho comum (*common header*) as mensagens RSVP, seguido de uma seqüência de objetos que compõem um tipo de mensagem RSVP, apresentada na figura 2.8.

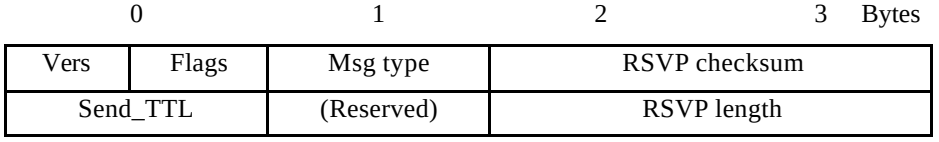

Figura 2.8 Formato do cabeçalho RSVP (*RSVP header*)

Vers (4 bits): número da versão RSVP;

Flags (4 bits): vários flags – (especificado na RFC2205);

Msg type (8 bits): mensagem RSVP (descrito em **2.2.5, tabela 2**);

1= **PATH**; 2=**RESV**; 3=**PATHErr**; 4= **RESVErr**;

5= **PATHTear**; 6=**RESVTear**; 7=**RESVConf**

RSVP Checksum (16 bits): verificação de erro da mensagem;

Send_TTL (8 bits): RSVP pode detectar nó de rede que não seja RSVP-*Aware* comparando este campo com o valor TTL (*TimeToLive*) quando receber a mensagem;

RSVP Length: o tamanho da mensagem RSVP em bytes, incluindo o cabeçalho e os objetos que a compõem;

2.2.7.1 Formato dos Objetos RSVP

Um objeto contém campos e são requeridos para descrição de uma mensagem. O objeto **NULL** tem a classe número 0 e o conteúdo é ignorado pelo receptor. O formato e descrição do cabeçalho do objeto são apresentados na figura 2.9.

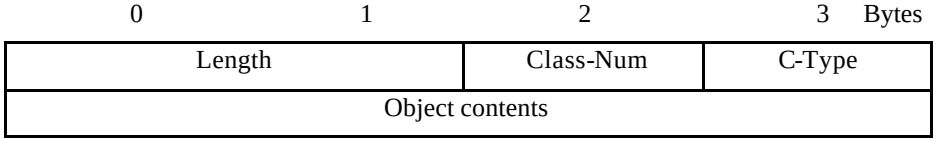

Figura 2.9 Formato de um objeto RSVP

Length (16 bits): tamanho do objeto em *bytes* e deve ser múltiplo de 4 e no mínimo com tamanho de 4 bytes;

Class-Num (8 bits): identifica a classe do objeto:

01=**SESSION** 03=**RSVP_HOP** 04=**INTEGRITY** 05=**TIME_VALUES**
06=**ERROR_SPEC** 07=**SCOPE** 08=**STYLE** 09=**FLOWSPEC**
10=**FILTER_SPEC** 11=**SENDER_TEMPLATE** 12=**SENDER_TSPEC** 13=**ADSPEC**
14=**POLICY_DATA** 15=**RESV_CONFIRM**

C-Type (8 bits): tipo do objeto e único dentro do *Class-Num*;

Object contents: conteúdo do objeto sendo limitado ao tamanho de 65.528 bytes;

Existem 14 classes de objetos definidos no padrão RSVP que são utilizados pelas Mensagens RSVP de acordo com as definições descritas por *Braden* [BRA97a]. Um resumo das funcionalidades é apresentado na tabela 2.3:

Tabela 2.3 Tipos de Classes e Objetos

Nº da Classe / Nome do Objeto	**Descrição**
01=**SESSION**	Contém o endereço IP do destino, *protocolID* {TCP=6, UDP=17} e a porta do destino. Especifica a sessão para o objeto que segue e requerido em todas mensagens RSVP.
03=**RSVP_HOP**	Transporta o endereço IP do nó RSVP-*Aware* de um PHOP (*previous hop*) para mensagem *downstream* ou NHOP(*next hop*) para mensagem *upstream*.
04=**INTEGRITY**	Utilizado para autenticação do conteúdo da mensagem RSVP utilizando criptografia. Pode ser especificado nas mensagens.
05=**TIME_VALUES**	Contém o valor do período de atualização (manutenção). Requerido em cada mensagem PATH e RESV.
06=**ERROR_SPEC**	Especifica um erro em um PATH *error*, RESV *error* ou em uma confirmação RESV *confirmation*.
07=**SCOPE**	Transporta explicitamente uma lista de SENDERS Hosts em mensagens *upstream*. Pode ser usado em mensagens RESV, RESV *error* ou RESV *tear*, quando usados em multicast.
08=**STYLE**	Define a reserva de estilo.Requerida a cada mensagem RESV.
09=**FLOWSPEC**	Define um QoS desejado em uma mensagem RESV.
10=**FILTER_SPEC**	Usado pelo *RECEIVER* para identificar um fluxo de um *SENDER* para receber um QoS. Muito útil em comunicação *multicast* que possui vários *SENDERS*. Contém o endereço IP e porta do *SENDER*. A especificação de QoS é definida pelo ***flowspec***.
11=**SENDER_TEMPLATE**	Contém o endereço IP e porta do *SENDER* e assume o *protocolID* especificado na sessão {TCP=6,UDP=17}. Identifica o *SENDER*.
12=**SENDER_TSPEC**	Define as características do tráfego do fluxo de dados que o *SENDER* irá gerar.
13=**ADSPEC**	Pacote OPWA (*One Pass With Avertising*) conhecido como *Adspec*. São pacotes de controle passados ao Controle de Tráfego

	local a cada nó de rede *QoS-Aware*, que retorna como um *Adspec* atualizado. Esta versão atualizada segue no PATH downstream até o *RECEIVER*. Os resultados do *"avertising"* (anúncio) são usados para construir ou ajustar dinamicamente a mensagem RSVP RESV.
14=**POLICY_DATA**	Informações para uso do módulo de política local. Pode ser definido em mensagens: PATH, RESV, PATH *error* ou RESV *error*.
15=**RESV_CONFIRM**	Transporta o endereço IP do *RECEIVER* para requisitar a confirmação da instalação da reserva. Pode ser usada pelas mensagens RESV ou RESV *confirmation*.

Os campos *"**Class-Num**"* e *"**C-Type**"* podem ser utilizados em conjunto para definir um tipo único para cada objeto. O objeto **POLICY_DATA** não está especificado na RFC 2205. A extensão deste objeto é descrita na RFC 2750 [HER00b], para ser utilizado pelo módulo de controle de política. O **POLICY_DATA** não é utilizado nesta proposta porque a decisão não é tomada pelo controle de política local, mas pelo servidor de políticas.

Os objetos das classes: 09=**FLOWSPEC**, 12=**SENDER_TSPEC**, 13=**ADSPEC** são definidos na RFC 2210 [WRO97a] e suas definições são caracterizadas na RFC 2215 [SHE97b], e não são interpretados pelo protocolo RSVP da RFC 2205, mas utilizados para a sinalização em uma reserva de fluxo. Serão detalhados na seqüência.

O protocolo IP *Security* (IPSEC) suporta autenticação de pacotes descrita na RFC 1826 **[**ATK95a**]** e integridade e confidencialidade descritas na RFC 1827 **[**ATK95b**]**, através da adição de um novo cabeçalho entre a camada de rede IP e a camada de transporte TCP ou UDP. A utilização de IPSEC com RSVP foi especificada, uma vez que os objetos **SESSION**, **FILTER_SPEC**, **SENDER_TEMPLATE** utilizam dados contidos nos cabeçalhos, que sofrem alterações pelo IPSEC. As extensões dos objetos RSVP citados acima e seus respectivos conteúdos são descritos na RFC 2207, mapeando os novos cabeçalhos IPSEC.

2.2.7.2 Composição de uma mensagem

O formato final de uma mensagem RSVP depende do seu tipo (***Msg type***), especificado no *RSVP header*, seguido dos objetos que compõem esta mensagem (*RSVP Objects*), conforme mostra a figura 2.10. Os formatos dos objetos e seus detalhes estão descritos na RFC2205.

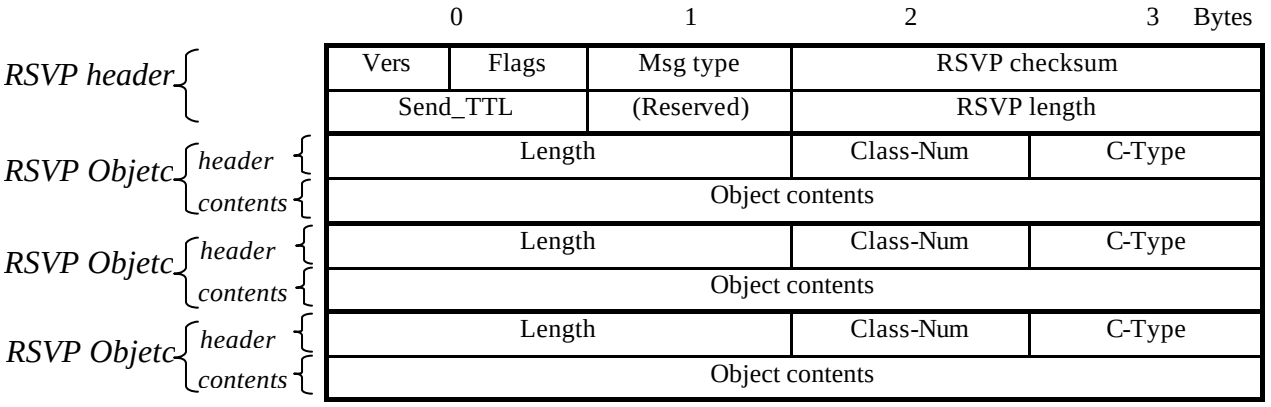

Figura 2.10 Composição de uma mensagem RSVP

2.2.7.3 Formato da Mensagem PATH

A mensagem PATH é formada pelos objetos apresentados na figura 2.11, que foram descritos na tabela 2.3.

PATH Message	**RSVP header**
	[INTEGRITY object]
	SESSION object
	RSVP_HOP object
	TIME_VALUES object
	[POLICY_DATA object…]
	SENDER_TEMPLATE object
	SENDER_TSPEC object
	[ADSPEC object]

Figura 2.11 Formato da mensagem PATH

Informação de objeto entre colchetes significa que pode estar ou não presente. O POLICY_DATA pode conter mais de um objeto.

Uma mensagem RSVP **PATH** e seus objetos é apresentada de forma generalizada na figura 2.12:

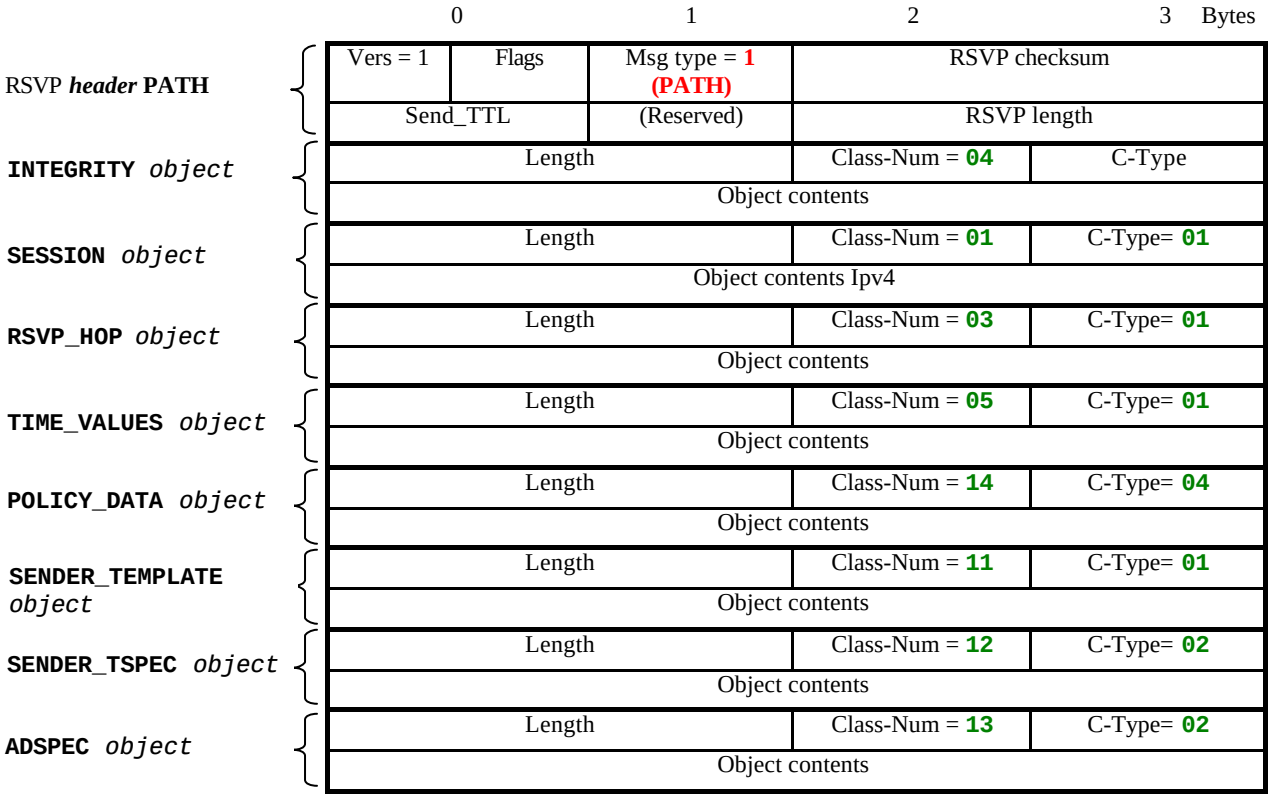

Figura 2.12 Formato da mensagem PATH com objetos

Objeto RSVP SESSION

O objeto Classe 01 "**SESSION**" é requerido para **todas mensagens** RSVP. Este objeto descreve no seu conteúdo os dados: <IP do destino, *protocolID* e porta do destino>. Suporta os protocolos IPv4 (**`C-Type=1`**) e IPv6 (**`C-Type=2`**). O protocolo utilizado TCP=6 ou UDP=17, para um fluxo de dados a ser estabelecido é definido pelo parâmetro *ProtocolID*. O objeto **SESSION** é apresentado na figura 2.13, como exemplo, com a descrição para IPv4 e UDP.

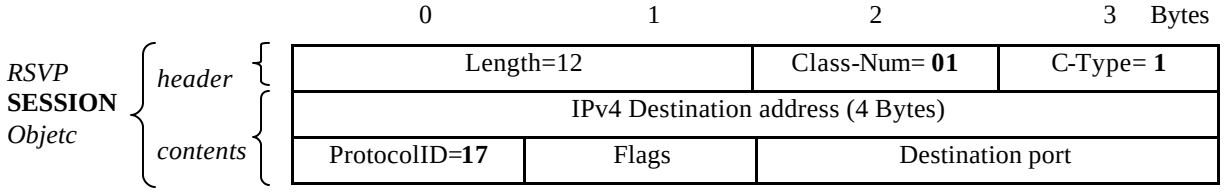

Figura 2.13 Objeto de Sessão IPv4 UDP

Objeto RSVP SENDER_TSPEC

O objeto RSVP **SENDER_TSPEC** contém um cabeçalho e no seu conteúdo as especificações ***Tspec***, para definição de um fluxo QoS, que serão obtidos do servidor de políticas usando XACML, através da consulta do mecanismo de controle de políticas do *Host*

RSVP. O objeto RSVP **SENDER_TSPEC** é apresentado na figura 2.14.

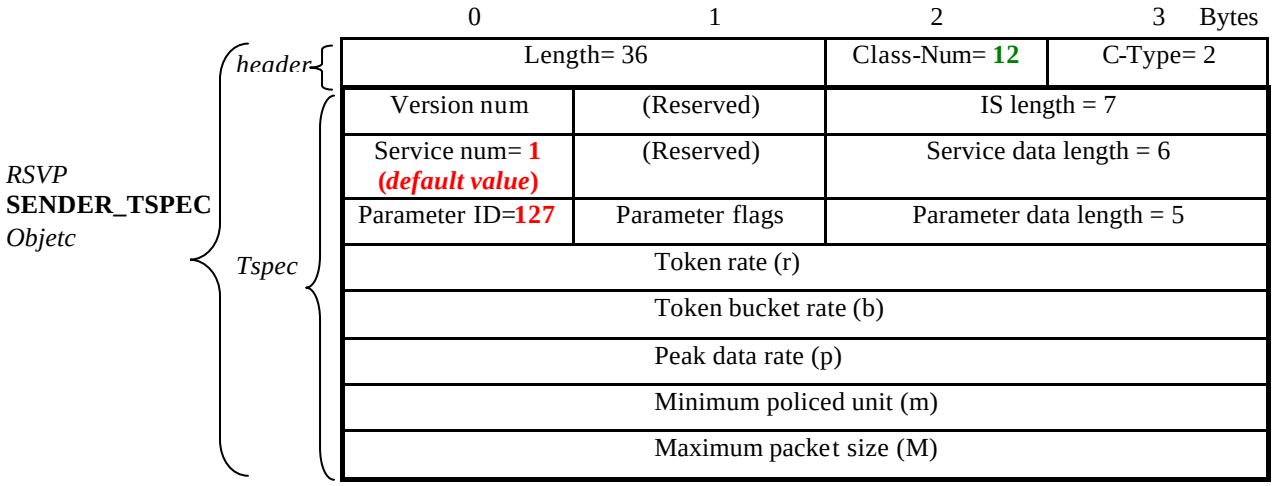

Figura 2.14 Objeto RSVP **SENDER_TSPEC**

Os números de serviços (*"service_number"*) são utilizados para alocação de serviços individuais de QoS. O número "**1**" indicado em <*service_number*> é sempre um serviço padrão e o número "127" em <*parameter_ID*> determina um *"Token_Bucket_TSpec"*, segundo RFC 2215.

O *Tspec*, encapsulado pelo objeto RSVP SENDER_TSPEC, define o tráfego para um fluxo do *SENDER* (emissor) para o *RECEIVER*, com os seguintes parâmetros:

***Token rate* (*r*)** (*bytes/second*): especifica a taxa que os *tokens* chegam no *token bucket*;

***Token bucket depth* (*b*)** (*bytes*): o tamanho do *token bucket*.

***Peak data rate* (*p*)** (*bytes/second*): taxa máxima que uma origem pode transmitir em rajadas de dados na rede (*burst traffic*).

Minimum policed unit* (*m*)** (*bytes/second*): qualquer pacote com um tamanho menor que 'm***' será considerado com '***m***' *bytes*.

Maximum packet size* (*M*)** (*bytes/second*): especifica o tamanho máximo de um pacote que pode ser aceito, ou seja, o valor ***M representa o maior pacote que o *SENDER* pode enviar.

Token Bucket

O mecanismo *Token Bucket* é um algoritmo para regular tráfego de dados que controla

e limita a largura de banda, utilizando os parâmetros contidos no ***Tspec***. Possui dois parâmetros: a taxa que gera os *tokens* é chamada de **(*r*)**, e o tamanho (profundidade) onde se armazenam os *tokens* é chamado de **(*b*)**. Metaforicamente, *Token Bucket* é comparado a um balde que possui uma profundidade **(*b*)** e uma torneira que controla **(*r*)** a capacidade de escoamento a uma taxa constante. A figura 2.15 apresenta um *Token Bucket* com taxa **(*r*)** e profundidade **(*b*)**.

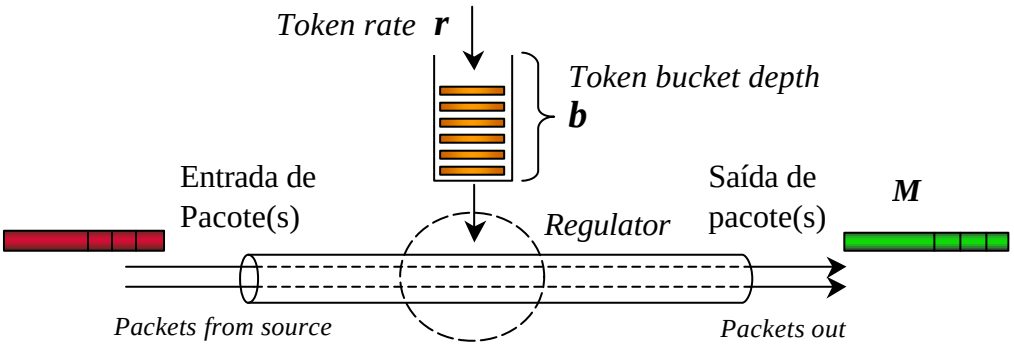

Figura 2.15 Mecanismo *Token Bucket* com taxa ***r*** e profundidade ***b***

Os *tokens* são gerados a uma taxa constante ***r***, sendo armazenados no balde (*token bucket depht*), e são consumidos a medida que chegam os pacotes. A profundidade ***b*** é o limite da quantidade máxima de *tokens* que pode ser armazenada. Os *tokens* gerados em excesso são descartados. Considera-se ***M*** o tamanho máximo do pacote. Para enviar um pacote de dados, um "regulador" (*regulator*) remove do balde um número de *tokens* equivalentes ao tamanho dos pacotes (***M***) que irão sair do regulador de tráfego, seguindo sua rota.

A quantidade de pacotes que chegam ao mecanismo e excedem o número de *tokens* disponíveis, são tratados de acordo com o tipo de controle de tráfego utilizado, como o *traffic policing* ou o *traffic shaping* que serão apresentados na seqüência.

Algumas propriedades do mecanismo regulador do *Token Bucket* [RFC2212] são:

1. O número total de bits que o *Token Bucket* permite enviar é limitado por uma função linear. Uma quantidade de bits considerada como $s(t)$ e transmitida em um intervalo de tempo **T**, define a equação: $\boxed{s(t) \leq (r \times T) + b}$;
2. A quantidade máxima de pacotes de uma rajada em uma taxa de pico ***p*** (*burst size*) que chegam ao regulador não pode ser maior que a 'profundidade' ***b***, então: $\boxed{s < b}$;

3. O *Data Sent* que é a quantidade de dados enviados pelo regulador, não pode exceder o limite a equação abaixo, onde ***M*** é tamanho máximo do pacote, **T** um período de tempo e ***p*** a taxa de pico:

$$\boxed{Data\ \ Sent \leq (M + pT)\ \text{ sendo }\ (p \geq r)}$$

Controle de tráfego

Os mecanismos de controle de tráfego utilizam o *Traffic Policing* nos *hosts SENDER / RECEIVER* e *Traffic Shaping* nos roteadores (nós). Ambos mecanismos utilizam o *Token Bucket*.

O *Traffic Policing* atua quando o tráfego atinge a taxa máxima configurada, descartando ou remarcando o pacote, desta forma, impede atrasos pela característica de evitar o atraso em filas pelo uso de *buffers*. São requeridos os parâmetros do *Token Bucket* {***r***, ***b***, ***p***, ***M***} para que o *Data Sent* não exceda o limite segundo a equação abaixo:

$$\boxed{Data\ \ Sent \leq M + min(pT, rT + b - M)}$$

O *Traffic Shaping* atua de forma diferente. O tráfego excedido em relação ao *Tspec* é retido em um *buffer*, e este excesso é programado para uma transmissão posterior denominada *Reshaping*, que aumenta o tempo de atraso do pacote até o destino.

A utilização do modelo *Shaping* implica na existência de uma fila e memória suficiente para armazenar pacotes atrasados ou ainda em trânsito na rede. Quando os pacotes não excedem os limites estabelecidos pelo ***Tspec*** é necessária à existência do *buffer* para aguardar o último bit do pacote chegar antes de possibilitar o seu envio adiante, o que adiciona um atraso fixo.

Para evitar perda de pacotes, o espaço do *buffer* representado por **B** é obtido pela combinação dos parâmetros do *Token Bucket*{***r***,***b***} e taxas de erro de termos de roteadores $\{C_{sum}, D_{sum}\}$[7], definido pela equação a seguir:

$$\boxed{B = b + C_{sum} + (D_{sum} \times r)}$$

Caso seja considerado a taxa de pico ***p***, o *buffer* **B** pode ter o seu espaço reduzido pelo fato de ***p*** limitar a taxa de rajada gerada por ***b*** na rede, descrito no tópico *'Guidelines for*

[7] C_{sum}, D_{sum}: representam a soma parcial entre pontos de armazenamento de pacotes, ou seja, desde o último *shaping points* de taxas de erro de de um termo dependente C e um termo independente D.

Implementators' na RFC 2212.

Para garantia de serviços em um fluxo de dados somente deve ser utilizado o *Policing* nas extremidades da rede e o *Shaping* nos pontos intermediários.

O *token bucket* com uma área de armazenamento para pacotes, *buffer*, é apresentado na figura 2.16.

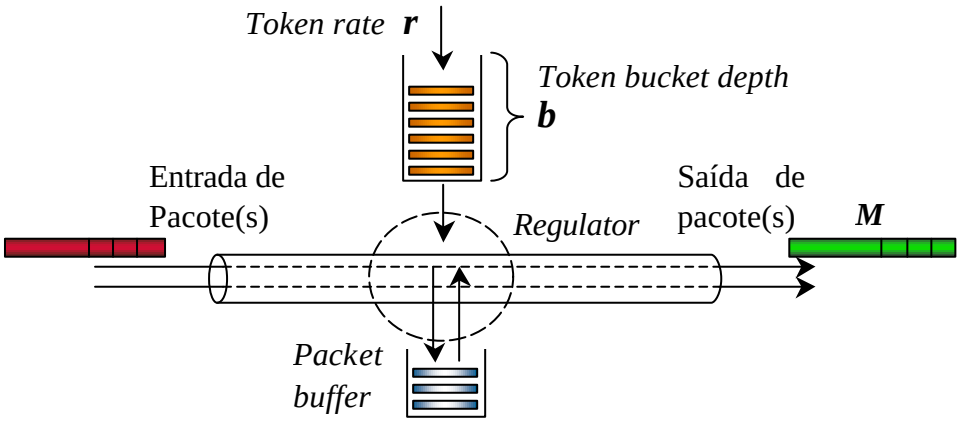

Figura 2.16 Shaping utilizando Token Bucket

Objeto Adspec

O objeto *Adspec* é transportado na mensagem PATH e contém informações que são geradas em cada nó de rede *RSVP aware* ao longo da rota, sendo usadas ou modificadas pelos nós, antes do objeto ser entregue ao *RECEIVER*. As informações recebidas pelo *Adspec* podem ser utilizadas pelo *RECEIVER* para determinar se há disponibilidade de serviço e as capacidades mínimas oferecidas pelos nós até o *SENDER*.

O objeto *Adspec* especifica um cabeçalho e três blocos funcionais chamados fragmentos, sendo: 1) fragmento de caracterização geral; 2) fragmento de serviço *Controlled load* e 3) fragmento de serviço *Guaranteed*, sendo apresentado o cabeçalho e o formato na figura 2.17.

3	2	1	0 Bytes
Vers =0	(Reserved)	Message length	
Default General Parameters fragment (Service **1**) (**Always Present**)			
Guaranteed Service Fragment (Service **2**)			
Controlled-Load Service Fragment (Service **5**)			

Figura 2.17 Formato RSVP ADSPEC

O serviço padrão é também denominado de *"override values"* e um elemento de rede

utiliza o valor do serviço especificado, se este existir, caso contrário usa-se o valor padrão. Para *Guaranteed Service* é alocado o número "**2**" e *Controlled-Load* o número "**5**", descritas na RFC 2215.

Um *SENDER* pode indicar quais os tipos de serviços que não serão oferecidos aos *RECEIVERS* em uma sessão RSVP, omitindo os fragmentos dos serviços. A ausência completa no ***Adspec*** de um fragmento de um serviço específico indica ao *RECEIVER* que este serviço não deverá ser utilizado. Em uma sessão com vários *RECEIVERS*, todos devem usar o mesmo tipo de serviço. Esta restrição é imposta pela dificuldade de intercalar várias requisições de reservas com diferentes tipos de serviços QoS, descrita na sessão 2.2 da RFC 2210.

Default General Parameters fragment: Este fragmento padrão sempre está presente no objeto e o serviço especificado é o número "**1**", apresentado na figura 2.18:

Service num= **1**	x	Reserved	Length block – header word= **8**	(8)
Param ID= 4		flag byte	Length=1	1
NUMBER_OF_IS_HOPS				2
Param ID= 6		flag byte	Length=1	3
AVAILABLE_PATH_BANDWIDTH				4
Param ID= 8		flag byte	Length=1	5
MINIMUM_PATH_LATENCY				6
Param ID= 10		flag byte	Length=1	7
PATH_MTU				8

Figura 2.18 Fragmento de padrão geral de parâmetros

Todos os parâmetros descritos abaixo, caracterizados na RFC 2215, são identificados pela composição **<*Serv_Num*,*Param_Number*>**.

NON_IS_HOP (x) <**1**,2>: indica a existência de um elemento de rede que não suporta o serviço de QoS caso o valor de ***x***=1, tipo lógico (*boolean*: 0 ou 1). O parâmetros ***x*** é definido no cabeçalho do fragmento.

NUMBER_OF_IS_HOPS <**1**,4>: número de elementos de rede ao longo da rota, em conformidade com o RSVP;

AVAILABLE_PATH_BANDWIDTH <**1**,6>: maior taxa possível de dados na rota que o *RECEIVER* pode receber;

MINIMUM_PATH_LATENCY **<1,** 8>: atraso mínimo que um *RECEIVER* pode esperar ao longo da rota. Aumenta a cada dispositivo na rota para refletir o atraso total, ou seja, é um **atraso fixo** introduzido no processamento de pacotes de cada dispositivo e/ou atraso de propagação experimentados na rede;

PATH_MTU **<1,** 10>: unidade de transmissão máxima suportada na rota, passado ao serviço de QoS, para limitar o tamanho do pacote, ou seja, o parâmetro ***M*** definido no ***Tspec***;

Guaranteed Service Fragment: quando utilizado, é definido o serviço número "**2**" e fornece informações sobre o atraso ao longo da rota. O fragmento *Guaranteed* com o fragmento padrão, são apresentados na figura 2.19, e neste exemplo, o *SENDER* não oferece o serviço *Controlled-Load*.

3 2 1 0 Bytes

Vers =0	**Reserved**		**Message length – header word = 18**	*header* ← (18)
Service num= **1**	x	Reserved	Length block –header word= **8**	← (8) 1
Param ID= 4	Flag byte		Length=1	1 2
NUMBER_OF_IS_HOPS				2 3
Param ID= 6	Flag byte		Length=1	3 4
AVAILABLE_PATH_BANDWIDTH				4 5
Param ID= 8	Flag byte		Length=1	5 6
MINIMUM_PATH_LATENCY				6 7
Param ID= 10	Flag byte		Length=1	7 8
PATH_MTU				8 9
Service num= **2**	x	Reserved	Length block –header word = **8**	← (8) 10
Param ID= 133	Flag byte		Length=1	1 11
End-to-end composed value for C [Ctot]				2 12
Param ID= 134	Flag byte		Length=1	3 13
End-to-end composed value for D [Dtot]				4 14
Param ID= 135	Flag byte		Length=1	5 15
Since-last-reshaping point composed C [Csum]				6 16
Param ID= 136	Flag byte		Length=1	7 17
Since-last-reshaping point composed D [Dsum]				8 18

Figura 2.19 Fragmento do serviço *Guaranteed*

Segue a descrição dos campos do fragmento *Guaranteed*, serviço número "**2**":

***X* (*Guaranteed Service Break Bit*)** **<2,** 2>: cada elemento de rede (nó) informa para

o fragmento *Guaranteed* se não o suporta, apesar de poder suportar outro tipo de serviço QoS. Se ***x*** =1, tipo lógico (*boolean*: 0 ou 1), o *RECEIVER* recebe informação que não terá serviço garantido em algum nó ao longo da rota até o *SENDER*. O parâmetro ***x*** é definido no cabeçalho do segmento *Guaranteed*.

Ctot <2,133>: a soma total de ***C*** ao longo da rota, ou seja, a soma total de uma taxa de erro de um termo dependente ***C*** que representa o atraso contribuido pelos valores dos parâmetros ***Tspec*** e o tamanho dos pacotes;

Dtot <2,134>: a soma total de ***D*** ao longo da rota, ou seja, a soma total de uma taxa de erro de um termo independente ***D*** que representa o atraso nos *buffers* (atraso variável) dos nós de rede;

Csum <2,136>: a soma parcial de ***C*** entre pontos de armazenamento de pacotes, ou seja, desde o último *shaping points*. O *Shaping* é um controle de política utilizado em nós de rede descrito no tópico ***Token Bucket***.

Dsum <2,136>: a soma parcial de ***D*** entre pontos de armazenamento de pacotes, ou seja, desde o último *shaping points*;

C <2,131>: representa uma taxa de erro de um termo dependente, medida em bytes. São utilizados nos roteadores para cálculo de atraso.

D <2,132>: representa uma taxa de erro de um termo independente, medida em microsegundos. São utilizados nos roteadores para cálculo de atraso.

Controlled-Load Service Fragment: quando utilizado, é definido o serviço número "5" no cabeçalho deste fragmento sem parâmetros adicionais no conteúdo do objeto, apresentado na figura 2.20. Este fragmento acompanha o fragmento padrão, ou seja, *Default General Parameters fragment*.

Service num= 5	x	Reserved	Length block –header word= **0**

Figura 2.20 Fragmento do serviço *Controlled-Load*

***X* (*Break bit*)** <5, 2>: definido no cabeçalho do fragmento e anuncia ao destino, que o serviço *Controlled-Load* é suportado ao longo da rota caso o seu valor seja igual a "0", tipo lógico (*boolean*: 0 ou 1);

Comportamento de atraso "end-to-end"

I - Modelo "*fluid model*"

O modelo conhecido como "*fluid model*" concebe um serviço com uma taxa ***R*** (*rate R*), considerando que um serviço deveria ser fornecido através de uma linha dedicada com largura de banda ***R*** (*bandwidth R*) entre um emissor e um receptor, definidos no tópico '*End-to-End Behavior*' na RFC 2212.

Neste modelo com taxa ***R*** fixa, o serviço para um fluxo é completamente independente de qualquer outro fluxo, ou seja, não há compartilhamento no meio físico. Portanto, o atraso variável máximo ou "*end-to-end worst-case queuing delay*" para reserva de um fluxo é determinado por valores do ***token bucket*** e taxa ***R***, definido na equação a seguir:

$$Delay_{max} = \frac{b}{R} \quad \text{sendo} \quad (p \to \infty \quad \text{e} \quad R \geq r)$$

Caso o "*peak rate*" ***p*** seja comparável aos valores de ***R*** e ***r***, então:

$$Delay_{max} = \frac{b \times (p - R)}{R \times (p - r)} \quad \text{sendo} \quad (p \to \infty \quad \text{e} \quad R \geq r)$$

II -Atraso variável no Elemento de Rede

Um roteador deve assegurar um serviço similar ao modelo "*fluid model*", considerando que o seu atraso variável não exceda o referido modelo, sendo que ***C*** e ***D*** descrevem o atraso variável local do roteador. Portanto, um roteador (nó ou elemento de rede) DEVE assegurar que o atraso de qualquer datagrama seja menor que o valor determinado pela equação 01:

(Equação 01) $$Delay_{\max} < \frac{b}{R} + \frac{C}{R} + D \quad \text{sendo } R \geq r$$

Um atraso entre um emissor e um receptor denominado *end-to-end delay*, utilizando o modelo "*fluid model*", sendo que $\boldsymbol{C}_{tot}$ e $\boldsymbol{D}_{\boldsymbol{tot}}$ descrevem o atraso variável total dos roteadores, é determinado pela equação 02:

(Equação 02) $$Delay_{max} < \frac{b}{R} + \frac{C_{tot}}{R} + D_{tot} \quad \text{sendo} \quad R \geq r$$

III- Atraso variável máximo "*end-to-end*"

O comportamento do atraso máximo entre um *SENDER* e um *RECEIVER*, com

elementos de rede ao longo da rota, uma taxa ***p*** e uma política para controle de fluxos que assegurem um serviço com limite máximo de atraso, é definido a seguir, considerando não haver falhas de componentes de rede ou mudança de rota em uma reserva de fluxo.

O atraso variável máximo ("*maximum end-to-end queuing delay*") é caracterizado por C_{tot}, D_{tot} e a largura de banda caracterizada por ***R***. A definição do atraso neste caso é determinada pela equação 03:

(Equação 03) $$Delay_{max} = \left[\frac{(p-M)}{R} \times \frac{(p-R)}{(p-r)}\right] + \frac{(M+C_{tot})}{R} + D_{tot} \quad \text{sendo } (p > R \geq r)$$

Se considerar na equação acima $p = R$, isto significa que não existe atraso por taxa de pico, não havendo necessidade para armazenar dados no *buffer* e usar *traffic shaping*, o atraso variável máximo será determinado pela equação 04.

(Equação 04) $$Delay_{max} = \frac{(M+C_{tot})}{R} + D_{tot} \quad \text{sendo} (r \leq p \leq R)$$

Estes parâmetros fornecidos pelo ***Adspec***, quando disponíveis, geralmente são utilizados para definir o ***Rspec*** na mensagem RESV, para serviço *Guaranteed*. O ***Rspec*** será descrito no tópico a seguir.

Como o atraso variável depende dos parâmetros do ***Token Bucket*** e da requisição da taxa de dados de aplicação chamada ***R***, estes valores estão sob o controle da aplicação, que pode estimar qual será o valor máximo do atraso variável que o serviço *Guaranteed* pode oferecer.

Para atingir um atraso com um valor baixo, a aplicação pode alterar os valores do *token bucket* e a taxa de dados ***R***. Como muitas aplicações têm requerimentos conhecidos para o ***token bucket*** **(*r*,*b*)**, a aplicação pode aumentar a taxa ***R*** para diminuir o atraso.

2.2.7.4 Formato da Mensagem RESV

A mensagem RESV é formada por objetos descritos na figura 2.21 que foram descritos na tabela 2.3.

RESV Message	**RSVP header**
	[INTEGRITY object]
	SESSION object
	RSVP_HOP object
	TIME_VALUES object
	[RESV_CONFIRM object]
	[SCOPE object]
	[POLICY_DATA object]
	STYLE object
	FLOWSPEC object
	FILTER_SPEC object

Figura 2.21 Formato da mensagem RESV

Informação de objeto entre colchetes significa que pode ou não estar presente.

Uma mensagem RSVP RESV deve apresentar no seu objeto **FLOWSPEC** a especificação do fluxo QoS, a ser instalado nos nós de rede e no *SENDER* através da requisição RESV. O formato do Flowspec é diferente quando usados os serviços *Guaranteed* e *Controlled-Load*, citados no item **2.2.6**.

O formato para um fluxo *Controlled-Load* tem somente a especificação do ***Tspec***, e a mensagem RSVP RESV obtém o *Tspec* da mensagem PATH. A figura 2.22 apresenta um objeto **FLOWSPEC** para o serviço citado.

RSVP
FLOWSPEC
Objetc

	0	1	2	3 Bytes
header	Length= 36		Class-Num= **9**	C-Type= **2**
Tspec	Version num	(Reserved)	IS length = 7	
	Service num= **5** ***(controlled- load)***	(Reserved)	Service data length = 6	
	Parameter ID=127	Parameter flags	Parameter data length = 5	
	Token rate (r)			
	Token bucket rate (b)			
	Peak data rate (p)			
	Minimum policed unit (m)			
	Maximum packet size (M)			

Figura 2.22 Objeto **FLOWSPEC** para especificação *"Controlled-Load"*

O formato para um fluxo *Guaranteed* deve ter a especificação do *Tspec* e o *Rspec* na mensagem RSVP RESV, que obtém o ***Tspec*** da mensagem PATH e ***Rspec***, calculados pelas informações do objeto ***Adspec***. Caso a mensagem PATH não especifique o objeto Adspec, o *RECEIVER* deverá definir o ***Rspec*** para utilização de serviço garantido. A figura 2.23

apresenta um objeto **FLOWSPEC** para o serviço *Guaranteed*.

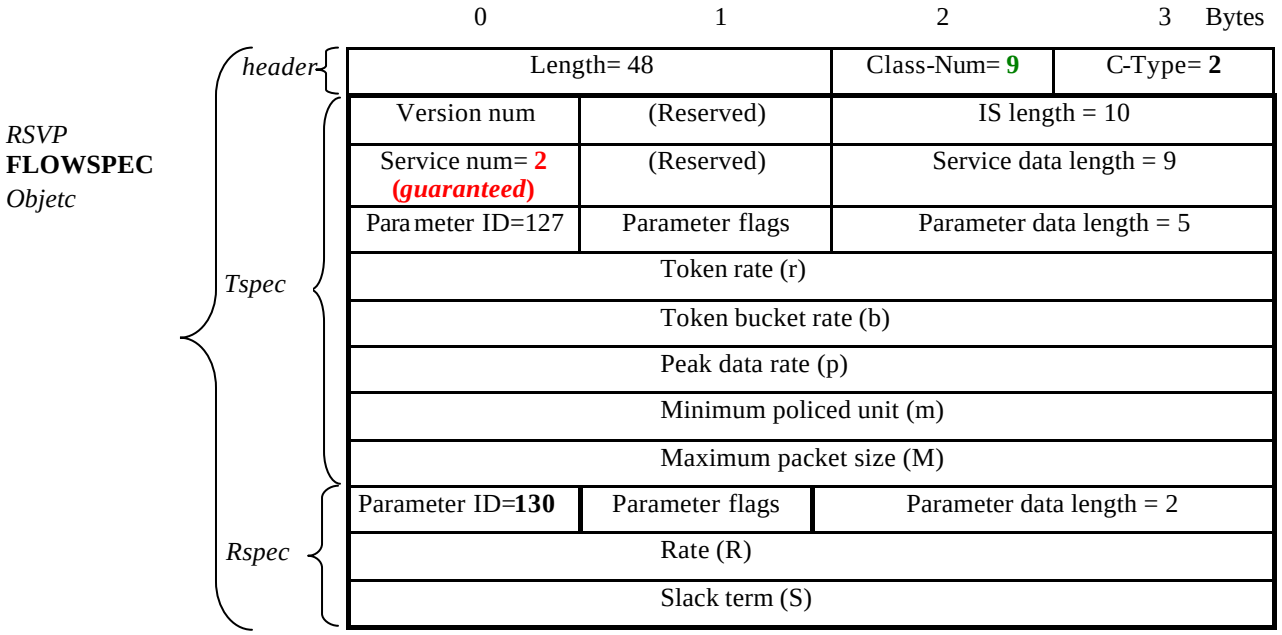

Figura 2.23 Objeto **FLOWSPEC** para especificação *Guaranteed*

O ***Rspec*** tem dois parâmetros: *"Rate"* ***R*** (bytes/sec) e *"Slack term"* ***S*** (microsegundos) que são utilizados para especificar características de atraso do fluxo, e são necessários para configuração de um serviço *Guaranteed*. O ***Rspec*** é usado para atingir um limite de atraso em particular, geralmente as aplicações utilizam os dados do ***Adspec*** e do ***Tspec*** para definí-lo.

O cálculo dos parâmetros ***R*** e *slack* ***S*** pode ser baseado no objeto ***Adspec*** quando recebido na mensagem PATH, desde que a mensagem não tenha sido danificada por algum elemento de rede, conforme descrito na RFC 2210.

A taxa ***R*** é definida pelo *RECEIVER* como a largura de banda a ser reservada. Pode ser obtida igualando-se ao parâmetro ***r*** do ***Tspec*** ou adicionando uma taxa de reserva ***R*** maior para compensar o atraso informado pelos parâmetros do ***Adspec***, ou seja, se aumentar a taxa ***R***, minimizará a necessidade de recursos dos *buffers*, reduzindo o atraso do fluxo de dados.

O ***Slack*** *term* é uma quantidade adicional de atraso que um nó de rede pode utilizar, sem comprometer o requerimento de atraso total definido para a aplicação. O ***Slack*** *term* significa a diferença entre o atraso desejado e o obtido na reserva da largura de banda, em outras palavras, é uma quantidade de atraso creditada para ser utilizada pelos nós de rede.

Com um *Slack term* ***S*** igual a "0" (zero), cada roteador ao longo da rota DEVE reservar a largura de banda ***R***. Caso ***S*** seja diferente de 0, oferece para cada roteador uma maior flexibilidade para executar sua reserva local, permitindo reservar menor largura de

banda, observando certas limitações, como o atraso descrito pela **equação 01** e as regras de processamento do ***Rspec***, definidas no tópico *"Ordering and Merging"* na RFC 2212.

Um exemplo de cálculo para o *Slack term* (***S***), considerando o atraso requerido *end-to-end*, chamado $\boldsymbol{D_{REQ}}$, é maior que o atraso máximo em comparação a um modelo *"fluid model"* apresentado na **equação 01**. Assumindo um valor ***R*** = ***r*** na fórmula *"fluid model"*, o atraso é definido pela equação 05.

(Equação 05) $$Delay_{max} < \frac{b}{r} + \frac{C_{tot}}{r} + D_{tot} \quad \text{sendo } r = R$$

Neste caso, o *Slack* é determinado pela equação a seguir:

$$S = D_{REQ} - \left(\frac{b}{r} + \frac{C_{tot}}{r} + D_{tot} \right) \quad \text{sendo } R \geq r$$

Seguem exemplos de reservas utilizando o ***Slack***, com o valor 'S=0' na primeira reserva de recursos que retorna um erro por falta de recursos, e uma segunda reserva com valor 'S>0' que permite ao nó ajustar seus recursos durante a rota, segundo regras estabelecidas e definidas na **RFC 2212**, critérios observados na **RFC 2210** e regras de reserva no processamento de mensagens descritos na **RFC 2209**. O exemplo é apresentado na figura 2.24.

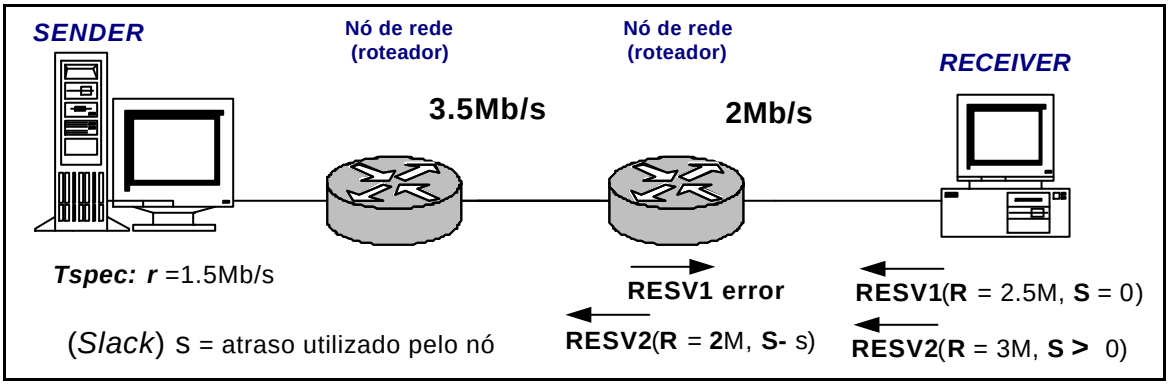

Figura 2.24 Exemplo de uma reserva de fluxo utilizando *Rspec*

2.2.8 *Mapeamento das mensagens PATH e RESV*

As informações recebidas em uma mensagem PATH pelo *RECEIVER*, geralmente são utilizadas para obter as informações sobre o comportamento da rede contidas no objeto **ADSPEC** e as especificações de QoS no objeto **SENDER_TSPEC**, possibilitando o *RECEIVER* formatar uma mensagem de requisição de reserva RESV. Para fins de demonstração deste

processo serão utilizados os nomes dos parâmetros dos objetos **FLOWSPEC**, **SENDER_TSPEC** e **SENDER_TSPEC** definidos na RFC 2210, apresentada na figura 2.25.

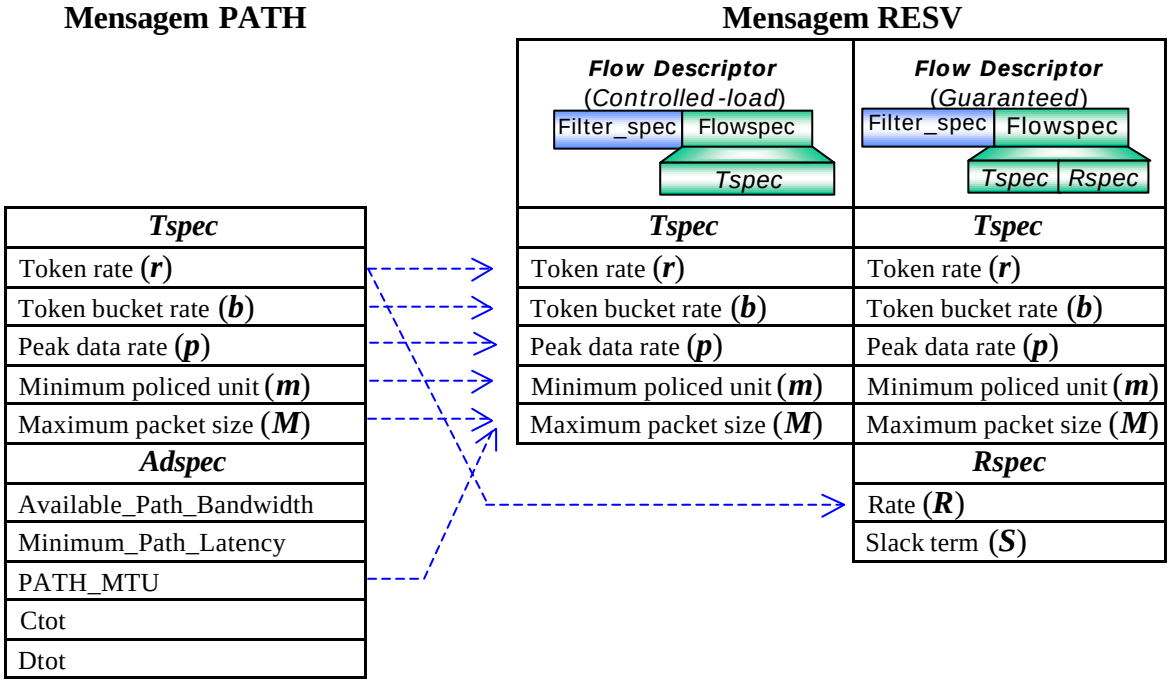

Figura 2.25 Mapeamento de mensagens PATH e RESV

Para obtenção dos parâmetros e formatação da mensagem RESV, geralmente os parâmetros do PATH ***Tspec*** são os mesmos para a estrutura RESV ***Tspec***, para os serviços *Controlled Load* ou *Guaranteed*, mas para uma requisição de um serviço *Guaranteed*, ainda é necessário definir os parâmetros do ***Rspec***.

O *RECEIVER* pode alterar os dados conforme a necessidade para a requisição RESV, obedecendo critérios de serviços oferecidos no objeto *Adspec*, descritos na sessão 3.3 da RFC2210. Caso seja utilizado o modelo de reserva *"one pass"*, o *RECEIVER* definirá seus parâmetros e tipo de serviço, e a reserva pode ser negada por indisponibilidade de recursos da rede ou do serviço do próprio *SENDER*, sendo recomendado utilizar sempre o modelo OPWA, transportado no ***Adspec***.

Os parâmetros do ***Tspec*** para o objeto FLOWSPEC devem observar as restrições descritas nas RFC 2210, RFC 2211 e RFC 2212. Para os serviços *Guaranteed* e *Controlled Load*, o *RECEIVER* pode copiar os valores ***Tspec*** {***r***,***b***,***p***,***m***,***M***} da mensagem PATH para o objeto FLOWSPEC da mensagem RESV.

Segue a descrição para formatar uma mensagem RESV.

1 Configuração do ***Tspec***{***M***} da mensagem RESV:

- O parâmetro ***Adspec PATH_MTU*** indica o tamanho máximo do pacote suportado na rede, podendo limitar o ***M*** = ***PATH_MTU***, porém ***M*** ≥ ***m***;
- Caso o ***M*** não seja especificado no PATH ***Tspec*** e por algum motivo não contenha o valor ***Adspec PATH_MTU***, o RSVP *daemon* estabelece um valor padrão de 1.500 bytes, o tamanho máximo de um pacote;

2 Configuração do ***Tspec***{***m***} da mensagem RESV:
- Caso ***m*** não seja especificado no PATH ***Tspec***, o RSVP *daemon* estabelece um valor padrão de 128 bytes;
- O valor de ***m*** nunca deve ser igual a '0', já que poderia caracterizar não existir dados ou cabeçalho IP presentes, descritos na sessão 3.1 da RFC 2210;

3 Configuração do ***Tspec***{***p***} da mensagem RESV:
- O valor da taxa ***p*** deve ser maior ou igual a taxa ***r***;
- Caso o ***p*** não seja especificado no PATH ***Tspec***, deve ser ajustado para o valor da taxa suportada pela rede (*link*) se este for conhecido, caso contrário, ***p*** receberá um valor positivo infinito, limitado pelo valor máximo definido na RFC 2212, 40 *terabytes*;

4 Configuração do ***Tspec***{***r,b***} da mensagem RESV:
- Os valores ***r*** e ***b*** assumem os parâmetros do PATH ***Tspec***{***r,b***};

5 Reserva de estilo do fluxo pelo *RECEIVER*:
- O *RECEIVER* indica o estilo de acordo com os fragmentos recebidos do objeto ***Adspec***;

6 Objeto FILTER_SPEC assume os valores do SENDER_SPEC;

7 Objeto RESV_CONF pode receber o endereço IP do *RECEIVER* para indicação da probabilidade da reserva do fluxo estar instalado entre *SENDER* e *RECEIVER.*

O *RECEIVER* quando especificar o serviço *Guaranteed*, deve fornecer os parâmetros do ***Rspec***{***R***,***S***}. Utilizando os parâmetros do objeto ***Adspec*** *Minimum_Path_Latency* e *Available_Path_Bandwidth* e as equações 03 e 04 definidas no tópico ***Comportamento de atraso "end-to-end"***, é possível calcular respectivamente ***R*** e ***S*** para a mensagem RESV, descrito no tópico '*Guidelines for Implementators*' na RFC 2212.

2.3 Reservas de RSVP (Estilos)

2.3.1 Reservas Distintas (FF)

Cada fluxo de dados possui uma reserva independente. Este tipo de reserva é utilizada para sessões *unicast* e *multicast*. O estilo *Fixed Filter* (FF) é representado por:

$$FF(S\{Q\}) \quad \text{ou} \quad FF(S_1\{Q_1\},S_2\{Q_2\},\ldots,S_n\{Q_n\})$$

Para simplificar, o exemplo abaixo apresenta uma taxa expressa em *Kbps*[8] ao invés de utilizar parâmetros do *token bucket*. O estilo distinto (*Fixed Filter*) é apropriado para aplicações semelhantes a videoconferência, onde uma janela é requerida para cada emissor e todas as janelas devem ser atualizadas simultaneamente. Este estilo requer que o *SENDER* seja identificado pelos *RECEIVERS* para receberem a reserva com a largura de banda solicitada. Se uma ou mais reservas forem feitas para um *SENDER* em particular, não haverá compartilhamento da largura de banda da entre outros *SENDERS*.

No exemplo apresentado na figura 2.26, uma sessão *multicast* utiliza três *SENDERS* caracterizados por S_1, S_2, S_3, três computadores (*hosts*) H_1, H_2, H_3 como *RECEIVERS* e dois roteadores.

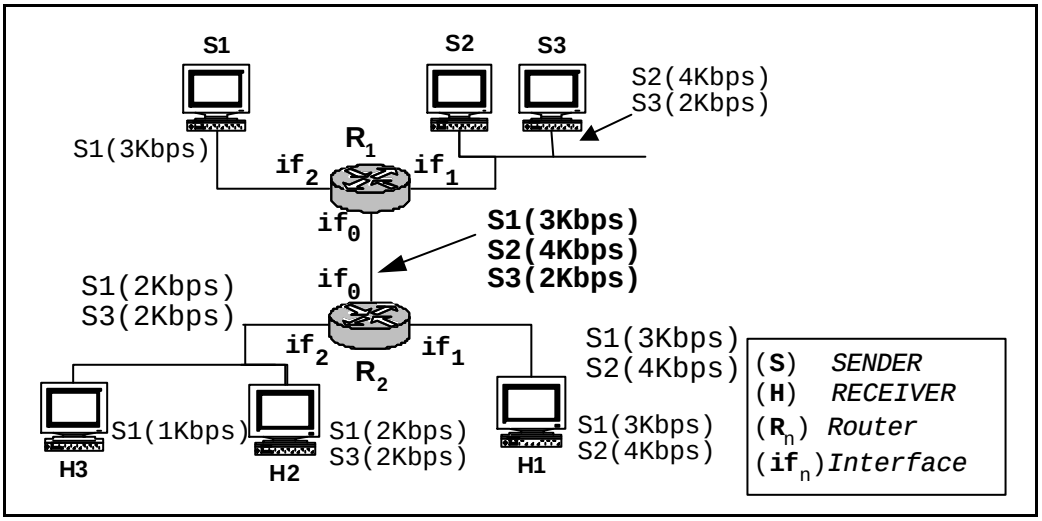

Figura 2.26 Exemplo de um estilo *Fixed Filter* (FF)

Os seguintes requerimentos são solicitados pelos *RECEIVERS*:

- H_1 quer reservar 3Kbps para S_1 e 4Kpbs para S_2;

[8] Kbps: *Kilobits per second*. *K* representa 10^3 ou 1000. Por exemplo, 64Kbps equivale a 64000 bits por segundos de dados.

- H_2 quer reservar 2Kbps para S_1 e 2Kbps para S_3;
- H_3 quer reservar 1Kbps para S_1;

A reserva dos *RECEIVERS* H_2 e H_3 para o *SENDER* S_1 são intercaladas para uma reserva de 2Kbps em (if_2, R_2). A reserva de H_2 com um fluxo de 2Kbps para o *SENDER* S_3 chega ao roteador também na interface (if_2, R_2). Outra requisição procedente do *RECEIVER* H_1 chega a interface (if_1, R_2) com uma reserva para S_1 de 3Kbps e 4Kbps para S_2. As requisições do roteador R_2 das interfaces (if_1, if_2) são intercaladas e enviadas para interface (if_0, R_2) com as definições dos fluxos: **3Kbps** para **S_1**, **4Kbps** para **S_2** e **2Kbps** para **S_3**.

A requisição recebida pelo roteador R_1 na interface if_0 são enviadas aos *SENDERS* pelas interfaces if_1 e if_2 de R1, com as seguintes taxas dos fluxos recebidos: na interface if_2 uma taxa de **3Kbps** ao *SENDER* **S_1**, na interface if_1 com **4Kbps** ao *SENDER* **S_2** e **2Kbps** ao *SENDER* **S_3**.

2.3.2 Compartilhadas

As reservas compartilhadas são adequadas para sessões *multicast* e uma mesma reserva pode ser compartilhada por vários fluxos. Um exemplo deste tipo de aplicação pode ser a transmissão de um evento (*SENDER*) e acompanhada por vários participantes (*RECEIVERS*).

As reservas compartilhadas podem ser classificadas como:

- Estilo *Wildcard-Filter* (WF), representado por $\boxed{WF(*\{Q\})}$;
- Estilo *Shared-Explicit* (SE), representado por $\boxed{SE((S_1, S_2, \ldots, S_n), \{Q_n\})}$;

2.3.2.1 *Wildcard Filter* (WF)

A reserva do estilo WF não possui o objeto **`FILTER_SPEC`**. Portanto, como os *SENDERS* não são definidos, os *RECEIVERS* não necessitam fornecer os endereços IP dos *SENDERS*. As mensagens RESV com estivo WF são aplicadas para todos *SENDERS* da sessão.

O exemplo apresentado na figura 2.27, é uma sessão *multicast* contendo três *SENDERS* caracterizados por S_1, S_2, S_3, três *RECEIVERS* representados H_1, H_2, H_3 e dois roteadores.

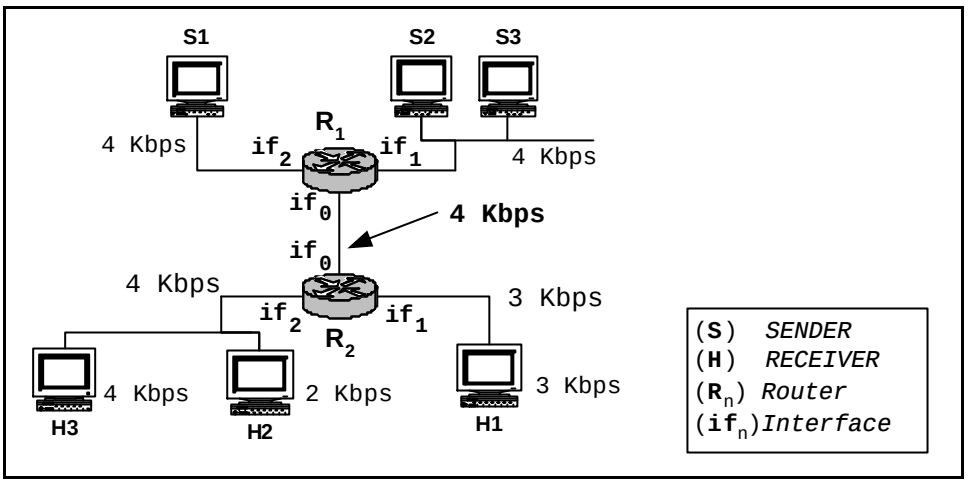

Figura 2.27 Exemplo de um estilo *Wildcard Filter* (WF)

Os seguintes requerimentos são solicitados pelos *RECEIVERS*:

- H1 quer reservar 3Kbps
- H2 quer reservar 2Kbps
- H3 quer reservar 4Kbps

É importante notar que a origem não é identificada e a intercalação das requisições nos roteadores não somam as requisições que chegam, mas utilizam a maior taxa dos valores solicitados.

As reservas de H3 e H2 são intercaladas para uma reserva de 4Kbps em (if_2, R_2). Outra requisição de H1 vem de (if_1, R_2) com uma taxa de 3Kbps. O roteador R_2 intercala a requisição das interfaces if_2 e if_1 para interface if_0 reservando de 4Kbps em (if_0,R_2) enviando a requisição para o roteador R_1 na interface if_0. O roteador R_1 envia a requisição as interfaces if_1 e if_2 aos *SENDERS* S1, S2, S3.

2.3.2.2 Compartilhamento Explícito (SE)

A diferença entre a reserva dos estilos WF e SE, é a identificação dos *SENDERS* pelos *RECEIVERS*. A figura 2.28 apresenta um exemplo de reserva com estilo SE e a descrição a seguir.

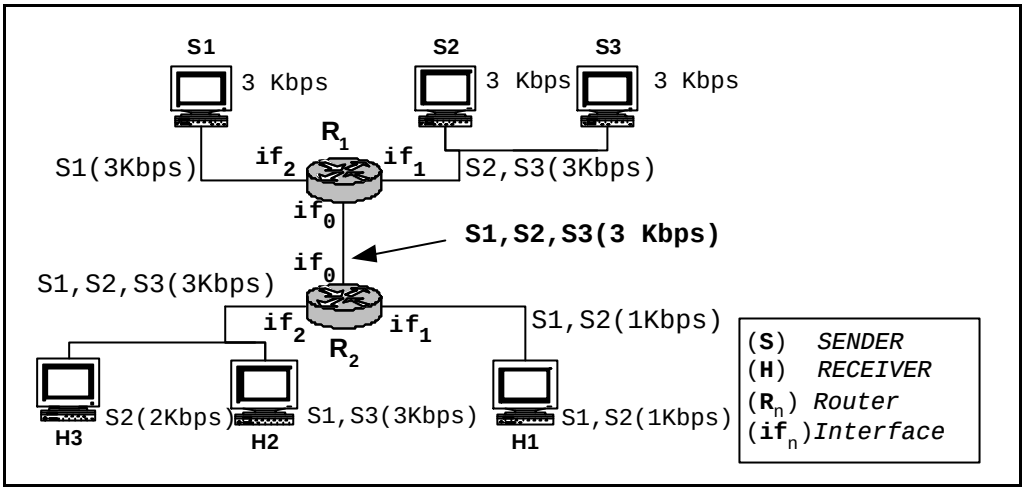

Figura 2.28 Exemplo de um estilo *Shared-Explicit* (SE)

Os seguintes requerimentos são solicitados pelos *RECEIVERS*:

- H_1 deseja reservar 1Kbps para S_1 e S_2;
- H_2 deseja reservar 3Kbps para S_1 e S_3;
- H_3 deseja reservar 2Kbps para S_2;

As reservas S_1, S_2 e S_3 provenientes de H_3 e H_2 na interface (if_2, R_2) são intercaladas em uma reserva de 3Kbps. Outra requisição de reserva 1Kbps de H_1 para S_1 e S_2 chegam a interface (if_1,R_2). As requisições das interfaces (if_1,if_2) do roteador R_2 são intercaladas na interface (if_0, R_2) com um fluxo de 3Kbps para os *SENDERS* S_1, S_2 e S_3, enviadas ao roteador R_1.

A requisição recebida pelo roteador R_1 na interface (if_0, R_1) são enviadas com um fluxo compartilhado de 3Kbps na interface (if_2, R_1) ao *SENDER* S_1 e pela interface (if_1, R_1) aos *SENDERS* S_1 e S_2.

2.4 Classes de Aplicações

A importância e o impacto de uma aplicação em relação ao negócio, geralmente são considerados para definições de garantia e performance, nas políticas em uma rede. Aplicações qualitativas não possuem requerimentos de tráfego definidos, tal como *Enterprise Resource Planning* (ERP), variam entre organizações e devem ser mensuradas em cada caso, para definição correta dos fluxos de dados necessários ao estabelecimento de tráfego QoS.

Entre os padrões de multimídia que definem serviços quantitativos, a principal fornecedora de padrões de áudio é a *International Telecommunication Union* (ITU), que é

uma agência especializada no campo das telecomunicações e, a ITU *Telecommunication Standardization Sector* (ITU-T), que é o órgão responsável pelas recomendações que visam, entre outras, padronizar este setor. As recomendações da ITU-T são divididas em séries classificadas por ordem alfabética de A-Z.

A Tabela 2.4 apresenta como exemplo, alguns dados de duas séries de serviços: a **G** e **H**, que serão utilizados no presente estudo. A série **G** possui recomendações para sistemas de transmissão e mídia, sistemas digitais e redes e a série **H** possui recomendações para sistemas audiovisuais e multimídia [ITU01].

Tabela 2.4 Descrição de parâmetros ***Tspec*** para classes de aplicações

Classes de Serviço	**G711**	**G729**	**H263CIF**	**H261QCIF**
Tipo de Serviço	*Guaranteed*	*Guaranteed*	*Controlled Load*	*Controlled Load*
Token rate (***r***)	9250	2000	16000	12000
Token bucket rate (***b***)	680	80	8192	6000
Peak data rate (***p***)	13875	4000	-	-
Maximum packet size (***M***)	340	40	8192	2500
Minimum policed unit (***m***)	340	40	80	80

Unidades em Bytes

Descrição das classes de serviços utilizadas na tabela 2.4:

- **G.711**: aplicações de voz em redes utilizando a modulação de código de pulso de freqüências de voz - *Pulse code modulation* (PCM). O atraso dos dados de áudio aceitável em redes privadas é de 200ms e o limite é de 250ms [CIS01b].
- **G.729**: aplicações de voz com taxa de 8Kbps para transmissão, usando algoritmo *Coding of Speech - Adaptive Code-Excited Linear-Predictive* (CS-ACELP). Amplamente utilizada em muitas aplicações multimídia.
- **H.261**: algoritmo de codificação e decodificação (codec) de vídeo para serviços audiovisuais a taxas de **p** x **64Kbps**, (**p** varia de 1 a 30), com largura de banda variando de 64Kbps à aproximadamente 2Mbps, dependendo do algoritmo de compressão e resolução utilizados.
- **H261QCIF**: Serviço audiovisual com resolução de 177x144 *pixeis*, com os parâmetros de serviço de QoS apresentados na tabela 4. Está classificado como um serviço *Controlled-load*, ou seja, aplicação tolerante de tempo real que requer uma quantidade suficiente de largura de banda e pode tolerar atrasos ocasionais e perdas

de pacotes.

- **H.263**: Codificação de vídeo para comunicação com baixas taxas de dados (*low bit-rate*), semelhantes ao MPEG-4. Este tipo de aplicação é destinada para serviços tipo vídeo-fone, controle e monitoramento remoto entre outras aplicações. Classificado como um serviço *Controlled-load*.

Com as definições dos parâmetros do *Token Bucket*, como as apresentadas na tabela 4, que serão armazenados em classes de serviços, utilizando o XACML para descrição de políticas de controle de acesso de serviços QoS RSVP, através de um servidor de políticas, possibilitará controlar quais serviços RSVP os *SENDERS* poderão fornecer aos *RECEIVERS*.

2.5 Conclusões do Capítulo

Este capítulo apresentou conceitos sobre QoS e RSVP. Foi discutido QoS e a sua classificação em níveis de aplicações, que são configuradas no controle de admissão, como aplicações em tempo-real, que são medidas em dimensões de latência e fidelidade. Estas conhecidas como aplicações *playback*, classificadas como tolerantes e intolerantes, visaram introduzir conceitos para facilitar a compreensão, na seqüência, do protocolo RSVP e o suporte deste em QoS.

O protocolo RSVP suporta três tipos de tráfego para QoS: o *best-effort, controlled-load* e *guaranteed service*. A especificação do fluxo de dados orienta como os roteadores ou *hosts* devem tratar o tráfego de dados e propiciar o QoS desejado.

O RSVP trata o emissor distinto do receptor, e uma mesma aplicação pode atuar como um emissor e um receptor ao mesmo tempo, mantendo o estado da conexão com atualizações periódicas chamadas de "*soft state*", através de sinalização do RSVP.

O protocolo RSVP utiliza mensagens de sinalização para configuração e manutenção do fluxo de dados. As principais mensagens RSVP são PATH e RESV. A mensagem PATH é sempre enviada pelo *SENDER*, e a mensagem RESV pelo *RECEIVER*.

Um fluxo de dados em RSVP é uma seqüência de mensagens que possuem a mesma origem, um ou mais destinos e qualidade de serviço desejada. Uma requisição de reserva de RSVP (RESV) consiste de um ***Flow Descriptor***, formado por um objeto ***flowspec*** e um objeto ***filter spec***, que atua no nó RSVP. O ***filter spec*** junto com a especificação da sessão definem os pacotes de dados (fluxo RSVP) para receberem o QoS. A especificação de QoS

desejado é definida pelo ***flowspec***.

O ***flowspec*** geralmente inclui uma classe de serviço e dois conjuntos de parâmetros numéricos: ***Rspec*** e ***Tspec***. O ***Rspec*** (*Request Specification*) descreve o nível de QoS desejado para um fluxo requisitado a um *router* ou *host*. O ***Tspec*** (*Traffic Specification*) descreve o tráfego de dados para qual o serviço está sendo requisitado e que será reservado pelo *router* ou *host*. A estrutura e as informações contidas no objeto ***Tspec*** são utilizadas em ambos serviços *Guaranteed* e *Controlled-Load*.

Foi apresentado o mecanismo do *Token Bucket*, que é um algoritmo para regular tráfego de dados que controla e limita a largura de banda, utilizando os parâmetros contidos no ***Tspec*** e os mecanismos de controle de tráfego que o utilizam.

Foi discutido o objeto Adspec e seus parâmetros. Este é transportado na mensagem PATH e contém informações que são geradas em cada nó de rede *RSVP aware* ao longo da rota, sendo usadas ou modificadas pelos nós, antes do objeto ser entregue ao *RECEIVER*. As informações recebidas pelo objeto *Adspec* podem ser utilizadas pelo *RECEIVER* para determinar se há disponibilidade de serviço e as capacidades mínimas oferecidas pelos nós até o *SENDER*. Recomenda-se a utilização do objeto *Adspec* para impedir refutar a instalação de reserva de recursos ao longo da rota.

Nesta proposta de dissertação a aplicação *host RSVP* reserva fluxos de QoS utilizando o mecanismo de Controle de Políticas do prórpio *host RSVP*, o que estende a responsabilidade deste para efetivação das políticas (PEP-*Policy Enforcement Point*), através da execução de chamadas ao *RSVP daemon*, e um servidor de políticas para consultas de autorização, obtenção de parâmetros de QoS e tipos de serviço (PDP-*Policy Decision Point*). A comunicação é realizada em um ambiente cliente/servidor entre o mecanismo de Controle de Políticas e um Servidor de Políticas, e este descreve as classes de serviço de QoS de acordo com a aplicação, utilizando XACML. É necessário usar um protocolo transacional de políticas para troca de mensagens entre o *host RSVP* e o servidor de políticas, como COPS (*Commom Open Policy Service Protocol*).

O COPS é um protocolo de consultas e respostas que suporta de forma eficiente o controle de políticas. O COPS, PEP e PDP serão esmiuçados a seguir, no capítulo 3.

Capítulo 3

Gerência de QoS Baseada em Políticas

3.1 Gerência de Redes Baseadas em Políticas

Em redes tradicionais, novos serviços são desenvolvidos e atualizados lentamente, adicionando novo *hardware*. A rede física é então configurada novamente para refletir as novas alterações. Administradores de rede destes ambientes geralmente gerenciam a rede manualmente, configurando um equipamento por vez, tal como roteadores, *switches* e outros dispositivos de rede que são projetados para interação manual e não para uma interação de gerenciamento automático de equipamentos.

A rede tradicional é lenta para mudanças, ineficiente, e também difícil para proteger, escalar e atualizar, tornando-se um problema. A solução para este envolve uma mudança nos dispositivos de *hardware*, configuração da rede com dispositivos manuais para uma rede programável e automática, estruturada em gerência de rede baseada em política, ou seja, PBNM (*Policy-Based Network Management*) que padroniza e centraliza as políticas.

A PBNM é um conjunto de regras que assegura a validação da política em elementos de rede que controlam políticas para operar e gerenciar requisições para recursos da rede. É um mecanismo que codifica as regras de negócio na utilização apropriada dos recursos da rede. Essencialmente a PBNM provê uma "inteligência" de gerenciamento, permitindo às políticas serem alteradas, adicionadas e removidas, através da manipulação de políticas. A hierarquia em um sistema distribuído é necessária para saber quais os requerimentos de redes multi-serviço e poder propagar a política entre domínios.

A estrutura da PBNM proposta pela IETF na RFC 2753 [YAV00] têm dois elementos para controle de políticas, definidos como PEP (*Policy Enforcement Point*) e PDP (*Policy*

Decision Point).

3.1.1 Descrição dos Elementos da arquitetura PBNM

O **PEP** (*Policy Enforcement Point*): é um dispositivo de rede que aplica a política quando invocado por um evento externo, que será monitorado pelo próprio PEP, garantindo que uma decisão de política seja obedecida [WES01].

O **PDP** (*Policy Decision Point*): é um dispositivo lógico, onde as decisões de política são tomadas para si mesmo ou para outro elemento de rede que requisite semelhantes decisões, ou seja, um servidor de políticas [WES01].

Na arquitetura proposta pela RFC 2753, o PDP pode utilizar outros mecanismos e protocolos, não especificados nesta RFC, com o propósito de armazenamento das políticas ou comunicação com outros servidores para obter dados de autenticação, autorização e contabilizar acessos.

3.1.2 Modelo de processamento PBNM

A arquitetura definida na RFC 2753 [YAV00] possibilita três modelos de processamento para os elementos de controle de políticas PEP e PDP, descritos abaixo:

1. O PEP recebe uma notificação ou mensagem que requer uma decisão baseada em política e deve solicitá-la ao PDP, que retorna a decisão de autorização ao PEP que obedece-la-á.
2. Um evento local ou mensagem invoca o PEP para tomar uma decisão baseada em política, o mesmo consultará sua base local chamada LPDP (*Local Police Decision Point*). O PEP solicitará uma decisão ao PDP somente se nenhuma política local relevante for encontrada, descrito no item 4 - *Architectural Elements* na RFC 2753.
3. O PEP e PDP podem estar no mesmo elemento de rede e executar uma decisão local.

3.2 Protocolo Transacional de Políticas - COPS

O **COPS** (*Common Open Policy Service*) é um protocolo de comunicação que emprega um modelo cliente/servidor. Permite que um servidor de política PDP comunique as decisões de política ao dispositivo de rede PEP.

O COPS é um protocolo transacional *"query/response"* de políticas definido pela IETF

e que suporta dois modelos para controle de política: *"Outsourcing"* e *"Provisioning"* descrito na RFC 2748 [BOY00].

O ***Outsourcing*** é um modelo de execução onde um PEP remete uma consulta (*"query"*) ao PDP, delegando a decisão de um evento específico de política (*outsourced policy*) para o servidor de política PDP.

O ***Provisioning*** é um modelo de execução onde os elementos de rede (PEP) são pré-configurados, baseados em uma política, antes de processar os eventos provenientes da rede. A configuração é enviada de um PDP para um PEP e, isto pode ocorrer quando um PEP é conectado a rede ou haja alguma alteração de política no servidor de políticas. O objetivo deste modelo é a distribuição de configuração de informações para um PEP.

O COPS é um protocolo que mantém o estado das conexões ativas durante a comunicação entre o PEP e o PDP, e por esta característica é chamado de *statefull* e utiliza o protocolo de transporte TCP (*Transmission Control Protocol*). Os servidores PDP devem aguardar pedidos de conexão utilizando TCP na porta 3288, padronizada pela *Internet Assigned Numbers Authority* (IANA)[9]. As solicitações do PEP são instaladas no PDP e, através de uma mensagem do PEP devem ser excluídas do PDP.

O protocolo apresenta tolerância a falhas através do envio constante de mensagens de monitoração entre o PDP e PEP, chamada de *"keep-alive"*. Quando uma conexão TCP de um PEP é perdida, o PDP pode excluir todos os seus estados instalados.

O PEP detectando a perda da conexão deve tentar restabelecê-la para sinalizar ao PDP, enviando uma mensagem contendo o objeto de erro e notificar eventuais solicitações efetuadas durante o período que a conexão estava perdida. Uma mensagem pode ser solicitada pelo PDP ao PEP para sincronizar os estados internos do PEP com os estados instalados no PDP.

As mensagens COPS e o sentido que podem ocorrer são apresentadas abaixo, de acordo com a RFC 2748, descritas no item 3 [BOY00]:

> ***REQ – Request*** (PEP → PDP): o PEP solicita ao PDP uma decisão de política e este mantém o estado associando o *client-handle* recebido, ou seja, valor único que identifica um estado, recebido na mensagem do PEP.

---

[9] *Internet Assigned Numbers Authority* (IANA): autoridade para coordenação de trabalhos de administração global de endereços, nomes de domínio e outros detalhes dos protocolos usados na tecnologia Internet. Atualmente realizada pela *Internet Corporation for Assigned Names and Numbers* (ICANN). URL http://www.icann.org

DEC – Decision (PDP → PEP): envia uma decisão de política ao PEP em resposta a uma requisição ***REQ***.

RPT - Report State (PEP → PDP): comunica ao PDP o resultado da efetivação da decisão da política recebida do PDP, por uma mensagem ***DEC***.

DRQ - Delete Request State (PEP → PDP): objeto que especifica ao PDP que o estado identificado por um *client-handle* não é mais necessário. PDP remove o estado indicado.

SSQ - Synchronize State Request (PDP → PEP): mensagem solicitada pelo PDP ao PEP para sincronizar os estados internos do PEP com os estados instalados no PDP.

OPN - Client-Open (PEP → PDP): após o estabelecimento da conexão TCP entre PEP e PDP, indica o início de um serviço de política para um tipo específico de cliente. Um PEP é identificado por um objeto único **<PEPID>** em um domínio de política.

CAT - Client-Accept (PDP → PEP): usada pelo PDP para confirmar uma mensagem ***OPN***, retornando ao PEP um objeto com um intervalo de tempo para mensagens ***KA***.

CC - Client-Close (PEP → PDP, PDP → PEP): usada tanto pelo PDP como pelo PEP, para notificar que um determinado tipo de cliente (*client-type*)[10] não é mais suportado.

KA - Keep-Alive (PEP → PDP, PDP → PEP): a mensagem deve ser enviada pelo PEP dentro de um período definido na mensagem ***CAT*** recebida do PDP. Quando o PDP recebe a mensagem *keep-alive* do PEP, deve responder ao PEP validando para ambos que a conexão está ativa, mesmo se não houver troca de mensagens.

SSC - Synchronize State Complete (PEP → PDP): quando a sincronização de estado é terminada pelo PEP, após o recebimento de uma mensagem ***SSQ***, envia a mensagem ao PDP, indicando que o PEP e PDP estão sincronizados.

A IETF[11] tem desenvolvido e padronizado serviços de rede aplicáveis para diferentes tipos de QoS, utilizando os modelos de arquitetura definidos na RFC 2753 [YAV00] em conjunção com os modelos para controle de políticas do protocolo COPS definidos na RFC

[10] Campo *client-type* de 16 bits que identifica o tipo de cliente de política. Clientes não padronizados utilizam a faixa de 0x800 a 0xFFFF.

[11] A IETF definiu um grupo de trabalho para padronizar serviços de rede, como QoS, e engenharia de tráfego chamado RAP-WG, *Resource Allocation Protocol Working Group*.

2748 [BOY00].

Outro modelo apresentando COPS utilizando o mecanismo *provisioning* é descrita na RFC 3084 [CHA01], mas sua especificação é independente do tipo de política, tais como políticas de QoS e segurança, entre outras. Um PEP (dispositivo de rede) é provido (*provisioned*) pelo PDP com políticas através de uma estrutura de dados enviada, chamada de *Policy Information Base* (PIB)[12], que definem as políticas a serem aplicadas (*enforced*) pelo PEP.

Uma arquitetura para PIB usando COPS foi descrita recentemente na RFC 3318 [SAH03][13] e uma estrutura de informação de política de aprovisionamento (*Structure of Policy Provisioning Information* - SPPI) foi descrita na RFC 3159 [MCC01][14].

Um modelo utilizando o protocolo COPS para fornecer um mecanismo *outsourcing* de decisão baseada em política usando o protocolo QoS RSVP é apresentado na RFC 2749 [HER00a].

3.3 Cenário de Modelo de Processamento de Políticas

Em um cenário utilizando um modelo ***outsourcing*** RSVP, apresentado por Ponnappan et al. [PON02], quando o RSVP *daemon* recebe uma mensagem PATH ou RESV, emulando o PEP, este envia uma mensagem de requisição COPS(**REQ)** para o servidor de políticas PDP.

O servidor PDP obtém os dados das regras de política referentes ao identificador do usuário ou aplicação recebida na mensagem de requisição do PEP. O PDP envia uma mensagem COPS(**DEC**) indicando a decisão de autorização para o PEP, e dependendo da avaliação recebida, o PEP (*RSVP daemon*) estabelece ou não o fluxo RSVP.

A cada início de sessão RSVP, é necessário o PEP enviar uma mensagem de Requisição COPS(**REQ)**. O Servidor PDP instala uma sessão para identificar o PEP e monitorar o estado da mesma. Quando se encerra uma sessão, O PEP envia uma mensagem COPS(**DRQ**) ao PDP significando que o estado do PEP instalado no PDP não é mais

---

[12] PIB: Coleção de classes que identificam um dispositivo. As classes são um conjunto de dados ordenados representando um tipo de política.

[13] RFC 3318 de[Sahita/FwkPIB]: Neste modelo, a informação de políticas é vista como uma coleção de *Provisioning Classes* (PRCs) e *Provisioning Instances* (PRIs) residindo em uma informação armazenada virtualmente, expressa como *Policy Information Base* (PIB).

[14] RFC 3159 de[MCCLOGHRIE/SPPI]: descreve a estrutura para especificar informações de política que pode ser transmitida para dispositivos de rede com o propósito de sua própria configuração. A definição dos módulos PIB neste formato permite utilizar o conhecimento da comunidade, experiência e ferramentas de SMI e módulos MIB.

necessário. O PDP remove o estado indicado. Situação representada na figura 3.1:

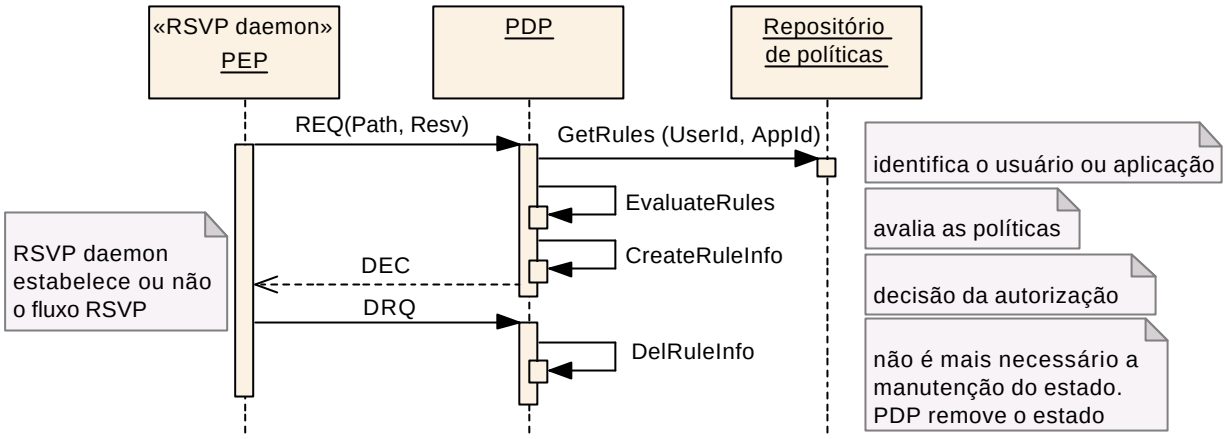

Figura 3.1 Cenário COPS- RSVP *Outsourcing*

No cenário COPS *provisioning*, apresentado por Ponnappan et al. [PON02], foi utilizado o protocolo QoS de Priorização ou Serviços Diferenciados (*DiffServ*). Quando o PEP inicia seu funcionamento, reporta as interfaces físicas e os papéis (*Roles*) que são designadas àquelas interfaces, através de uma mensagem de requisição COPS(**REQ**) para o PDP.

O servidor de políticas PDP extrai todas as regras de política para cada uma das diferentes combinações dos papéis reportados pelo PEP. O PDP converte as regras para informar a configuração específica de cada dispositivo, através de uma PIB *DiffServ*. O PDP envia a decisão na mensagem COPS(**DEC**) ao PEP para instalação de cada regra (*rule*) que se aplica à combinação dos papéis (*roles*) reportados, em mensagens COPS(**DEC**) separadas. Situação representada na figura 3.2:

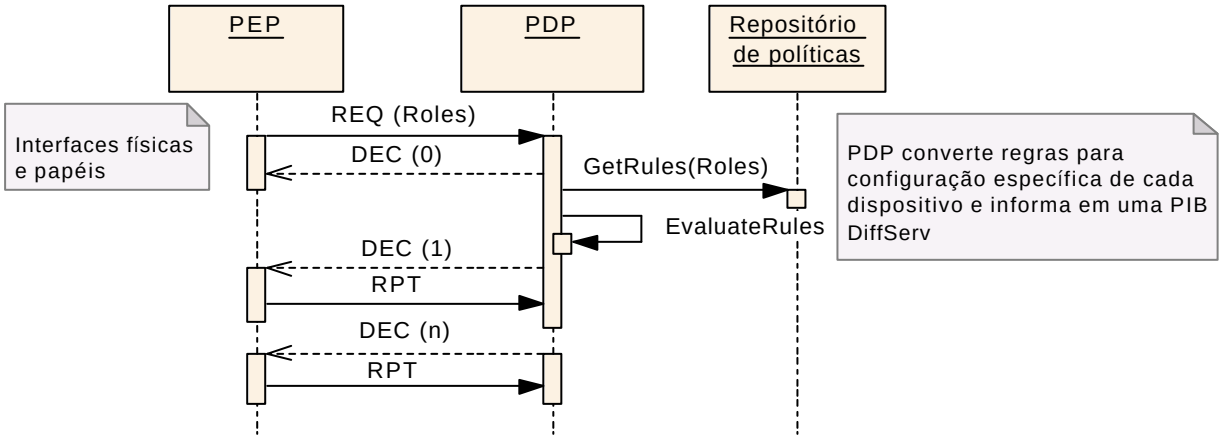

Figura 3.2 Cenário COPS *Provisioning*

Estes dois cenários apresentados com protocolos de QoS utilizando COPS no modelo

outsourcing (COPS-RSVP) e no modelo *provisioning* (COPS-PR), demonstram a necessidade de controle para cada evento em um modelo *outsourcing* e a conversão de regras através de uma PIB no modelo *provisioning*.

Atualmente, não há uma estrutura de PIB definida para RSVP para ser utilizada no modelo *provisioning* de controle de políticas, e minimizar o excesso de requisições de decisões de políticas para RSVP. Portanto, a proposta utilizará o mecanismo *outsourcing* para consultas a um servidor de políticas PDP, utilizando o modelo XACML para suportar o protocolo RSVP.

3.4 Representação de Políticas orientadas a objetos – PCIMe - QPIM

Os atuais modelos de representação de políticas orientadas a objetos herdaram a arquitetura básica do CIM (*Common Information Model*), definido pela *Distributed Management Task Force* (DMTF) [DMT99].

O CIM é um modelo conceitual de informação para representação de elementos gerenciados e não é limitada a uma implementação em particular. Permite a troca de informações de gerenciamento entre sistemas e aplicações de gerenciamentos heterogêneos. O padrão CIM é composto de duas partes: a especificação CIM (***CIM Specification***) e o esquema CIM (***CIM Schema***). A especificação define detalhes para integração com outros modelos de gerenciamento, enquanto o esquema fornece as descrições do próprio modelo.

O CIM é estruturado em três diferentes camadas [DMT99], [WES00]:

- *Core Model*: conjuntos de classes e associações representam o núcleo e modelos comuns do CIM que são aplicáveis a todas áreas do gerenciamento.
- *Common Model*: conjunto básico de classes que definem várias tecnologias de áreas independentes. As áreas comuns são respectivamente: sistemas, aplicações, redes e dispositivos. As classes, propriedades, associações e métodos fornecem uma visão da área que é detalhada o suficiente para uso como base para um projeto de programa ou implementação. Extensões usam este esquema para expansão de classes e implementações de plataformas específicas. O *Core Model* e o *Common Model* são chamados de CIM *Schema*.
- *Extension Schemas*: utilizado por plataformas específicas para mapear as informações e proporcionar o gerenciamento utilizando o *Common Model* como base. Esses esquemas são específicos para os ambientes, como sistemas

operacionais, por exemplo, UNIX ou Microsoft Windows.

A troca de informações entre aplicações de gerenciamento é essencial para o CIM e o mecanismo corrente para troca de informações de gerenciamento é o *Management Object Format* (MOF)[15]. Portanto, um sistema que suporte a implementação CIM deve ser capaz de importar e exportar dados utilizando o MOF de acordo com a especificação CIM.

O padrão CIM da DMTF, foi estendido pela *Internet Engineering Task Force* (IETF), chamado-o de PCIM (*Policy Core Information Model*) definido na RFC 3060 [MOO01a], em fevereiro de 2001, que permite a representação da informação de política de qualquer natureza.

O *Policy Core Information Model Extensions* (PCIMe) proposto pela IETF e apresentado na RFC 3460, descrita por Moore [MOO03] em janeiro de 2003, é um modelo de representação de políticas orientada a objetos compatível com o PCIM, diferenciando-se deste por ter introduzido em sua arquitetura novas classes em áreas que o PCIM previamente não cobria. Apesar do trabalho em conjunto da IETF com a DMTF, algumas das especificações do modelo PCIMe ainda não estão incluídas na especificação final do modelo CIM versão 2.7 da DMTF, definida em maio de 2003, ainda pendentes para análise e implementação pela DMTF.

O PCIMe adicionou elementos, associações, prioridades e estratégias de decisão, com a origem de algumas mudanças baseadas em alguns trabalhos em andamento (*drafts*), como: ICPM (*IPsec Configuration Policy Model*) [JAS01], QPIM (*QoS Policy Information model*) [SNI01] e outros. As alterações no modelo PCIM foram necessárias para prover maior flexibilidade na representação de políticas e evitar conflitos de interpretação em determinados componentes, como por exemplo, os componentes do repositório de políticas (*PolicyRepository*) que podiam dificultar o discernimento das definições de arquitetura de política (*Policy Framework*), tais como, ferramentas de gerenciamento de políticas, repositório de políticas, PDP e PEP.

O QPIM (*QoS Policy Information model*) descrito no *draft*[16] de maio de 2003 [SNI03], é uma proposta da IETF que vem sendo desenvolvida há muito tempo, tendo sua descrição anterior publicada no *draft* de novembro de 2001 [SNI01], que serviu de auxílio

---

[15] Arquivo MOF (*managed object format*). Este formato é usado para descrever informações CIM e é definida pela DMTF na especificação CIM. URL: http://www.dmtf.org/standards/cim/cim_spec_v22/

[16] *Draft*: trabalho em progresso sendo desenvolvida pela IETF, e o *dratft* QPIM aborda o "modelo de informação de política para QoS".

para adicionar associações e elementos reutilizáveis adicionais para descrição do PCIMe na RFC 3460. A evolução do PCIMe [MOO03] pode ser observada na figura 3.3.

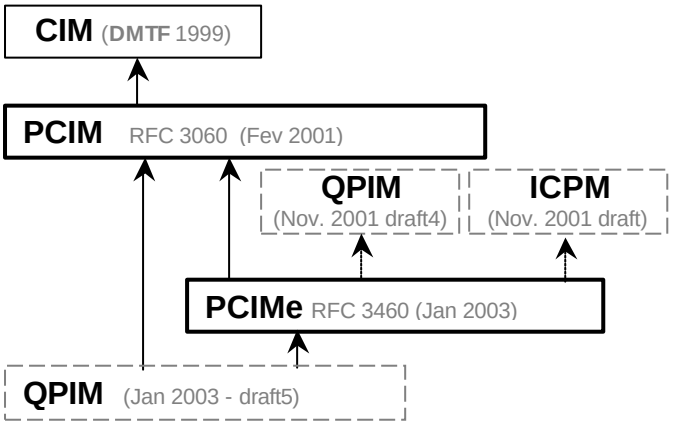

Figura 3.3 Política orientada a objetos – evolução PCIMe

O QPIM é um modelo que pretende estabelecer uma estrutura padronizada de construção para especificar e representar políticas que administram e controlam o acesso a recursos de QoS na rede. A estrutura é formada por um conjunto de classes e relacionamentos que são organizados em um modelo de informação independente de qualquer implementação de PDP, PEP ou mecanismos particulares de QoS. Portanto, o QPIM ainda encontra-se em desenvolvimento e não há uma padronização definida.

3.5 Trabalhos Relacionados

Um projeto, implementação e avaliação de desempenho de uma PBNM, utilizando COPS no modelo *outsourcing* com RSVP (COPS-RSVP) e o protocolo LDAP (*Lightweight Directory Access Protocol*) foi descrito por Ponnappan et al. [PON02] esquematizados a seguir.

O protocolo COPS foi utilizado como protocolo de transação de políticas e o LDAP como interface para o repositório de políticas. O LDAP é um serviço de diretório distribuído, orientado a objetos e específico para protocolo TCP/IP, e a sua evolução foi documentada pela IETF em uma série de padrões descritos por RFCs. Por ser um padrão aberto, é amplamente utilizado tanto em sistemas operacionais como repositório de informações e em aplicações em ambientes distribuídos.

Este projeto utilizou roteadores baseados no Linux e foi projetado em CORBA (*Common Object Request Broker Architecture*) para interação de componentes. O padrão

CORBA é um conjunto de especificações criado pela OMG (*Object Management Group*), um consórcio de empresas da indústria de informática que produz e mantêm especificações para operações entre aplicações corporativas [OMG02]. O CORBA define uma camada intermediária (*middleware*) que permite aos programas de aplicação se comunicarem entre si, independente de suas linguagens de programação, plataforma de *software* e de *hardware* e rede de dados sobre a qual eles se comunicam.

Cada informação da política é representada utilizando regras de políticas (*policy rules*) e cada regra consiste de "condições" e "ações", apresentadas por [MOO01a]. O editor de políticas (*Policy Editor* - PE) é uma entidade responsável para criar, modificar ou excluir regras no servidor LDAP e notificar as mudanças para o Servidor de Políticas (PS). O editor utiliza as interfaces fornecidas por um módulo chamado de adaptador de repositório de políticas (*Policy Repository Adapter* – PRA).

O módulo PRA atua como uma interface para o repositório de políticas, encapsulando suas informações e isolando-o do editor de políticas e do próprio servidor de políticas. O repositório de políticas, chamado de DS (*Directory Server*), implementa o esquema de políticas de QoS em: QPIM (*QoS Policy Information model*) descrito no *draft* de novembro de 2001 [SNI01] e LDAP descrito em [STR02]. Um esboço da arquitetura utilizada por Ponnappan é apresentado na figura 3.4.

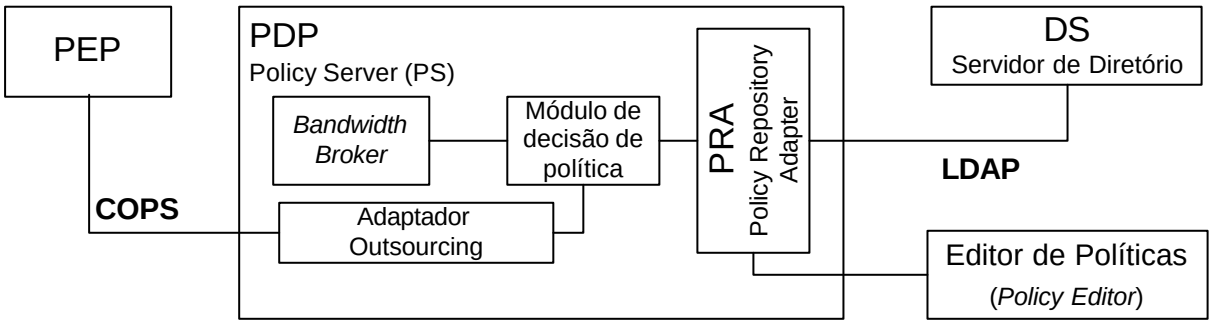

Figura 3.4 Arquitetura COPS- RSVP *Outsourcing*

Este trabalho avaliou o desempenho para um servidor de políticas de QoS utilizando conceitos de políticas orientadas a objetos, apresentou suas limitações e estudos a serem elaborados no futuro.

O trabalho de Sungjune Hong et al. [HON98] descreveu um projeto e implementação para aplicações de multimídia em tempo-real, utilizando o protocolo *Inter-ORB* (*Object Request Broker*) do ambiente CORBA e protocolo RSVP. Discute-se neste trabalho, que a

plataforma CORBA não possui um ambiente completo para aplicações de tempo-real, o que implica em definições de mecanismos de políticas e de garantias de QoS, e definição de caracerísticas do ambiente para suportar programação em tempo-real.

O protocolo CORBA GIOP (*General Inter-ORB Protocol*) define a interoperabilidade entre ORBs heterogêneos. A integração do protocolo GIOP/IIOP (*Internet Inter-ORB Protocol*) com RSVP permite a uma aplicação QoS (RECEIVER) transferir seus parâmetros de QoS para o SENDER, através de roteadores RSVP-*aware*.

A integração do IIOP com RSVP denominada de RIOP (*Real-Time Inter-ORB protocol*), foi desenvolvida utilizando como base o projeto TAO (*The ACE* [17] *ORB*) desenvolvido por [SCH97]. Este é um sistema ORB de tempo-real que fornece QoS em sistemas fim-a-fim, integrando aplicação com CORBA e este com subsistemas de entrada/saída de sistemas operacionais, protocolos de comunicação e interfaces de rede.

A desvantagem do projeto TAO é a falta do mecanismo de efetivação de QoS (*enforcement*). Para suportar o QoS *enforcement*, este trabalho integrou o GIOP/IIOP com RSVP. A arquitetura é apresentada na figura 3.5.

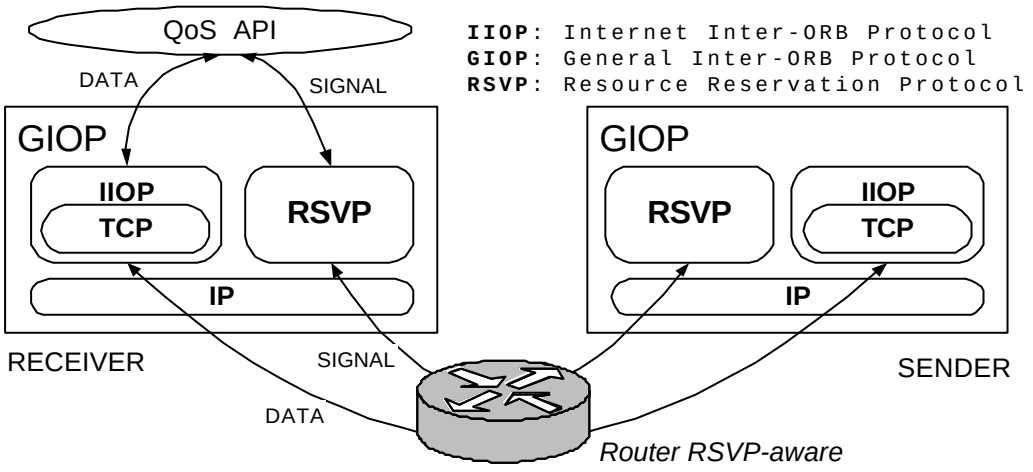

Figura 3.5 Arquitetura RIOP com garantias de QoS

A comunicação entre o SENDER e RECEIVER foi estabelecida pelo módulo RIOP que é o responsável para efetivar o processo de QoS *enforcement*. Este módulo contém o processo RSVP (RSVP *daemon*) no ORB, local onde reside o mecanismo de controle de

[17] ACE: O ACE (*ADAPTIVE Communication Environment*) é um conjunto de ferramentas de *software* desenvolvido para programação de rede. Escrito em linguagem C++, este ambiente propicia o aumento das das potencialidades dos sistemas operacionais.

admissão. O protocolo RSVP foi utilizado para reservar e controlar recursos ao longo da rota. Portanto, o RIOP foi instalado nos SENDERS e RECEIVERS.

O RIOP foi desenvolvido baseado no ORB do TAO no sistema operacional Solaris 2.5. Devido à característica do RSVP, este trabalho proporcionou o QoS *enforcement* em relação ao TAO, ao custo do excesso de comunicação do RSVP (*overhead*), devido à sinalização que o protocolo exige, havendo um efetivo controle de admissão. Não se observou a utilização de uma estrutura de políticas para controle de acesso de QoS.

O trabalho de Min Cai et al. [CAI00] descreveu um projeto e implementação para assegurar requerimentos de QoS, utilizando o modelo de implementação do serviço de fluxo CORBA (CORBA *stream service*) e RSVP, que fornece suporte para diferentes tipos de serviços de QoS.

Discute-se a limitação de CORBA, que suporta somente a semântica de requisição/ resposta e processamento de codificação e decodificação para transmissão de multimídia. Esta limitação diminui a taxa de transmissão de dados, logo, os sistemas ORB (*Object Request Broker*) têm limitações para construção de aplicações multimídia. Para solucionar estes problemas, a OMG propôs a especificação para controle e gerenciamento de fluxo de áudio e vídeo (*CORBA A/V Stream Service Specification*), baseados no modelo de referência CORBA, porém, não indica qual o mecanismo de rede deve ser empregado para assegurar garantia de QoS para fluxos de dados de multimídia.

Os autores se propuseram a utilizar o modelo de implementação IntServ/RSVP e serviço de fluxo CORBA no projeto chamado Multimídia ORBUS (MMORBUS). As aplicações reservam recursos de QoS invocando a biblioteca de APIs (*Application Program Interface*) RSVP, que podem forencer serviço controlado ou garantido. Os dispositivos de multimídia possuem interface para os módulos chamados *"Flow Endpoint"*, apresentados a seguir.

Os objetos que descrevem, os parâmetros do dispositivo multimídia são representados pela interface *MMDevice* que é componente da arquitetura de serviços de áudio e vídeo CORBA. Houve extensão da interface *StreamEndPoint* com operações para mapear QoS solicitado pela aplicação e negociar recursos de rede, representada pelo Objeto de Controle de Fluxo. A arquitetura é apresentada na figura 3.6.

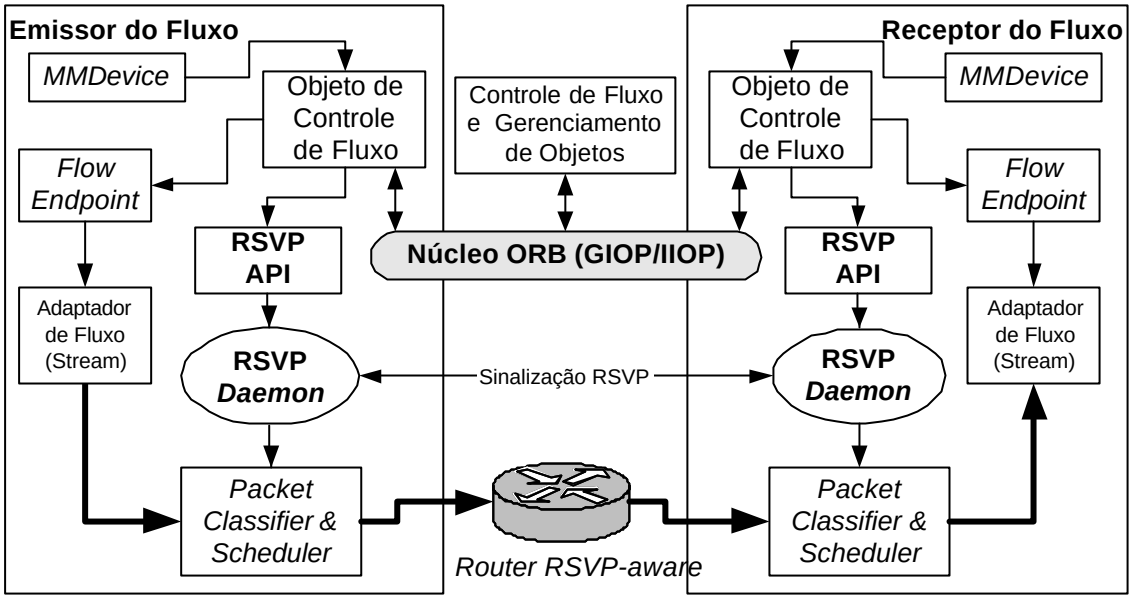

Figura 3.6 Modelo de implementação RSVP baseado em serviço CORBA

Na implementação do modelo MMORBUS, os programadores de aplicativos podem especificar os parâmetros para o nível de serviço de QoS desejado. O MMORBUS mapeia os parâmetros para a rede, negocia o QoS com a camada de rede e reserva os recursos através de API RSVP. O protocolo RSVP assegura o QoS para o fluxo de dados e o protocolo CORBA IIOP provê aos programadores flexibilidade para controlar e gerenciar os fluxos para aplicações multimídia.

Foi expandida a capacidade dos serviços CORBA de áudio e vídeo, porém não foi observada a utilização de uma estrutura de políticas para controle de acesso de QoS ou realização de testes em alguma plataforma.

3.6 Conclusões do Capítulo

Este capítulo apresentou conceitos sobre PBNM (*Policy-Based Network Management*), os elementos de sua arquitetura e modelo de processamento, o protocolo COPS e a evolução da representação de políticas orientadas a objetos.

A PBNM é um conjunto de regras que assegura a validação da política em elementos de rede que controlam políticas para operar e gerenciar requisições para recursos da rede. A PBNM padroniza e centraliza as políticas, e sua estrutura tem dois elementos para controle de políticas, definidos como PEP (*Policy Enforcement Point*) e PDP (*Policy Decision Point*).

O PEP é um dispositivo de rede que aplica a política quando invocado por um evento

externo, garantindo que uma decisão de política seja obedecida.

O PDP é um dispositivo lógico, onde as decisões de política são tomadas para si mesmo ou para outro elemento de rede que requisite semelhantes decisões.

O COPS (*Common Open Policy Service*) é um protocolo de comunicação que emprega um modelo cliente/servidor. Permite que um servidor de política PDP comunique as decisões de política ao dispositivo de rede PEP. O COPS utiliza o protocolo de transporte TCP. O protocolo COPS suporta dois modelos para controle de política: *"Outsourcing"* e "*Provisioning*".

O *Outsourcing* é um modelo de execução onde um PEP remete uma consulta ao PDP, delegando a decisão de um evento específico de política para o servidor de política PDP.

O *Provisioning* é um modelo de execução onde os elementos de rede (PEP) são pré-configurados, baseados em uma política, antes de processar os eventos provenientes da rede. O objetivo deste modelo é a distribuição da configuração de informações para um PEP.

Os atuais modelos de representação de políticas orientadas a objetos herdaram a arquitetura básica do CIM, que é um modelo conceitual de informação para representação de elementos gerenciados e não é limitada a uma implementação em particular.

O padrão CIM foi estendido pela IETF, chamado-o de PCIM, que permite a representação da informação de política de qualquer natureza. O PCIMe é um modelo de representação de políticas orientada a objetos compatível com o PCIM, diferenciando-se deste por ter introduzido em sua arquitetura novas classes em áreas que o PCIM previamente não cobria. O PCIMe adicionou elementos, associações, prioridades e estratégias de decisão.

O QPIM é um modelo que pretende estabelecer uma estrutura padronizada de construção para especificar e representar políticas que administram e controlam o acesso a recursos de QoS na rede. O QPIM ainda encontra-se em desenvolvimento e não há uma padronização definida.

Esta proposta utilizará o mecanismo *outsourcing* para consultas a um servidor de políticas PDP, utilizando o modelo XACML para suportar o protocolo RSVP. No próximo capítulo serão apresentados os modelos para representação de políticas e as especificações do XACML e a descrição de uma política.

Capítulo 4

XACML - eXtensible Access Control Markup Language

O XACML (*eXtensible Access Control Markup Language*) é uma proposta da OASIS de configuração para modelar, armazenar e distribuir políticas, sendo um modelo voltado para representação de políticas descritivas. Possui como qualidades a flexibilidade para atender a maioria das necessidades de controle de acesso e ser extensível no quesito de suporte a novas exigências no modelo de políticas.

Por se tratar de um padrão aberto e utilizar XML como base de desenvolvimento, facilitará a criação de ferramentas de controle para administração e manutenção de políticas no mercado.

A meta do XACML é definir padrões para armazenamento das políticas em serviços de diretórios e base de dados, entre outros. Devido ao padrão XACML ter sido definido recentemente, existem propostas de trabalhos de desenvolvimento com outras tecnologias de comunicação, autenticação e armazenamento de dados. Apesar de facilitar a operação com outros sistemas pelo uso do XML, ainda existem indefinições quanto a adoção de protocolos para comunicação entre o PEP e PDP utilizando XACML.

No presente estudo, será apresentado, definições de políticas de controle de acesso com XACML e protocolo RSVP para servidores de QoS, utilizado o modelo *outsourcing* do protocolo COPS, atuando como mecanismo de transporte de políticas. O armazenamento das políticas será realizado em arquivo XML e não serão definidos mecanismos para utilizar o modelo *provisioning* para controle de políticas RSVP.

4.1 Especificações XACML

O XACML utiliza para definição de sintaxe um *Schema* associado a um *namespace* na sua organização, sendo que o XML *Schema* estabelece regras para definir e validar um documento, incluindo uma linguagem de definição de tipos de dados junto com a linguagem de especificação estrutural e, o XML *namespace* é uma recomendação que descreve a sintaxe de espaço de nomes chamado de *namespace*, tornando clara as definições de nomes para as *tags* (forma de marcação de nomes), mesmo que sejam nomes iguais e que definam dados diferentes.

Segue a descrição da organização do XACML:

1. Sintaxe XACML ***policy***: utilizado para definir um modelo de política, representado por: **urn:oasis:names:tc:xacml:1.0:policy** ;
2. Sintaxe XACML ***context***: identificador de contexto, utilizado para definir um modelo de requisição e resposta, que serve tanto a uma requisição de decisão (***decision request***) enviado de um PEP a um PDP, como a decisão de autorização (***authorization decision***) encaminhada do PDP ao PEP, representado por: **urn:oasis:names:tc:xacml:1.0:context** ;
3. O XML *Signature* [DS]: padrão W3C[18] para assinatura de *namespace* importado para o XACML *Schema*, representado por: **http://www.w3.org/2000/09/xmldsig#** ;

4.2 Modelos XACML

4.2.1 Modelo do fluxo de dados

O XACML define o fluxo de dados envolvendo duas entidades principais, o PEP e o PDP. As políticas ou conjunto delas são descritas por um administrador chamado **PAP** (*Policy Administration Point*) que serão utilizadas pelo PDP.

Um evento externo invoca um PEP gerando uma requisição de acesso e envia ao ***context handler***, contendo dados necessários para uma avaliação de política. O ***context handler*** gera uma requisição no formato XACML *Request* ***context***, chamada ***decision request***, e disponibiliza ao PDP para avaliação.

O ***context handler*** é uma entidade do sistema responsável para converter os dados

[18] W3C: O World Wide Web Consortium, conhecido por W3C, é um consórcio internacional envolvendo 450 organizações para padronizar e desenvolver tecnologias para Web. URL: http://www.w3.org/

requisitados pelo PEP em forma nativa para o modelo XACML *Request* ***context***, possibilitando o processamento do PDP. A avaliação da decisão do PDP, chamada de ***authorization decision***, é gerada no modelo XACML *Response* ***context*** e encaminhada ao ***context handler*** que converte para o formato nativo encaminhando a resposta ao PEP. O ***context handler*** sempre intermediará qualquer transação entre o PEP e o PDP, garantindo a conversão de informações entre o formato XACML ***context*** e dados nativos, recebidos ou enviados ao PEP. A figura 4.1 apresenta os eventos do fluxo de dados entre PEP e PDP.

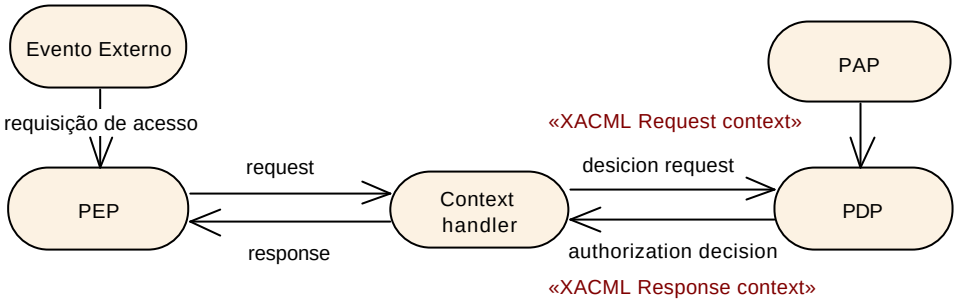

Figura 4.1 Fluxo de dados

4.2.2 Descrição de um contexto XACML (XACML context)

A entrada e saída de dados em ambientes de aplicações são definidas pelo XACML ***context*** em um XML *Schema*, isolando o núcleo XACML para validações de regras de políticas. As aplicações devem converter a representação de atributos do seu ambiente em representação de atributos do XACML ***context*** para requisições e respostas. O escopo do XACML não é especificar como as aplicações irão obter esta conversão. A figura 4.2 exemplifica o isolamento do ambiente externo com o núcleo XACML utilizando o XACML ***context***.

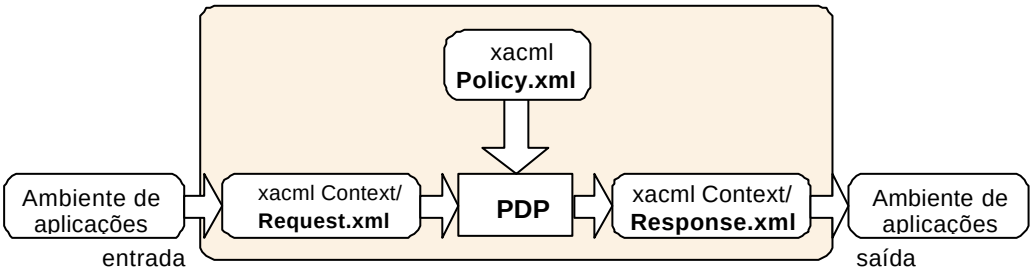

Figura 4.2 Representação XACML ***context***

4.2.3 Representação de *Policy* e *Context*

Uma Requisição <Request> é o elemento que se encontra no topo da cadeia do XACML ***context*** *schema*. O elemento <Request> é uma camada de abstração usada pela linguagem de política. Qualquer sistema proprietário DEVE gerar uma requisição de decisão chamada de ***decision request*** para o formato XACML ***context*** <Request>.

Em uma Política, o elemento de alto nível é o <PolicySet> de um XACML ***policy*** *schema*. O <PolicySet> é uma agregação de conjuntos de políticas chamadas de ***policy sets*** e políticas chamadas de ***policy***. Os elementos ***policy sets*** e ***policy*** serão detalhados no tópico a seguir.

4.2.4 Modelo de Linguagem de Política XACML

Os principais componentes do modelo de política que se encontram no topo da cadeia do XACML ***policy*** *schema* são: ***Rule***, ***Policy*** e ***Policy set***. O modelo é apresentado na figura 4.3, e descrito na seqüência.

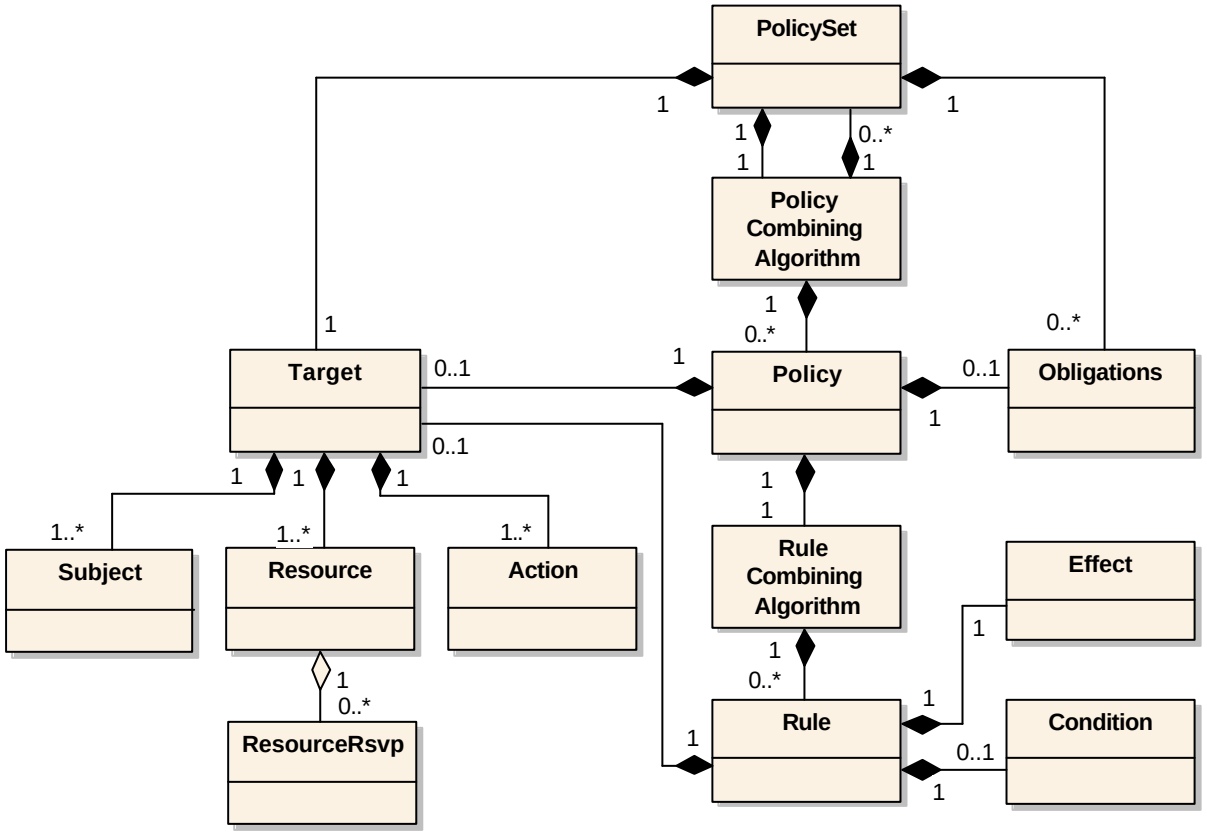

Figura 4.3 Modelo de política XACML

Rule: é um elemento que contém expressões lógicas que podem ser avaliadas isoladamente, ou seja, possibilita a uma entidade (PAP) definir regras isoladas e que serão utilizadas por um PDP na avaliação de uma ***authorization decision***.

A cardinalidade da classe ***Rule*** com outras classes é apresentado na figura 4.3. A classe ***Effect*** é definida como um atributo da classe ***Rule***. As representações de classes em XACML são definidas por elementos, e os principais componentes do elemento <Rule> são os elementos <Target> , <Effect> e <Condition>.

Uma regra <Rule> requer obrigatoriamente a definição de dois atributos, o atributo "RuleId", que é uma URN[19] para identificar a ***Rule*** e o atributo "Effect", que define o resultado da regra podendo ser "*Permit*" ou "*Deny*". Um exemplo de uma estrutura básica de uma ***Rule*** é apresentada na tabela 4.1.

Tabela 4.1 Estrutura básica de uma <Rule>.

```
<Rule RuleId=" " Effect=" ">
        <Target>...</Target>
        <Condition FunctionId=" ">...</Condition>
</Rule>
```

Policy: é um elemento ***policy*** é a menor entidade que deve estar presente para a avaliação de uma ***authorization decision*** do PDP. A Policy se baseia em uma ou mais regras em conjunto com um algoritmo que especifica um procedimento, para avaliação do resultado.

O modelo apresentado na figura 4.3 descreve a cardinalidade da classe ***Policy*** com outras classes. A classe ***Rule Combining Algorithm*** é definida como um atributo da classe ***Policy***. As classes são representadas por elementos em XACML e os principais componentes da ***Policy*** são os seguintes elementos: <Target>, <Rule>, <Obligations> e o atributo "RuleCombiningAlgId".

A definição de uma ***policy*** requer sua identificação descrita pelo atributo "PolicyId" e a especificação do algoritmo no atributo "RuleCombiningAlgId", para avaliação das regras.

O elemento <Target> é requerido na estrutura da <Policy> e define a aplicabilidade da mesma para um conjunto de ***decision requests*** definido em um XACML *Request*. Se os elementos do <Target>, descritos a seguir neste tópico, forem definidos como <AnySubjects>,

[19] **URN**: "***Uniform Resource Name***" refere-se a um subconjunto da URI (*Uniform Resource Identifier*) que identifica recursos, visando a sua persistência e independência de localização e facilitando a migração para outro *namespace*, definido na [RFC 2141]. A URI é um conjunto de caracteres utilizado para identificar um recurso físico ou abstrato definido na [RFC2396].

<AnyResources> e <AnyActions>, a <Policy> avalia a requisição pelas regras e condições sem considerar algum sujeito, recurso ou ação para o resultado, já que não foram especificados.

O elemento <Rule> define uma seqüência de autorizações que devem ser combinadas de acordo com a definição do algoritmo da ***policy***, e pode ser definido uma ou mais vezes na política.

O elemento <Obligations> é um elemento opcional e descreve uma ***conjunctive sequence***[20] de uma ou mais obrigações (*obligation*) que DEVEM ser executadas pelo PEP em conjunção com uma ***authorization decision*** do PDP, que não interpreta o conteúdo das obrigações. O PEP deve interpretar as informações contidas nas <Obligations> e aplicá-las. Um exemplo de estrutura básica de uma ***Policy*** é apresentada na tabela 4.2.

Tabela 4.2 Estrutura básica de uma <Policy>.

```
<Policy PolicyId=" " RuleCombiningAlgId=" ">
    <Target>
        <Subjects>...</Subjects>
        <Resources>...</Resources>
        <Actions>...</Actions>
    </Target>
    <Rule RuleId=" " Effect=" ">
        <Target>...</Target>
    </Rule>
</Policy>
```

Policy set: é um elemento que pode conter um conjunto de ***Policy*** ou ***PolicySet***, e deve especificar um algoritmo que descreve um procedimento para avaliação de resultado.

O modelo apresentado na figura 4.3, apresenta a cardinalidade da classe ***PolicySet*** com as outras classes. A classe ***Policy Combining Algorithm*** é definida como um atributo da classe ***PolicySet***. As representações das classes XACML seguem os mesmos procedimentos descritos anteriormente. Os quatro principais componentes da ***PolicySet*** são os elementos <Target>, <Policy>, <Obligations>, descritos acima, e o atributo "PolicyCombiningAlgId".

A definição de uma ***PolicySet*** requer sua identificação descrita pelo atributo "PolicySetId" e a especificação do algoritmo no atributo "PolicyCombiningAlgId" a ser utilizado para avaliação das regras. Para formar a estrutura de uma <PolicySet> são requeridos: o elemento <Target>, os elementos <Policy> e <PolicySet> que podem ser usados uma ou mais vezes na política, sendo opcional o uso de <Obligations>.

---

[20] ***conjunctive sequence***: uma seqüência de elementos de um sistema combinatório lógico que representa simbolicamente relações entre entidades, usando um operador lógico '**AND**'.

Um exemplo de uma estrutura ***PolicySet*** identificada como "...:**policyset**", contendo uma ***Policy*** identificada como "...:**policy1**" e os elementos de *Obligations*, é apresentada na tabela 4.3.

Tabela 4.3 Estrutura de uma <PolicySet>.

```
<PolicySet PolicySetId="...:policyset" PolicyCombiningAlgId=" ">
    <Target>      <... .....>
    </Target>

    <!-- Início da descrição da Política 1 com a descrição de uma Obrigação -->
    <Policy PolicyId="...:policy1" RuleCombiningAlgId=" ">
        <Target>      <... .....>
        </Target>
    <Rule RuleId=" " Effect=" ">
        <Target>      <... .....>
        </Target>
    </Rule>
    <Obligations>
        <Obligation ObligationId="...:policy1:obligation-1"FulfillOn="Permit"></Obligation>
    </Obligations>
    </Policy>
    <!-- Início de descrição de outras Políticas... -->

    <!-- Início da descrição de uma Obrigação da PolicySet -->
    <Obligations>
       <Obligation  ObligationId="...:policyset:obligation-2" FulfillOn="Deny"></Obligation>
    </Obligations>
</PolicySet>
```

Target: é o elemento que identifica um conjunto de ***decisions requests*** que um elemento de uma hierarquia superior deseja avaliar. O <Target> deve estar definido nos elementos <PolicySet>, <PolicySet> e <Rule>.

O target contém as definições para os seus elementos <Subjects>, <Resources> e <Actions> que são requeridos na especificação do XACML, interpretados como:

- Um elemento <Subjects> especifica atributos de um ***subject*** que pode ser encontrado em um XACML ***context***. Um *subject* é definido como um "ator" e pode ser referenciado por seus atributos.
- Um elemento <Resources> especifica atributos de um ***resource*** que pode ser encontrado em um XACML ***context***. Um ***resource*** é definido como um dado, dispositivo ou componente de um sistema.
- Um elemento <Actions> especifica atributos de uma ***action*** que pode ser encontrada em um XACML ***context***. Uma ***action*** define uma operação requerida em um ***resource***.

4.2.5 Algoritmos para decisão de autorização

Os algoritmos especificados pelo XACML são usados para determinar o resultado de várias ***rules*** que podem estar dentro de uma ***policy***.

O algoritmo ***rule-combining algorithm*** é um atributo da ***Policy*** e o algoritmo ***policy-combining algorithm*** é um atributo da ***PolicySet***, discutidas anteriormente. Segue uma breve descrição dos algoritmos:

- *Deny-overrides*: se em uma avaliação de um conjunto de regras o resultado é uma negação (*Deny*), esta tem precedência sobre qualquer outro resultado. Pode ser utilizada tanto por ***rule-combining*** como por ***policy-combining***.
- *Permit-overrides*: se em uma avaliação de um conjunto de regras o resultado é uma permissão (*Permit*), esta tem precedência sobre qualquer outro resultado. Pode ser utilizada tanto por ***rule-combining*** como por ***policy-combining***.
- *First-applicable*: as avaliações das regras seguem a seqüência que está descrita na *policy*. Se em uma avaliação de um conjunto de regras, a primeira regra possa ser aplicada, prevalece seu resultado, podendo ser uma permissão (*Permit*) ou uma negação (*Deny*). Pode ser utilizada tanto por ***rule-combining*** como por ***policy-combining***.
- *Only-one-applicable*: se em uma avaliação de um ***PolicySet***, possa ser aplicada a uma *policy*, prevalece seu resultado, caso contrário pode retornar "*Indeterminate*" ou "*NotApplicable*" se não for aplicada a nenhuma das políticas da ***policySet***. Este algoritmo pode ser aplicado somente a uma ***PolicySet***.

A avaliação que o algoritmo utiliza através de uma ***Policy*** e por uma ***PolicySet*** são diferentes. Uma descrição de *pseudo*-códigos são apresentados na especificação da OASIS, no anexo 'C' [OAS03].

4.3 Descrição de uma Política

Um controle de acesso de política utilizando XACML deve conter um XACML *Request* ***context***, um XACML *Policy* com um algoritmo especificado para avaliação das regras, e um XACML *Response* ***context***, descritos e comentados na seqüência, através de um

exemplo que terá por regra: *"um usuário autorizado pode acessar um recurso (login) de um servidor de multimídia, identificado pelo nome e no período entre às 09:00 horas e às 17:00 horas"*. A descrição completa da política deste exemplo se encontra no Anexo A.

4.3.1 Descrição de um XACML *Request context*

Uma requisição de decisão, ***decision request***, deve seguir a estrutura padronizada pela OASIS e submeter-se ao PDP para decisão de autorização, ***authorization decision***. A estrutura <Request> requer um cabeçalho com uma estrutura semelhante à definição de uma estrutura de ***policy***, contendo os elementos <Subject>, <Resource>, <Action> e <Enviroment>, e cada elemento contém uma seqüência de atributos associados para detalhar a solicitação que descrevem a requisição ao PDP.

Somente o elemento <Subject> pode ser referenciado mais de uma vez em uma requisição. O elemento <Enviroment> é opcional. Quando não é necessário especificar um valor para um determinado elemento na requisição, deve-se representá-lo como um elemento vazio </element_name>.

Descrição do cabeçalho da requisição

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
[05] http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
```

1. [01] Linha padrão XML com uma *tag* indicando a versão e codificação utilizada.
2. [02] Uma *tag* que indica uma ***Request*** e insere a referência do XACML ***context*** *schema*.
3. [03-04-05] Declarações de *namespace* do XACML.
4. [05] Define a URL[21] para o XACML ***context*** *schema*..

Na seqüência, devem ser definidos os elementos e atributos necessários para uma requisição de decisão para o PDP processar. Segue um exemplo resumido com o formato e descrição dos elementos necessários a uma requisição: <Subject>, <Resource>, <Action>.

[21] URL: *Uniform Resource Locator*, refere-se a um subconjunto da URI (*Uniform Resource Identifier*) que identifica recursos via representação do seu mecanismo primários de acesso, como por exemplo, localização na rede, arquivo, web, entre outros. A URI é um conjunto de caracteres utilizado para identificar um recurso físico ou abstrato definido na RFC2396.

Representação do elemento <Subject>: este elemento especifica um sujeito ou ator, seguido de uma seqüência de seus atributos.

```
[06] <Subject>
[07]   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[08]     DataType="http://www.w3.org/2001/XMLSchema#string">
[09]       <AttributeValue>Ben-Hur</AttributeValue>
[10]    </Attribute>
[11] </Subject>
```

5. [06] *Tag* indicando um elemento <Subject>. Em uma requisição pode existir vários *subjects* e cada um contendo um ou mais atributos <Attribute>. Neste exemplo está representado somente um *subject* com um atributo na seqüência.
6. [07-08] Um atributo representa uma característica de um elemento e a definição do tipo do atributo é determinado pelo "DataType", no caso acima, como caracteres (*string*). Neste caso, o identificador AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" representa a identidade do *subject*.
7. [09] O valor entre as *tags* do <AttributeValue>, neste exemplo, "**Ben-Hur**", expressa a identidade do *subject*.
8. [10-11] Representa o fim dos *tags* do *attribute* e *subject*.

Representação do elemento <Resource>: este elemento especifica informações do *resource* para qual o acesso é requisitado, seguido de uma seqüência de seus atributos.

```
[12] <Resource>
[13]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
[14]         DataType="http://www.w3.org/2001/XMLSchema#string">
[15]         <AttributeValue>MultimidiaServer</AttributeValue>
[16]    </Attribute>
[17] </Resource>
```

9. [12] *Tag* indicando um elemento <Resource>. Em uma requisição existe somente um *resource*. Neste exemplo está representado um *resource* com um atributo.
10. [13-14] Um atributo com o identificador AttributeId="...:resource-id" representa a identidade do *resource* e a definição do tipo do atributo é determinado pelo "DataType", no caso acima, como caracteres (*string*). Este atributo especifica a identidade do *resource* para o acesso requisitado.
11. [15] O valor entre as *tags* do <AttributeValue>, neste exemplo,

"**MultimidiaServer**", é o nome do *resource*.

12. [16-17] Representa o fim dos *tags* do *attribute* e *resource*.

Representação do elemento <Action>: o elemento *action* especifica a ação requisitada para o *resource*, descrito em um atributo ou mais, associados ao elemento *action*.

```
[18] <Action>
[19]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action-id:ServerAction"
[20]        DataType="http://www.w3.org/2001/XMLSchema#string">
[21]        <AttributeValue>login</AttributeValue>
[22]    </Attribute>
[23] </Action>
[24]</Request>
```

13. [18] *Tag* indicando um elemento <Action>. Em uma requisição existe somente um elemento *action* seguido de um ou mais atributos. Neste exemplo está representado apenas um *action* com um atributo.
14. [19-20] Um atributo com o identificador AttributeId="...: ServerAction" representa a identidade do *action* e a definição do tipo do atributo é determinado pelo "DataType", no caso acima, como caracteres (*string*).
15. [21] O valor entre as *tags* do <AttributeValue>, neste exemplo, "*login*", expressa o nome do *action* e a ação para o *resource*.
16. [22-23-24] Representa o fim da requisição <Request> pelos *tags* dos *attribute*, *action* e *request*.

4.3.2 Descrição de um XACML *Policy*

Uma ***Policy*** é composta por um cabeçalho, descrição de texto opcional, *target*, uma ou mais *rules* e conjunto opcional de obrigações (*obligations*).

Descrição do cabeçalho de uma *policy*: é semelhante ao cabeçalho de requisição.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
[05] http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-policy-01.xsd"
[06] PolicyId="urn:oasis:names:tc:xacml:1.0:example:PolicyServer01"
[07] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm: first-applicable">
```

1. [01] Linha padrão XML com uma *tag* indicando a versão e codificação utilizada.

2. [02] Insere a referência da política XACML.
3. [03-04-05] Declarações de *namespace* do XACML.
4. [05] Define a URL para o XACML *Policy schema*.
5. [06] Atribui um nome para a ***policy*** e deve ser única para um PDP, mas pode ser referenciada por outras ***policies***.
6. [07] Especifica o algoritmo que será utilizado para decisão de uma autorização e pode conter várias regras em uma política. Neste exemplo, a primeira regra aplicável (*Permit* ou *Deny*), encerra a avaliação e retorna o resultado.

Descrição da *decision requests* que se aplica a esta *policy*:

```
[08] <!-- Esta política se aplica somente para requisições no servidor chamado MultimidiaServer -->
[09] <Target>
[10]    <Subjects>
[11]        <AnySubject/>
[12]    </Subjects>
[13]    <Resources>
[14]        <Resource>
[15]            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[16]                <AttributeValue
[17]                    DataType="http://www.w3.org/2001/XMLSchema#string">MultimidiaServer
[18]                </AttributeValue>
[19]                <ResourceAttributeDesignator
[20]                    DataType="http://www.w3.org/2001/XMLSchema#string"
[21]                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
[22]            </ResourceMatch>
[23]        </Resource>
[24]    </Resources>
[25]    <Actions>
[26]        <AnyAction/>
[27]    </Actions>
[28] </Target>
```

7. [08] Indica uma *tag* com comentário.
8. [09 a 28] A sessão *Target* é útil para criar um índice para uma ***PolicySet***. Neste caso não há referências para <Subjects> nem para <Actions>. Se não existir nenhuma referência nesta sessão, significa que a *policy* é aplicável a qualquer ***decision requests***.
9. [13 a 24] As *tags* <Resources> possuem informações para uma requisição de decisão.
10. [14-15] Uma *tag* <Resource> contém um elemento "ResourceMatch", e este contêm atributos e elementos. O atributo "MatchId" define uma função para comparar o valor <AttributeValue> da ***policy*** com o valor do *resource* definido

no XACML *request* ***context***.

11. [16-17-18] O elemento <AttributeValue> possui um "DataType" tipo *string* com o valor, neste exemplo, "**MultimidiaServer**".
12. [19-20-21] O elemento <ResourceAttributeDesignator> identifica um ou mais valores de atributos em um elemento <Resource> de um XACML *request* ***context***. O valor será identificado pelo "DataType" e pelo AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" que indica uma URI do *resource* no XACML *request* ***context***.
13. [22 a 28] Representa o fim da sessão *Target* indicando fim de *tags*.

Descrição de uma Regra (*Rule*): uma regra deve possuir um identificador único. Os principais componentes de uma *rule* são: uma ***target***, um ***effect*** e uma ***condition***, descritos a seguir.

```
[29] <Rule RuleId="urn:oasis:names:tc:xacml:1.0:LoginRule" Effect="Permit">
[30] <!-- Regra para ação de login especificada no XACML request context -->
[31] <Target>
```

14. [29] Indica o nome de uma Regra (*Rule*) chamada "...xacml:1.0:LoginRule" e o efeito "**Permit**", caso seja validada.
15. [30] Insere um comentário na *Rule*.
16. [31] Insere um elemento <Target> que descreve um conjunto de ***decision requests*** aplicáveis pela *Rule*. Se os elementos não encontram valores especificados na "***Rule Target***", a ***Rule*** não pode ser avaliada e retorna um valor de acordo com o tipo de algoritmo especificado no cabeçalho da ***Policy***.

    Nota: Uma "***Target***"contêm os elementos "***Subjects***", "***Resources***" e "***Actions***" e possui uma ***disjunctive sequence***[22] destes elementos {"***Subjects***" OR "***Resources***" OR "***Actions***"} para avaliação da regra (***rule***).

[22] ***disjunctive sequence***: uma seqüência de elementos de um sistema combinatório lógico que representa simbolicamente relações entre entidades, usando um operador lógico '**OR**'.

```
[32]  <Subjects>
[33]     <Subject>
[34]        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[35]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Ben-Hur
[36]           </AttributeValue>
[37]           <SubjectAttributeDesignator
[38]              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[39]              DataType="http://www.w3.org/2001/XMLSchema#string"/>
[40]        </SubjectMatch >
[41]     </Subject>
[42]  </Subjects>
```

17. [32] Especifica a *tag* de início de uma <Subjects>.
18. [33 a 41] Insere a *tag* do elemento <Subject> que especifica qual é o ***subject*** que deve ser encontrado na ***decision requests***. Este elemento tem uma ***conjunctive sequence*** [23], neste caso, de um elemento "**SubjectMatch**".
19. [34 a 40] O elemento <SubjectMatch> deve identificar o valor de um ***subject*** e pesquisar em seus dois elementos, declarados entre as tags <SubjectMatch>, de acordo com a função declarada no atributo "MatchId", neste caso: "function:string-equal".
20. [35-36] A *tag* <AttributeValue> contém um "DataType", neste caso tipo *string*, que será utilizado pela função e o valor do atributo a ser pesquisado, ou seja, o valor "**Ben-Hur**".
21. [37-38-39] O elemento <SubjectAttributeDesignator> é usado para localizar a identidade do ***subject***, utilizando o "AttributeId="...:subject:subject-id" ".
22. [41-42] Representa o fim do elemento <Subjects> pelos *tags*: *Subjects* e *Subjects*.

[23] ***conjunctive sequence***: uma seqüência de elementos de um sistema combinatório lógico que representa simbolicamente relações entre entidades, usando um operador lógico '**AND**'.

```
[43]    <Resources>
[44]        <AnyResource/>
[45]    </Resources>
[46]    <Actions>
[47]        <Action>
[48]            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[49]                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login
[50]                </AttributeValue>
[51]                <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
[52]                    AttributeId="urn:oasis:names:tc:xacml:1.0:action-id:ServerAction"/>
[53]            </ActionMatch>
[54]        </Action>
[55]    </Actions>
[56] </Target>
```

23. [43 a 45] A *tag* <Resources> pode conter uma ***disjunctive sequence*** de *Resource* ou um elemento <AnyResource>.
24. [44] Um elemento <AnyResource> é um elemento especial que procura algum ***subject*** em um XACML *Request* ***context***.
25. [46 a 55] As *tags* <Actions> podem conter uma ***disjunctive sequence*** de *Actions* ou um elemento <AnyAction>.
26. [47 a 54] As *tags* <Action>, neste exemplo, têm uma ***conjunctive sequence*** de elemento "**ActionMatch**".
27. [48 a 53] O elemento <ActionMatch> deve identificar atributos de uma <Action> em um XACML *request* ***context***, pesquisando um valor em seus dois elementos, declarados entre as *tags* <ActionMatch>, de acordo com a função declarada pelo "MatchId".
28. [49-50] A *tag* <AttributeValue> contém a descrição de um "DataType", neste exemplo tipo *string*, que será utilizado pela função e o valor do atributo a ser pesquisado, ou seja, o valor "**login**".
29. [51-52] O elemento <ActionAttributeDesignator> é usado localizar um ou mais atributos do ***action*** especificado no "AttributeId="...: action-id:ServerAction "".
30. [55-56] Representa o fim dos elementos identificados pelas *tags Actions* e *Target*.

```
[57] <!-- Permite executar logins compreendido no horário de 09:00 às 17:00 horas -->
[58] <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[59]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">  f(1)
[60]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">  f(2)
[61]       <EnvironmentAttributeDesignator
[62]       DataType="http://www.w3.org/2001/XMLSchema#time"
[63]       AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
[64]     </Apply>
[65]     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00
[66]     </AttributeValue>
[67]   </Apply>
[68]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">  f(3)
[69]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">  f(4)
[70]       <EnvironmentAttributeDesignator
[71]       DataType="http://www.w3.org/2001/XMLSchema#time"
[72]       AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
[73]     </Apply>
[74]     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00
[75]     </AttributeValue>
[76]   </Apply>
[77] </Condition>
[78] </Rule>
[79] <Rule RuleId="urn:oasis:names:tc:xacml:1.0:FinalRule" Effect="Deny"/>
[80] </Policy>
```

$f(n)$ = *representa uma função para exemplificar a aplicação da <condition>*

31. [57] Insere um comentário sobre as condições da *Rule*.
32. [58 a 77] Descreve uma *tag* <Condition>. A condição deve ser avaliada como verdadeira para uma regra ser aplicada. A condição determina se a requisição a ser avaliada está compreendida no horário de 09:00 horas às 17:00 horas.
33. [58] O atributo "FunctionId"da <Condition> utiliza uma função lógica 'AND' a ser avaliada pela expressão das funções descritas na seqüência.
34. [59 a 67] O elemento <Apply> indica a aplicação de uma função para argumentos, e pode ser aplicada em alguma combinação com outros elementos <Apply>.

Nota: Um elemento <Apply> possui um atributo tipo função chamado de "FunctionId" e uma opção para escolha de um entre vários elementos declarados, definidos entre suas *tags* de início e de fim. Foi convencionado uma referência numérica para elementos <Apply>, atributos e elementos escolhidos, apresentados entre parênteses "$_{(n)}$".

a. Um elemento escolhido, definido entre as tags <Apply>, é chamado de argumento.
b. Um atributo "FunctionId" é uma função aplicada ao argumento

escolhido para validar o resultado.

35. [59] O elemento <Apply>$_{(1)}$ é combinado com o <Apply>$_{(2)}$ contido entre suas *tags*.
36. [59] O atributo "FunctionId"$_{(1)}$ especifica uma função $f(1)$ "...:function:time-greater-than-or-equal" para validar o valor do argumento$_{(1)}$ e demais que estejam definidos entre suas *tags* de início e de fim.
37. [60 a 64] A *tag* do elemento <Apply>$_{(2)}$ define a função $f(2)$ "FunctionId" como "..:function:time-one-and-only" para ser aplicada ao argumento$_{(2)}$.
38. [61-62-63] O elemento <EnvironmentAttributeDesignator>, argumento$_{(2)}$, obtém o horário local e possui um tipo de dado definido como "*time*".
39. [65-66] A *tag* <AttributeValue>, argumento$_{(1)}$, contém o valor definido na política igual a "**09:00:00**" horas.
40. [67] Indica fim de *tag* do elemento <Apply>$_{(1)}$ e o resultado é válido se:
    a. A $f(2)$ possuir somente um argumento$_{(2)}$ válido (horário do sistema);
    b. A $f(1)$ compara se argumento$_{(2)}$ = argumento$_{(1)}$. Será um resultado válido se o horário atual do sistema for = **09:00:00** horas.
41. [68] O elemento <Apply>$_{(3)}$ é combinado com o elemento <Apply>$_{(4)}$ contido entre suas *tags*.
42. [68] O atributo "FunctionId" especifica uma função $f(3)$ "...:function:time-greater-than-or-equal" para validar o valor do argumento$_{(3)}$ e outros argumentos que estejam definidos entre suas *tags* de início e de fim.
43. [69 a 73] A *tag* do elemento <Apply>$_{(2)}$ define a função $f(4)$ "FunctionId" como "..:function:time-one-and-only" para ser aplicada ao argumento$_{(4)}$.
44. [70-71-72] O elemento <EnvironmentAttributeDesignator>, argumento$_{(4)}$, obtém o horário local e possui um "DataType", no caso tipo "*time*".
45. [74-75] A *tag* <AttributeValue>, argumento$_{(3)}$, contém o valor definido na política igual a "**17:00:00**" horas.
46. [76] Indica fim de *tag* do elemento <Apply>$_{(1)}$ e o resultado é válido se:
    a. A função $f(4)$ possuir somente um argumento$_{(4)}$ válido (horário do sistema);
    b. A função $f(3)$ resultar em argumento$_{(4)}$ = argumento$_{(3)}$. Será um

resultado válido se o horário atual do sistema for = a **17:00:00** horas.

47. [77] Representa o fim da *tag* <Condition>. A ***condition*** avalia o resultado final com a operação lógica de: $f(1)$ 'AND' $f(3)$.
    a. Será considerada uma <Condition> válida se a ***decision requests*** for recebida no horário de **09:00:00** às **17:00:00** horas.
48. [78] Representa o fim dos elemento ***Rule*** representado por uma tag.
49. [79] Indica o nome de outra Regra (*Rule*) chamada "...:xacml:1.0:**FinalRule**" e o efeito "**Deny**" caso a primeira regra não possa ser avaliada.
50. [80] Representa o fim da política com a *tag* <Policy>.

4.3.3 Resposta de um XACML *Response context*

O elemento <Response> encapsula a resposta da ***authorization decisions*** produzido pelo PDP. Pode conter uma seqüência de um ou mais resultados por requisição. Um sistema proprietário que utilize a especificação XACML deve transformar um XACML ***context*** <Response> para um formato de *authorization decisions*.

Descrição do cabeçalho da resposta: segue o mesmo modelo de uma requisição, indicando o elemento *Response* no cabeçalho.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Response xmlns="urn:oasis:names:tc:xacml:1.0:context"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
[05] http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
[06]    <Result>
[07]    <Decision>Permit</Decision>
[08]        <Status>
[09]            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
[10]        </Status>
[11]    </Result>
[12] </Response>
```

1. [01] Linha padrão XML com uma *tag* indicando a versão e codificação utilizada.
2. [02] Uma *tag* que indica uma ***Response*** e insere a referência do XACML ***context*** *schema*.
3. [03-04-05] Declarações de *namespace* do XACML.
4. [05] Define a URL para o XACML ***context*** *schema*.
5. [06 a 11] O elemento <Result> possui o resultado da avaliação da ***decision***

requests submetida à ***policy***, um elemento <Status> e opcionalmente elemento <Obligations>.

6. [07] O elemento <Decision> informa a decisão do PDP. Neste exemplo, o resultado é "**Permit**". Uma política pode retornar uma avaliação como "**Permit**", "**Deny**", "**NotApplicable**" or "**Indeterminate**".
7. [08-09-10] Declaração do elemento <Status> que indica a avaliação com erros se existirem.
8. [11-12] *Tags* de finalização do elemento *Result* e *Response*.

4.3.4 XACML e COPS

A OASIS ainda não definiu qual protocolo de comunicação a ser usado para troca de mensagens de requisições entre o PDP e o PEP. Nesta proposta, será utilizada a seqüência de comunicação e controle através do protocolo COPS no modelo *outsourcing* devido ao padrão já estabelecido pela IETF.

As mensagens deverão ser encapsuladas em um objeto nas mensagens COPS, embora importante, não é o enfoque deste trabalho definir qual objeto irá encapsulá-las, visto este processo ser conhecido, experimentado e já utilizado por vários autores. Esta proposta restringe a utilização do COPS na seqüência necessária e correta para a utilização entre o PEP e PDP com o XACML, mapeando a solução como uma proposta de comunicação a ser definida pela OASIS.

Esta proposta não abordará em qual objeto do protocolo COPS será encapsulada a mensagem de requisição XACML, podendo ser uma expansão futura deste modelo.

4.4 Conclusões do Capítulo

Este capítulo apresentou o XACML, modelo proposto pela OASIS para representação de políticas baseado em XML. O XACML encontra-se em desenvolvimento e, até o momento da apresentação desta dissertação, ainda não possui definições quanto ao protocolo de comunicação para troca de mensagens entre PEP e PDP, nem definições quanto ao local para o armazenamento das políticas.

O XACML especifica um identificador de contexto XACML (XACML *context*) e um modelo de política XACML (XACML *policy*). O XACML *context* é um modelo para

representação de requisição e resposta, isolando o núcleo XACML para validações de regras e políticas, e o XACML *policy* é utilizado para descrição das políticas. A especificação XACML é suportada por uma série de definições de: tipos, atributos, identificadores, funções e outros, utilizados no XACML *context* e *policy*, que definem as sintaxes utilizadas na descrição de requisições, respostas e definição de políticas.

Embora a sintaxe do XACML seja longa para muitas das definições descritas acima, o XACML mostrou-se prático em sua utilização, tanto para simples descrições de políticas como para modelar representações de requisições. Entretanto, em uma utilização mais complexa, por exemplo, aumentar a quantidade de restrições de acessos e regras em uma definição de política, seu uso é dispendioso e dificulta a compreensão se não houver suporte de ferramentas para o ambiente XML. A utilização de uma ferramenta que ofereça um ambiente de desenvolvimento XML facilita a descrição em XACML, as definições de regras, funções e validações com o XML *Schema*, tal como uso da ferramenta XMLSpy® [24].

Conclui-se que o XACML não suporta o processamento para o modelo *provisioning*. O retorno de informações para o PEP é especificado através da definição do algoritmo na *policy* para determinar um resultado e, opcionalmente, informações de elemento *Obligations*, que não são processadas pelo PDP.

Os algoritmos para decisão do modelo XACML, que determinam o resultado da avaliação de regras, proporcionam maior flexibilidade para definição de políticas, sendo esta uma vantagem do modelo de arquitetura oferecido pela OASIS.

No próximo capítulo, será apresentada a proposta desta dissertação utilizando o protocolo RSVP e XACML.

[24] XMLSpy: ferramenta para ambiente de desenvolvimento XML da Altova GmbH. O XMLSpy® oferece um ambiente de desenvolvimento para XML, XML Schema, XSL/XSLT, SOAP, WSDL e tecnologias de Web services. Foi utilizado o XMLSpy v5 para definição e validação de tipos de elementos nesta proposta.

Capítulo 5

Proposta

O presente estudo apresenta um modelo *outsourcing* para controle de políticas em um cenário de QoS RSVP, utilizando o PEP em servidores QoS RSVP, o PDP em um servidor de políticas para decisão de controle de admissão e o XACML para definição de políticas.

Faz-se mister que o XACML seja estendido para suportar os dados da configuração RSVP. Como o XACML não especifica um protocolo para transação de políticas entre PEP e PDP, será utilizado o protocolo COPS para mapear as funcionalidades dos eventos RSVP *daemon* entre o PEP e o PDP.

Em uma reserva de recursos RSVP, tradicionalmente a aplicação invoca o RSVP *daemon* informando os dados do fluxo desejado. Nesta proposta, a aplicação solicita ao PEP a reserva de um fluxo RSVP. O PDP avalia o direito de acesso do usuário e do servidor de aplicação QoS, através de uma consulta de política definida em XACML, e informa ao PEP os parâmetros para a reserva do fluxo. O PEP invoca o RSVP *daemon*, tornando a aplicação passiva neste processo, determinando a principal mudança conceitual em uma reserva de fluxo RSVP. Portanto, este modelo não se aplica a roteadores RSVP-*Aware*.

O armazenamento das políticas ocorre em um arquivo XML, devido à especificação XACML estar em andamento para definição de uma ou mais soluções de armazenamentos.

O protocolo COPS é utilizado na seqüência de eventos, porém não será abordado em qual objeto do protocolo COPS será encapsulada a mensagem de requisição XACML, podendo ser uma expansão futura deste modelo, assunto já discutido anteriormente.

5.1 Extensão do XACML

Seguindo as especificações XACML para suportar a descrição de políticas de RSVP, foi estendida a classe ***Resource***, definida no modelo de linguagem de política XACML. A classe estendida é chamada de *ResourceRsvp* e possui descrição de dados para suportar classes de aplicações QoS, tipo de serviço, estilo e especificação para o tráfego de dados ***Tspec***, necessários à configuração em Servidores QoS RSVP. Segue a descrição da classe estendida *ResourceRsvp,* apresentada na figura 5.1.

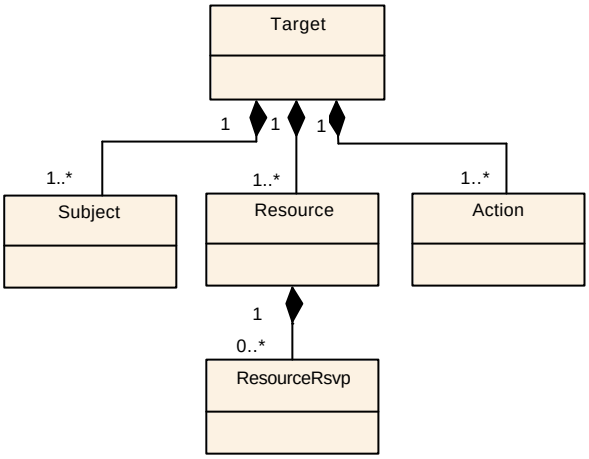

Figura 5.1 Extensão da classe *ResourceRsvp*

A classe *ResourceRsvp* esta contida na classe ***Resource***, e foi definida uma cardinalidade '0..*', ou seja, podendo ou não ser declarado em uma ***Policy***.

As classes em XACML são definidas como elementos. O elemento "***Target***" contêm "***Subjects***", "***Resources***" e "***Actions***", elementos estes que definem o modelo de classes e já foram apresentados no capítulo anterior. O "***Resources***" contém o elemento ***Resource***, sendo obrigatória sua definição em uma política, podendo ser declarado uma ou mais vezes. O *ResourceRsvp* foi definido como um elemento opcional dentro do ***Resource***, e quando aquele é utilizado pode ser declarado mais de uma vez. Isto se torna útil, quando um ***Resource*** apresenta mais de um tipo de serviço RSVP em um mesmo servidor.

Esta flexibilidade facilita a administração e definição de uma política, possibilitando conter definições de vários ***Resource*** (Servidores) com suporte para mais de um serviço.

Foram introduzidos elementos no XACML ***policy*** *schema* para definição dos parâmetros RSVP, com restrições aos tipos de dados suportados para cada elemento, e os limites mínimos e máximos de dados suportados no ***Tspec*** descritos nas RFC 2210 e RFC

2215. O objeto ***Tspec*** é apresentado na tabela 5.1, segundo definição XDR (*External Data Representation*) para tipos de dados definidos pela IEEE (*Institute of Electrical and Electronic Engineers*), descritas na RFC1832.

Tabela 5.1 Tipo de dados e limites para o ***Tspec***

Definição ***Tspec*** (XDR)	Unidade de medida	Tipo de dado	Limites definidos
struct {			
float r;	bytes	float	[1 - 40TB]
float b;	bits/sec	float	[1 - 250GB]
float p;	bytes	float	[1 - 40TB]
unsigned m;	microseconds	unsigned integer	[1 - 4294967296]
unsigned M;	bytes	unsigned integer	[1 – 4294967296]
}TOKEN_BUCKET_TSPEC;			

Descrição do ***Tspec***:

```
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)
```

5.1.1 Descrição dos elementos definidos no XACML *policy* ***schema*:**

O XACML ***policy*** *schema* define a estrutura para validação de definição de políticas no *schema* chamado *"**cs-xacml-schema-policy-01.xsd**"*. Sua atual versão é a 1.0 de acordo com a especificação OASIS *standard* publicada em Fev/2003 [OAS03]. A extensão do XACML ***policy*** *schema* descrita na proposta será denominada *"**cs-xacml-schema-policy-01-rsvp.xsd**"*, apresentada no anexo D.

Definição do elemento *Resources*: contém uma ***disjunctive sequence*** para elementos <Resource>, apresentado na seqüência.

```
[01] <xs:element name="Resources" type="xacml:ResourcesType"/>
[02]    <xs:complexType name="ResourcesType">
[03]        <xs:choice>
[04]            <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
[05]            <xs:element ref="xacml:AnyResource"/>
[06]        </xs:choice>
[07] </xs:complexType>
```

1. [01 a 07] Define um elemento "Resources" utilizando um tipo *complexType*, e este contêm elementos em sua definição.
2. [03 a 06] O elemento <choice> determina que deve ser escolhido um elemento declarado entre suas *tags* de início e de fim.
3. [04] Define um elemento "Resource" com o atributo maxOccurs="unbounded",

que determina não existir um número máximo para ocorrência do elemento.

4. [05] Define um elemento "AnyResource" como opção quando não é desejado definir um elemento ***resource***.
5. [07] Uma *tag* de finalização do elemento *Resources*.

Definição do elemento *Resource*: este elemento contém uma ***conjunctive sequence*** para os elementos tipo <**ResourceMatch**>, que são requeridos na declaração do elemento *Resource.*

```
[01] <!--  -->
[02] <xs:element name="Resource" type="xacml:ResourceType"/>
[03]    <xs:complexType name="ResourceType">
[04]        <xs:sequence>
[05]            <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
[06]            <!--  Extended ResourceRsvp  -->
[07]            <xs:choice minOccurs="0" maxOccurs="unbounded">
[08]                <xs:element ref="xacml:ResourceRsvp"/>
[09]            </xs:choice>
[10]        </xs:sequence>
[11] </xs:complexType>
```

6. [01] *Tags* para comentários.
7. [02 a 11] Define um elemento "Resource" utilizando um tipo *complexType*, e este contêm elementos em sua definição.
8. [03] *Tag* de um elemento *complexType* e o seu nome "ResourceType".
9. [04] *Tag* indicando uma seqüência de elementos.
10. [05] Define o elemento "ResourceMatch" com o atributo maxOccurs= "unbounded", determinando que não existe um número máximo para ocorrência do elemento. O "ResourceMatch" é utilizado para identificar o valor de um atributo de um ***resource*** em um XACML ***context***.
11. [06] *Tags* de comentário indicando a extensão da proposta, neste caso a declaração do elemento *ResourceRsvp.*
12. [07 a 09] Definição de um elemento <choice> com dois atributos. O elemento <choice> define que um dos elementos declarados entre suas *tags* de início e de fim deve ser escolhido. Neste caso existe apenas um elemento declarado entre suas *tags*, o elemento ***ResourceRsvp*** que estende o modelo XACML nesta proposta.
    a. O atributo minOccurs="0" com valor igual a "0" define que o elemento a ser escolhido é opcional e não necessita ser descrito na ***Policy***.

b. O atributo maxOccurs="unbounded" indica que não existe um número máximo para ocorrência do elemento *ResourceRsvp* em uma ***Policy***.

13. [08] Declaração do elemento <**ResourceRsvp**>.
14. [09-10] *Tags* de finalização dos elementos *choice* e *sequence*.
15. [11] *Tag* de finalização do elemento *complexType* chamado *ResourceType*.

5.1.2 Extensão de elementos para RSVP no XACML *policy* ***schema*:**

Definição do elemento <ResourceRsvp>: este elemento contém uma estrutura para definição de tráfego de dados, necessários a uma configuração de um fluxo de dados RSVP.

```
[01] <xs:element name="ResourceRsvp" type="xacml:ResourceRsvpType"/>
[02] <xs:complexType name="ResourceRsvpType">
[03]    <xs:sequence>
[04]        <xs:element ref="xacml:TspecBucketRate_r"/>
[05]        <xs:element ref="xacml:TspecBucketSize_b"/>
[06]        <xs:element ref="xacml:TspecPeakRate_p"/>
[07]        <xs:element ref="xacml:TspecMinPoliceUnit_m"/>
[08]        <xs:element ref="xacml:TspecMaxPacketSize_M"/>
[09]        <xs:choice minOccurs="0" maxOccurs="unbounded">
[10]            <xs:element ref="xacml:RsvpService"/>
[11]            <xs:element ref="xacml:RsvpStyle"/>
[12]        </xs:choice>
[13]    </xs:sequence>
[14]    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
[15]    <xs:attribute name="RsvpClass" type="xacml:RsvpClassType"  use="required"/>
[16] </xs:complexType>
```

1. [01] Define um elemento " ResourceRsvp" e utiliza um tipo *complexType* chamado "xacml:ResourceRsvpType".
2. [02 a 16] *Tags* de um elemento tipo *complexType* devido a existência de elementos e atributos em sua definição, e seu nome é name="ResourceRsvpType".
3. [03 a 13] *Tags* tipo <sequence> indicando uma seqüência de elementos.
4. [04 a 08] *Tags* com referência a vários elementos que são requeridos em uma definição do elemento <ResourceRsvp> quando utilizado em uma ***policy***. Estes elementos, quando utilizados pelo <ResourceRsvp> seguem a mesma ordem da declaração neste *complexType*. Os elementos referenciados, seguindo a ordem da declaração são: <TspecBucketRate_r>, <TspecBucketSize_b>, <TspecPeakRate_p>, <TspecMinPoliceUnit_m> e <TspecMaxPacketSize_M>.
5. [09 a 12] Definição de um elemento <choice> com dois atributos. O elemento <choice> define que um dos elementos declarados entre suas *tags* de início e

de fim deve ser escolhido. Existem dois elementos declarados entre suas *tags*, os elementos <RsvpService> e <RsvpStyle>, descritos na seqüência.

   a. O atributo minOccurs="0" com valor igual a "0" define que o elemento a ser escolhido é opcional e não necessita ser descrito na ***Policy***.
   b. O atributo maxOccurs="unbounded" indica que não existe um número máximo para ocorrência do elemento escolhido em uma ***Policy***.

6. [12-13] *Tags* de finalização dos elementos *choice* e *sequence*.
7. [14] Define um atributo com o nome "AttributeId" para identificar o elemento <ResourceRsvp>, e a utilização deste atributo é obrigatório.
8. [15] Define um atributo com o nome "RsvpClass" para identificar o tipo da classe de um serviço QoS, já discutido no capítulo 2, classes de aplicações RSVP, tabela 2.4.
9. [16] *Tag* de finalização do elemento *complexType* identificado como *ResourceRsvpType*.

Elementos <simpleType>: são elementos que não possuem declarações de elementos ou atributos em seu conteúdo. São utilizados, nesta proposta, para restrições de nomes e valores de atributos na definição de um elemento **<ResourceRsvp>** em uma ***policy***, evitando erros através da validação do documento. Um erro crasso na configuração do ***Tspec***, por exemplo, seria trocar os valores dos atributos de um Token Bucket Rate (r) com o Token Bucket Size (b), que apesar de serem dados definidos como 32-bit IEEE tipo *float*, possuem faixas máximas diferentes, apresentadas na tabela 5.1.

Definição do elemento identificado por "RsvpClassType": um elemento <simpleType> usado para definir nomes de classes de QoS. Este elemento é um atributo do <ResourceRsvp>. Os nomes sugeridos neste *schema* seguem o exemplo apresentado no capítulo 2, RSVP, classes de serviços RSVP, tabela 2.4.

```
[01] <!-- Class Qos RSVP -->
[02] <xs:simpleType name="RsvpClassType">
[03]    <xs:restriction base="xs:string">
[04]        <xs:enumeration value="G711"/>
[05]        <xs:enumeration value="G729"/>
[06]        <xs:enumeration value="H263CIF"/>
[07]        <xs:enumeration value="H261QCIF"/>
[08]    </xs:restriction>
[09] </xs:simpleType>
```

10. [01] *Tags* de comentário na definição do tipo Classes Qos de RSVP.
11. [02 a 09] *Tags* declarando um elemento tipo <simpleType> identificado pelo nome "RsvpClassType" e deve ser único no *schema*.
12. [03] Definição de um elemento <restriction> e o tipo da base é "**string**". Qualquer valor declarado que não seja do mesmo tipo da base, não é validado através do ***policy*** *schema*.
13. [04 a 07] *Tags* de elementos "enumeration" e a definição de valores válidos na declaração de uma ***policy***.
14. [08] *Tag* de finalização do elemento <restriction>.
15. [09] *Tag* de finalização do elemento *simpleType* chamado *RsvpClassType*.

Definição do elemento "RsvpService": um elemento que utiliza <simpleType> para restrição de valores do Tipo de Serviço oferecido pelo RSVP, padronizando as declarações do elemento <ResourceRsvp> nas ***policies***.

```
[01] <!—Service Qos RSVP -->
[02] <xs:element name="RsvpService">
[03]    <xs:simpleType>
[04]        <xs:restriction base="xs:string">
[05]            <xs:enumeration value="Null"/>
[06]            <xs:enumeration value="Guaranteed"/>
[07]            <xs:enumeration value="Controlled-load"/>
[08]        </xs:restriction>
[09]    </xs:simpleType>
[10] </xs:element>
```

16. [01] *Tags* de comentário na definição do tipo *Service Qos RSVP*.
17. [02] Declaração de um elemento chamado "RsvpService".
18. [03 a 09] *Tags* declarando um elemento tipo <simpleType> sem identificação.
19. [04] Definição de um elemento <restriction> e o tipo da base é "**string**". Qualquer valor declarado que não seja do mesmo tipo da base, não é validado através do ***policy*** *schema*.
20. [05 a 07] *Tags* de elementos "enumeration" e a definição de valores válidos na declaração de uma ***policy***.
21. [08] *Tag* de finalização do elemento <restriction>.
22. [09] *Tag* de finalização do elemento *simpleType*.
23. [10] *Tag* de finalização do elemento "RsvpService".

Definição do elemento "RsvpStyle": um elemento que utiliza <simpleType> para restrição de valores dos estilos de serviços utilizados pelo RSVP, padronizando as declarações do elemento <ResourceRsvp> nas ***policies***.

```
[01] <!-- Style Qos RSVP -->
[02] <xs:element name="RsvpStyle">
[03]    <xs:simpleType>
[04]        <xs:restriction base="xs:string">
[05]            <xs:enumeration value="SE"/>
[06]            <xs:enumeration value="WF"/>
[07]            <xs:enumeration value="FF"/>
[08]        </xs:restriction>
[09]    </xs:simpleType>
[10] </xs:element>
```

24. [01] *Tags* de comentário na definição do Estilo de Serviço Qos RSVP.
25. [02] Declaração de um elemento chamado "RsvpStyle".
26. [03 a 09] *Tags* declarando um elemento tipo <simpleType> sem identificação.
27. [04] Definição de um elemento <restriction> e o tipo da base é "**string**". Qualquer valor declarado que não seja o mesmo tipo da base, não é validado através do ***policy*** *schema*.
28. [05 a 07] *Tags* de elementos "enumeration" e a definição de valores válidos na declaração de uma ***policy***.
29. [08] *Tag* de finalização do elemento <restriction>.
30. [09] *Tag* de finalização do elemento *simpleType*.
31. [10] *Tag* de finalização do elemento "RsvpStyle".

Definição do elemento " TspecBucketRate_r": um elemento que utiliza <simpleType> para restrição de valores dos ***Tspec*** transportado em mensagens RSVP.

```
[01] <!--  RFC 2212 -->
[02] <!-- TSPEC QoS RSVP -->
[03] <!-- measured in bits/sec   float [1 - 40TB] -->
[04] <!-- base="xs:float" -->
[05] <xs:element name="TspecBucketRate_r">
[06]    <xs:simpleType>
[07]        <xs:restriction base="xs:double">
[08]            <xs:minInclusive value="1"/>
[09]            <xs:maxInclusive value="40000000000000"/>
[10]        </xs:restriction>
[11]    </xs:simpleType>
[12] </xs:element>
```

32. [01 a 04] *Tags* de comentários declarando RFC, *Tspec*, especificação e limites para declaração do elemento, limites declarados e a base do tipo de dado.

33. [05] Declaração de um elemento chamado "TspecBucketRate_r".
34. [06 a 11] *Tags* declarando um elemento tipo <simpleType> sem identificação.
35. [07] Definição de um elemento <restriction> e o tipo da base é "**double**". Qualquer valor declarado que não seja o mesmo tipo da base, não é validado através do ***policy*** *schema*.
36. [08-09] *Tags* de elementos <minInclusive> e <maxInclusive> restringindo o limite mínimo e o limite máximo para definição de valores ***Tspec*** válidos na declaração de uma ***policy***.
37. [10] *Tag* de finalização do elemento <restriction>.
38. [11] *Tag* de finalização do elemento *simpleType*.
39. [12] *Tag* de finalização do elemento "TspecBucketRate_r".

Definição do elemento "TspecBucketSize_b": um elemento que utiliza <simpleType> para restrição de valores dos ***Tspec*** transportado em mensagens RSVP.

```
[01] <!-- measured in bits/sec   float [1 - 250GB]-->
[02] <!-- base="xs:float" -->
[03] <xs:element name="TspecBucketSize_b">
[04]    <xs:simpleType>
[05]        <xs:restriction base="xs:double">
[06]            <xs:minInclusive value="1"/>
[07]            <xs:maxInclusive value="250000000000"/>
[08]        </xs:restriction>
[09]    </xs:simpleType>
[10] </xs:element>
```

40. [01-02] *Tags* de comentários declarando o tipo de medida, limite mínimo e máximo para declaração do elemento e a base do tipo de dado.
41. [03] Declaração de um elemento chamado "TspecBucketSize_b".
42. [04 a 09] *Tags* declarando um elemento tipo <simpleType> sem identificação.
43. [05] Definição de um elemento <restriction> e o tipo da base é "**double**". Qualquer valor declarado que não seja o mesmo tipo da base, não é validado através do ***policy*** *schema*.
44. [06-07] *Tags* de elementos <minInclusive> e <maxInclusive> restringindo o limite mínimo e o limite máximo para definição de valores ***Tspec*** válidos na declaração de uma ***policy***.
45. [08] *Tag* de finalização do elemento <restriction>.
46. [09] *Tag* de finalização do elemento *simpleType*.

47. [10] *Tag* de finalização do elemento "TspecBucketSize_b".

Definição do elemento "TspecPeakRate_p": um elemento que utiliza <simpleType> para restrição de valores dos ***Tspec*** transportado em mensagens RSVP.

```
[01] <!-- measured in bytes  float [1 - 40TB] -->
[02] <!-- base="xs:float" -->
[03] <xs:element name="TspecPeakRate_p">
[04]    <xs:simpleType>
[05]        <xs:restriction base="xs:double">
[06]            <xs:minInclusive value="1"/>
[07]            <xs:maxInclusive value="40000000000000"/>
[08]        </xs:restriction>
[09]    </xs:simpleType>
[10] </xs:element>
```

48. [01-02] *Tags* de comentários declarando o tipo de medida, limite mínimo e máximo para declaração do elemento e a base do tipo de dado.
49. [03] Declaração de um elemento chamado "TspecPeakRate_p".
50. [04 a 09] *Tags* declarando um elemento tipo <simpleType> sem identificação.
51. [05] Definição de um elemento <restriction> e o tipo da base é "**double**". Qualquer valor declarado que não seja o mesmo tipo da base, não é validado através do ***policy*** *schema*.
52. [06-07] *Tags* de elementos <minInclusive> e <maxInclusive> restringindo o limite mínimo e o limite máximo para definição de valores ***Tspec*** válidos na declaração de uma ***policy***.
53. [08] *Tag* de finalização do elemento <restriction>.
54. [09] *Tag* de finalização do elemento *simpleType*.
55. [10] *Tag* de finalização do elemento "TspecPeakRate_p".

Definição do elemento "TspecMinPoliceUnit_m": um elemento que utiliza <simpleType> para restrição de valores dos ***Tspec*** transportado em mensagens RSVP.

```
[01] <!-- measured in microseconds unsigned integer [4294967296] -->
[02] <!-- base="xs:unsignedInt" -->
[03] <xs:element name="TspecMinPoliceUnit_m">
[04]    <xs:simpleType>
[05]        <xs:restriction base="xs:integer">
[06]            <xs:minInclusive value="1"/>
[07]            <xs:maxInclusive value="4294967295"/>
[08]        </xs:restriction>
[09]    </xs:simpleType>
[10] </xs:element>
```

56. [01-02] *Tags* de comentários declarando o tipo de medida, limite mínimo e máximo para declaração do elemento e a base do tipo de dado.
57. [03] Declaração de um elemento chamado "TspecMinPoliceUnit_m".
58. [04 a 09] *Tags* declarando um elemento tipo <simpleType> sem identificação.
59. [05] Definição de um elemento <restriction> e o tipo da base é "**integer**". Qualquer valor declarado que não seja o mesmo tipo da base, não é validado através do ***policy*** *schema*.
60. [06-07] *Tags* de elementos <minInclusive> e <maxInclusive> restringindo o limite mínimo e o limite máximo para definição de valores ***Tspec*** válidos na declaração de uma ***policy***.
61. [08] *Tag* de finalização do elemento <restriction>.
62. [09] *Tag* de finalização do elemento *simpleType*.
63. [10] *Tag* de finalização do elemento "TspecMinPoliceUnit_m".

Definição do elemento "TspecMaxPacketSize_M": um elemento que utiliza <simpleType> para restrição de valores dos ***Tspec*** transportado em mensagens RSVP.

```
[01] <!-- measured in bytes  unsigned integer [4294967296] -->
[02] <!-- base="xs:unsignedInt" -->
[03] <xs:element name="TspecMaxPacketSize_M">
[04]    <xs:simpleType>
[05]        <xs:restriction base="xs:integer">
[06]            <xs:minInclusive value="1"/>
[07]            <xs:maxInclusive value="4294967295"/>
[08]        </xs:restriction>
[09]    </xs:simpleType>
[10] </xs:element>
```

64. [01-02] *Tags* de comentários declarando o tipo de medida, limite mínimo e máximo para declaração do elemento e a base do tipo de dado.
65. [03] Declaração de um elemento chamado "TspecMaxPacketSize_M".
66. [04 a 09] *Tags* declarando um elemento tipo <simpleType> sem identificação.
67. [05] Definição de um elemento <restriction> e o tipo da base é "**integer**". Qualquer valor declarado que não seja o mesmo tipo da base, não é validado através do ***policy*** *schema*.
68. [06-07] *Tags* de elementos <minInclusive> e <maxInclusive> restringindo o limite mínimo e o limite máximo para definição de valores ***Tspec*** válidos na declaração de uma ***policy***.

69. [08] *Tag* de finalização do elemento <restriction>.
70. [09] *Tag* de finalização do elemento *simpleType*.
71. [10] *Tag* de finalização do elemento "TspecMaxPacketSize_M".

EXTENSÃO DE ATRIBUTO DE LOCALIZAÇÃO DA POLICY

Uma ***policy*** para avaliar uma ***authorization decision*** de um ***Subject*** ou ***Resource***, deve se basear na identificação destes ou por alguma característica que os descrevam.

O XACML suporta o requerimento de identificação para ***policy*** através de um elemento chamado <AttributeDesignator>, que contém um atributo URN para comparação de atributos de um ***Subject*** ou ***Resource***, em um XACML *Request* **context**. O requerimento para descrição de características para a ***policy***, é fornecido através do elemento XACML chamado <AttributeSelector>, que contém a expressão XPath, descrita na sequência, que é utilizada para localizar as características dos atributos em um XACML *Request* **context**.

Nesta proposta, foi extendido o <AttributeSelector>, característica de localização, para que a ***Policy*** possa utilizá-lo na pesquisa de localização de atributos de QoS declarados em seus elementos ***Resource***, possibilitando fornecer dados do ***Tspec*** no elemento *Obligations* e retornar dados ao PEP. Antes da instalação de um fluxo QoS, o PEP deve solicitar a validação ao PDP dos dados ***Tspec***, recebidos do RECEIVER, e que devem ser validados pelo PDP através de pesquisa na política em elementos ***Resources***.

Portanto, a extensão do elemento <AttributeSelector>, denominado <AttributeSelectorRsvp>, possibilita a localização de dados QoS na ***policy***, tanto para o fornecer, quanto para validar os parâmetros ***Tspec***, em um XACML *Response* ***context***.

Definição do elemento <AttributeSelectorRsvp>:

O elemento <AttributeSelectorRsvp> possibilita, tanto a validação como o retorno de valores do ***Tspec*** definidos na ***policy*** pelo elemento <ResourceRsvp>. Se solicitado um XACML *Request* pelo PEP que requer parâmetros de *Tspec*, estes serão informados através do elemento <Obligations> enviado pelo PDP ao PEP.

O elemento <AttributeSelectorRsvp> contém uma estrutura com uma expressão XPath para pesquisa de um nó declarado na ***policy***. Este elemento deve avaliar, conforme a especificação do seu "DataType", o valor do nó pesquisado, conforme descrição a seguir no algoritmo 5.1. O "DataType" contém um tipo de dado definido pela W3C em [W3C02] e

[W3C01], para que o XPath retorne o valor do nó pesquisado para o elemento <AttributeSelectorRsvp> e com o tipo especificado no "DataType" deste.

O XPath fornece facilidades para manipulação de caracteres, números e expressões lógicas definidas pela W3C [W3C99]. A conversão do XPath utiliza funções definidas na sessão 4 [W3C02].

```
[01] <xs:element name="AttributeSelectorRsvp" type="xacml:AttributeSelectorRsvpType"/>
[02]    <xs:complexType name="AttributeSelectorRsvpType">
[03]        <xs:attribute name="PolicyPath" type="xs:string" use="required"/>
[04]        <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
[05]    </xs:complexType>
```

72. [01] Define um elemento "AttributeSelectorRsvp" e utiliza um tipo *complexType* chamado "xacml: "AttributeSelectorRsvpType".
73. [02] *Tags* de um elemento tipo *complexType* devido a existência de atributos em sua definição e seu nome é name=" AttributeSelectorRsvpType".
74. [03] Declaração de atributo com o nome: name="PolicyPath". Uma expressão XPath para pesquisa em nós de um XACML ***policy***.
75. [04] Declaração de atributo com o nome: name="DataType". Define o tipo do valor de retorno da expressão XPath.
76. [05] *Tags* de finalização do *complexType*.

Descrição do Algoritmo de Pesquisa "AttributeSelectorRsvp"

ENTRADA DE DADOS:

PolicyPath: caminho do elemento a ser pesquisado segundo hierarquia dos elementos declarados na policy, iniciado pelo primeiro elemento: Policy ou PolicySet, declarados com o formato: "//Policy/Target/Resources/ NóFinadePesquisa";

Parâmetro recebido pelo "PolicyPath" com sintaxe definida em [W3C99].

DataType : descrição do tipo de dado a ser pesquisado definidos pelo XACML, como tipo: *string*, *integer* ou *double*, utilizados no *Tspec*;

RETORNO DE DADOS:

Retorno do valor do atributo do elemento pesquisado na política ao elemento <AttributeSelectorRsvp>, tipo *double*, *integer* ou *string*;

INÍCIO

Receber valor da estrutura para pesquisa do atributo "PolicyPath";
Receber valor do tipo de dado a ser pesquisado do atributo "DataType";
Se ("DataType" não é válido nos tipos definidos em em [W3C01] e [W3C02]) então
    Informe erro e encerre pesquisa;

```
Se (estrutura informada dos nós é válida) então
        percorrer até o (NóFinadePesquisa) na estrutura da policy;
        converter valor do conteúdo pesquisado para o mesmo tipo do "DataType";
Senão
        informe erro e encerre pesquisa;
 Retornar valor do elemento (NóFinadePesquisa);
FIM DO ALGORITMO DE PESQUISA.
```

Algoritmo 5.1 Pesquisa do AttributeSelectorRsvp

5.2 Framework de Implementação: PDP, PEP e COPS

Uma arquitetura básica para controle de acesso RSVP utilizando XACML, é exemplificada nesta proposta, utilizando uma rede local (LAN) com um computador RSVP *client* (*RECEIVER*), um servidor de Multimídia RSVP (*SENDER*) atuando como o PEP e um servidor PDP que utiliza políticas definidas em XACML. Os roteadores existentes na rede são dispositivos RSVP-*Aware*.

Os serviços de multimídia podem ser executados por um cliente QoS RSVP que tenha autorização para acesso, e a decisão de autorização do serviço QoS é obtida através do PEP que envia uma ***decision request*** ao servidor de políticas PDP, utilizando para comunicação entre PEP e PDP o protocolo COPS. O cenário é apresentado na figura 5.2 com a descrição do fluxo abaixo:

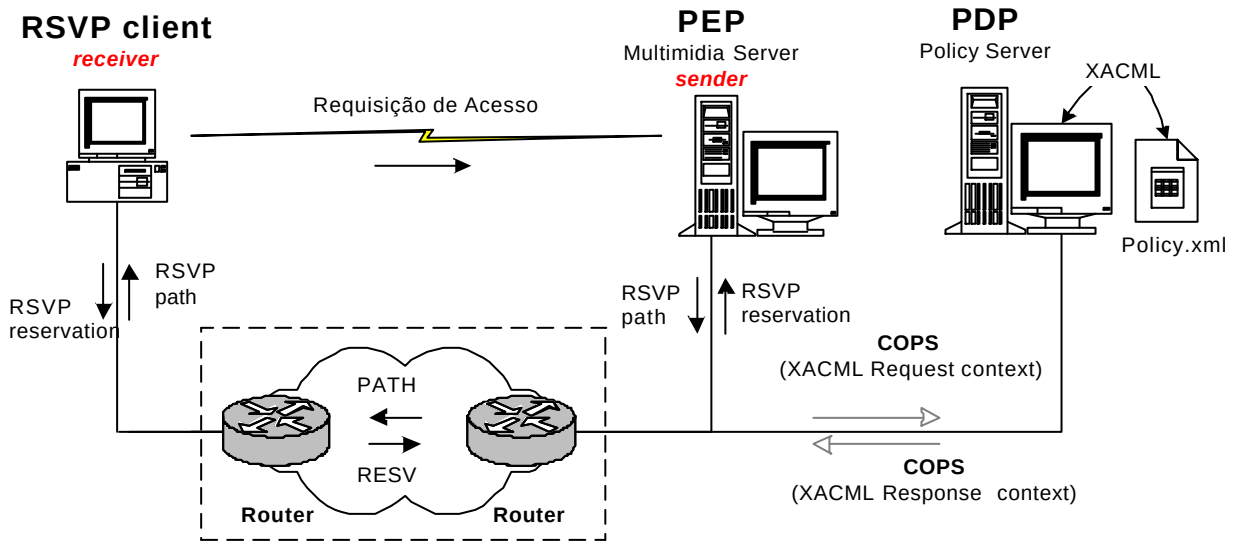

Figura 5.2 Cenário RSVP com XACML

1. O cliente QoS RSVP solicita uma conexão a um servidor multimídia para

obtenção de serviços QoS.

2. O PEP recebe a solicitação de serviço do cliente. O PEP informa o seu endereço IP e o IP do cliente, enviando um XACML *Request* ***context*** encapsulado em uma mensagem COPS ao PDP.
3. O PDP avalia as restrições na política definida em XACML, neste caso, endereços IP autorizados a serviços de RSVP e restrições de horários, e envia uma ***authorization decision*** através do XACML *Reponse* ***context***, encapsulado em uma mensagem COPS, ao PEP contendo informações de especificação de tráfego *Tspec*.
4. O PEP com a ***authorization decision*** e invoca seu serviço RSVP *daemon*, informando os parâmetros *Tspec.* O serviço RSVP *daemon* do PEP envia uma mensagem RSVP PATH ao *RECEIVER,* neste caso o cliente RSVP.
5. O cliente RSVP, o *RECEIVER,* ao receber a mensagem RSVP PATH invoca seu RSVP *daemon*, que obtém os parâmetros da mensagem PATH e formata a mensagem RSVP RESV, retornando ao *SENDER,* neste caso o PEP.
6. O PEP, Servidor de Multimídia, recebe a mensagem RSVP RESV do *RECEIVER* e analisa se o tráfego de dados ***Tspec*** não foi alterado e estabelece o fluxo RSVP.
7. Confirmada uma alteração do ***Tspec***, o PEP informa o seu endereço IP, o IP do cliente e o ***Tspec*** recebido na mensagem RSVP RESV, e envia uma ***decision request*** ao PDP via COPS, conforme descrito no passo 2.
8. O servidor PDP consulta a política definida em XACML para permissão de acesso do usuário e aplicação em relação às restrições, respondendo conforme descrito no passo 3. O PEP estabelecerá ou não a comunicação de acordo com a decisão recebida.

As aplicações são classificadas como emissoras ou receptoras, baseadas nos requerimentos de QoS. Apesar das aplicações não serem limitadas somente a transmitirem ou receberem dados, a utilização de QoS para muitas das aplicações são mapeadas para transmitirem com QoS ou somente receberem com QoS. Esta é uma característica de aplicações *Playback*, semelhantes a servidores de aplicações de Áudio ou Vídeo, na qual o servidor envia tráfego QoS e os receptores recebem tráfego QoS em um fluxo unidirecional.

A apresentação do cenário a seguir, estabelece funcionalidade para servidores de aplicações de QoS em um fluxo unidirecional, controladas por um servidor de política para definição de acesso e tráfego.

5.3 Fluxo Completo para RSVP e XACML

Será apresentado um cenário de eventos simulando uma solicitação de Serviço QoS de um servidor de aplicação QoS, utilizando protocolo RSVP e a representação do fluxo de transação de políticas entre PEP e PDP usando COPS.

O diagrama de Seqüência apresenta o cenário envolvido em uma requisição de um serviço QoS. Os objetos do diagrama representam os dispositivos de rede identificados através do estereótipo. Os cenários representam a seqüência para solicitar, requisitar, autorizar, instalar e encerrar um serviço de QoS e será. O diagrama será dividido em fases para exemplificação do processo de reserva de recursos. Um cenário completo mapeando inclusive as mensagens COPS encontra-se no Anexo E.

Uma aplicação para transmitir tráfego com QoS, deve invocar o processo RSVP. O *RSVP daemon* responderá iniciando a transmissão de uma mensagem PATH. Esta descreve os dados do tráfego com a estrutura ***Tspec***, endereço do emissor (*SenderTemplate*) e o destino do tráfego (*session*). A aplicação para receber tráfego QoS deverá também invocar o RSVP *daemon*, que responderá iniciando a transmissão da mensagem RESV. Neste cenário, para garantia e controle da aplicação das políticas, a chamada ao RSVP *daemon* é executada pelo PEP.

5.3.1 Conexão para uma Aplicação QoS-Aware

Um cliente chamado RECEIVER, executa um acesso a um servidor de QoS RSVP solicitando um serviço de QoS, conforme descrito na figura 5.3.

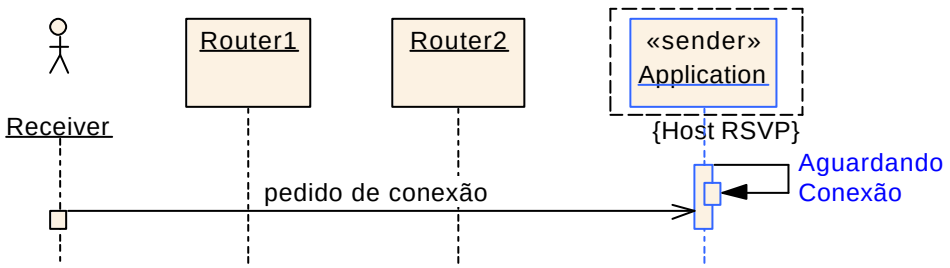

Figura 5.3 Solicitação de conexão

1. O Aplicativo RSVP QoS-Aware atuando como servidor de aplicação abre uma comunicação TCP através de *socket*, aguardando requisições. Permanece ativo aguardando uma chamada.
2. Usuário abre outra comunicação através de *socket* e solicita uma requisição de conexão para obtenção de um serviço QoS, representado na figura 5.3, por uma mensagem chamada de pedido de conexão.

5.3.2 Solicitação de um Serviço de QoS pela Aplicação

Como no modelo descrito na RFC 2205, a aplicação servidora ao receber um pedido de conexão de serviço QoS invoca o RSVP *daemon* para estabelecer um fluxo RSVP. A principal mudança conceitual do processo de reserva de fluxo QoS nesta proposta, é a solicitação da aplicação ao PEP para um fluxo de QoS, apresentada na figura 5.4.

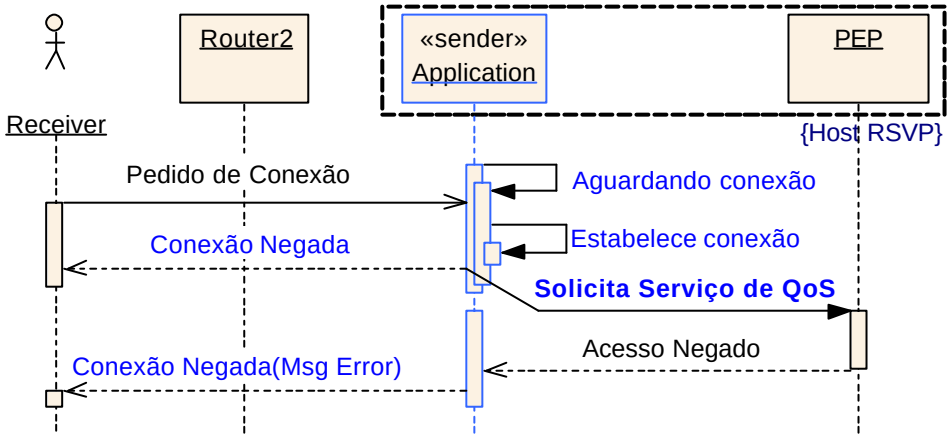

Figura 5.4 Solicitação de serviço QoS ao PEP

3. Aplicação servidora de multimídia estabelece a conexão com o *RECEIVER*, e caso ocorra algum erro ou exceção na abertura da sessão de comunicação, será rejeitada a conexão entre Aplicação e *RECEIVER* retornando um erro, representado na figura acima, pela mensagem “conexão negada”.
4. A Aplicação solicita os parâmetros de QoS ao PEP para estabelecer uma comunicação QoS, representada pela mensagem “Solicita Serviço de QoS”, descrito no algoritmo 5.2, informando ao PEP o endereço IP do *RECEIVER* e o endereço IP da aplicação, utilizados neste cenário como identificadores para controle de acesso. Aguarda

resposta via PEP.

```
Mensagem: "Solicita Serviço de QoS" [App → PEP]
// (1) Msg Enviada pela Aplicação ao PEP
{    Informar IP Receiver e IP Sender
     Informar hostName da Aplicação
}
```

Algoritmo 5.2 Solicita Serviço de QoS

5.3.3 Requisição de decisão ao PDP

O PEP ao receber uma mensagem "Solicitação de serviço de QoS", estabelece uma conexão com o PDP utilizando o protocolo COPS, conforme descrito no cenário apresentado na figura 5.5.

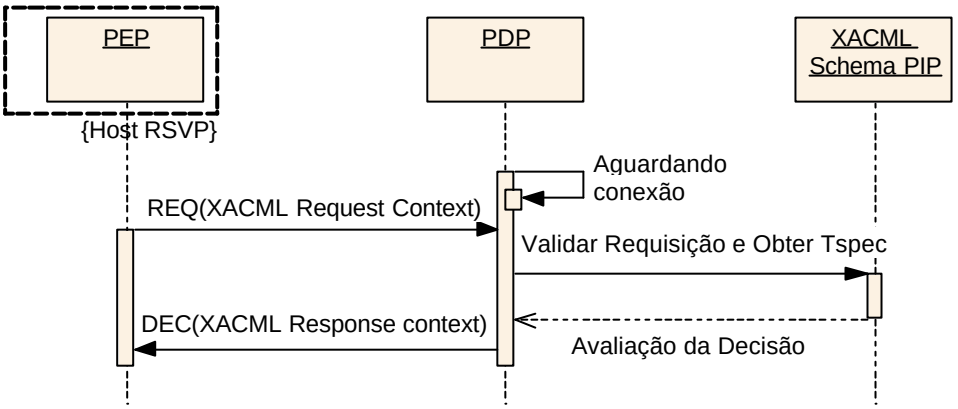

Figura 5.5 Solicitação de decisão ao PDP

5. O PEP gera uma mensagem "REQ(XACML *Request* ***context***)" requisitando a decisão ao PDP para uma autorização de acesso e dados *Tspec*. O PEP encapsula os dados XACML no protocolo COPS REQ( ) e envia ao PDP, descrito no algoritmo 5.3.

```
Mensagem: "REQ(XACML Request Context)"     [PEP → PDP]
// (1) Processada pelo PEP após Solicitação de QoS
// (2) PEP envia Requisição de Decisão e Solicita Recursos de QoS ao PDP
{  Gerar a mensagem "XACML Request context" definindo os elementos:
       <Subject>   IP RECEIVER  /  IP SENDER
       <Resource>  hostname SENDER
       <Action>    getResourceQoS
   Encapsular "XACML Request context" no COPS (REQ);
   Abrir conexão PDP e enviar COPS(REQ); // (XACML Request Context)
   Aguardar Decisão do PDP;
}
```

Algoritmo 5.3 REQ(XACML *Request* ***context***)

6. O servidor PDP recebe uma conexão de um PEP utilizando o protocolo COPS. A descrição do cenário completo mapeando as mensagens COPS encontra-se no Anexo E. O PEP envia uma mensagem "REQ(XACML *Request* ***context***)" ao PDP, contendo basicamente as seguintes informações: dados dos <Subject>, nome do recurso <Resource> e ação desejada <Action>.

```
<Request>
  <Subject> "authn-locality:ip-address:receiver/sender"
      <AttributeValue>192.168.200.192</AttributeValue>
      <AttributeValue>192.168.200.1</AttributeValue>
  </Subject>
  <Resource>   "resource:resource-id"
      <AttributeValue>MultimidiaServer</AttributeValue>
  </Resource>
  <Action>   "action-id:ServerAction"
      <AttributeValue>getResourceQoS</AttributeValue>
  </Action>
</Request>
```

7. A explanação de um XACML *Request* ***context*** foi abordada no capítulo 4 e será apresentado neste cenário somente dados relevantes a exemplificação para obtenção do fluxo RSVP. A descrição com detalhes das mensagens XACML deste cenário encontra-se no capítulo 6, descrevendo um exemplo de política em um cenário típico.

8. A mensagem "Validar Requisição e Obter Tspec" é utilizada pelo PDP para processar uma ***decision request*** e gerar uma ***authorization decision*** formatando a resposta ao PEP, procedimento descrito no algoritmo 5.4.

```
Mensagem: "Validar Requisição e Obter Tspec"    [PDP]
// (1) Processada pelo PDP para Decisão de Autorização
{ Obter "XACML Request context" encapsulados no COPS (REQ) e Processar
  a Política:
Validar Policy <Target>
  Policy <Resource> hostnameServer  (igual)  XACML Request <Resource>
Validar <Rule> // expressões lógicas
  Rule   <Subject>  IP Address      (igual)  XACML Request <Subject>
  Rule   <Action>   getResourceQoS  (igual)  XACML Request <Action>
  Validar <Conditions>, se definidas;
  // ex: Restrição de Horário  [09h00 – 17h00]
Se (<Rule> = "Permit") então
{ PDP deve Processar <Obligations> e Obter "Tspec" e  "RsvpService";
  // Utilizado elemento <ResourceRsvp>, extensão do XACML
  Formatar Decisão de Autorização("Permit") com <Obligations> incluindo os
     parâmetros {Tspec,RsvpService};
}
Senão
  Formatar Decisão de Autorização ("Deny");
}
```

Algoritmo 5.4 Validar Requisição e Obter *Tspec*

9. O PDP avalia na política quais <Subject> tem direito de acesso aos recursos <Resource>, representado no exemplo abaixo pelos identificadores "**receiver/sender**" e os atributos sendo os endereços IP descritos em <Subject> e qual é ação desejada, neste caso representado por "**getResourceQoS**". O PDP avalia restrições descritas na política, neste caso a restrição se refere ao horário disponível para acesso ao Servidor de Aplicações Multimídia entre às **09:00** horas e às **17:00** horas, conforme o esboço da política representada abaixo.

```
<Target>
    <Resources> "resource:resource-id"
            <AttributeValue>MultimidiaServer</AttributeValue>
    </Resources>
</Target>
<Rule Effect="Permit">
    <Target>
        <Subjects> "authn-locality:ip-address:receiver/sender"
            192.168.200.192 /  192.168.200.1
        </Subjects>
        <Actions> "action-id:ServerAction"
            getResourceQoS
        </Actions>
    </Target>
    <Condition> 09:00:00 às 17:00:00 // restrição de horário
    </Condition>
<Obligations> FulfillOn="Permit"
        <TokenBucketRate_r>9250          <TspecBucketRate_r>9250
        <TokenBucketSize_b>680           <PeakRate_p>13875
        <MinimumPoliceUnit_m>340         <MaximumPacketSize_M>340
        <RsvpService>Guaranteed
</Obligations>
```

10. O PDP valida a ***authorization decision*** verificando se na descrição do <Target> existe um recurso descrito na política em relação ao XACML *Request* ***context***, neste caso o "**MultimidiaServer**". O PDP aplica a regra verificando quais os direitos de acesso e se existe uma ação definida, neste caso os endereços IPs e o nome da ação "**getResourceQoS**", verifica a condição que é uma restrição de horário de acesso.

11. Neste exemplo, a regra é valida e o PDP retorna os valores do ***Tspec*** ao PEP encapsulando a mensagem XACML *Response* ***context*** com a decisão "*Permit*". A mensagem "Avaliação da Decisão" gerada pelo PDP é apresentada no algoritmo 5.5.

```
Mensagem "Avaliação da Decisão" [PDP]
// (1) Decisão de Autorização do PDP
// (2) Função de encapsular avaliação do PDP no COPS DEC e enviar ao PEP
{  Encapsular na mensagem COPS (DEC) o "XACML Response Context";
   Responder ao PEP via COPS (DEC);
}
```

Algoritmo 5.5 Avaliação da Decisão para solicitação de RSVP QoS

12. O PEP ao receber a mensagem "DEC(XACML *Response* ***context***)", via protocolo COPS DEC( ), obtém os dados XACML com a avaliação de decisão do PDP. Se a resposta do PDP é válida, o PEP deve armazenar os dados *Tspec* e tipo de Serviço recebidos na <Obligations>, invocando o RSVP *daemon* para iniciar a sinalização RSVP, caso contrário, deve informar uma mensagem de erro ao SENDER, e será discutido com mais detalhes nos itens a seguir. O algoritmo 5.6 apresenta a mensagem de decisão enviada do PDP ao PEP.

```
Mensagem: "DEC(XACML Request Context)" [PDP → PEP]
// (1) Processada pelo PEP em resposta a solicitação
{ Obter dados do "XACML Response context" encapsulados no COPS (DEC);
  Se (decisão = "DENY") então
    Informar condição de erro para a aplicação (SENDER);
  Senão {// RSVP SessionId para fluxos RSVP
    Obter e Armazenar dados de <Obligations> {Tspec,RsvpService};
    Executar chamada SENDER; // API RFC 2205
  }
}
```

Algoritmo 5.6 DEC(XACML *Request* ***context***)

5.3.4 Reserva de fluxo de Dados

O PEP recebendo uma resposta válida através da mensagem XACML *Response* ***context*** retornada do PDP, invoca o RSVP *daemon* para estabelecer o fluxo de dados. Segue abaixo a descrição apresentada na figura 5.6.

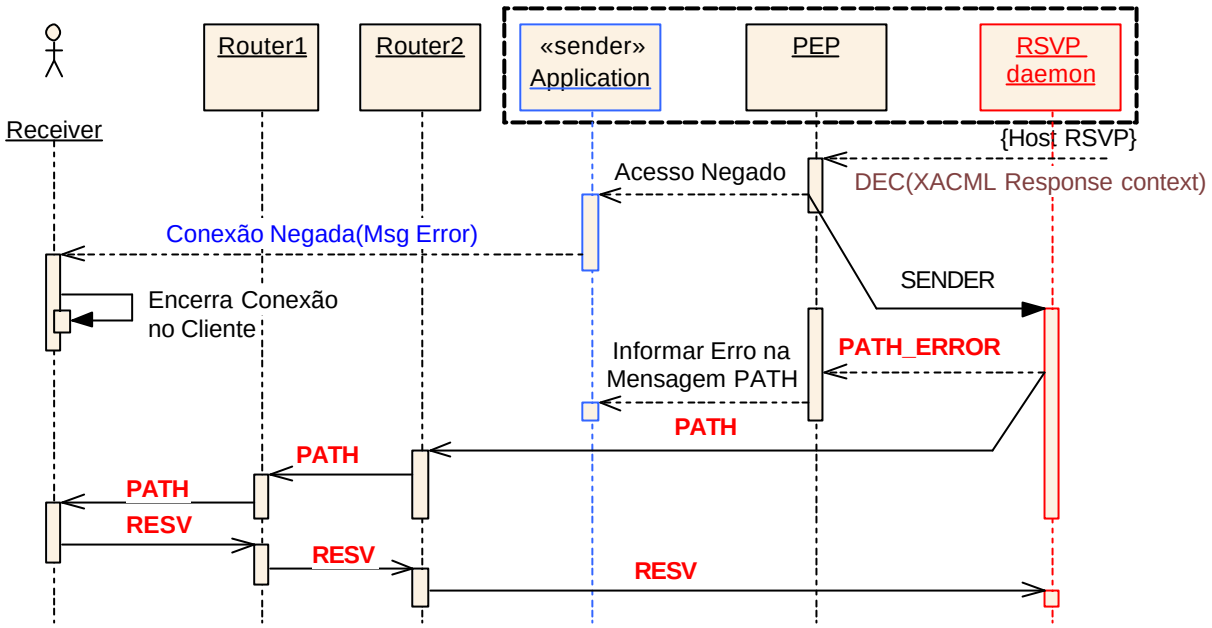

Figura 5.6 Reserva de fluxo de QoS invocado pelo PEP

O *RSVP daemon* deve consultar o Controle de Políticas para autorização e o Controle de Admissão para verificação de recursos antes do estabelecimento do fluxo. Antes de estabelecer o fluxo, este deve ser definido com parâmetros de tráfego (Tspec). A solicitação do ***Tspec*** ao servidor de políticas PDP visa assegurar que a aplicação, via programador, não irá alterar dados ou criar seu próprio padrão de tráfego solicitando diretamente ao *RSVP daemon*. **A alteração neste procedimento assegura o estabelecimento da política de segurança para obtenção do *Tspec* e um procedimento de reserva de fluxos via PEP**.

13. Se o acesso informado pelo PDP é permitido, inicia o processo de reserva de fluxo QoS, caso contrário, uma mensagem é enviada do PEP à Aplicação informando o acesso negado, indicada pela mensagem “Acesso Negado” descrita na figura 5.6. A aplicação informa ao RECEIVER com a mensagem “Conexão Negada” e encerra a conexão. Neste cenário, o acesso é permitido e continua o procedimento para a reserva de fluxo RSVP.

14. O PEP invoca o RSVP *daemon* que inicia o processo de sinalização RSVP PATH, descrito no capítulo 2, mensagens RSVP.

15. O PEP ao invocar o RSVP *daemon*, apresentada na figura 5.6 pela mensagem SENDER (aplicação), informa o ***Tspec*** e o tipo de serviço que será oferecido ao RECEIVER, neste exemplo o *Guaranteed*, que é formatado no objeto *Adspec* e

enviado junto na mensagem RSVP PATH do *SENDER* ao *RECEIVER*.

16. A mensagem SENDER é uma sugestão de API descrita na RFC 2205 para uma aplicação invocar um processso RSVP ou RSVP *daemon*, neste cenário é o PEP que invoca o RSVP *daemon*. O algoritmo 5.7 apresenta a chamada "SENDER".

```
Mensagem:  "SENDER" [PEP → RSVP Daemon]
//(1)  Call: SENDER é uma API sugerida na RFC 2205 – pg63
//(2)  Invocar RSVP Daemon com parâmetros definidos no item 3.11.1 RFC2205
{ Se chamada (SENDER == Sucesso) então
    RSVP Daemon Inicia Sinalização ao Receiver "RSVP PATH";[RSVPD → RSVPD]
  Senão
    RSVP Daemon Gera Evento "PATH_ERROR" ao PEP; [RSVP Daemon → PEP]
}
```

Algoritmo 5.7 SENDER – API RFC 2205

17. Se ocorrer algum erro para enviar a mensagem PATH, O *RSVP daemon* gera um evento com uma mensagem "PATH-ERROR" e informa ao PEP o motivo do erro. O PEP informa o erro ao servidor de aplicação com a mensagem "Informar Erro na Mensagem PATH". O algoritmo 5.8 apresenta a mensagem de erro do PEP ao servidor de aplicação. O cliente, *RECEIVER*, tem a opção de estabelecer novamente a reserva ou encerrar a conexão. Neste cenário a sinalização PATH segue *downstream* ao RECEIVER. Este processo de sinalização de reserva e manutenção de estados nos roteadores, como o encerramento da reserva de fluxos foi esmiuçado no capítulo 2.

```
Mensagem: "Informar Erro na Mensagem PATH" [PEP → SENDER]
{ PEP informa Tipo de Erro ao SENDER;
  // SENDER pode encerrar conexão com RECEIVER
}
```

Algoritmo 5.8 Informar Erro na Mensagem PATH

18. O *RECEIVER* recebe a sinalização PATH e formata a mensagem RSVP RESV reservando o fluxo QoS ao longo do caminho até o SENDER (aplicação).

5.3.5 Validando Reserva de QoS

Uma mensagem RSVP RESV chegando ao *SENDER* (aplicação) solicita uma reserva de fluxo QoS. Um evento RESV_EVENT é gerado pelo RSVP *daemon* do SENDER ao PEP, solicitando validação de fluxo de dados antes de sua instalação. O PEP deve avaliar se houve

alteração do fluxo ***Tspec***, informado na mensagem PATH enviada ao *RECEIVER*, com a descrição do tráfego de dados recebido no FLOWSPEC da mensagem RSVP RESV.

O objeto *Flowspec* contém a descrição do tráfego que pode ter sido alterado pelo RECEIVER ou roteadores ao longo da rota, conforme apresentado no capítulo 2, mensagens RSVP. O fluxo de validação da reserva é apresentado na figura 5.7 e discutido a seguir.

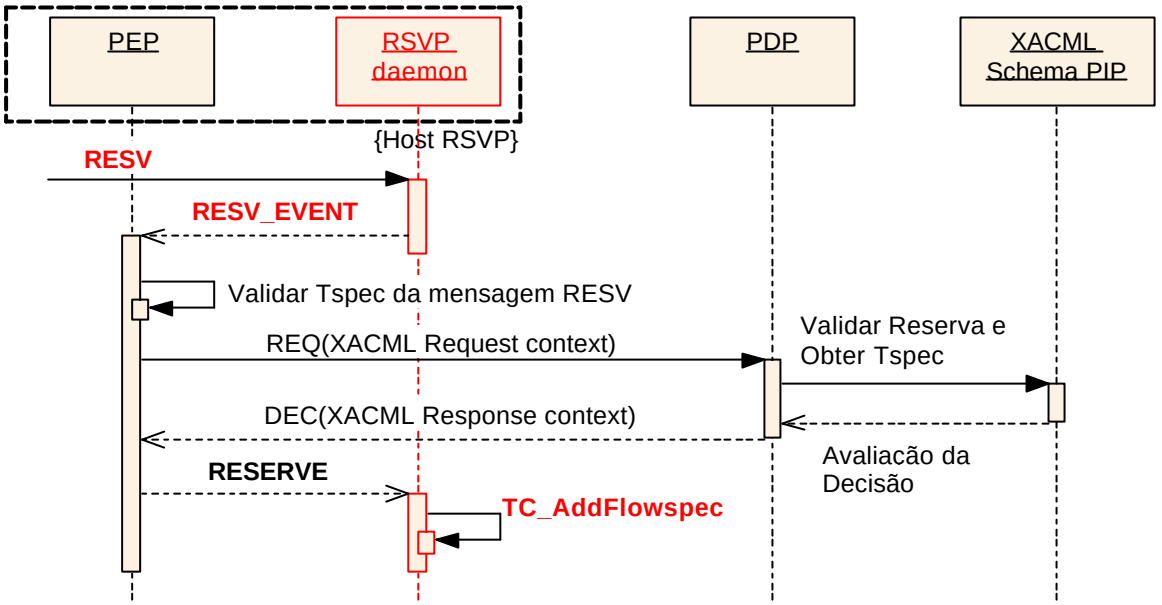

Figura 5.7 Validação da reserva de fluxo de dados no SENDER

19. O RSVP *daemon* gera um evento RESV_EVENT ao receber uma solicitação de reserva de fluxo de dados do RECEIVER. O PEP deve validar o fluxo de dados antes de estabelecer uma reserva de fluxo RSVP, através da mensagem "Validar Tspec da mensagem RESV", analisando o *Tspec* e tipo de Serviço RSVP enviados ao RECEIVER com o *Tspec* e número de serviço RSVP recebidos, identificados através da sessão RSVP e armazenados conforme descrição do algoritmo 5.6. O algoritmo 5.9 descreve o procedimento de validação do *Tspec*.

```
Mensagem: "Validar Tspec da Mensagem RESV"  [PEP]  Triggered:RSVPD
// (1) Comparar TspecSender armazenado no PEP com TspecReceiver
//     [evento RSVPD]
// (2) Número RsvpService pode ser: Controlled[5];Guaranteed[2]; RFC 2210
{ Se((TspecReceiver_menor_que_ou_igual_TpecSender(Tspec1,Tscpec2)=
     Verdadeiro) AND (SenderRsvpService = ReceiverServiceNum)) então
  {   Invocar RESERVE; // API RSVP RFC2205
  }
  Senão
      Requisitar Decisão ao PDP;//Mensagem: "REQ(XACML Request Context)"
}

Boolean TspecReceiver_menor_que_ou_igual_TpecSender(Tspec1, Tscpec2);
// (1) Algoritmo "less_than_or_equal"  RFC 2210  Tspec{r, b, p, m, M}
// (2) Tspec2: Recebidos da "Authorization Decision" e armazenados no PEP;
// (3) Tspec1: Dados do RECEIVER recebidos através do evento "RESV_ENVENT"
//             RFC2205 pg 66
{ Se ((r1 = r2)AND(b1 = b2)AND(p1 = p2)AND(m1 =  m2)AND(M1 = M2))então
      Retorne(Verdadeiro);
  senão
      Retorne(Falso);
}
```

Algoritmo 5.9 Validar *Tspec* da Mensagem RESV

20. Caso a validação dos dados seja diferente, o PEP deve solicitar a confirmação ao PDP, descrita através da mensagem "REQ (XACML *Request* ***context***)", informando os dados de endereços IP do *RECEIVER* e *SENDER* (aplicação), dados do tráfego e qual é a ação desejada, neste caso a confirmação de Reserva declarada no elemento <Action>. É utilizado o protocolo COPS REQ( ) para encapsular a mensagem XACML e que mantém o estado da sessão COPS com o servidor PDP. O algoritmo 5.10 apresenta o procedimento da mensagem de requisição de decisão de autorização enviado do PEP ao PDP.

```
Mensagem: "REQ(XACML Request Context)"  [PEP → PDP]
// (1) Mensagem "Validar Tspec da Mensagem RESV" não validou "RESV"
{ Gerar a mensagem "XACML Request context" definindo os elementos:
       <Subject>        IP RECEIVER  / IP SENDER
       <Resource>       hostname SENDER
       <ResourceRsvp>   Tspec{r,b,p,m,M}, RsvpService // NumService
       <Action>         setInstallReserve
   Encapsular "XACML Request context" no COPS (REQ);
   Enviar Requisição COPS(REQ)
   Aguardar Decisão do PDP;
}
```

Algoritmo 5.10 REQ (XACML *Request* ***context***)

21. A lógica para validação de informações foi discutida no capítulo 4, orientando a

formação de uma política na composição de regras. Neste caso, para validar os parâmetros do ***Tspec***, definidos no elemento <Rule>, deverá ser utilizada a extensão do elemento <AttributeSelectorRsvp>, descrito neste capítulo, possibilitando localizar os elementos do ***Tspec*** do <Resource>, ou seja, do Servidor Multimídia, e retornar a ***authorization decision***, permitindo ou negando a instalação da reserva de fluxo de dados. O algoritmo 5.11 descreve o procedimento de validação de reserva de fluxo.

```
Mensagem "Validar Reserva e Obter Tspec" [PDP]
// (1) Não necessita processar Obligations. Não há retorno de Tspec.
// (2) Comparar valores na Policy utilizando o elemento
//     <AttributeSelectorRsvp>
{Obter "XACML Request context" encapsulados no COPS REQ() e Processar a
 Política:

 Validar Policy <Target>
   Policy <Resource> hostnameServer (igual) XACML Request <Resource>
 Validar <Rule>
   // expressões lógicas
   Rule <Subject>  IP Address      (igual) XACML Request <Subject>
   Rule <Resource> Tspec{r,b,p,m,M}, RsvpService (válido) XACML Request <Resource>
   // Recursos de Policy Tspec limitados pelo Administrador ou,
   // Administrador define Reserva de Recurso para <Resource>. O PDP deve
   // controlar.
   Rule <Action> setInstallReserve (igual)XACML Request <Action>
   Validar <Conditions> , se definidas;
   // ex: Restrição de Horário  [09h00 – 17h00]
 Se Regra = "Permit" então
    Formatar Decisão de Autorização("Permit");
 senão
    Formatar Decisão de Autorização ("Deny");
}
```

Algoritmo 5.11 Validar Reserva e Obter *Tspec*

22. No exemplo abaixo, é validado o nome do servidor **MultimidiaServer**, utilizando o elemento <ResourceAttributeDesignator> para pesquisar na descrição do XACML *Request* ***context*** com o nome do Recurso descrito na ***policy*** e, na seqüência a utilização do elemento <AttributeSelectorRsvp> que pesquisa na própria ***policy*** o nome do Serviço RSVP comparado ao nome do Serviço informado no XACML *Request* ***context*** através do elemento <ResourceAttributeDesignator>, devendo validar os demais atributos, como o *Tspec*, segundo exemplo abaixo e descrição da lógica para validação, já discutidos anteriormente.

```
<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
   <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">MultimidiaServer
```

```
        </AttributeValue>
        <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeSelectorRsvp
            PolicyPath="//Policy/Target/Resource/ResourceRsvp/RsvpService"
            DataType="http://www.w3.org/2001/XMLSchema#string">
        </AttributeSelectorRsvp>
        <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:RsvpService"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </ResourceMatch>
```

23. O resultado é retornado pelo PDP com a decisão ao PEP. A mensagem “Avaliação da Decisão” gerada pelo PDP é apresentada no algoritmo 5.12.

```
Mensagem “Avaliação da Decisão” [PDP]
// (1) Decisão de Autorização do PDP
{  Encapsular na mensagem COPS (DEC) o “XACML Response Context”;
   Responder ao PEP via COPS (DEC);
}
```

Algoritmo 5.12 Avaliação da Decisão para RSVP RESV

24. O PEP ao receber a mensagem “DEC (XACML *Response* ***context***)”, via protocolo COPS DEC( ), obtém os dados XACML com a avaliação de decisão do PDP. Se a resposta do PDP é válida, o PEP confirma a instalação do fluxo RSVP através da mensagem RESERVE, uma API sugerida na RFC 2205. Os parâmetros desta API estão definidos no item 3.11.1 da RFC 2205 e utiliza dados como a sessão RSVP, dados do *Tspec* e tipo de serviço definido para o fluxo. Os procedimentos são descritos no algoritmo 5.13.

```
Mensagem “DEC(XACML Response Context)” [PDP → PEP]
// (1) Processada pelo PEP em resposta a solicitação
{ Obter dados do “XACML Response context” encapsulados no COPS (DEC);
  Se (decisão = "DENY") senão
    Informar condição de erro para a aplicação (SENDER);
  Senão
    Chamar mensagem RESERVE; // Dados definidos na API RFC 2205
}
Mensagem “RESERVE” [PEP → RSVP Daemon]
// (1) PEP invoca o RSVP Daemon para INSTALAR RESERVA de fluxo QoS RSVP
// (2) Msg RESERVE é uma API sugerida na RFC 2205 CALL RESERVER
{ Invocar RSVP Daemon
} // parâmetros definidos no item 3.11.1 RFC 2205 - pg 65.
```

Algoritmo 5.13 Mensagens: DEC (XACML *Request* ***context***) e *RESERVE*

25. A mensagem “RESERVE” descrita no algoritmo 5.13 invoca um procedimento *“TC_AddFlowspec”* ao controle de tráfego utilizando parâmetros do ***flowspec***, discutido no capítulo 2, através de uma API sugerida na RFC 2205.

As mensagens são validadas sempre pelo PDP seguindo a seqüência descrita neste tópico.

5.4 Cenário completo

O cenário completo de mensagens entre RECEIVER, SENDER s(PEP) e o Servidor de Políticas (PDP) utilizando o mapeamento das mensagens COPS, consta no Anexo E.

5.5 Conclusões do Capítulo

Nesta proposta, a aplicação solicita ao PEP a reserva de um fluxo RSVP que se comunica com o PDP. Este avalia o direito de acesso do usuário e do servidor de aplicação QoS através de uma consulta de política definida em XACML, e informa ao PEP os parâmetros para a reserva do fluxo. O PEP invoca o RSVP *daemon*, tornando a aplicação passiva neste processo, determinando a principal mudança conceitual em uma reserva de fluxo RSVP. Portanto, este modelo não se aplica a roteadores RSVP-*Aware*.

A utilização de um servidor de políticas, com definições de políticas de acesso centralizadas, para obtenção de parâmetros para tráfego de dados QoS RSVP propicia melhorias na área de segurança. Isto possibilita controlar o acesso, o tipo de serviço e a quantidade de tráfego que será disponibilizada para servidores de aplicações QoS RSVP, descritas em políticas, utilizando XACML.

As aplicações de QoS podem transmitir e/ou receber dados, dependendo da característica de sua utilização. Este trabalho estabeleceu funcionalidades para servidores de aplicações de QoS RSVP com fluxo unidirecional, controlados por um servidor de políticas para definição de controle de acesso e parâmetros para tráfego de dados QoS.

A utilização de XACML para definição de políticas de servidores de QoS, necessita de uma estrutura de dados, ou elementos em XACML, para definição de tráfego QoS e classes de aplicações. A estrutura definida na política deve permitir a sua reutilização na pesquisa de parâmetros de QoS durante uma decisão de autorização, o que não é suportada atualmente pelo XACML. Portanto, a extensão do elemento <AttributeSelector>, nesta proposta denominado <AttributeSelectorRsvp>, possibilita a localização de dados QoS em um XACML

policy, tanto para fornecer quanto para validar os parâmetros ***Tspec***, em um XACML *Response* ***context***.

A utilização de uma estrutura PBNM proporciona um controle melhor nas definições de políticas de controle de acesso, e juntamente com o XACML, oferece uma alternativa viável para representação de políticas, apesar das limitações discutidas em relação ao modelo de políticas orientado a objetos, proposta pela IETF.

Capítulo 6

Estudo de caso e avaliação

O processamento do PDP na proposta da OASIS não especifica execução de procedimentos nas definições dos elementos *"Obligations"*. Utilizando a arquitetura de implementação da SUN XACML[25] que segue a definição da OASIS, não foi possível, neste trabalho, a execução de testes de retorno de dados *Tspec* após validação de condições de requisições do PEP, sem que houvesse a necessidade de declarar novamente os dados de tráfego de dados no elemento <Obligations>. Portanto, não foi utilizado este procedimento por necessitar a declaração dos mesmos parâmetros de QoS em uma política evitando reutilizar os parâmetros definidos em <Resources>. Logo, há necessidade de acrescentar algoritmos de processamento do PDP da arquitetura e manter sua funcionalidade. Como não foram computados dados para avaliação de desempenho é irrelevante citar os equipamentos utilizados e suas capacidades.

A avaliação das mensagens entre o RECEIVER, SENDER (PEP) e PDP descritas no cenário completo, apresentado no Anexo E, foi usada para validação da proposta. Utilizou-se a linguagem Java com a plataforma de desenvolvimento Java™ 2 SDK, *Standard Edition* 1.4.2, ambiente de programação Sun™ ONE Studio 4 *update* 1 e a arquitetura de implementação Sun XACML, tecnologias da SUN *Microsystems.*

Para avaliar a versatilidade na descrição de políticas utilizando o XACML, foram executados os seguintes procedimentos: descrição de uma política; requisição do PEP de uma ***decision request*** ao PDP; processamento do PDP retornando uma ***authorization request*** ao PEP e a validação dos XACML *request* ***context*** e XACML *policy* com o XACML *Schema*

[25] Sun XACML: arquitetura proposta em Java para suportar XACML no endereço URL: http://sourceforge.net/projects/sunxacml .

"*cs-xacml-schema-policy-01-rsvp.xsd*" contendo os elementos QoS RSVP, que foram extendidos nesta proposta e descritos no capítulo 5, utilizando a ferramenta XMLSpy® v.5.0, *release* 4.

O exemplo de política deste estudo de caso descreve a ação do cliente *RECEIVER* que deseja serviços de QoS do *SENDER*, chamado ***MultimidiaServer***, e seguiu as definições para especificar os parâmetros, conforme apresentado no Anexo C. As definições do anexo C têm como objetivo apenas sugerir uma padronização para nomes de parâmetros e facilitar a definição de políticas para troca de informações com o PEP.

A política utilizada segue a seguinte premissa: *"um usuário autorizado pode acessar um recurso de um servidor de multimídia, identificado pelo endereço de rede IP e no período entre às 09:00 horas e às 17:00 horas"*.

O usuário é o RECEIVER identificado pelo endereço IP "**192.168.200.192**" e o Servidor de Multimídia, referido como SENDER, tem o endereço IP "**192.168.200.1**" e o nome da máquina sendo ***MultimidiaServer*** (*hostname*). A ação que será avaliada é declarada na requisição como "***getResourceQoS***". A condição testada será a restrição de horário de acesso ao recurso, estabelecida entre **09:00:00** e às **17:00:00** horas.

6.1 Descrição de um XACML *Request context*

O PEP deve informar na ***decision request***: Cabeçalho, *Subject*, *Resource* e *Action*. A lógica de validação está detalhada no capítulo 4. A formatação completa deste XACML *Request* ***context*** encontra-se no Anexo B.

Cabeçalho: descrição do Request.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
D:\XACML\qos\xacml-word\cs-xacml-schema-context-01.xsd">
```

Elemento <Subject>: dois atributos, o IP RECEIVER "192.168.200.192" e o IP SENDER "192.168.200.1".

```
  <Subject>
    <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:receiver"
        DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>192.168.200.192</AttributeValue>
    </Attribute>
    <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:sender"
```

```
            DataType="http://www.w3.org/2001/XMLSchema#string">
          <AttributeValue>192.168.200.1</AttributeValue>
        </Attribute>
      </Subject>
```

Elemento <Resource>: o nome do recurso, "**MultimidiaServer**".

```
    <Resource>
      <Attribute
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">
          <AttributeValue>MultimidiaServer</AttributeValue>
      </Attribute>
    </Resource>
```

Elemento <Action>: ação que será validada para o recurso, "**getResourceQoS**".

```
    <Action>
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id:ServerAction"
            DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>getResourceQoS</AttributeValue>
      </Attribute>
    </Action>
</Request>
```

6.2 Descrição de um XACML *Policy*

Política definida para o Servidor PEP – Multimidia Server – avalia uma ***decision request*** recebida na mensagem COPS.

A política define acesso ao <Resource> chamado de **MultimidiaServer** para usuários autorizados, identificados pelo endereço IP.

Uma ***authorization request*** somente é valida (*PERMIT*) se o recurso for para o servidor **MultimidiaServer** e o usuário *RECEIVER* for autorizado a utilizar recursos do Servidor QoS, no período definido pela política. A formatação completa desta ***policy*** encontra-se no Anexo B.

A ***Policy*** deve conter uma estrutura com um cabeçalho, *Target*, *Rules* e *Obligations*, descritas a seguir.

Cabeçalho: contém o nome da *Policy* "...:policy:MultimidiaPolicy"" e o algoritmo utilizado para avaliação RuleCombiningAlgId="....:**rule-combining-algorithm:first-applicable**">.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="urn:oasis:names:tc:xacml:1.0:policy:MultimidiaPolicy"
xmlns="urn:oasis:names:tc:xacml:1.0:policy"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy  cs-xacml-schema-policy-01-rsvp.xsd"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">
  <!-- Esta politica se aplica somente para requisições no servidor chamado MultimidiaServer -->
```

Elemento <Target>: o nome do recurso que a política validará, o <Resource> "MultimidiaServer".

```
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">MultimidiaServer
            </AttributeValue>
          <ResourceAttributeDesignator
            DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
        <ResourceRsvp
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource-id:multimidiaserver"
            RsvpClass="G711">
              <TspecBucketRate_r>9250</TspecBucketRate_r>
              <TspecBucketSize_b>680</TspecBucketSize_b>
              <TspecPeakRate_p>13875</TspecPeakRate_p>
              <TspecMinPoliceUnit_m>340</TspecMinPoliceUnit_m>
              <TspecMaxPacketSize_M>340</TspecMaxPacketSize_M>
              <RsvpService>Guaranteed</RsvpService>
         </ResourceRsvp>
        <!-- <AnyResourceRSVP/> -->
      </Resource>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
```

Elemento <Rule>: contém o nome da primeira regra chamada "ResourceQoS" e três elementos para validação das regras, o <Subject> com os endereços a serem validados, o IP RECEIVER "**192.168.200.192**" e o do IP SENDER "**192.168.200.1**", a <Action> que será avaliada para o recurso declarado como "**getResourceQoS**" e a <Condition> **09:00:00 - 17:00:00** que restringe o horário de acesso ao recurso.

```
<!—Regra para recursos de QoS no SENDER MultimidiaServer e com restrições de horário -->
  <Rule RuleId="urn:oasis:names:tc:xacml:1.0:ResourceQoS" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
        <!-- Identificador do subject RECEIVER: urn:oasis:names:tc:xacml:1.0:subject:
```

```
                authn-locality:ip-address:receiver -->
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">192.168.200.192
                </AttributeValue>
                <SubjectAttributeDesignator
                        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:
                                ip-address:receiver"
                        DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </SubjectMatch>
            <!-- Identificador do subject SENDER: urn:oasis:names:tc:xacml:1.0:subject:
                 authn-locality:ip-address:sender -->
            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">192.168.200.1
                </AttributeValue>
                <SubjectAttributeDesignator
                    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:
                            ip-address:sender"
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </SubjectMatch>
            </Subject>
            <!-- <AnySubject/> -->
        </Subjects>
        <Resources>
            <AnyResource/>
        </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
                    DataType="http://www.w3.org/ 2001/XMLSchema#string">getResourceQoS
                    </AttributeValue>
                    <ActionAttributeDesignator
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id:ServerAction"/>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
    <!--Permite serviços de QoS compreendido no horário de 09:00 às 17:00 horas-->
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
                <EnvironmentAttributeDesignator
                    DataType="http://www.w3.org/2001/XMLSchema#time"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
            </Apply>
            <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00
                </AttributeValue>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
                <EnvironmentAttributeDesignator
                    DataType="http://www.w3.org/2001/XMLSchema#time"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
            </Apply>
```

```
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00
            </AttributeValue>
        </Apply>
    </Condition>
</Rule>
```

Elemento <Rule>: contém o nome da segunda regra chamada "***FinalQoSRule***", e se a avaliação da regra anterior for inconclusiva, aplica a avaliação com valor "***Deny***".

```
<!-- Caso o recurso não seja possível de avaliação, retorna um Deny ao PEP
    (evita resposta NotApplicable) -->
<Rule RuleId="urn:oasis:names:tc:xacml:1.0:FinalQoSRule" Effect="Deny"/>
```

Elemento <Obligations>: foi definido com o nome ObligationId="**MultimidiaServer**" e definido o atributo FulfillOn="**Permit**". Caso uma avaliação seja autorizada, o PDP retornará ao PEP a descrição dos valores ***Tspec*** declarados no <Resource> da ***policy***, utilizando os elementos <AttributeSelectorRsvp>. Cada elemento define o local para obtenção do valor do ***Tspec*** descrito pelo atributo PolicyPath.

```
<Obligations>
    <Obligation ObligationId=" urn:oasis:names:tc:xacml:1.0:obligation:MultimidiaServer"
            FulfillOn="Permit">
    <AttributeAssignment AttributeId="TspecBucketRate_r"
        DataType="http://www.w3.org/2001/XMLSchema#double">
        <AttributeSelectorRsvp
            PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecBucketRate_r"
            DataType="http://www.w3.org/2001/XMLSchema#double">
        </AttributeSelectorRsvp>
    </AttributeAssignment>
    <AttributeAssignment AttributeId="TspecBucketSize_b"
        DataType="http://www.w3.org/2001/XMLSchema#double">
        <AttributeSelectorRsvp
            PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecBucketSize_b"
            DataType="http://www.w3.org/2001/XMLSchema#double">
        </AttributeSelectorRsvp>
    </AttributeAssignment>
    <AttributeAssignment AttributeId="TspecPeakRate_p"
        DataType="http://www.w3.org/2001/XMLSchema#double">
        <AttributeSelectorRsvp
            PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecPeakRate_p"
            DataType="http://www.w3.org/2001/XMLSchema#double">
        </AttributeSelectorRsvp>
    </AttributeAssignment>
    <AttributeAssignment AttributeId="TspecMinPoliceUnit_m"
        DataType="http://www.w3.org/2001/XMLSchema#integer">
        <AttributeSelectorRsvp
            PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecMinPoliceUnit_m"
            DataType="http://www.w3.org/2001/XMLSchema#integer">
        </AttributeSelectorRsvp>
```

```
        </AttributeAssignment>
        <AttributeAssignment AttributeId="TspecMaxPacketSize_M"
            DataType="http://www.w3.org/2001/XMLSchema#integer">
            <AttributeSelectorRsvp
                PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecMaxPacketSize_M"
                DataType="http://www.w3.org/2001/XMLSchema#integer">
            </AttributeSelectorRsvp>
        </AttributeAssignment>
        <AttributeAssignment AttributeId="RsvpService"
            DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeSelectorRsvp
                PolicyPath="//Policy/Target/Resource/ResourceRsvp/RsvpServi ce"
                DataType="http://www.w3.org/2001/XMLSchema#string">
            </AttributeSelectorRsvp>
        </AttributeAssignment>
    </Obligation>
  </Obligations>
</Policy>
```

6.3 Descrição de um XACML *Response context*

O PDP deve informar o resultado da ***authorization request*** ao PEP e se o resultado avaliado constar alguma *Obligation*, esta deve ser enviada com a avaliação em um XACML *Response* ***context***. A estrutura do XACML *Request* deve conter: Cabeçalho, *Result*, *Status* e *Obligations*, sendo opcional quando não definido no resultado da *policy*.

A lógica de validação está detalhada no capítulo 4. A formatação completa deste XACML *Request* ***context*** encontra-se no Anexo B.

Segue a descrição do XACML *Response* ***context***, indicando a decisão como "**Permit**", um Status descrito como "**...:status:ok**" e neste caso, possui Obligations, apresentados a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response
    xmlns="urn:oasis:names:tc:xacml:1.0:context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
        cs-xacml-schema-context-01.xsd">
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
      <Obligations  xmlns="urn:oasis:names:tc:xacml:1.0:policy">
        <Obligation ObligationId="urn:oasis:names:tc:xacml:1.0:obligation:MultimidiaServer"
        FulfillOn="Permit">
          <AttributeAssignment AttributeId="TspecBucketRate_r"
            DataType="http://www.w3.org/2001/XMLSchema#double">9250</AttributeAssignment>
          <AttributeAssignment AttributeId="TspecBucketSize_b"
            DataType="http://www.w3.org/2001/XMLSchema#double">680</AttributeAssignment>
```

```
            <AttributeAssignment AttributeId="TspecPeakRate_p"
              DataType="http://www.w3.org/2001/XMLSchema#double">13875</AttributeAssignment>
            <AttributeAssignment AttributeId="TspecMinPoliceUnit_m"
              DataType="http://www.w3.org/2001/XMLSchema#integer">340</AttributeAssignment>
            <AttributeAssignment AttributeId="TspecMaxPacketSize_M"
              DataType="http://www.w3.org/2001/XMLSchema#integer">340</AttributeAssignment>
            <AttributeAssignment AttributeId="RsvpService"
            DataType="http://www.w3.org/2001/XMLSchema#string">Guaranteed</AttributeAssignment>
        </Obligation>
    </Obligations>
    </Result>
</Response>
```

6.4 Conclusões do Capítulo

A definição de elementos para representação de classes de QoS e *Tspec* (tráfego de dados) em um XML *Schema* propicia um controle de fluxo efetivo na declaração de políticas, evitando a descrição de políticas pelo administrador com nomes de elementos diferentes na descrição de outras políticas de QoS. Esta característica é assegurada com análise e definições de nomenclaturas a serem utilizadas nos elementos XML, que oferecem controles usando tipos enumerados e restrições para os limites dos valores de parâmetros no próprio XML *Schema*. Portanto, a versatilidade que o XML propicia nas definições de elementos pode ser controlada na definição e uso do XML *Schema*.

A definição de nomes de elementos para parâmetros de QoS utilizados juntamente com a sintaxe XACML, e a unicidade de conceitos discutidos nos modelos de processamento PBNM propiciam melhorias na segurança de rede e facilitam o controle e administração de políticas pelo admistrador. Um exemplo de definições de nomes de elementos para parâmetros de QoS utilizando XACML é apresentado no anexo C.

O processamento do PDP na proposta da OASIS não especifica execução de procedimentos nas definições dos elementos "*Obligations*". Utilizando a arquitetura de implementação da SUN XACML que segue a definição da OASIS, não foi possível a execução de testes de retorno de dados *Tspec* após validação de condições de requisições do PEP, sem que houvesse a necessidade de declarar novamente os dados de tráfego de dados no elemento <Obligations>. Este procedimento evitaria a reutilização dos parâmetros definidos em <Resources> e é antagônico à descrição desta proposta de dissertação. Portanto, há necessidade de alterar os algoritmos de processamento do PDP da arquitetura SUN XACML mantendo suas funcionalidades originais. Por este motivo não foram apresentados resultados

para autorização de instalação de fluxos, sendo uma expansão futura deste trabalho e possibilitar a avaliação de resultados de desempenho com XACML.

Capítulo 7

Conclusão

Este trabalho apresentou uma proposta para controle de serviços de QoS com protocolo RSVP através da alteração de procedimentos para solicitação de QoS, obtenção de *Tspec* e reserva de fluxos por intermédio do PEP. O controle sobre os recursos disponíveis no Servidor de QoS, com a obtenção de parâmetros em um PDP utilizando o XACML e a garantia da aplicação das políticas, é possível através de uma Aplicação (*Sender*) que solicita um serviço de QoS ao PEP, que invoca os serviços de QoS ao *RSVP daemon*, após decisão do PDP.

O PEP não atua somente como um *enforcement* para aplicação de políticas, mas possui um papel ativo na chamada ao *RSVP daemon*, através de APIs sugeridas pela RFC 2205, e controle sobre sessões RSVP para administrar a instalação do fluxo de QoS, trabalhando em conjunto com o PDP.

O modelo apresentado no trabalho para controle de Servidores de QoS é apropriado para servidores de aplicações que fornecem serviços de QoS, o que restringe a aplicabilidade para controle de QoS RSVP, além da característica de escalabilidade do protocolo RSVP em redes locais e a necessidade de dispositivos RSVP-*Aware*.

Na proposta da OASIS, o XACML ainda encontra-se em desenvolvimento, embora limitado em alguns aspectos pela falta de padronização, mostra-se flexível para novas aplicações e com potencial de desenvolvimento para integração com outras tecnologias como protocolos de comunicação, local de armazenamento de políticas e serviços de autenticação. Um exemplo desta evolução é a definição de políticas para suportar representação de papéis

para controle de acesso (RBAC) [26], que apesar de ser um esboço no momento, o trabalho encontra-se em estágio avançado e, em breve, poderá ser parte integrante da especificação padrão do XACML.

O retorno de dados pelo PDP, no modelo da OASIS, não suporta a representação do tráfego de dados necessários à reserva de um fluxo QoS sem que haja reuso destes dados. Nesta dissertação foi proposta o processamento do PDP no elemento *Obligation* para suportar tal necessidade, uma característica há muito suportada na representação da IETF. Apesar disso, os algoritmos para decisão do modelo XACML proporcionam maior flexibilidade na definição de políticas, uma vantagem deste modelo de arquitetura em relação ao modelo orientado a objetos da IETF.

O modelo de descrição de políticas XACML da OASIS, apresentado como sendo genérico, flexível e um padrão aberto para representação de políticas, mas não se mostrou real em todos aspectos. A falta de elementos para representação de objetos de QoS e a devida manipulação de dados para avaliação em decisões de políticas é um ponto deficiente no modelo XACML. A reutilização de dados de QoS, através da representação do *Tspec*, demonstrou este tipo de limitação e a necessidade de extender o modelo, contribuição feita por este trabalho.

A extensão do modelo descritivo para suportar QoS com o protocolo RSVP, no modelo outsourcing, envolve a definição da estrutura de tráfego de dados *Tspec* e dados para classes de serviços definidos no modelo XACML, com restrições de tipos e validação de valores no XACML *Schema*, segundo definição do *Tspec* da RFC2210.

A utilização do XACML e RSVP para controle de admissão possibilitam ampliar a pesquisa em trabalhos futuros tais como: definição do local de encapsulamento do XACML na mensagem COPS; utilização do objeto RSVP POLICY_DATA para controle de autenticação; definição de políticas para mapear recursos de QoS RSVP na Rede e ainda realizar o estudo de viabilidade da expansão do XACML para suportar o modelo *Provisioning*.

Este trabalho propôs a utilização do protocolo QoS-RSVP para alocação de recursos submetidos a uma política descritiva centralizada XACML, e concluí-se que:

1. O protocolo RSVP é complexo, porém fornece serviço garantido. Possibilita o

[26] RBAC: *Role Based Access Control* - controle de acesso baseado em papéis. O controle de acesso é feito considerando-se a função (papel) de cada usuário e não sua identidade.

controle em servidores de QoS com o PEP invocando serviços de QoS, garantindo a segurança na reserva de recursos.

2. O XACML não possui padrão de protocolo de comunicação e local de armazenamento de políticas definido pelo OASIS.
3. O XACML é flexível e extensível, suportando novas exigências no modelo de políticas.
4. O XACML por ser um padrão aberto possibilita a criação de novas ferramentas de controle para administração de políticas.
5. O XACML ainda encontra-se em desenvolvimento.
6. O protocolo COPS é adequado a este modelo por permitir o encapsulamento do XACML em sua mensagem e padronizado pela IETF.
7. Não foi definido o local do encapsulamento do XACML na mensagem COPS, podendo ser uma expansão futura deste modelo.
8. Pode ser utilizado o objeto POLICY_DATA do RSVP para controle de autenticação, podendo ser extensão futura desta proposta.
9. Atualmente o XACML não suporta o modelo *Provisioning*.
10. O modelo de políticas orientado a objetos da IETF se encontra em uma fase muito avançada em relação ao modelo XACML da OASIS.

Referências Bibliográficas

[ALM92] Almquist, P. *Type of Service in the Internet Protocol Suite*, **RFC1349**, Jul. 1992.

[ATK95a] Atkinson, R. *IP Authentication Header*, **RFC1826**, NRL, Aug 1995.

[ATK95b] Atkinson, R. *IP Encapsulating Security Payload*, **RFC1827**, NRL, Aug 1995.

[BER97] Berger, L; O'Malley, T. *RSVP Extensions for IPSEC IPv4 Data Flows*, **RFC2207**, Sep. 1997.

[BER96] Berners-Lee, T.; Fielding, R.; Frystyk, H. *Hypertext Transfer Protocol -HTTP/1.0*, **RFC1945** MIT/LCS, UC Irvine, May. 1996.

[BAK01] Baker, F. et al. *Aggregation of RSVP for IPv4 and IPv6 Reservations*. **RFC3171**, Sep. 2001.

[BLA98] Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. *An Architecture for Differentiated Services*, **RFC2475**, Dec. 1998.

[BER98a] Berger, L. *RSVP over ATM Implementation Guidelines*, **RFC2379**, *Best Current Practice – (BCP) 24*, Aug. 1998.

[BER98b] Berger, L. *RSVP over ATM Implementation Requirements*, **RFC2380**, Aug. 1998.

[BOY00] Boyle, J.; Cohen, R.; Durham, D.; Herzog, S.; Rajan, R.; Sastry, A. *The COPS* (*Common Open Policy Service*) *Protocol*, **RFC2748**, Jan. 2000.

[BRA94] Braden, R.; Clark, D.; S. Shenker, *Integrated Services in the Internet Architecture: An Overview*, **RFC1633**, Jul. 1994.

[BRA97a] Braden, R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S. *Resource Reservation Protocol (RSVP) Version 1 Functional Specification*, **RFC2205**, Sep. 1997.

[BRA97b] Braden, R.; Zhang, L. *Resource Reservation Protocol (RSVP) Version 1 - Message Processing Rules*, **RFC2209**, Sep. 1997.

[BRA00] Bray , T.; Hollander, D.; Layman, A. *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C Recommendation, Oct. 2000. URL: http://www.w3.org/TR/REC-xml

[CAI00] Cai, M.; Xie, J.; Wang, Y.; Gu, G. *An Implementation Model of IntServ/RSVP based CORBA A/V Stream Service*, The 36th IEEE International Conference on Technology of Object-Oriented Languages and Systems, Xi'an, pp.94-99. IEEE Oct. 2000.

[CCI99] CCISTE. *Common Criteria for Information Technology Security Evaluation* (*CCITSE*) - *Common Criteria* (*CC*). Trusted Product Evaluation Program (TPEP). 1999. URL: http://www.radium.ncsc.mil/tpep/library/ccitse/index.html e URL: http://www.commoncriteria.org/

[CIS01a] Cisco. *Cisco White paper – Class-Based Weighted Fair Queueing*. CISCO. Jun. 2001. URL: http://www.cisco.com/

[CIS01b] Cisco. *Cisco White paper - Understanding Delay in Packet Voice Networks*. CISCO. 2001. URL: http://www.cisco.com/

[CIS02a] Cisco. *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.1*. CISCO. Jan. 2002. URL: http://www.cisco.com/univercd/

[CIS02b] Cisco. *Quality of Service Networking. Internetworking Technologies Handbook*. CISCO. Feb. 2002. URL: http://www.cisco.com

[CIS02c] Cisco. *Resource Reservation Protocol (RSVP) Internetworking Technologies Handbook. Chapter 48*. CISCO. Feb. 2002. URL: http://www.cisco.com/

[CIS02d] Cisco. *Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting*. CISCO. Nov. 2002. URL: http://www.cisco.com/univercd/

[CIS02e] Cisco. *Understanding and Configuring QoS. Chapter 20. Software Configuration Guide—Release 12.1(8a)EW*. CISCO. Nov. 2002. URL: http://www.cisco.com/

[COU96] Coulouris, G. et al. *Distributed Systems – Concepts and Design*. 3.ed. England, Addison-Wesley, 1996.

[CHA01] Chan K.; Seligson, J.; Durham, D.; Gai, S.; McCloghrie, K.; Herzog, S.; Reichmeyer, F.; Yavatkar, R.; Smith, A. *COPS Usage for Policy Provisioning* (*COPS-PR*), **RFC3084**, Mar. 2001.

[CHO97] Chow, R.; Johnson, T.; Chow, Y. *Distributed Operating Systems & Algorithms*. Addison Wesley, 1997, ISBN 0-201-49838-3.

[DOD85] US Department of Defense. *Trusted Computer System Evaluation Criteria, DoD 5200.28-STD*. Washington, D.C., US Department of Defense, USA. Dec. 1985.

[DMT99] Distributed Management Task Force, Inc. *Common Information Model (CIM) Specification Version 2.2*. DMTF. Jun. 1999. URL: http://www.dmtf.org/standards/documents/CIM/DSP0004.pdf

[FER00] Ferraiolo, D. F.; Sandhu, R.; Gavrila, S.; Chandramouli, R; Kuhn, D.R. *A Proposed Standard for Role-Based Access Control*. National Institute of Standards and Technology. USA. Dez. 2000.

[FIE99] Fielding, R. et al. *Hypertext Transfer Protocol - HTTP/1.1*. **RFC2616**, Jun. 1999.

[HER00a] Herzog, S.; Rajan, R.; Sastry, A. *COPS usage for RSVP*, **RFC2749**, Jan. 2000.

[HER00b] Herzog, S. *RSVP Extensions for Policy Control*, **RFC2750**, Jan. 2000.

[IEE98] The Institute of Electrical and Electronics Engineers, Inc. *Logical Link Control (802.2), ANSI/IEEE Std 802.2, 1998 Edition*. IEEE Computer Society. USA, 1998. URL: http://standards.ieee.org/reading/ieee/std/lanman/802.2-1998.pdf

[ITU01] ITU-T Recommendation G.1010. *Series G: Transmission Systems And Media, Digital Systems And Networks Quality Of Service And Performance*. ITU - International Telecommunication Union. FR, Nov. 2001.

[JAS01] Jason, J.; Rafalow, L; Vyncke, E. *IPsec Configuration Policy Model - ICPM , Work in Progress, draft-ietf-ipsp-config-policy-model-04.txt*. IETF, Nov. 2001.

[McC01] McCloghrie, K.; Fine, M.; Reichmeyer, F.; Hahn, S.; Chan, K.; Seligson, J.; Sahita, R.; Smith, A. *Structure of Policy Provisioning Information (SPPI)*, **RFC3154**, Aug.2001.

[MOO01a] Moore, B.; Ellesson, E.; Strassner, J.; Westerinen, A. *Policy Core Information Model - Version 1 Specification*, **RFC3060**, Feb. 2001.

[MOO01b] Moore, B.; Durham, D.; Halpern, J. ; Strassner, J.; Westerinen, A.; Weiss, W. *Information Model for Describing Network Device QoS Datapath Mechanisms, work in progress, draft-ietf-policy-qos-device-info-model-06.txt*. IETF. Nov 2001.

[MOO03] Moore, B.; Ellesson, E.; Strassner, J.; Westerinen, A. *Policy Core Information Model -- Version 1 Specification*, **RFC3460**, Jan. 2003.

[NCS88] National Computer Security Center. *Glossary of Computer Security Terms, NCSC-TG-004-88*. Ft. Meade, Md. US. Department of Defense, USA. Oct. 1988.

[NIC98] Nichols, K.; Blake, S.; Baker, F.; Black, D. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, **RFC2474**, Dec. 1998.

[OAS03] OASIS. *eXtensible Access Control Markup Language* (XACML) *Version* 1.0. OASIS, Feb. 2003. URL http://www.oasis-open.org/committees/xacml/

[OMG03] OMG – Object Management Group. *CommonObject Request Broker Architecture: CoreSpecification*. OMG Inc., Mass., USA. Dez. 2002. URL http://www.omg.org

[PON02] Ponnappan, A.; Yang, L.; Pillai, R.; Braun, P. *A Policy Based QoS Management System for the IntServ/DiffServ Based Internet*. Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (POLICY.02). IEEE, 2002.

[POS80a] Postel, J. *DoD Standard Internet Protocol*, **RFC760**. USC/Information Sciences Institute, Jan. 1980.

[POS80b] Postel, J. *User Datagram Protocol, STD 6*, **RFC768**, USC/Information Sciences Institute, Aug. 1980.

[POS81a] Postel, J. *Internet Protocol - Darpa Internet Program Protocol Specification, Std 5*, **RFC791**, USC/Information Sciences Institute, Sep. 1981.

[POS81b] Postel, J. *Transmission Control Protocol*, **RFC793**. USC / Information Sciences Institute, Sep. 1981.

[RAD99] RADCOM Academy, *A World of Protocols, Millennium Edition*. 2.ed., Published by RADCOM Academy, Jul. 1999. ISBN 965-90199-1-2.

[RAW03] Rawlins, D.; Kulkarni, A.; Bokaemper, M.; Chan, K. Framework for Policy Usage Feedback for Common Open Policy Service with Policy Provisioning (COPS-PR), **RFC3483**. Mar. 2003.

[SAH03] Sahita, R.; Hahn, S.; Chan, K.; McCloghrie, K. *Framework Policy Information Base*. **RFC 3318**, Mar. 2003.

[SAN98] Sandhu, R; Ahn, G-J. Decentralized group hierarchies in UNIX: An experiment and lessions learned. In National Information Systems Security Conference, 1998.

[SCH97] Schmidt, Douglas C.; Gophale, A.; Harrison, T.; Parulker, G. *A highperformance end system architecture for real-time CORBA*, IEEE Communication Magazine, pp.72-77, Feb. 1997. URL: http://www.cs.wustl.edu/~schmidt/TAO.html

[SHE97a] Shenker, S.; Partridge, C.; Guerin, R. *Specification of Guaranteed Quality of Service*, **RFC 2212**, Sep. 1997.

[SHE97b] Shenker, S.; Wroclawski, J. *General Characterization Parameters for Integrated Service Network Elements*, **RFC 2215**, Sep. 1997.

[SHI00] Shirey, R. *Internet Security Glossary*. *FYI 36*, **RFC 2828**, May 2000.

[SHI94] Shinghal, M.; Shivaratri, N. G. *Advanced Concepts in Operating Systems, Distributed, Database, and Multiprocessor Operating Systems.* McGraw-Hill, Inc.1994. ISBN 0-07-057572-X.

[SNI01] Snir, Y.; Ramberg, Y.; Strassner, J.; Cohen, R. *Policy QoS Information Model, work in progress, draft-ietf-policy-qos-info-model-04.txt*. IETF, Nov. 2001.

[SNI03] Snir, Y.; Ramberg, Y.; Strassner, J.; Cohen, R. *Policy QoS Information Model, work in progress, draft-ietf-policy-qos-info-model-05.txt*. IETF, May. 2003.

[STA99a] Stardust.com. *White Paper - QoS protocols & architectures*. Stardust.com, Inc, California. USA, Jul. 1999. URL: http://www.qosforum.com

[STA99b] Stardust.com. *Technology Backgrounder - Quality of Service – Glossary of Terms.* Stardust.com, Inc, California. USA, May. 1999. URL: http://www.qosforum.com

[STA99c] Stardust.com. *White Paper - Introduction to QoS Policies*. Stardust.com, Inc, California. USA, Jul. 1999. URL: http://www.qosforum.com

[STA99d] Stardust.com. *White Paper - The Need for QoS*. Stardust.com, Inc, California. USA, Jul. 1999. URL: http://www.qosforum.com

[STR02] Strassner, J.; Ellesson, E.; Moore, B.; Moats, R. *Policy Core LDAP Schema*, IETF Internet Draft, Feb. 2002.

[TAN95] Tanenbaum, A. S. *Distributed Operating Systems.* Prentice-Hall, Inc. New Jersey. USA, 1995. ISBN 0-13-219908-4.

[TCS85] Trusted Computer Security Evaluation Criteria. *TCSEC - DOD 5200.28-STD.* Department of Defense, 1985. URL: http://www.radium.ncsc.mil/tpep/

[W3C99] W3C. *XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999*. World Wide Web Consortium-W3C, Nov 1999. URL: http://www.w3.org/TR/xpath

[W3C01] W3C. *XML Schema, parts 1 and 2*. World Wide Web Consortium-W3C, May 2001. URL:http://www.w3.org/TR/xmlschema-1/ e URL:http://www.w3.org/TR/xmlschema-2/

[W3C02] W3C. *XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Working Draft*. World Wide Web Consortium-W3C, Aug 2002. URL: http://www.w3.org/TR/2002/WD-xquery-operators-20020816

[WAN01] Wang, Z. *Internet QoS: Architecture and Mechanisms for Quality of Service*. Bell Labs, Lucent Technology. Morgan Kaufmann Publishers. USA, 2001.

[WBE01] WBEM. *Web-Based Enterprise Management (WBEM) Initiative.* Distributed Management Task Force, Inc. DMTF 2001. URL: http://www.dmtf.org/standards/standard_wbem.php

[WES01] Westerinen, A. et. al. *Terminology for Policy Based Management*. **RFC3198**, Nov. 2001.

[WES00] Westerinen, A., Strassner, J. *CIM Core Model White Paper: Common Information Model (CIM) Core Model, Version 2.4.* DSP111. Distributed Management Task Force, Aug. 2000.

[WRO97a] Wroclawski, J. *RSVP with INTSERV*, **RFC 2210**, Sep. 1997.

[WRO97b] Wroclawski, J. *Specification of the Controlled Load Quality of Service*, **RFC 2211**, Sep. 1997.

[YAV00] Yavatkar, R., Pendarakis, D.; Guerin, R. *A Framework for Policy-Based Admission Control*, **RFC2753**, Jan. 2000.

[ZHA93] Zhang, L.; Deering, S.; Estrin, D.; Shenker, S.; D. Zappala. *RSVP: A New Resource ReSerVation Protocol*. IEEE Network, Sep 1993.

Anexo A - Descrição de uma Política

Exemplo completo das descrições *Request*, *Policy* e *Response* da Política XACML apresentadas no capítulo 4.

Descrição de um XACML *Request context*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
[05] http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
[06] <Subject>
[07]   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[08]         DataType="http://www.w3.org/2001/XMLSchema#string">
[09]         <AttributeValue>Ben-Hur</AttributeValue>
[10]    </Attribute>
[11] </Subject>
[12] <Resource>
[13]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
[14]         DataType="http://www.w3.org/2001/XMLSchema#string">
[15]         <AttributeValue>MultimidiaServer</AttributeValue>
[16]    </Attribute>
[17] </Resource>
[18] <Action>
[19]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action-id:ServerAction"
[20]         DataType="http://www.w3.org/2001/XMLSchema#string">
[21]         <AttributeValue>login</AttributeValue>
[22]    </Attribute>
[23]  </Action>
[24] </Request>
```

Descrição de um XACML *Policy*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
[05] http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-policy-01.xsd"
[06] PolicyId="urn:oasis:names:tc:xacml:1.0:example:PolicyServer01"
[07] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm: first-applicable">
[08] <!-- Esta política se aplica somente para requisições no servidor chamado MultimidiaServer -->
[09] <Target>
[10]    <Subjects>
[11]         <AnySubject/>
[12]    </Subjects>
[13]    <Resources>
[14]         <Resource>
```

Continuação da Policy

```
[15]            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[16]                <AttributeValue
[17]                    DataType="http://www.w3.org/2001/XMLSchema#string">MultimidiaServer
[18]                </AttributeValue>
[19]                <ResourceAttributeDesignator
[20]                    DataType="http://www.w3.org/2001/XMLSchema#string"
[21]                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
[22]            </ResourceMatch>
[23]        </Resource>
[24]    </Resources>
[25]    <Actions>
[26]        <AnyAction/>
[27]    </Actions>
[28] </Target>
[29] <Rule RuleId="urn:oasis:names:tc:xacml:1.0:LoginRule" Effect="Permit">
[30] <!-- Regra para ação de login especificada no XACML request context -->
[31] <Target>
[32]    <Subjects>
[33]        <Subject>
[34]            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[35]                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Ben-Hur
[36]                </AttributeValue>
[37]                <SubjectAttributeDesignator
[38]                    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[39]                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
[40]            </SubjectMatch >
[41]        </Subject>
[42]    </Subjects>
[43]    <Resources>
[44]        <AnyResource/>
[45]    </Resources>
[46]    <Actions>
[47]        <Action>
[48]            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[49]                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login
[50]                </AttributeValue>
[51]                <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
[52]                    AttributeId="urn:oasis:names:tc:xacml:1.0:action-id:ServerAction"/>
[53]            </ActionMatch>
[54]        </Action>
[55]    </Actions>
[56] </Target>
[57] <!--Permite executar logins compreendido no horário de 09:00 às 17:00 horas -->
[58] <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[59]    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"> f(1)
[60]        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only"> f(2)
[61]            <EnvironmentAttributeDesignator
[62]            DataType="http://www.w3.org/2001/XMLSchema#time"
[63]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
[64]        </Apply>
[65]        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00
[66]        </AttributeValue>
[67]    </Apply>
[68]    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"> f(3)
```

Continuação da Policy

```
[69]        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">  f(4)
[70]            <EnvironmentAttributeDesignator
[71]            DataType="http://www.w3.org/2001/XMLSchema#time"
[72]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
[73]        </Apply>
[74]        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00
[75]        </AttributeValue>
[76]    </Apply>
[77]  </Condition>
[78]  </Rule>
[79]  <Rule RuleId="urn:oasis:names:tc:xacml:1.0:FinalRule" Effect="Deny"/>
[80] </Policy>
```

$f(n)$ = *representa uma função para exemplificar a aplicação da <condition>*

Resposta de um XACML *Response context*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Response xmlns="urn:oasis:names:tc:xacml:1.0:context"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
[05] http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
[07]    <Decision>Permit</Decision>
[08]        <Status>
[09]            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
[10]        </Status>
[11]    </Result>
[12] </Response>
```

Anexo B - Exemplo de política de QoS com XACML

Seguem as descrições completas das mensagens XACML apresentadas no exemplo de política de QoS em um cenário típico, descritas no capítulo 6.

Descrição do XACML *Request context*

```
[01] <?xml version=”1.0” encoding="UTF-8"?>
[02] <Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
[03] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[04] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
[05] D:\XACML\qos\xacml-word\cs-xacml-schema-context-01.xsd">
[06] <!-- context URL:  D:\XACML\qos\xacml-word\ -->
[07]     <Subject>
[08]         <Attribute
[09]      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:receiver"
[10]      DataType="http://www.w3.org/2001/XMLSchema#string">
[11]             <AttributeValue>192.168.200.192</AttributeValue>
[12]         </Attribute>
[13]         <Attribute
[14]      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:sender"
[15]      DataType="http://www.w3.org/2001/XMLSchema#string">
[16]             <AttributeValue>192.168.200.1</AttributeValue>
[17]         </Attribute>
[18]     </Subject>
[19]     <Resource>
[20]         <Attribute
[21]      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
[22]      DataType="http://www.w3.org/2001/XMLSchema#string">
[23]             <AttributeValue>MultimidiaServer</AttributeValue>
[24]         </Attribute>
[25]     </Resource>
[26]     <Action>
[27]         <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id:ServerAction"
[28]      DataType="http://www.w3.org/2001/XMLSchema#string">
[29]                 <AttributeValue>getResourceQoS</AttributeValue>
[30]         </Attribute>
[31]     </Action>
[32] </Request>
```

Descrição do XACML *Policy*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Policy PolicyId="urn:oasis:names:tc:xacml:1.0:policy:MultimidiaPolicy"
[03] xmlns="urn:oasis:names:tc:xacml:1.0:policy"
[04] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[05] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy  cs-xacml-schema-policy-01-rsvp.xsd"
[06] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">
[07] <!-- Esta política se aplica somente para requisições no servidor chamado MultimidiaServer -->
[08]   <Target>
[09]     <Subjects>
[10]         <AnySubject/>
[11]     </Subjects>
[12]     <Resources>
[13]       <Resource>
[14]         <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[15]           <AttributeValue
[16]             DataType="http://www.w3.org/2001/XMLSchema#string">MultimidiaServer
[17]             </AttributeValue>
[18]           <ResourceAttributeDesignator
[19]             DataType="http://www.w3.org/2001/XMLSchema#string"
[20]             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
[21]         </ResourceMatch>
[22]         <ResourceRsvp
[23]             AttributeId="urn:oasis:names:tc:xacml:1.0:resource-id:multimidiaserver"
[24]             RsvpClass="G711">
[25]               <TspecBucketRate_r>9250</TspecBucketRate_r>
[26]               <TspecBucketSize_b>680</TspecBucketSize_b>
[27]               <TspecPeakRate_p>13875</TspecPeakRate_p>
[28]               <TspecMinPoliceUnit_m>340</TspecMinPoliceUnit_m>
[29]               <TspecMaxPacketSize_M>340</TspecMaxPacketSize_M>
[30]               <RsvpService>Guaranteed</RsvpService>
[31]          </ResourceRsvp>
[32]         <!-- <AnyResourceRSVP/> -->
[33]       </Resource>
[34]     </Resources>
[35]     <Actions>
[36]         <AnyAction/>
[37]     </Actions>
[38]   </Target>
[39] <!-- Regra para recursos de QoS no SENDER MultimidiaServer e com restrições de horário -->
[40]   <Rule RuleId="urn:oasis:names:tc:xacml:1.0:ResourceQoS" Effect="Permit">
[41]     <Target>
[42]         <Subjects>
[43]           <Subject>
[44]           <!-- Identificador do subject RECEIVER: urn:oasis:names:tc:xacml:1.0:subject:
[45]               authn-locality:ip-address:receiver -->
[46]               <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[47]                   <AttributeValue
[48]                   DataType="http://www.w3.org/2001/XMLSchema#string">192.168.200.192
[49]                   </AttributeValue>
[50]               <SubjectAttributeDesignator
[51]                   AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:
[52]                                ip-address:receiver"
[53]                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
[54]               </SubjectMatch>
```

Continuação da Policy

```
[55]
[56]    <!-- Identificador do subject SENDER: urn:oasis:names:tc:xacml:1.0:subject:
[57]  authn-locality:ip-address:sender -->
[58]              <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[59]                  <AttributeValue
[60]                      DataType="http://www.w3.org/2001/XMLSchema#string">192.168.200.1
[61]                  </AttributeValue>
[62]                  <SubjectAttributeDesignator
[63]                      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:
[64]                                  ip-address:sender"
[65]                      DataType="http://www.w3.org/2001/XM LSchema#string"/>
[66]                  </SubjectMatch>
[67]              </Subject>
[68]              <!-- <AnySubject/> -->
[69]          </Subjects>
[70]          <Resources>
[71]              <AnyResource/>
[72]          </Resources>
[73]          <Actions>
[74]              <Action>
[75]                  <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[76]                      <AttributeValue
[77]                      DataType="http://www.w3.org/2001/XMLSchema#string">getResourceQoS
[78]                      </AttributeValue>
[79]                      <ActionAttributeDesignator
[80]                          DataType="http://www.w3.org/2001/XMLSchema#string"
[81]                          AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id:ServerAction"/>
[82]                  </ActionMatch>
[83]              </Action>
[84]          </Actions>
[85]      </Target>
[86]      <!--Permite serviços de QoS compreendido no horário de 09:00 às 17:00 horas -->
[87]      <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[88]          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
[89]              <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
[90]                  <EnvironmentAttributeDesignator
[91]                      DataType="http://www.w3.org/2001/XMLSchema#time"
[92]                      AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
[93]              </Apply>
[94]              <AttributeValue
[95]                  DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00
[96]                  </AttributeValue>
[97]          </Apply>
[98]          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
[99]              <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
[100]                 <EnvironmentAttributeDesignator
[101]                     DataType="http://www.w3.org/2001/XMLSchema#time"
[102]                     AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
[103]             </Apply>
[104]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00
[105]             </AttributeValue>
[106]         </Apply>
[107]     </Condition>
[108]  </Rule>
```

Continuação da Policy

```
[109] <!-- Caso o recurso não seja possível de avaliação, retorna um Deny ao PEP (evita resposta
[110]      NotApplicable) -->
[111] <Rule RuleId="urn:oasis:names:tc:xacml:1.0:FinalQoSRule" Effect="Deny"/>
[122]   <Obligations>
[123]     <Obligation ObligationId=" urn:oasis:names:tc:xacml:1.0:obligation:MultimidiaServer"
[124]           FulfillOn="Permit">
[125]     <AttributeAssignment AttributeId="TspecBucketRate_r"
[126]         DataType="http://www.w3.org/2001/XMLSchema#double">
[127]         <AttributeSelectorRsvp
[128]             PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecBucketRate_r"
[129]             DataType="http://www.w3.org/2001/XMLSchema#double">
[130]         </AttributeSelectorRsvp>
[131]     </AttributeAssignment>
[132]     <AttributeAssignment AttributeId="TspecBucketSize_b"
[133]         DataType="http://www.w3.org/2001/XMLSchema#double">
[134]         <AttributeSelectorRsvp
[135]             PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecBucketSize_b"
[136]             DataType="http://www.w3.org/2001/XMLSchema#double">
[137]         </AttributeSelectorRsvp>
[138]     </AttributeAssignment>
[139]     <AttributeAssignment AttributeId="TspecPeakRate_p"
[140]         DataType="http://www.w3.org/2001/XMLSchema#double">
[141]         <AttributeSelectorRsvp
[142]             PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecPeakRate_p"
[143]             DataType="http://www.w3.org/2001/XMLSchema#double">
[144]         </AttributeSelectorRsvp>
[145]     </AttributeAssignment>
[146]     <AttributeAssignment AttributeId="TspecMinPoliceUnit_m"
[147]         DataType="http://www.w3.org/2001/XMLSchema#integer">
[148]         <AttributeSelectorRsvp
[149]             PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecMinPoliceUnit_m"
[150]             DataType="http://www.w3.org/2001/XMLSchema#integer">
[151]         </AttributeSelectorRsvp>
[152]     </AttributeAssignment>
[153]     <AttributeAssignment AttributeId="TspecMaxPacketSize_M"
[154]         DataType="http://www.w3.org/2001/XMLSchema#integer">
[155]         <AttributeSelectorRsvp
[156]             PolicyPath="//Policy/Target/Resource/ResourceRsvp/TspecMaxPacketSize_M"
[157]             DataType="http://www.w3.org/2001/XMLSchema#integer">
[158]         </AttributeSelectorRsvp>
[159]     </AttributeAssignment>
[160]     <AttributeAssignment AttributeId="RsvpService"
[161]         DataType="http://www.w3.org/2001/XMLSchema#string">
[162]         <AttributeSelectorRsvp
[163]             PolicyPath="//Policy/Target/Resource/ResourceRsvp/RsvpService"
[164]             DataType="http://www.w3.org/2001/XMLSchema#string">
[165]         </AttributeSelectorRsvp>
[166]     </AttributeAssignment>
[167]   </Obligation>
[168]  </Obligations>
[169] </Policy>
[170]
```

Descrição do XACML *Request* ***context***

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <Response
[03] xmlns="urn:oasis:names:tc:xacml:1.0:context"
[04] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[05] xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context   cs-xacml-schema-context-01.xsd">
[06]   <Result>
[07]     <Decision>Permit</Decision>
[08]     <Status>
[09]         <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
[10]     </Status>
[11]     <Obligations  xmlns="urn:oasis:names:tc:xacml:1.0:policy">
[12]         <Obligation ObligationId="urn:oasis:names:tc:xacml:1.0:obligation:MultimidiaServer"
[13]         FulfillOn="Permit">
[14]           <AttributeAssignment AttributeId="TspecBucketRate_r"
[15]             DataType="http://www.w3.org/2001/XMLSchema#double">9250</AttributeAssignment>
[16]           <AttributeAssignment AttributeId="TspecBucketSize_b"
[17]             DataType="http://www.w3.org/2001/XMLSchema#double">680</AttributeAssignment>
[18]           <AttributeAssignment AttributeId="TspecPeakRate_p"
[19]             DataType="http://www.w3.org/2001/XMLSchema#double">13875</AttributeAssignment>
[20]           <AttributeAssignment AttributeId="TspecMinPoliceUnit_m"
[21]             DataType="http://www.w3.org/2001/XMLSchema#integer">340</AttributeAssignment>
[22]           <AttributeAssignment AttributeId="TspecMaxPacketSize_M"
[23]             DataType="http://www.w3.org/2001/XMLSchema#integer">340</AttributeAssignment>
[24]           <AttributeAssignment AttributeId="RsvpService"
[25]             DataType="http://www.w3.org/2001/XMLSchema#string">Guaranteed
[26]             </AttributeAssignment>
[27]        </Obligation>
[28]     </Obligations>
[29]   </Result>
[30] </Response>
```

Anexo C – Especificação de QoS RSVP com XACML

Segue a descrição de uma especificação sugerida para uso na declaração de políticas QoS RSVP em XACML, utilizadas no estudo de caso do capítulo6.

Definições para uma XACML *request context* obter os parâmetros de QoS.

<Subject>: identifica um ou mais elementos <**Subject**> em um XACML *request context*.

1. AttributeId: Identificador para um RECEIVER:

   urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:**receiver**

2. AttributeId: Identificador para um SENDER:

   urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:**sender**

   Observação: Como alternativa, pode ser definido um nome DNS, alterando a sintaxe para: urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name:**{sender/receiver}**

3. <AttributeValue>: Valor do atributo sugerido como os endereços IPs do SENDER e RECEIVER, seguindo o formato: **X.Y.Z.W**

   Exemplo: endereço IP ‘192.168.0.100’ {sem as ‘aspas’}.

<Resource>: identifica o elemento <Resource> em um XACML *request* ***context***, neste caso o SENDER (PEP).

1. AttributeId: Identificador do *Resource*, ou seja, o XACML ***policy*** utilizará este identificador para validação de um XACML *request*. Identifica o SENDER (PEP) na ***policy***.

   urn:oasis:names:tc:xacml:1.0:resource:resource-id

2. <AttributeValue>: Valor do atributo sugerido, NetBIOS name do SENDER (PEP).

   hostnameSENDER

   Exemplo:{ MultimidiaServer }

<**Action**>: identifica que ação é desejada para o elemento <Resource> em um XACML *request* ***context***, neste caso o SENDER (PEP).

1. AttributeId: Identificador da ação desejada para um SENDER:

   urn:oasis:names:tc:xacml:1.0:action:action-id:***hostNameSENDER***

   Exemplo:{urn:oasis:names:tc:xacml:1.0:action:action-id:**MultimidiaServer**}

Segue a estrutura básica de um **XACML** *Request* ***context***:

```
<Request ...>
   <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:receiver"
          DataType="#string">
          <AttributeValue>IP_Address_RECEIVER</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:sender"
          DataType="#string">
          <AttributeValue>IP_Address_SENDER</AttributeValue>
    </Attribute>
   </Subject>
   <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="http://www.w3.org/2001/XMLSchema#string">
          <AttributeValue>hostNameSENDER</AttributeValue>
    </Attribute>
   </Resource>
   <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id:hostNameSENDER"
          DataType="#string">
          <AttributeValue>getResourceQoS</AttributeValue>
    </Attribute>
   </Action>
</Request>
```

Definições para uma XACML *policy*

<**Resource**>: identifica se existe o elemento <Resource> em um XACML *request* ***context***, neste caso o SENDER (PEP).

1. MatchId: Define a função para comparar o valor do atributo da ***policy*** com o valor definido no XACML *request*.

   Função: “function:string-equal”

2. AttributeId: Este identificador representa a configuração de um SENDER e utilizado para avaliação de acesso e recursos de um XACML *request* com as definições da ***policy***. Caso no XACML *request* ***context*** não seja encontrada a referência, significa que não existe definições na política que atenda o requerimento solicitado.

   urn:oasis:names:tc:xacml:1.0:resource:resource-id

2.1 <AttributeValue>: Valor do atributo sugerido, NetBIOS name do SENDER (PEP).

hostnameSENDER

2.2 <ResourceRsvp>: Define os parâmetros Tspec de um SENDER.

⇒ AttributeId: Identificador de um SENDER com as configurações de QoS. Segue o mesmo padrão do identificador. Sugerido utilizar o mesmo valor definido no <AttributeValue> do elemento <Resource>, neste caso, *hostnameSENDER.*

Exemplo: {"urn:oasis:names:tc:xacml:1.0:resource-id:MultimidiaServer"}

<Rule>: regra que será aplicada em uma ***policy***

<Subject>: identifica um ou mais elementos <Subject> que devem estar descritos na política para utilização de controle de acessso em avaliações de um XACML *request context*.

1. <SubjectMatch> define a função a ser utilização em um XACML *request*. A função definida para fins de comparação é "function:string-equal". As definições de AttributeId sugeridas, definem o modo de identificação de um <Subject> e orientam a criação de uma ***policy*** e um XML *request*.
2. AttributeId: Identificador para um RECEIVER:
   urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:**receiver**
3. AttributeId: Identificador para um SENDER:
   urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:**sender**

<Action>: definem ações permitidas a um determinado <Resource> na ***policy***, neste verifica que se trata de uma solicitação de recursos QoS para um <Resource> em um XML *request*.

1. <AttributeValue>: Valor do atributo sugerido para determinação de uma regra para avaliação de controle de acesso para obter recursos de QoS. Deverá ser utilizado o mesmo tipo de solicitação em um XML *request*.
   GetResourceQoS

<Condition>: condições definidas na ***policy*** utilizadas em uma ***authorization decision***.

Observação: No exemplo citado, verifica restrições quanto a horário de acesso permitido para recursos de QoS.

<**Obligations**>: obrigações definidas na ***policy*** e que devem ser obedecidas pelo PEP.

Nesta proposta, em caso de uma ***authorization decision*** permitida deve retornar os valores do tráfego de dados Tspec e Classe de Serviço definida na ***policy*** para o PEP invocar o RSVP *daemon*, informando valores do ***Tspec*** e a classe {Guaranteed/Controlled-load/Null} para definição do objeto ***Adspec***.

⇒ A presente versão do XACML não prevê processamento no PDP para retornar valores, e por isto deve ser estendido um elemento que execute uma referência ao <ResourceRsvp> possibilitando o retorno dos dados sem necessidade de configuração manual de parâmetros informados na Obligatios, uma opção possível atualmente.

Segue uma estrutura básica de ***Policy*** para controle de acesso e descrição de tráfego de dados ao SENDER:

```
<Policy PolicyId="urn:oasis:names:tc:xacml:1.0:policy:NomedaPolicy"
   <Target>
      <Subjects>
         <AnySubject/>
      </Subjects>
      <Resources>
         <Resource>
            <ResourceMatch MatchId="....:function: string-equal">
               <AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string">hostName
               </AttributeValue>
            <ResourceAttributeDesignator
               DataType="... #string"
               AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
            </ResourceMatch>
            <ResourceRsvp
               AttributeId="urn:oasis:names:tc:xacml:1.0:resource-id:hostName"
               RsvpClass="DefinidaNoSchema - ex: G711">
               <TspecBucketRate_r>9250</TspecBucketRate_r>
               <TspecBucketSize_b>680</TspecBucketSize_b>
               <TspecPeakRate_p>13875</TspecPeakRate_p>
               <TspecMinPoliceUnit_m>340</TspecMinPoliceUnit_m>
               <TspecMaxPacketSize_M>340</TspecMaxPacketSize_M>
               <RsvpService>Guaranteed</RsvpService>
            </ResourceRsvp>
         </Resource>
      </Resources>
      <Actions>
         <AnyAction/>
      </Actions>
   </Target>
```

```
    <!-- Regra para recursos de QoS no SENDER MultimidiaServer e com restrições de horário -->
    <Rule RuleId="urn:oasis:names:tc:xacml:1.0:ResourceQoS" Effect="Permit">
        <Target>
            <Subjects>
                <Subject>
                <!-- Identificador do subject SENDER/RECEIVER: Deve existir duas definições!
                urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:receiver -->
                <SubjectMatch MatchId="......:function:string-equal">
                    <AttributeValue
                    DataType="…. #string">192.168.200.192</AttributeValue>
                    <SubjectAttributeDesignator
                    AttributeId=
                    "urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address:receiver/sender"
                    DataType="......#string"/>
                </SubjectMatch>
            </Subjects>
            <Resources>
                <AnyResource/>
            </Resources>
            <Actions>
                <Action>
                    <ActionMatch MatchId="....:function:string-equal">
                        <AttributeValue
                        DataType="....#string">getResourceQoS</AttributeValue>
                        <ActionAttributeDesignator
                            DataType=".........#string"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id:hostName"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
        <!--Permite serviços de QoS compreendido no horário de 09:00 às 17:00 horas -->
        <Condition>
                <ConditionsToCheck...>
        </Condition>
    </Rule>
    <Obligations>
        <Obligation ObligationId="urn:oasis:names:tc:xacml:1.0:obligation:hostName"
            FulfillOn="Permit">
        <!-- Dados enviados após o processamento do PDP. Exemplo ilustrativo  -->
            <AttributeAssignment AttributeId="TspecBucketRate_r"
                DataType=".....#double">9250</AttributeAssignment>
            <AttributeAssignment AttributeId="TspecBucketSize_b"
                DataType=".....#double">680</AttributeAssignment>
            <AttributeAssignment AttributeId="TspecPeakRate_p"
                DataType="......#double">13875</AttributeAssignment>
            <AttributeAssignment AttributeId="TspecMinPoliceUnit_m"
                DataType="......#integer">340</AttributeAssignment>
            <AttributeAssignment AttributeId="TspecMaxPacketSize_M"
                DataType="........#integer">340</AttributeAssignment>
            <AttributeAssignment AttributeId="RsvpService"
                DataType="......#string">Guaranteed</AttributeAssignment>
            <AttributeAssignment AttributeId="ServiceQoS"
                DataType="........#string">G711</AttributeAssignment>
    </Obligation>
  </Obligations>
</Policy>
```

Anexo D – Extensão do XACML *policy schema*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <!-- -->
  <xs:element name="PolicySet" type="xacml:PolicySetType"/>
  <xs:complexType name="PolicySetType">
    <xs:sequence>
        <xs:element ref="xacml:Description" minOccurs="0"/>
        <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
        <xs:element ref="xacml:Target"/>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xacml:PolicySet"/>
            <xs:element ref="xacml:Policy"/>
            <xs:element ref="xacml:PolicySetIdReference"/>
            <xs:element ref="xacml:PolicyIdReference"/>
        </xs:choice>
        <xs:element ref="xacml:Obligations" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
    <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="PolicySetIdReference" type="xs:anyURI"/>
  <xs:element name="PolicyIdReference" type="xs:anyURI"/>
  <!-- -->
  <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
  <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
  <xs:complexType name="DefaultsType">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="xacml:XPathVersion"/>
        </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="XPathVersion" type="xs:anyURI"/>
  <!-- -->
  <xs:element name="Policy" type="xacml:PolicyType"/>
  <xs:complexType name="PolicyType">
    <xs:sequence>
        <xs:element ref="xacml:Description" minOccurs="0"/>
        <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
        <xs:element ref="xacml:Target"/>
        <xs:element ref="xacml:Rule" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="xacml:Obligations" minOccurs="0"/>
    </xs:sequence>
```

```
  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="Description" type="xs:string"/>
<!-- -->
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
  <xs:sequence>
      <xs:element ref="xacml:Description" minOccurs="0"/>
      <xs:element ref="xacml:Target" minOccurs="0"/>
      <xs:element ref="xacml:Condition" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="RuleId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
</xs:complexType>
<!-- -->
<xs:simpleType name="EffectType">
  <xs:restriction base="xs:string">
      <xs:enumeration value="Permit"/>
      <xs:enumeration value="Deny"/>
  </xs:restriction>
</xs:simpleType>
<!-- -->
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence>
      <xs:element ref="xacml:Subjects"/>
      <xs:element ref="xacml:Resources"/>
      <xs:element ref="xacml:Actions"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
  <xs:choice>
      <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
      <xs:element ref="xacml:AnySubject"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
      <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="AnySubject"/>
<!-- -->
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
  <xs:choice>
      <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
      <xs:element ref="xacml:AnyResource"/>
  </xs:choice>
</xs:complexType>
<!-- -->
```

```
<xs:element name="AnyResource"/>
<!-- -->
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
      <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
         <xs:element ref="xacml:ResourceRsvp"/>
      </xs:choice>
      <!--  Extended ResourceRsvp  -->
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
  <xs:choice>
      <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
      <xs:element ref="xacml:AnyAction"/>
  </xs:choice>
</xs:complexType>
<!-- -->
<xs:element name="AnyAction"/>
<!-- -->
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
      <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
      <xs:element ref="xacml:AttributeValue"/>
      <xs:choice>
         <xs:element ref="xacml:SubjectAttributeDesignator"/>
         <xs:element ref="xacml:AttributeSelector"/>
      </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
      <xs:element ref="xacml:AttributeValue"/>
      <xs:choice>
         <xs:element ref="xacml:ResourceAttributeDesignator"/>
         <xs:element ref="xacml:AttributeSelector"/>
      </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
      <xs:element ref="xacml:AttributeValue"/>
```

```
      <xs:choice>
        <xs:element ref="xacml:ActionAttributeDesignator"/>
        <xs:element ref="xacml:AttributeSelector"/>
      </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"/>
<xs:complexType name="AttributeSelectorType">
  <xs:attribute name="RequestContextPath" type="xs:string" use="required"/>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
</xs:complexType>
<!-- -->
<xs:element name="ResourceAttributeDesignator" type="xacml:AttributeDesignatorType"/>
<xs:element name="ActionAttributeDesignator" type="xacml:AttributeDesignatorType"/>
<xs:element name="EnvironmentAttributeDesignator" type="xacml:AttributeDesignatorType"/>
<!-- -->
<xs:complexType name="AttributeDesignatorType">
  <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="Issuer" type="xs:string" use="optional"/>
  <xs:attribute name="MustBePresent" type="xs:boolean" use="optional" default="false"/>
</xs:complexType>
<!-- -->
<xs:element name="SubjectAttributeDesignator" type="xacml:SubjectAttributeDesignatorType"/>
<xs:complexType name="SubjectAttributeDesignatorType">
  <xs:complexContent>
      <xs:extension base="xacml:AttributeDesignatorType">
        <xs:attribute name="SubjectCategory" type="xs:anyURI" use="optional"
        default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
      </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="AttributeValue" type="xacml:AttributeValueType"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
<!-- -->
<xs:element name="Function" type="xacml:FunctionType"/>
<xs:complexType name="FunctionType">
  <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="Apply" type="xacml:ApplyType"/>
<xs:element name="Condition" type="xacml:ApplyType"/>
<!-- -->
<xs:complexType name="ApplyType">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xacml:Apply"/>
      <xs:element ref="xacml:Function"/>
```

```
        <xs:element ref="xacml:AttributeValue"/>
        <xs:element ref="xacml:SubjectAttributeDesignator"/>
        <xs:element ref="xacml:ResourceAttributeDesignator"/>
        <xs:element ref="xacml:ActionAttributeDesignator"/>
        <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
        <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
    <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
    <!-- Legal types for the first and subsequent operands are defined in the accompanying table -->
</xs:complexType>
<!-- -->
<xs:element name="Obligations" type="xacml:ObligationsType"/>
<xs:complexType name="ObligationsType">
    <xs:sequence>
        <xs:choice>
            <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Obligation" type="xacml:ObligationType"/>
<xs:complexType name="ObligationType">
    <xs:sequence>
        <xs:element ref="xacml:AttributeAssignment" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
    <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
<xs:complexType name="AttributeAssignmentType" mixed="true">
    <xs:complexContent mixed="true">
        <xs:extension base="xacml:AttributeValueType">
            <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- -->
<!-- Extended by Toktar -->
<!-- -->
<xs:element name="AttributeSelectorRsvp" type="xacml:AttributeSelectorRsvpType"/>
<xs:complexType name="AttributeSelectorRsvpType">
    <xs:attribute name="PolicyPath" type="xs:string" use="required"/>
    <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
</xs:complexType>
<!--  -->
<xs:element name="ResourceRsvp" type="xacml:ResourceRsvpType"/>
<xs:complexType name="ResourceRsvpType">
    <xs:sequence>
        <xs:element ref="xacml:TspecBucketRate_r"/>
        <xs:element ref="xacml:TspecBucketSize_b"/>
        <xs:element ref="xacml:TspecPeakRate_p"/>
        <xs:element ref="xacml:TspecMinPoliceUnit_m"/>
        <xs:element ref="xacml:TspecMaxPacketSize_M"/>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xacml:RsvpService"/>
            <xs:element ref="xacml:RsvpStyle"/>
            <!--   <xs:element ref="xacml:RsvpClass"/> -->
```

```
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
    <xs:attribute name="RsvpClass" type="xacml:RsvpClassType" use="required"/>
</xs:complexType>
<!-- -->
<!-- Class Qos RSVP -->
<xs:simpleType name="RsvpClassType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="G711"/>
        <xs:enumeration value="G729"/>
        <xs:enumeration value="H263CIF"/>
        <xs:enumeration value="H261QCIF"/>
    </xs:restriction>
</xs:simpleType>
<!-- -->
<!-- Service Qos RSVP -->
<xs:element name="RsvpService">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Null"/>
            <xs:enumeration value="Guaranteed"/>
            <xs:enumeration value="Controlled-load"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<!-- -->
<!-- Style Qos RSVP -->
<xs:element name="RsvpStyle">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="SE"/>
            <xs:enumeration value="WF"/>
            <xs:enumeration value="FF"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<!--  RFC 2212 -->
<!-- TSPEC QoS RSVP by Toktar-->
<!-- measured in bits/sec   float [1 - 40TB] -->
<!-- base="xs:float" -->
<xs:element name="TspecBucketRate_r">
    <xs:simpleType>
        <xs:restriction base="xs:double">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="40000000000000"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<!-- -->
<!-- measured in bits/sec   float [1 - 250GB]-->
<!-- base="xs:float" -->
<xs:element name="TspecBucketSize_b">
    <xs:simpleType>
        <xs:restriction base="xs:double">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="250000000000"/>
        </xs:restriction>
```

```
    </xs:simpleType>
  </xs:element>
  <!-- -->
  <!-- measured in bytes  float [1 - 40TB] -->
  <!-- base="xs:float" -->
  <xs:element name="TspecPeakRate_p">
    <xs:simpleType>
        <xs:restriction base="xs:double">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="40000000000000"/>
        </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <!-- -->
  <!-- measured in microseconds unsigned integer [4294967296] -->
  <!-- base="xs:unsignedInt" -->
  <xs:element name="TspecMinPoliceUnit_m">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="4294967295"/>
        </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <!-- -->
  <!-- measured in bytes  unsigned integer [4294967296] -->
  <!-- base="xs:unsignedInt" -->
  <xs:element name="TspecMaxPacketSize_M">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="4294967295"/>
        </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <!-- -->
</xs:schema>
```

Anexo E - Cenário Completo