Uma Arquitetura baseada em XML para Políticas RSVP

Emir Toktar, Edgard Jamhour, Altair Olivo Santin

Pontifícia Universidade Católica do Paraná, PUCPR, PPGIA, Rua Imaculada Conceição 1155, CEP 80215-901, Curitiba, Brasil

{toktar, jamhour, santin}@ppgia.pucpr.br, emir.toktar@computer.org

Abstract. *This work proposes a XML-based framework for distributing and enforcing RSVP access control policies, for RSVP-aware application servers. Policies RSVP are represented by extending XACML, a general purpose access control language proposed by OASIS. The extensions had been required for representation and transport of information of politics for control of access RSVP. Using appropriate XACML extensions, this can be used to define complex access control policies for specific domains, such as RSVP.*

Resumo. *Este trabalho apresenta uma arquitetura baseada em XML para distribuir e aplicar políticas de controle de acesso para servidores de aplicações RSVP. As políticas RSVP são representadas estendendo o modelo XACML, uma linguagem genérica para controle de acesso proposto pelo OASIS. As extensões foram requeridas para representação e transporte de informações de política para controle de acesso RSVP. Usando extensões XACML apropriadas, este pode ser utilizado para definir políticas complexas.*

1. Introdução

A gerência baseada em políticas está se tornando uma abordagem importante para diferentes áreas de gerência de redes IP. Muitos trabalhos recentes do IETF estão sendo direcionados para criar um modelo padronizado para representar políticas em diferentes áreas da gerência de redes. A base para este modelo é o PCIM (*Policy Core Information Model*), definido pela RFC 3060 [5]. O PCIM é um modelo de informação orientado a objetos, independente de plataforma, e não possui elementos suficientes para descrever políticas para áreas específicas de gerência de redes. O modelo define uma estratégia padronizada para representação de políticas de redes na forma de uma agregação de regras, formadas por condições e ações. Para endereçar áreas específicas, o PCIM precisa ser estendido. O próprio IETF já apresentou extensões do PCIM para representação de políticas IPsec e QoS (*Quality of Service*) [10]. Outros trabalhos também exploraram extensões do PCIM para a área de controle de acesso [6].

Os modelos propostos pelo IETF, todavia, não são os únicos a abordar a gerência de redes baseada em políticas (*Policy Based Network Managemen*t- PBNM). O OASIS (*Organization for the Advancement of Structured Information Standards*) propôs uma linguagem para representar políticas de controle de acesso, de propósito geral, denominada XACML (*eXtensible Access Control Markup Language*). Existem várias diferenças nas abordagens do PCIM e do XACML. Enquanto o PCIM é a base para criar políticas para qualquer área de gerência, o XACML é dedicado ao controle de acesso. Devido ao seu propósito genérico, a implementação de modelos de políticas

baseados em PCIM é uma tarefa bastante complexa. O XACML, por outro lado, é muito mais simples de ser interpretado e implementado.

Baseado nessa argumentação, este trabalho teve por objetivo avaliar o uso do XACML para modelar e distribuir políticas de controle de acesso para aplicações RSVP (*Resource Reservation Protocol*) em servidores. Como o RSVP é uma aplicação de domínio específico, este não é suportado pelo padrão XACML. Por esta razão, este trabalho define as extensões XACML requeridas para representar e transportar informações de políticas de controle de acesso RSVP. Este trabalho permitiu avaliar as vantagens e desvantagens relativas da abordagem do XACML em relação ao modelo PCIM com relação à implementação e distribuição de políticas. Estabelecendo paralelos com a abordagem baseada em PCIM, este trabalho define extensões futuras requeridas para estender esta proposta para outros elementos de redes, como roteadores.

Este artigo está estruturado da seguinte forma: A seção 2 apresenta uma revisão dos principais aspectos relacionados ao controle de políticas em RSVP. A seção 3 apresenta uma análise dos modelos para descrever políticas de controle de acesso para RSVP e estratégias para desenvolvimento de sistemas para distribuição e execução dessas políticas. A seção 4 apresenta uma breve revisão do modelo XACML. A seção 5 descreve como o XACML pode ser usado para descrever políticas RSVP e apresenta a extensão para adaptá-lo ao RSVP. A seção 6 descreve como implementar a estrutura para distribuir e efetivar as políticas RSVP descritas em XACML. Finalmente, a conclusão resume os principais aspectos deste estudo e aponta para trabalhos futuros.

2. Controle de Política em RSVP

Esta seção apresenta uma breve revisão do protocolo RSVP, definindo o conceito "controle de política" e apresentando os termos importantes que serão utilizados nas próximas seções.

O RSVP é um protocolo de sinalização utilizado na estratégia de QoS para redes IP denominada "serviços integrados" [2]. Um *receptor* que deseja requisitar um QoS específico, para uma aplicação em particular, utiliza a sinalização do RSVP para negociar a reserva de recursos com a rede. A sinalização do RSVP é composta por um conjunto de mensagens padronizadas, sendo que as duas principais mensagens PATH e RESV. A negociação de QoS sempre é iniciada pelo *emissor* enviando a mensagem PATH ao *receptor*. A mensagem PATH define os parâmetros de QoS que o *receptor* deve solicitar à rede a fim de atender os requisitos da aplicação e também define o caminho que as demais mensagens RSVP e o fluxo de dados irão percorrer entre o *emissor* e o *receptor*. 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.

O *receptor* ao receber a mensagem PATH, inicia o processo de reserva de fluxo enviando a mensagem RESV no caminho inverso do PATH até o *emissor*. A mensagem RESV consiste de um descritor de fluxo (*flow descriptor)*, formado pelos objetos *flowspec* e *filterspec*. O *filterspec* junto com a especificação da sessão, define quais pacotes de dados (fluxo RSVP) deverão se beneficiar da reserva de QoS. A especificação do QoS desejado é definida pelo *flowspec*, através de duas estruturas de dados: *Rspec* (*Reserve Spec*), que indica a classe de serviço desejada e *Tspec* (*Traffic Spec*), que especifica os parâmetros de QoS que será transmitido.

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** (*Admissions Control*). O módulo de **Controle de Admissão** determina se o nó (*host* ou roteador) tem recursos disponíveis suficientes para atender a solicitação de QoS. O mecanismo de **Controle de Política** é responsável por determinar quando um usuário está autorizado a fazer a reserva [2]. 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.

De acordo com a RFC 2205, a aplicação *emissora* deve escolher o tipo de serviço mais apropriado para seus requisitos de transmissão, especificando a descrição do fluxo de dados para o RSVP *daemon* [2]. O *RSVP daemon* após ser invocado pela aplicação, consulta os módulos de decisão locais verificando recursos e autorização e, sendo permitida, inicia a troca de mensagens RSVP e estabelece a reserva de recursos nos elementos de rede envolvidos (*hosts* e *routers*) entre o *emissor* e *receptor*.

A proposta deste trabalho, conforme será detalhada nos próximos tópicos, consiste em definir e implementar um mecanismo para configurar as políticas de controle de acesso em um servidor de aplicação RSVP utilizando XACML. Para reserva de fluxo RSVP, ao invés da aplicação *emissora* invocar diretamente o RSVP *daemon* especificando os parâmetros para reserva do fluxo, a aplicação deverá solicitar ao PEP que invocará o RSVP *daemon*, passando os dados de fluxos autorizados e informados pelo PDP, tornando a aplicação passiva neste processo, determinando a principal mudança conceitual em uma reserva de fluxo RSVP, ilustrada na figura 1. Contudo, esta proposta também fornece informações para definição dos parâmetros *Tspec* e *Rspec* transportada nas mensagens PATH e RESV. Portanto, a política XACML também fornece a informação usada pelo módulo de controle de admissão dos elementos de rede ao longo do caminho entre o emissor e o receptor.

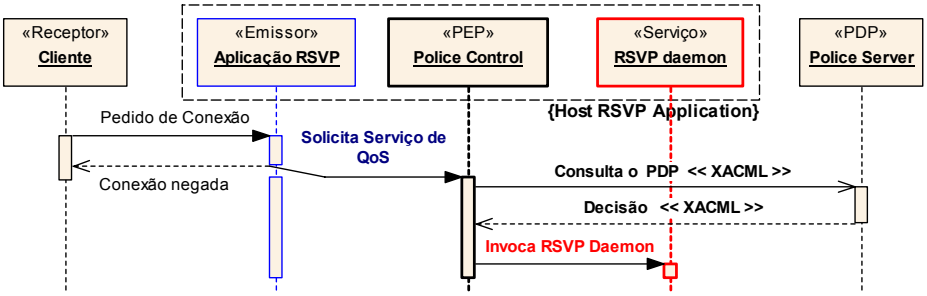


Figura 1. Invocando o RSVP *daemon* em um *host* de aplicação RSVP

3. Modelo para Controle de Políticas RSVP

A estratégia para controle de políticas de acesso a requisições RSVP neste trabalho segue a abordagem de gerência de redes baseada em políticas (PBNM). O conceito de PBNM já é adotado amplamente por organizações que elaboram padrões para Internet, como IETF [14] e o OASIS [7]. Apesar das definições de PBNM poderem divergir de acordo com a organização, os principais conceitos são relativamente universais. A idéia básica da abordagem PBNM é oferecer uma estratégia para configurar políticas em diferentes dispositivos de rede usando partir de uma infra-estrutura comum de gerência, composto por um servidor de políticas, denominado PDP (*Policy Decision Point*) e

vários clientes de políticas, denominados PEP (*Policy Enforcement Point*) [12]. O PDP é a entidade responsável por armazenar e distribuir as políticas para os diversos nós da rede. Um PEP é, usualmente, o componente de um nó da rede responsável por receber e efetivar as políticas enviadas pelo PDP. A abordagem PBNM pode ser aplicada a diversos aspectos da gerência de redes, nesta seção, será aplicada para o controle de políticas de acesso em servidores de aplicações RSVP.

O IETF explora o conceito de PBNM de acordo com duas estratégias, denominadas *outsourcing* e *provisioning*. Na estratégia *outsourcing*, o PEP faz uma consulta ao PDP quando precisa tomar uma decisão. Por exemplo, considerando o problema de acesso em RSVP, o PEP representaria o servidor de aplicação. Ao receber um pedido de reserva de um cliente, o PEP faria uma consulta ao PDP perguntando se o cliente tem ou não direito de pedir a reserva. O PDP então interpretaria as políticas e enviaria uma decisão final dizendo se o acesso é permitido ou negado. Na abordagem *provisioning*, o PEP (incluso no servidor de aplicação), ao ser iniciado, receberia do PDP o conjunto de políticas necessárias para sua tomada de decisão. A informação de políticas recebidas do PDP é armazenada localmente pelo PEP de acordo com a localização definida pelo esquema chamado PIB (*Policy Information Base*). Ao receber um pedido de reserva, o PEP consultaria suas políticas armazenadas localmente e tomaria a decisão. A comunicação entre o PEP e o PDP, neste tipo de abordagem, ocorre somente se houver necessidade de atualizar as políticas armazenadas nos PEPs.

Os trabalhos do IETF definem também um protocolo padronizado para comunicação entre o PEP e o PDP. Este protocolo é denominado COPS (*Common Open Policy Service*). A estrutura básica deste protocolo está descrita na RFC 2748 [1]. O protocolo COPS suporta ambos os modelos de controle de políticas "*outsourcing*" e "*provisioning*". No caso da abordagem *provisioning* foram feitas especificações adicionais ao protocolo COPS e renomeado para COPS-PR, descritas na RFC 3084.

O IETF já apresentou vários trabalhos referentes à utilização da abordagem PBNM para controle de políticas RSVP. Os trabalhos cobrem a definição de um framework para controle de admissão [14] e a utilização de protocolos de QoS com COPS nos modelos *outsourcing* (COPS-RSVP) [4] e *provisioning* (COPS-PR) [3]. A abordagem de *provisioning* ainda está em desenvolvimento, necessitando definições adicionais para sua especificação completa.

A proposta XACML do OASIS também descreve que sua implementação deve seguir a abordagem PDP/PEP. Todavia, o OASIS não faz a distinção entre os modelos *outsourcing* e *provisioning*, nem define um protocolo padronizado para comunicação entre o PEP e o PDP. Analisada a proposta do OASIS para XACML indica que sua utilização se adapta a estratégia ***outsourcing*** (ver a seção 4).

Uma diferença importante nas abordagens do OASIS e do IETF está também na relação à forma de concepção e armazenamento do modelo de políticas. O OASIS propõe o XACML como um modelo de políticas para área de controle de acesso, representado e armazenado na forma de documentos XML. Por outro lado, o IETF define o PCIM como um modelo de políticas genérico, independente da forma como será implementado e armazenado. O PCIM é um modelo abstrato e necessita ser estendido para suprir áreas específicas de gerência, por exemplo, QoS [10]. O IETF aponta estratégias para mapear os modelos de informação em LDAP (*Lightweight*

Directory Access Protocol), mas esta forma de armazenamento requer um esforço adicional por parte dos desenvolvedores. Um projeto de Ponnappan [8], apresentou a implementação e avaliação de desempenho de uma PBNM, utilizando COPS no modelo *outsourcing* com RSVP (COPS-RSVP) e utilizou a arquitetura CORBA (*Common Object Request Broker Architecture*) para interação de componentes, avaliando o desempenho para um servidor de políticas de QoS, utilizando o modelo de políticas de QoS QPIM (QoS Policy Information model) descrito por Snir [10].

4. XACML

O XACML (*eXtensible Access Control Markup Language*) é uma proposta do OASIS para modelar, armazenar e distribuir políticas descritivas de controle de acesso [7]. A proposta para uso do XACML segue a arquitetura PDP/PEP no modelo ***outsourcing***. A linguagem XACML é definida por dois esquemas (*schema*) XML: "*xacml context"* e "*xacml policy*". O "*xacml context*" é um modelo para representação das mensagens de requisição e resposta trocadas entre o PEP e o PDP. O "*xacml policy*" define como representar as políticas de controle de acesso.

A figura 2 mostra o diagrama UML para o esquema "*xacml policy*". A figura representa as classes e associações entre os elementos XACML, mas omite seus atributos. De acordo com a estratégia XACML, uma política determina um conjunto de permissões (ou negações) de acesso através de associações denominadas *Targets* (alvo). Uma *Target* é expressa pela sintaxe: "usuários (*Subjects*) podem (ou não podem) aplicar ações (*Action class*) sobre recursos (*Resource class*)".

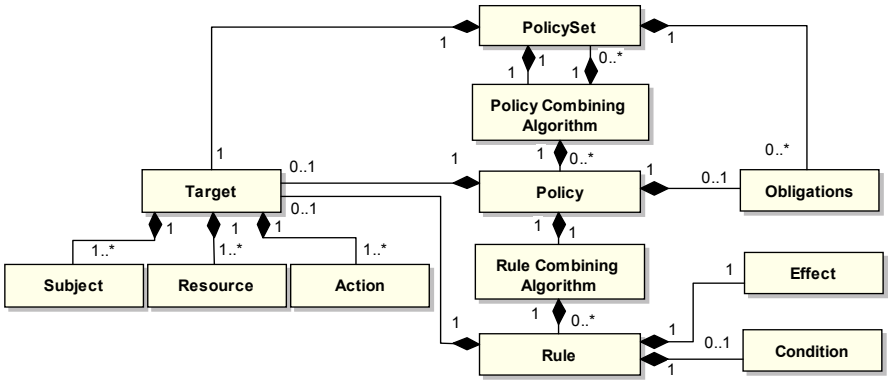


Figura 2. Modelo de política XACML

Os alvos (*Targets*) podem ser associados a uma política (*Policy*), a um conjunto de políticas (*Policy Set*) ou a uma regra (*Rule*). A *Target* associada a uma política ou um conjunto de políticas atua como um "seletor", por exemplo, quando um PEP requisita uma decisão referente a uma *Target*, somente as políticas ou conjunto destas que contenham os elementos necessários de uma *Target* para serem avaliadas serão processadas. As *Targets* associadas a regras (*Rules*) permitem expressar permissões (ou negações) condicionais. Uma regra é expressa pela sintaxe: "Se as condições *(Condition class*) forem satisfeitas, então aplicar o *efeito* (*Effect*) sobre o *alvo* (*Target*)". Os valores possíveis para *Effect* são: permitir (*Permit*) e negar (*Deny*).

Quando um PEP efetua uma requisição ao PDP, ele fornece os atributos que permitem identificar os elementos de uma *Target* (*Subject*, *Resource*, *Action*). O PDP

avalia as regras da política e determina se existe uma *Target* com esses atributos, e então retorna o efeito correspondente: "permitir" ou "negar". Se ele não conseguir encontrar uma *Target* em suas políticas que satisfaça os atributos fornecidos pelo PEP, ele retornará como resposta: não-aplicável (*NotApplicable*).

A classe *Obligations*, quando definida, é passada para o PEP juntamente com o resultado da avaliação da política. A versão 1.0 do XACML, utilizada neste estudo, não especifica os tipos de ações passadas em *Obligations*, apenas define que o PEP deverá interpretá-las, mas não especifica como será feito. Conforme descrito adiante, este trabalho utiliza a classe *Obligations* para informar parâmetros de QoS para um nó servidor de RSVP, possibilitando o processamento pelo PEP.

Apesar da classe *Obligations* oferecer uma alternativa para implementar algum tipo de política *provisioning*, observa-se que o XACML basicamente implementa o modelo ***outsourcing***. Isto é devido porque o PDP retorna praticamente decisões do tipo "*Permit*" ou "*Deny*" para os PEPs. Como será descrita na próxima seção, a definição da classe Obligations é bastante limitada, mas o "conceito" é flexível o suficiente para fornecer "informações de configuração" para nós de rede em diversos domínios de aplicações. Outras limitações da especificação XACML 1.0 são: ausência da definição referente ao protocolo de comunicação para suportar a troca de mensagens entre PDP e PEP e a definição da estratégia a ser adotada para armazenar os documentos XACML que representam as políticas na rede.

A estratégia adotada para avaliar um conjunto de regras associadas a uma mesma política é determinada pela classe "*Rule Combining Algorithm*" (algoritmo de combinação de regras). Da mesma forma que para as regras (*Rules*), as políticas são interpretadas de acordo com a classe *Policy Combining Algorithm* (algoritmo para combinação de políticas). O XACML define, ainda, que o administrador de políticas pode acrescentar suas próprias estratégias de algoritmos de combinação. A figura 3 ilustra como o modelo UML é representado em um documento XML.

```
<Policy PolicyId="..." RuleCombiningAlgId="...">
  <Target>
     <Subjects>...</Subjects>
     <Resources>...</Resources>
     <Actions>...</Actions>
  </Target>
  <Rule RuleId=" " Effect=" ">
     <Target>...</Target>
     <Condition FunctionId=" ">...</Condition>
  </Rule>
  <Obligations>
     <Obligation ObligationId=" " FulfillOn=" "> </Obligation>
  </Obligations> <!-- Em Obligations, o atributo FulfillOn indica se o obligation  -->
</Policy>       <!--deve ser exectuado no caso de um efeito "Permit" ou "Deny" -->
```

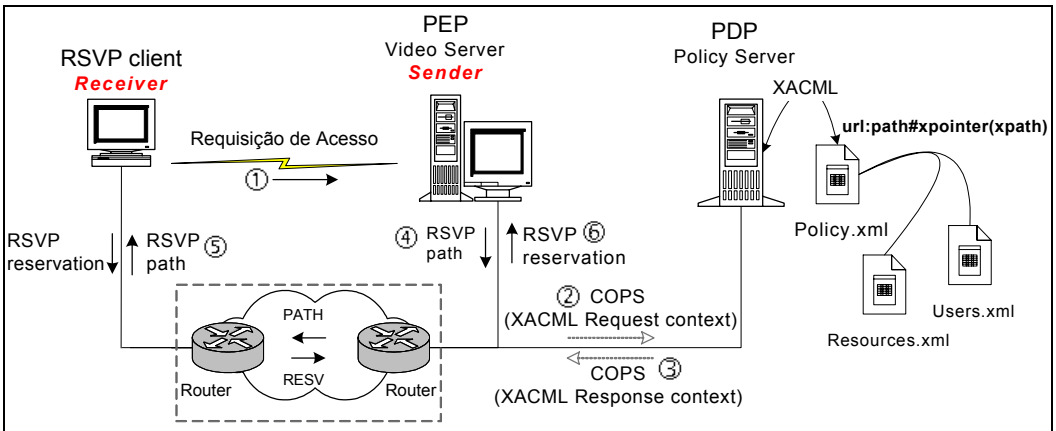
Figura 3. Representação de uma Política em XACML

5. Proposta

Este trabalho propõe um mecanismo de controle de políticas RSVP para servidores de aplicação que suportam RSVP (*RSVP-aware*), utilizando o modelo ***outsourcing*** e a especificação XACML 1.0. A figura 4 ilustra um cenário típico para aplicação deste mecanismo. O PEP representa um componente da aplicação servidora, responsável por solicitar as decisões ao PDP e interagir com o processo RSVP no servidor. O código do PEP deve ser integrado à aplicação servidora, conforme descrito na seção 6 deste

trabalho. Na arquitetura proposta, o PEP não atua somente como um **'executor'** de políticas, ele é o responsável por todas interações com o RSVP *daemon* liberando a aplicação de qualquer negociação de QoS. Esta interação inclui recuperar as informações para construção das mensagens PATH e garantir ou não a requisição de reserva na recepção de uma mensagem RESV. Esta abordagem pode ser implementada em qualquer sistema que suporte as APIs RSVP descritas na RFC 2205.

O armazenamento das políticas no PDP ocorre em um arquivo XML devido à especificação XACML estar em andamento quanto à definição de uma ou mais soluções de armazenamento. Como o XACML não especifica um protocolo para transação de políticas entre PEP e PDP, este trabalho sugere que as mensagens *xacml context* sejam transportadas utilizando o protocolo COPS. A seqüência de eventos relacionados ao estabelecimento de uma reserva RSVP para o cenário utilizando o mecanismo de controle de acesso proposto neste trabalho, é ilustrado na figura 4.



1. Um cliente RSVP solicita uma conexão a um servidor de vídeo para obtenção de serviços com QoS.

2. No servidor de vídeo, o PEP recebe a solicitação de serviço para avaliar a requisição. O PEP envia ao PDP uma mensagem *XACML request context* indicado a "*Target*" contendo seu endereço IP (*Resource*) e o IP do cliente (*Subject*) e a operação requisitada (*Action*).

3. O PDP avalia as restrições na política definida em XACML, e retorna ao PEP uma *decisão de autorização,* através de uma mensagem *XACML response context*, contendo, além do resultado, "permitir" ou "negar", as informações de especificação de tráfego Tspec, utilizando a estrutura "*Obligations*".

4. Em caso de uma decisão positiva, o PEP invoca seu serviço RSVP *daemon*, informando os parâmetros Tspec. O serviço RSVP *daemon*, na aplicação servidora RSVP, envia uma mensagem RSVP PATH ao receptor (neste caso o cliente RSVP). O parâmetros Tspec são armazenados no PEP para análise posterior (ver passo 6).

5. O cliente RSVP, ao receber a mensagem RSVP PATH invoca seu próprio RSVP *daemon*, o qual obtém os parâmetros da mensagem PATH e formata uma mensagem RSVP RESV, retornando-a ao emissor (aplicação servidora RSVP).

6. Ao receber a mensagem RESV do cliente, o RSVP *daemon* do servidor gera um evento ao PEP com o tráfego de dados Tspec. O PEP analisa se os parâmetros do Tspec são iguais ou menores ao que foi fornecido pela mensagem PATH (algoritmo **less_than_or_equal** - ver RFC 2210) e, em caso positivo, encaminha o pedido de reserva ao RSVP *daemon*. Nesse passo, cabe ao RSVP *daemon* analisar se possui ou não recursos suficientes para efetuar a reserva, através do módulo de controle de admissão.

Figura 4. Controle de Políticas RSVP com XACML

Os passos 1 a 6 referem-se a um cenário bem sucedido de reserva, e os tratamentos de erro foram omitidos. Uma solicitação de acesso RSVP difere de uma solicitação de acesso convencional (por exemplo, acesso a um recurso), porque o PDP precisa retornar o conjunto de informações necessárias para que o PEP possa construir a mensagem PATH. Por esta razão, diversas extensões no modelo XACML foram necessárias a fim de descrever e transportar informações de QoS. Como a descrição completa das extensões feitas no esquema XACML é bastante extensa, este artigo se limitará a descrever apenas os aspectos principais necessários para a compreensão dos

fundamentos da proposta. Para uma descrição completa do trabalho, favor referir-se a [11]. A estratégia adotada neste trabalho para descrever uma política RSVP em termos de XACML é ilustrada na Figura 5.

```
<PolicySet PolicySetId="RSVP_Aware_Server_Application">
    <Target> <!-- Defines the services (resources) to which the policy applies -->
    </Target>
    <Policy PolicyId="Service Level 1"> <!-- e.g. GOLD -->
        <Rule>
            <Target> <!-- Subjects to which the policy applies --> </Target>
            <Condition> <!-- Time and client's IP addresses restrictions -->
            </Condition>
        </Rule>
        <Obligations> <!-- TSpec specification for service level 1 -->
        </Obligations>
    </Policy>
    <Policy PolicyId="Service Level 2"> … </Policy> <!--  e.g. SILVER -->
    <Policy PolicyId="Service Level N"> … </Policy> <!-- e.g. BRONZE -->
    <Policy PolicyId="Default Policy"> <!-- usually denies all -->…</Policy>
</PolicySet>
```

Figura 5. Estrutura de política XACML para RSVP

Na estratégia proposta, cada servidor de aplicação "RSVP-aware" (ou grupo de aplicações) é mapeada em um XACML <PolicySet>. Aplicações servidoras podem ser descritas pelo mesmo conjunto de políticas somente se eles oferecerem os mesmos níveis de serviço de QoS e o mesmo conjunto de usuários, todos com as mesmas restrições. Por exemplo, servidores distintos de *Vídeo Streaming* dentro de uma empresa que ofereça um serviço "GOLD" para usuários registrados e serviços "SILVER" para visitantes, podem ser representados por uma política <PolicySet>.

As políticas são mapeadas para serviços através do elemento <Target> dentro da estrutura <PolicySet> (ver o exemplo na seção 6). O elemento <Policy> dentro da <PolicySet> são usadas para definir distintos níveis de serviços de QoS oferecidos pela mesma aplicação. Por exemplo, "GOLD", "SILVER", etc. O elemento <Rule> define os usuários (*Subjects*) que possuem autorização para receber o nível de serviço e os elementos <Obligations> descrevem os parâmetros *Tspec*.

A razão para definir uma política RSVP em termos de uma <PolicySet> e não em termos de uma simples <Policy> é relacionada pela definição XACML. Observe no diagrama XACML em UML da figura 2, que o elemento <Obligations> é mapeado para <Policy> or <PolicySet>, mas não pode ser mapeada para o elemento <Rule>, ou seja, todas regras <Rules> dentro de uma política define o mesmo <Obligations>. Portanto, serviços distintos não podem ser representados dentro de uma simples <Policy>.

Outro ponto importante é definir onde as informações dos usuários e serviços serão localizadas. Se considerarmos a proposta PCIM, definida pela IETF, uma proposta lógica seria representá-las usando objetos CIM. Supõe-se que o CIM é para ser suportado por um importante conjunto de empresas de hardware e software, é uma escolha interessante para compartilhar a mesma informação entre sistemas heterogêneos. Ambas informações de CIM[1] e PCIM podem ser armazenadas em servidores LDAP.

[1] CIM (Common Information Model), proposta pela DMTF (Distributed Management Task Force) é um modelo de informações compatível com o PCIM que define classes e associações para representar usuários, elementos de redes e serviços.

As definições XACML permitem definir todas as informações a respeito de políticas (*subjects, resources* e *actions*) dentro do mesmo documento XML, definidos pelo "*xacml policy scheme*". Contudo, o OASIS faz referência que será possível escrever políticas XACML que se refiram a elementos de informações armazenadas em um repositório LDAP, porém, na especificação 1.0 não define como será feito. Apesar disso, pela razão do XACML ser baseado em um padrão XML, uma possível solução seria criar referências em uma política para documentos externos usando a estratégia de *XML Pointer Language* (*XPointer*) da W3C [15]. Existem algumas referências a respeito do uso de *XPointer* na especificação 1.0 XACML, mas a informação é limitada pelo documento de requisição XACML (*Request context scheme*) e o seu uso nos documentos em uma política não é suportada. Porém, usando *XPointer* para criar políticas com reutilização de descrição de usuários (*Subjects*) e informações de serviços, é uma extensão lógica para futuras versões da especificação XACML.

Este trabalho adotou o uso de *XPointer* para definir políticas para a reutilização da descrição de usuários (*Subjects*) e informações de serviços. No futuro, esta proposta poderá ser substituída por LDAP sem modificar a estratégia da política. A figura 6 ilustra o relacionamento entre o repositório XML de documentos usados para descrever uma política. Exemplo de como estes documentos podem ser definidos são apresentados no estudo de caso na próxima seção (ver figuras 8 e 9).

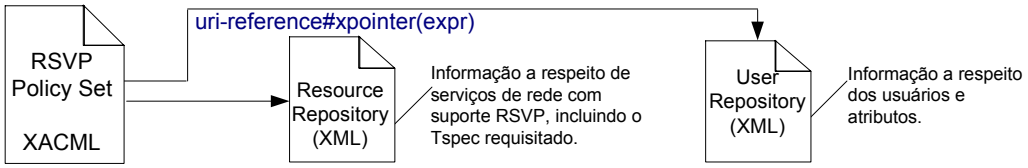


Figura 6. Documentos da Política, Recursos e Usuários.

A estrutura dos documentos de recursos e de usuários, descritos neste trabalho, foi escolhido intencionalmente para simples propósito didático. As informações de QoS estão descritas no documento XML de recursos (resources.xml). Um recurso é definido como um serviço de rede que suporta negociações RSVP. Daí, o documento de recursos descreve parâmetros de RSVP requeridos para construção da mensagem PATH, isto é, *Tspec* {r,b,p,m,M}, tipo de serviço (GS – *guaranteed service* ou CL – *controlled load*) e estilo de reserva descritos na RFC2210 [13] e RFC2215 [9].

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"   elementFormDefault="qualified">
 <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="1"><xs:element ref="xacml:RsvpService"/></xs:choice>
        <xs:choice minOccurs="0" maxOccurs="1"><xs:element ref="xacml:RsvpStyle"/> </xs:choice>
    </xs:sequence>
    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
    <xs:attribute name="RsvpClass" type="xacml:RsvpClassType" use="required"/>
 </xs:complexType>
 <!-- [RFC 2212 spec] definitions to elements above: TspecBucketRate_r , TspecBucketSize_b , QoS class, etc -->
```

Figura 7. Extensão do Esquema XACML para representação de políticas RSVP

A figura 7 mostra o esquema XML correspondente aos parâmetros RSVP. Esta proposta assume que um simples serviço pode oferecer diferentes níveis. Por exemplo,

um servidor de vídeo pode definir vários modos de QoS para *video streaming* em ordem para suportar diferentes resoluções. Neste caso, cada modo deve receber uma distinta classe específica (atributo *RsvpClass*). As classes enumeradas no esquema são definidas pela ITU-T e estão incluídas no esquema somente para fins de ilustração. Observe na figura 7, que o esquema do recurso RSVP não inclui os parâmetros do *Rspec*. Este trabalho sugere que o PEP poderia rejeitar a mensagem RESV recebida se os parâmetros do *Rspec* forem muito maiores que os especificados no *Tspec*, não necessitando uma nova consulta ao PDP para validar os recursos na mensagem RESV.

6. Caso de Estudo e Implementação

6.1 Caso de Estudo

Na seqüência para ilustrar o uso da proposta XACML para descrição de políticas RSVP, o seguinte cenário foi considerado. Um conjunto de servidores de *video streaming* em uma rede corporativa oferece um "tutorial" sobre equipamentos da empresa para usuários registrados e não registrados (visitantes em treinamento). A política adotada para ter acesso no serviço de "*VideoStreaming*" é definida como:

a) Usuários registrados têm permissão para acessar qualquer servidor dentro da rede corporativa oferecendo um serviço de "*VideoStreaming*" sem restrições quanto ao horário. Se um usuário se conectar ao servidor usando um computador cliente dentro da rede corporativa, ele receberá níveis de serviço "GOLD" ou "SILVER", caso contrário, receberá um nível de serviço "BRONZE".

b) Visitantes podem ter acesso ao serviço de "*VideoStreaming*" somente na rede interna dos laboratórios para treinamento e somente no horário comercial. Eles podem receber somente o nível de serviço "BRONZE".

As informações de serviços de vídeo são representadas no documento ilustrado na figura 8. Note que a estrutura <sap> define os serviços disponíveis na rede interna da empresa que estão sujeitados à política. A informação *Tspec* diz respeito dos níveis de serviços e também são definidos nas estruturas <serviceLevel> dentro do arquivo XML.

```
<service serviceId="VideoStreaming"> <!-- Resources.xml -->
<description>tutorial videos in the company network</description>
   <sap>
         <inetaddress>172.16.200.10</inetaddress>
         <inetaddress>172.16.5.3</inetaddress>
         <protocol>TCP</protocol>
         <port>8000</port>
   </sap>
   <serviceLevel serviceId="Gold">
     <ResourceRsvp AttributeId="qosG711" 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>
         <RsvpStyle>FF</RsvpStyle>
     </ResourceRsvp>
   </serviceLevel>
  <!-- definitions to others ResourceRsvp elements -->
   <serviceLevel serviceId="Silver"> … </serviceLevel>
   <serviceLevel serviceId="Bronze"> … </serviceLevel>
</service>
```

Figura 8. Informação de Serviços

```
<subjects>  <!-- Users.xml -->
    <user>
        <cn>Emir Toktar</cn>
        <sn>Toktar</sn>
        <uid>etoktar</uid>
        <mail>toktar@company.com</mail>
        <businessCategory>RegisteredUser
        </businessCategory>
    </user>
    <!-- definitions to others USERS -->
    <user>
        < …  >
    </user>
    <user>…
        < …  >
    </user>
    <user>
        <cn>Guest</cn>
        <uid>guest</uid>
        <businessCategory>UnregisteredUser
        </businessCategory>
    </user>
</subjects>
```

Figura 9. Informação dos Usuários

Uma requisição XACML para o servidor de vídeo (i.e., um PEP) usualmente identificará o usuário pelo seu login (*uid*). Contudo, a política dentro do PDP será descrita em termos do status do usuário (registrado ou não registrado). O mapeamento entre o "*user id*" e o usuário correspondente, é representado em um documento XML ilustrado na figura 9 e referenciado em um repositório XML chamado "Users.xml".

Conforme descrito na seção 5, a política RSVP é definida em termos de XACML <PolicySet>. A estrutura <PolicySet>, para o estudo de caso, define quatro políticas mostradas na figura 10. A estrutura *Target* dentro da <PolicySet> define recursos para qual a política se aplica. Note que os elementos <ResouceMatch> dentro da <Target> definem funções que comparam a informação fornecida pelo PEP através da mensagem *Request* (***resource-id*** e ***ip-address:sender***) com a informação descrita em XML no arquivo de informações de serviços, "Resources.xml" (ver figura 8).

```
<PolicySet PolicySetId="TutorialVideoServer" PolicyCombiningAlgId=":policy-combining-algorithm:first-applicable">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId=":function:string-equal">
              <AttributeValue DataType="#string">TutorialVideo</AttributeValue>
              <ResourceAttributeDesignator DataType="#string" AttributeId=":resource:resource-id"/>
        </ResourceMatch>
        <ResourceMatch MatchId=":function:xpath-node-match">
              <AttributeValue DataType="#string">http://pdp/resources.xml #xpointer(//service[@serviceId=
              "VideoStreaming"]/sap/inetaddress/text())</AttributeValue>
              <ResourceAttributeDesignator DataType="#string" AttributeId=":resource:authn-locality:ip-address:sender"/>
        </ResourceMatch>  <!-- Policy 1: Registered Users from inside of the network company -->
      </Resource>         <!-- Policy 02: Registered Users from outside of the network company -->
    </Resources>          <!-- Policy 03: Unregiestered Users (visitants) from inside of the network company -->
  </Target>               <!-- Policy 04 - Deny for All -->
  <Policy PolicyId=":TutorialRegUsersInternal" RuleCombiningAlgId=":rule-combining-algorithm:first-applicable">..</Policy>
  <Policy PolicyId=":TutorialRegUsersExternal" RuleCombiningAlgId=":rule-combining-algorithm:first-applicable">.</Policy>
  <Policy PolicyId=":TutorialRegUsersGuest" RuleCombiningAlgId=":rule-combining-algorithm:first-applicable">....</Policy>
  <Policy PolicyId=":TutorialDenyForOthers" RuleCombiningAlgId=":rule-combining-algorithm:first-applicable">
  <Rule RuleId=":Tutorial_Deny_Rule_For_Others" Effect="Deny"/></Policy>
</PolicySet>
```

Figura 10. Estrutura da "Policy Set"

Uma requisição de decisão de um PEP para um PDP, utilizando o "*xacml Request context"* é descrito a seguir. Uma requisição é composta de um elemento <Request> que possui a descrição dos elementos <Subject>, <Resource> e <Action> que definem os elementos de uma <Target> para avaliação de uma política no PDP. No presente estudo, o elemento <Subject> fornecerá informações a respeito do receptor (*receiver*) informando o "*user-id*" do usuário e endereço IP do computador cliente, ou seja, "***etoktar***" e "***172.16.0.1***", respectivamente. O elemento <Resource> fornece informações sobre o servidor de vídeo (*sender*), informando o "*resource-id*" do recurso "***VideoStreaming***" e o endereço IP, "***172.16.200.10***". O tipo de ação requisitado é definido no elemento <Action> como "***getResorceQoS***" e que será avaliado pela descrição da política no PDP.

A estrutura da "Política 1" dentro da <PolicySet> define condições aplicadas para o acesso de usuários registrados dentro da rede da empresa. O elemento <Subject> dentro da <Rule> define que políticas são aplicadas somente aos usuários registrados usando a função "*xpath-node-match*". Foi desenvolvida esta função para estender as funcionalidades do XACML possibilitando localizar informações em documentos externos à <Policy> (ver figura 6), utilizando dois parâmetros, um de localização

"*http://pdp/subjects.xml*" e outro da expressão *XPath* para pesquisa da informação "*#xpointer(//subjects/user[businessCategory='RegisteredUsers']/uid/text())*", segundo a sintaxe *XPointer*. O elemento <Condition> dentro da <Policy>, determina que políticas são aplicadas somente para a requisição onde o computador cliente está localizado dentro da rede interna da empresa, identificado pelo endereço IP do computador.

Finalmente, a figura 11 ilustra a resposta do PDP para o PEP. A permissão (*Permit*) do PDP informa que existem serviços oferecidos ao cliente. Os serviços são descritos na estrutura dentro de <Obligations>. Neste exemplo, duas especificações *Tspec* são retornadas para o PEP. Estas especificações correspondem aos níveis de serviços "GOLD" e "SILVER" oferecidos pelo servidor "*VideoStreaming*". Uma possível alternativa seria retornar somente o serviço de mais alto nível. A estrutura apresentada dentro de <Obligations> é definido pelo "XACML *context-schema*".

```
<Response>
  <Result>
    <Decision>Permit</Decision>
    <Status> <StatusCode Value="...:status:ok"/> </Status>
    <Obligations xmlns="urn:oasis:names:tc:xacml:1.0:policy">
      <Obligation ObligationId=":GoldSilverUsersInternal" FulfillOn="Permit">
        <AttributeAssignment AttributeId="RsvpClass#1"          DataType="#string"> G711</AttributeAssignment>
        <AttributeAssignment AttributeId="TokenBucketRate_r#1"  DataType="#double"> 9250.0</AttributeAssignment>
        <AttributeAssignment AttributeId="TokenBucketSize_b#1"  DataType="#double"> 680.0</AttributeAssignment>
        <AttributeAssignment AttributeId="PeakRate_p#1"         DataType="#double">13875.0</AttributeAssignment>
        <AttributeAssignment AttributeId="MinimumPoliceUnit_m#1" DataType="#integer">13875</AttributeAssignment>
        <AttributeAssignment AttributeId="MaximumPacketSize_M#1" DataType="#integer">13875</AttributeAssignment>
        <AttributeAssignment AttributeId="RsvpService#1"        DataType="#string">Guaranteed</AttributeAssignment>
        <AttributeAssignment AttributeId="ServiceQoS#1"         DataType="#string">FF</AttributeAssignment>
        <AttributeAssignment AttributeId="RsvpClass#2"          DataType="#string">H261QCIF</AttributeAssignment>
        <!-- definitions for others "Tspec" in the AttributeAssignment  elements -->
      </Obligation>
    </Obligations>
  </Result>
</Response>
```

Figura 11. Exemplo de uma resposta de avaliação de política

6.2 Implementação

Uma importante vantagem na abordagem do XACML em relação ao PCIM se refere a sua implementação. Pela razão que o XACML é definido em termos de XML, a implementação se beneficia de ferramentas existentes para desenvolvimento de aplicações XML. As implementações públicas disponíveis para XACML são oferecidas em linguagem JAVA, no projeto SUN XACML, e em C++ pela Jiffy Software.

A arquitetura descrita neste trabalho foi implementada usando a linguagem Java™ 2 SDK, *Standard Edition* 1.4.2 e o *pacote* de implementação *Sun XACML*. O pacote Sun XACML inclui os módulos dos esquemas "*Policy*" e "*Context*". O primeiro módulo suporta a interpretação de políticas XACML (requeridas para implementar um PDP) e a segunda, a troca de mensagens XACML entre PDP e PEP.

A implementação permitiu avaliar a proposta das extensões XACML que são compatíveis com a implementação *Sun XACML*. A estratégia adotada consiste em adicionar novas funcionalidades para a arquitetura XAMCL sem modificar o esquema. Observamos que o *pacote* XACML da Sun é flexível para adicionar novas funcionalidades sem alteração dos códigos, exceto no caso do tratamento da estrutura <Obligations>. Foi desenvolvida a função para o uso de *XPointer*, para referenciar arquivos externos, e foi adicionado as funções existentes do pacote através das APIs fornecidas no pacote *Sun XACML*, o que simplifica significativamente o processo de desenvolvimento do PDP e de PEPs integráveis com aplicações existentes.

A seguir são apresentados alguns exemplos de utilização do pacote XACML da Sun para desenvolvimento do PDP. O fragmento de código a seguir ilustra a seqüência de passos para criar uma instância de um PDP, inicializado com um arquivo de políticas definido por "*PolicyQoS.xml*". A chamada *policyModule.addPolicy* permite validar a política em relação ao esquema *xacml policy*. Esta chamada foi utilizada para validar a sintaxe das extensões de esquema propostas neste trabalho.

```
FilePolicyModule policyModule = new FilePolicyModule();
policyModule.addPolicy("Path/PolicyQoS.xml");
```

O pacote da Sun oferece classes que, através de tabelas de *Hashing*, simplificam o processo de procura de políticas (*PolicyFinder*) e atributos (*AttributeFinder*). O fragmento de código típico para criação de uma instância do PDP é ilustrado a seguir.

```
PolicyFinder policyFinder = new PolicyFinder();
Set policyModules = new HashSet();
policyModules.add(policyModule);
policyFinder.setModules(policyModules);
AttributeFinder attrFinder = new AttributeFinder();
List attrModules = new ArrayList();
attrFinder.setModules(attrModules);
PDP pdp = new PDP(new PDPConfig(attrFinder, policyFinder, null));
```

A seguir, é mostrado o fragmento de código para criação do PEP. A classe *RequestCtx* implementa um PEP requisitando para um PDP. Os atributos passados dentro da classe construtora se referem ao elemento *Target* (*Subject*, *Resource* e *Action*). O atributo de ambiente (*attrEnvironment*) é usado para passar outra informação relevante, por exemplo, como referência da data e hora do sistema.

```
RequestCtx request = new RequestCtx(attrSub,attrRes,attrAction,attrEnvironment);
```

O objeto *ResponseCtx* recebe a avaliação do PDP através da chamada ***evaluate***. O resultado contém a decisão, código de status e opcionalmente uma *Obligations*, apresentada no código: *ResponseCtx* response = pdp.**evaluate**(request);

7. Conclusão

Neste trabalho, estendeu-se a utilização do modelo de XACML para além das funcionalidades de controle de acesso, pois foram incluídas nas respostas enviadas do PDP para o PEP, as especificações de tráfego *Tspec*, necessárias para construção das mensagens PATH. O retorno de parâmetros para configuração de dispositivos juntamente com as decisões do PDP é uma característica importante para as abordagens de gerência de redes baseada em políticas. Essa funcionalidade, facilmente suportada nos modelo de políticas baseados na proposta PCIM do IETF, foi de difícil implementação no XACML por adotar uma estratégia de *outsourcing* e retorno de parâmetros limitados.

Para suportar o cenário RSVP, modificações na estrutura <Obligations> foram requiridas, incluido funcionalidades não suportadas pelo pacote de implementação XACML da Sun. A especificação XACML versão 1.0 e o pacote correspontdente são deficientes no retorno dos resultados que não são simplesmente decisões de permitir ou negar. Nesta proposta de trabalho, algumas características foram adicionadas para a

arquitetura XACML sem modificar o esquema: elementos <Obligations> são dinamicamente processados e referências *XPointer* para documentos externos foram usadas para criação de políticas com objetos reutilizávies de "*resources*" e "*subjects*".

O XACML ainda encontra-se em desenvolvimento e, embora limitado em alguns aspectos pela falta de padronização, mostrou-se um modelo flexível para descrição de políticas de acesso em diferentes domínios de aplicação. Algumas modificações no esquema XACML seriam bastante úteis. Sugere-se um modo mais flexível de mapear <Obligations> para políticas. Como no diagrama UML não existe associação entre estas <Obligations> e <Rules>, este mapeamento permitiria definir níveis de serviço em uma simples política, flexibilizando totalmente o modelo e muito útil para outros domínios de aplicações. Outra modificação sugerida é formalizar o uso de referências *XPointer* no esquema XAMCL.

8. Referências Bibliográficas

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

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

[3] Chan K.; Seligson, J.; Durham, D.; Gai, S.; McCloghrie, K.; Herzog, S.; Reichmeyer, F.; Yavatkar, R.; Smith, A. "COPS Usage for Policy Provisioning", RFC3084, 2001.

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

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

[6] Nabhen, R., Jamhour, E., Maziero C. "Policy-Based Framework for RBAC", 14th IFIP/IEEE: Workshop on Distributed Systems: Operations and Management, DSOM'03.

[7] OASIS, *eXtensible Access Control Markup Language* - XACML v 1.0, Feb. 2003.

[8] Ponnappan, A.; Yang, L.; Pillai, R.; Braun, P. "A Policy Based QoS Management System for the IntServ/DiffServ Based Internet". Proceedings - IEEE, POLICY 2002.

[9] Shenker, S.; Wroclawski, J. "General Characteri-zation Parameters for Integrated Service Network Elements", RFC 2215, Sep. 1997.

[10] 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.

[11] Toktar, E. "Controle de Admissão de RSVP utilizando XACML". Dissertação de Mestrado, PPGIA, PUCPR. Aug. 2003.

[12] Westerinen, A. et. al. "Terminology for PBMN". RFC3198, Nov. 2001.

[13] Wroclawski, J. "RSVP with INTSERV", RFC 2210, Sep. 1997.

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

[15] W3C, XPointer Framework, W3C Recommendation, 25 March 2003.