LARCES_PBM: Um Framework para Gerenciamento de QoS e Segurança Baseado em Políticas[1]

Ana Luiza Bessa de P. Barros, Marcial Porto Fernandez, Joaquim Celestino Junior, Laure Mendouga, César Augusto O. Soares, Marcos Dantas Ortiz, Antônia Diana Braga, Rafael Ramos, André Luís de O. Campos, Abner Rodrigues Neto

Departamento de Estatística e Computação – Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – Campus do Itaperi
CEP 60.740-000 – Fortaleza – CE – Brasil

{analuiza,marcial,celestino,laure,cesar,marcos,diana,rafael,andre,abner}@larces.uece.br

Abstract. *This paper presents the development of a frameowrk to manage QoS and security based on policies. The framework's goal is to manage different types of networks, wired or wireless, where it has to be adapted for each type. The framework is based on DMTF/IETF policy architecture. All modules were developed: PMT (Policy Management Tool), PR (Policy Repository), PDP (Policy Decision Point) e PEP (Policy Enforcement Point). An IEEE 802.11 network was used as a case study for this work.*

Resumo*. Esse artigo apresenta o desenvolvimento de um framework para gerenciamento de QoS e segurança baseado em políticas. O objetivo do framework é gerenciar diversos tipos de redes, cabeadas ou sem fio, bastando que o mesmo seja adaptado para cada tipo. O framework baseia-se na arquitetura de políticas definida pelo DMTF/IETF. Foram implementados todos os módulos dessa arquitetura: PMT (Policy Management Tool), PR (Policy Repository), PDP (Policy Decision Point) e PEP (Policy Enforcement Point). Para esse trabalho, foi utilizada uma rede IEEE 802.11 como estudo de caso.*

1. Introdução

No mundo ideal, as redes teriam recursos suficientes. Similarmente, todos os usuários seriam confiáveis e executariam apenas as aplicações que necessitassem. No entanto, o que realmente acontece é bem diferente.

Em uma rede sem fio alguns problemas podem se tornar ainda mais críticos. A limitação de recursos, que pode afetar a qualidade de serviço (QoS), se torna ainda mais crucial, incluindo-se também outros problemas, como por exemplo, o tempo de vida da bateria de um dispositivo móvel. A questão da segurança também se agrava; além dos problemas convencionais de qualquer rede, tem-se ainda a questão da suscetibilidade do meio de transmissão utilizado, o ar, a possíveis ataques.

O uso de políticas está sendo empregado para aplicações como definição de estratégias para gerenciamento de qualidade de serviço e configuração. A gerência de redes, antes limitada aos aspectos físicos (equipamentos) ou aos serviços, passa a ser

[1] Esse trabalho foi parcialmente financiado por CNPq/FUNCAP e desenvolvido em colaboração com a HP Brasil P&D.

vista agora a partir da visão do negócio, via Acordos de Níveis de Serviço (*Service Level Agreements* – SLAs).

O LARCES desenvolveu o *framework* LARCES_PBM, para gerenciamento, através de políticas, de qualidade de serviço e segurança em diversos tipos de rede, cabeadas ou sem fio. As informações definidas nos contratos de serviço (SLAs), entre usuário e provedor do serviço, devem ser mapeadas para as diversas infraestruturas. Foi desenvolvido também um módulo de gerenciamento baseado em políticas para redes 802.11, implementado a partir do framework.

O modelo de informações de políticas utilizado no framework foi o PCIM (*Policy Common Information Model*), definido pelo DMTF/IETF. Uma das contribuições desse trabalho foi a estruturação das informações no modelo PCIM e o mapeamento das informações do PCIM para o modelo de informações do LDAP (*Lightweight Directory Access Protocol*), utilizado como repositório de políticas.

Na seção 2 é apresentado o framework de gerenciamento e a comunicação entre os módulos. A especificação de cada módulo e suas funcionalidades são mostradas nas seções seguintes, sendo seção 3 para o PMT, seção 4 para o repositório, seção 5 para o PDP e seção 6 para o PEP. Na seção 7 são discutidos alguns resultados relacionados ao desempenho do framework e na seção 8 são apresentadas as conclusões.

2. Framework LARCES_PBM

O *framework* é baseado na arquitetura do DMTF/IETF. Foram implementados todos os módulos presentes nessa arquitetura, sendo eles: PMT (*Policy Management Tool*), PR (*Policy Repository*), PDP (*Policy Decision Point*) e PEP (*Policy Enforcement Point*).

A arquitetura do framework é apresentada na figura 1. O repositório de políticas utilizado foi o OpenLDAP [OpenLDAP] e o modelo de informações foi o PCIM. Foram também utilizadas duas PIBs (*Policy Information Base*), uma genérica sobre políticas e uma desenvolvida especificamente para o Ponto de Acesso (*Access Point*–AP) utilizado nos experimentos.

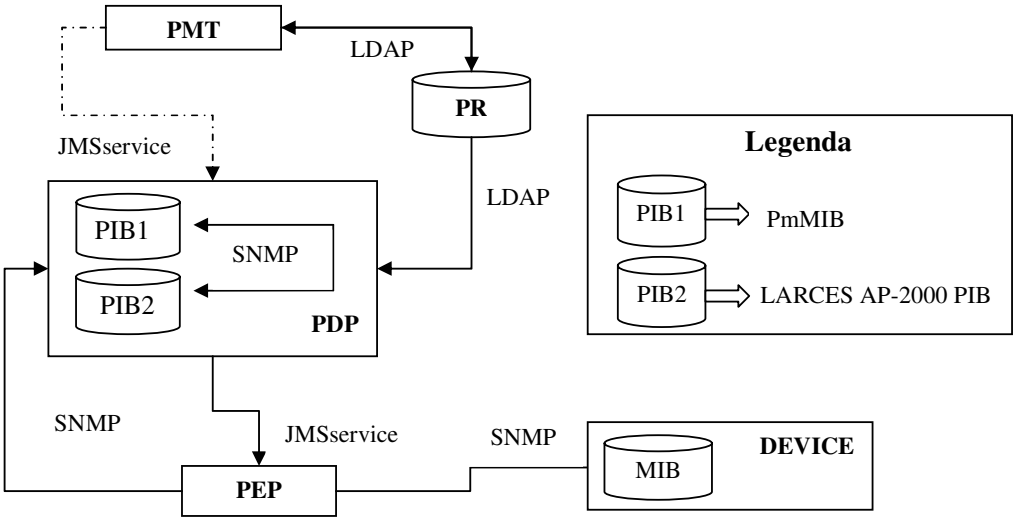

Figura 1 – Arquitetura do Framework LARCES_PBM

3. Policy Management Tool (PMT)

A ferramenta de gerenciamento de políticas permite ao administrador configurar as políticas em uma linguagem de alto nível e enviá-las ao repositório para que fiquem armazenadas e possam ser utilizadas, posteriormente, pelo sistema. A ferramenta foi desenvolvida em Java e é responsável pelo cadastro de dados para definição, tradução e atribuição de políticas. Através dessa ferramenta o administrador pode:

- ✓ Cadastrar SLAs, clientes, dispositivos, serviços e aplicações;
- ✓ Configurar os dados referentes ao serviço de comunicação dos módulos e do repositório de políticas.

Para cadastrar um SLA (figura 2), é especificado o contrato que deve ser provido para um determinado cliente ou grupo de clientes. O SLA é composto por parâmetros e pode ser implementado através de regras, as quais são formadas por condições e ações.

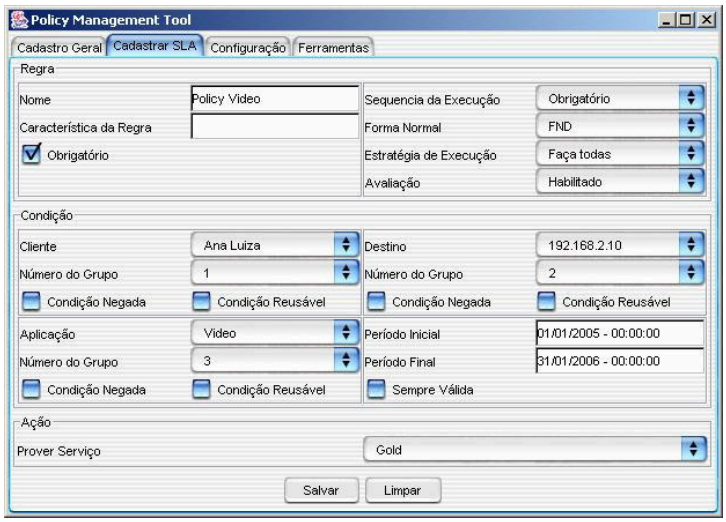

Figura 2 - Tela para Cadastrar SLA

Os campos da regra são:

- Nome: identificador da regra. Sempre deve ser editado.
- Característica da regra: campo opcional, que informa a função da regra.
- Seqüência de execução: especifica se a ordem de execução das ações deve seguir alguma estratégia.
- Forma normal: especifica a forma normal (disjuntiva, conjuntiva ou nenhuma das duas) na qual a regra está sendo montada. Toda regra é da forma **se** <condição> **então** <ação>.
- Estratégia de execução: Especifica a estratégia usada na execução das ações.
- Tipo de avaliação: Indica se a regra está atualmente habilitada ou não, sem ser necessário removê-la ou adicioná-la.
- Obrigatório: Indica se a regra deve ser avaliada ou não.

O SLA é formado por quatro condições: nome do cliente, nome da aplicação, endereço IP do destino e o período no qual a regra é válida.

As três primeiras condições podem ser negadas e reusáveis. Uma condição reusável não é propriedade de uma regra específica, ou seja, outras regras podem usar a mesma condição. A propriedade número do grupo, juntamente com forma normal,

indica a posição da condição na regra. No caso de quatro condições com a seguinte configuração:

C1: Número do Grupo = 1, Condição Negada = FALSO
C2: Número do Grupo = 1, Condição Negada = VERDADEIRO
C3: Número do Grupo = 2, Condição Negada = FALSO
C4: Número do Grupo = 3, Condição Negada = FALSO

Se a forma normal da regra for Disjuntiva (DNF), então as condições serão organizadas da seguinte forma:

(C1 AND (NOT C2)) OR (C3) OR (C4)

Por outro lado, se a forma normal for Conjuntiva, as condições serão organizadas de outra forma:

(C1 OR (NOT C2)) AND (C3) AND (C4)

A condição de tempo é formada pelo período inicial e final. O administrador pode marcar o campo "sempre válida", para regras que não possuam períodos específicos.

A ação, que será executada quando as condições da regra forem avaliadas como verdadeiras, é formada pelo serviço que deve ser provido.

Um SLA, após ser criado, é traduzido para uma regra, como mostrado na figura 3. Se todas as condições forem avaliadas como verdadeiras, a aplicação de vídeo será provida com o serviço *Gold*, no exemplo apresentado na figura 2. Após o administrador cadastrar o SLA, o PMT traduz as informações cadastradas de acordo com o *schema* de classes no LDAP (*Lightweight Directory Access Protocol*), antes de enviá-las ao repositório.

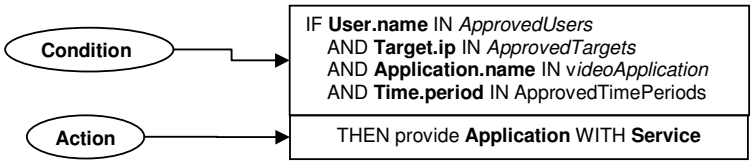

Figura 3 - SLA da Figura 2 traduzido em regra

Duas funcionalidades importantes também presentes no PMT são: configuração dos servidores utilizados pelo sistema e ferramentas para descoberta de recursos e para visualização do repositório de políticas – *LDAPBrowser*.

O *LDAPBrowser* permite ao administrador visualizar As regras existentes no repositório de políticas e os clientes, serviços, aplicações e dispositivos cadastrados. A figura 4 representa um exemplo dessa visualização.

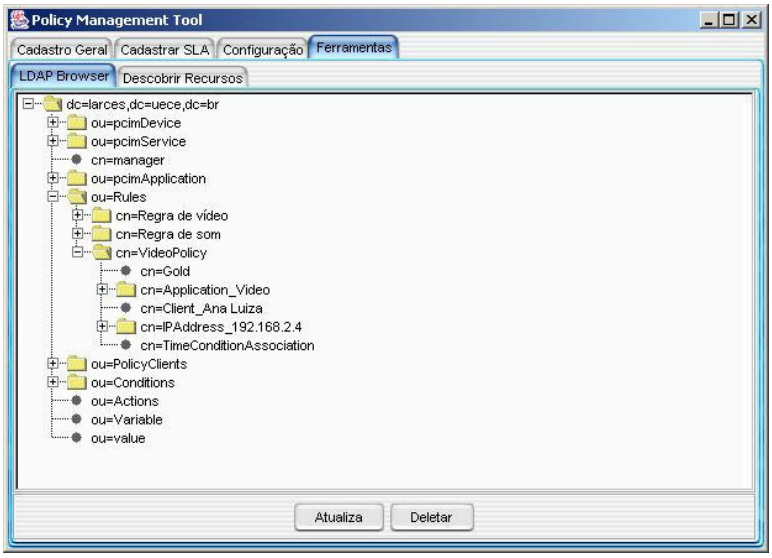

Figura 4 – LDAP Browser

4. Policy Repository (PR)

O repositório de políticas é o módulo do *framework* que armazena as políticas utilizadas no sistema. A ferramenta de gerenciamento (PMT) armazena as políticas no PR e o PDP as recupera. O repositório atua como um centralizador lógico das informações que gerenciam toda a rede. Além das políticas, o repositório também armazena as informações utilizadas para criação dos SLAs. Essas informações são referentes aos dispositivos e serviços que poderão ser utilizados, às aplicações e aos clientes da rede.

O repositório de políticas utiliza o LDAP (*Lightweight Directory Access Protocol*) como protocolo para comunicação com o PDP e com o PMT. No desenvolvimento desse *framework* foi utilizado o servidor de diretório da OpenLDAP.

4.1. Modelo de Informação de Políticas

As políticas criadas no PMT obedecem aos parâmetros definidos no PCIM, descritos na RFC 3060 [RFC_PCIM]. Para garantir que o PDP possa realizar a busca adequada dos dados no repositório foi implementado o mapeamento do PCIM para um tipo de dados que pudesse ser armazenado em um servidor LDAP. Esse mapeamento foi definido pela RFC 3703 [RFC_LDAP] e em seguida ampliado pelo DRAFT *Policy Core Schema* [DRAFT_ LDAP].

Também foram acrescentadas classes ao *schema* do LDAP, inexistentes na RFC 3060 e no DRAFT, com o objetivo de melhorar o desempenho das interações cliente/servidor LDAP e aperfeiçoar as recuperações de dados do LDAP.

O modelo de informações de políticas utilizado neste *framework* está baseado na RFC 3460 [RFC_PCIM_EXT] que define uma extensão para a RFC 3060. A RFC 3703 realiza, como mencionado anteriormente, o mapeamento do modelo descrito na RFC 3060. Logo, devido a essa desatualização da RFC 3703, tornou-se necessária a utilização do *DRAFT Policy Core Schema* que realiza o mapeamento baseado no modelo de políticas definido na RFC 3460. Como o DRAFT utiliza algumas classes

definidas no mapeamento descrito pela RFC 3703, é necessária a implementação de ambos os mapeamentos.

Há um número considerável de diferenças entre a especificação de classes no PCIM e no LDAP. Algumas das diferenças consideradas relevantes para a tarefa de mapeamento e utilização das classes mapeadas estão descritas abaixo:

- O CIM utiliza o termo propriedades para o que o LDAP chama de atributos;
- O CIM utiliza a notação "[]" para indicar que uma propriedade é multivalorada. Na definição de um atributo no LDAP, a não utilização do qualificador "SINGLE-VALUE" indica que o atributo é multivalorado;
- No CIM, classes e propriedades são identificadas pelo nome e não por OID's (*Object Identifiers*), como ocorre no LDAP;
- O CIM tem apenas dois tipos de classes (abstrata e instanciável), em vez dos três tipos de classes do LDAP: abstrata, auxiliar e estrutural. O tipo de uma classe do CIM é indicado pela utilização do qualificador "ABSTRACT" na sua definição. Além de tal qualificador, o LDAP utiliza o "AUXILIARY", tendo classes estruturais como padrão no caso da não especificação do tipo de classe;
- No CIM, a propriedade é definida dentro do escopo de uma simples definição de classe. Ela só pode aparecer nas classes ou em subclasses das quais essas foram definidas.No LDAP a definição de atributos é global e o mesmo atributo pode aparecer em muitas classes. Um efeito desta diferença é que as propriedades do CIM tendem a ser bem mais organizadas do que os atributos do LDAP.

Não foram utilizadas todas as classes de objetos mapeadas pela RFC 3703 e pelo *DRAFT Policy Core Schema* no armazenamento, pois a criação das políticas pela ferramenta de gerência não requisitou todos esses recursos disponíveis. Entretanto, devido aos relacionamentos existentes entre essas classes, tornou-se necessária a inserção, no repositório, de todo mapeamento descrito.

O mapeamento na RFC 3703 e no *DRAFT* é descrito de forma a capacitar o entendimento humano, tendo que ser reescrito em uma sintaxe adequada ao serviço de diretório utilizado (*schema*). Os arquivos com os *schemas* para o serviço de diretório da OpenLDAP estão disponíveis na implementação desse *framework*.

4.2 Armazenamento de Informações Adicionais

Além das classes definidas na RFC 3703 e no *DRAFT Policy Core Schema,* foram adicionadas à implementação do *framework* outras classes cujas instâncias armazenam as informações para criação dos contratos (SLAs). Tais informações, adicionadas pela ferramenta de gerência, geralmente utilizam atributos com sintaxe de *string,* mas há casos em que a sintaxe é booleana.

Como no caso das políticas, as entradas utilizadas para armazenar informações adicionais estão abaixo das entradas que são adicionadas inicialmente pelo administrador e indicadas para a ferramenta de gerência a partir de um arquivo XML. Essas classes de objetos adicionais são descritas abaixo:

- **pcimClient**. Classe estrutural que armazena informações dos clientes da rede. Esta classe tem todos os atributos com sintaxe de *string*. Como esta classe herda da classe inetOrgPerson, ela tem como atributos obrigatórios "cn" e "sn" e pode utilizar todo seus atributos opcionais. Essa herança é

utilizada para que objetos de pcimClient sejam identificados como usuários que podem se autenticar no repositório. A *objectClass* "inetOrgPerson" está mapeada e descrita no esquema fornecido com a instalação do serviço de diretório OpenLDAP.

- **pcimDevice**. Armazena informações sobre os dispositivos da rede cadastrados pela ferramenta de gerência. Essa classe, além de s*tring*, também utiliza atributos com sintaxe booleana: isPDP e isPEP.
- **pcimService**. Serviço que será aplicado pelo PDP em toda a rede. Todos os atributos dessa classe utilizam sintaxe de *string*.
- **pcimApplication**. As instâncias de pcimApplication descrevem as aplicações que executam na rede. Essas entradas têm apenas atributos com sintaxe de *string*.

5. Policy Decision Point (PDP)

O PDP é o bloco que se pode chamar de núcleo de todo o framework. Este bloco é responsável por interpretar as políticas armazenadas no repositório e enviá-las ao PEP para que possam ser executadas. A separação física entre o PDP e PEP é opcional. No trabalho realizado, essa separação foi feita considerando os benefícios que essa escolha traria no que diz respeito à agilidade de codificação, simplicidade e, principalmente, escalabilidade.

Para a comunicação entre PDP e PEP foram utilizados dois protocolos: SNMP e JMS. Este último nativo da linguagem Java, que foi utilizada em todo o framework. Para tal comunicação, foi tentado também utilizar o protocolo COPS, porém sua utilização ainda é baixa nos equipamentos comerciais e não foi possível encontrar uma biblioteca com implementação funcional.

As principais funções executadas pelo PDP são: tradução de políticas, configuração das bases de informação, distribuição de políticas, validação de políticas e detecção/resolução de conflitos. As três primeiras foram implementadas nessa versão do framework, enquanto que as duas últimas serão desenvolvidas em versões futuras.

A função de distribuição de políticas faz com que as políticas cheguem aos dispositivos nos quais elas serão aplicadas (PEPs). Para isso, há 2 principais métodos, denominados *provisioning* (provisionamento) e *outsourcing* (sob demanda).

No primeiro, as políticas são enviadas diretamente aos PEPs, fazendo com que estas sejam imediatamente aplicadas. Esse método deixa a cargo do PDP toda a complexidade e overhead de distribuição das políticas.

No segundo, as políticas são requisitadas pelo PEP ao PDP em situações em que este não saiba como tratar determinado tráfego.

O método atualmente adotado pelo framework propõe uma solução híbrida, onde apenas é enviada uma mensagem de notificação aos PEPs sobre a adição de uma nova política ou alteração de uma já existente. Esse então se encarrega de obter as políticas através de requisições para o PDP.

Independentemente do método utilizado, de acordo com [WALDBUSSER] algumas verificações devem ser feitas antes que haja realmente a distribuição como, por exemplo, se as políticas estão dentro dos seus horários de aplicação.

Essa checagem é feita através de um escalonador que, a cada intervalo de tempo pré-configurado, analisa o período de validade da política. Ainda segundo [WALDBUSSER], essa análise deve ser feita de forma "isolada", não levando em consideração resultados anteriores de avaliações sobre essa mesma política. Isto pode

implicar que, se em um dado instante a política é avaliada como dentro de seu período de atuação e consequentemente distribuída para aplicação nos dispositivos. Em um instante seguinte, no próximo intervalo de escalonamento (se curto o suficiente), o resultado dessa avaliação novamente será verdadeiro e, portanto, a política será novamente distribuída.

Daí a importância de fazer um balanceamento entre o intervalo de escalonamento e o período de validade da política, para que não haja repetidas distribuições.

5.1.Arquitetura

Diversos módulos compõem o PDP, separados conforme as suas funcionalidades, resultando na arquitetura apresentada na figura 5. São eles:

- RepositoryCommunication: Comunicação entre o PDP e o PR;
- PolicyTranslator: Obtenção das políticas armazenadas no PR e tradução de seus atributos para um nível mais baixo e detalhado. Envolve também a conversão do formato LDAP para novamente o modelo CIM;
- IBWriter: Envio de requisições de escrita das políticas nas PIBs;
- IBAgent: Atendimento de requisições de escrita (enviadas pelo IBWriter, IBScheduler) e leitura (enviadas pelo PEP) nas bases de informações;
- IBScheduler: Verificação do período de validade de uma determinada política, associada a ele após ter sido escrita com sucesso nas bases de informações;
- PolicyDispatcher: Interface responsável pela distribuição da política;
- MessageHandler: Interface responsável pelo recebimento de notificações enviadas através do servidor de mensagens.

RepositoryCommunication	
PolicyTranslator	IBScheduler
IBWriter	PolicyDispatcher
IBAgent	MessageHandler

Figura 5 – Arquitetura do PDP

6. Policy Enforcement Point (PEP)

O PEP é o bloco responsável por aplicar, de fato, as políticas nos dispositivos. Em uma rede gerenciável, vários dispositivos semelhantes podem ser submissos às ordens de um único PEP. Para cada tipo de dispositivo é preciso criar um PEP que saiba exatamente a maneira com que o dispositivo trabalha. No caso desse trabalho, foi desenvolvido um PEP para o Ponto de Acesso HP/Compaq WL510.

Como os dispositivos não trabalham usando políticas como parâmetros, este bloco precisa fazer uma tradução das políticas para a linguagem específica do dispositivo. A tradução consiste em receber as políticas oriundas do PDP, tratá-las e convertê-las para o formato que o dispositivo consiga trabalhar, isto é, no caso do AP, configurar um parâmetro através do protocolo SNMP.

A comunicação entre o PDP e o PEP é baseada no protocolo SNMP versão 2. O PDP recupera as políticas do repositório, as traduz e envia para o PEP. A comunicação entre PDP e PEP se dá através da PIB LARCES-AP2000-PIB. Para enviar as políticas

para o PEP, o PDP escreve nessa PIB e o PEP lê a partir dela. O PDP escreve na PIB e manda para o PEP uma mensagem através do protocolo JMS [JMS_SPECIF]. Esta mensagem contém somente os valores (OIDs) que foram modificados.

Ao saber quais OIDs foram modificadas, o PEP manda mensagens SNMP GET para a PIB para receber tais valores. Estes valores são então lidos, tratados, traduzidos, para enfim serem enviados, também na forma de mensagem SNMP, para o dispositivo.

6.1. PIBs Utilizadas

No desenvolvimento do framework foram utilizadas duas PIBs, a pmMib e a LARCES-AP2000-PIB. A pmMib é uma PIB genérica de políticas (tempo de duração de políticas, estado, etc) [DRAFT_PmMIB]. A outra PIB, a LARCES-AP2000-PIB, apresentada na figura 6, foi criada para armazenar os valores que podem ser manipulados no dispositivo de rede, no caso o AP HP/COMPAQ WL510.

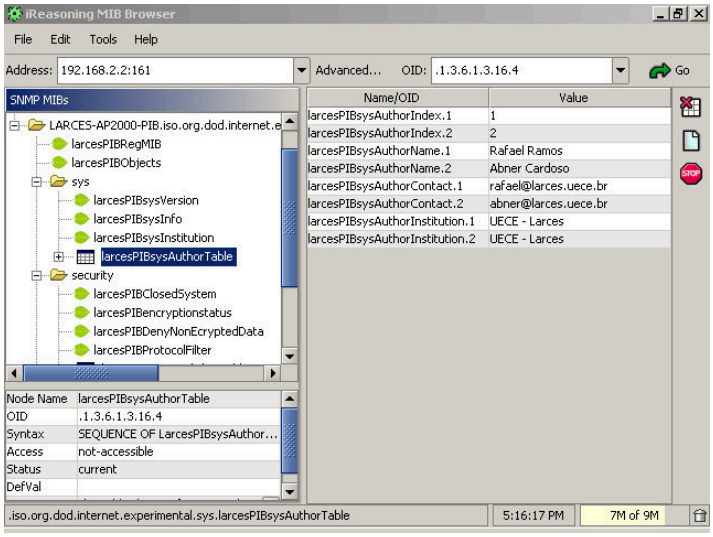

Figura 6- Ilustração da LARCES-AP2000-PIB

Na LARCES-AP2000-PIB, existem dois grupos básicos de objetos: sys e security. Todos os objetos de segurança estão agrupados em security. Em sys, são armazenadas informações referentes ao arquivo da PIB, como versão, autores e etc.

Os objetos relacionados com cada grupo são os seguintes:

- ✓ Grupo sys: Version, Info, Institution, AuthorTable, AuthorIndex, AuthorName, AuthorContact, AuthorInstitution;
- ✓ Grupo security: ClosedSystem, EncryptionStatus, DenyNonEncryptedData, ProtocolFilter, AccessControl, OperationType, ProtocolFilterTable, ProtocolIndex, ProtocolName, ProtocolComment, ProtocolStatus, AccessControlTable, AccessControlIndex, MACAddress, MACComment, MACStatus.

7. Testes

Para validação do framework, foram executados testes funcionais e de desempenho. Um dos objetivos dos testes foi medir o tempo que uma política leva desde o instante em que é inserida através da ferramenta de gerência até chegar ao PEP, passando pelo

repositório e pelo PDP. Outro objetivo foi avaliar cada módulo individualmente na manipulação de uma política. Toda a análise foi feita de acordo com o tipo de política inserida, a saber: políticas que possuem condições reusáveis (CR–essa mesma condição pode ser reutilizada por diversas outras políticas) e/ou condições de tempo (CT– determina o período de validade da política).

Os testes apresentados consistiram na inserção e processamento de 4 políticas, sendo uma de cada tipo: CR, ~CR, CT e ~CT. Foram calculados tempos médios para cada operação realizada no framework para cada política. A figura 7 indica que o fator preponderante no processo de inserção de políticas é a ausência de condições reusáveis.

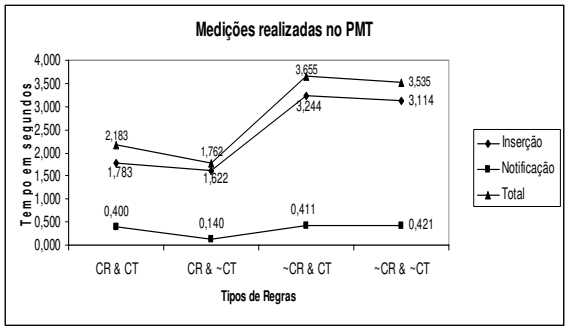

Figura 7 – Medições realizadas no PMT

No PDP, para o processo de tradução (figura 8), que efetua buscas no repositório, o tempo de obtenção das CRs é maior que o das *não CR* (~CR). No caso das condições de tempo, a ausência destas (~CT) implica que a política é sempre válida. A presença dessas condições acarreta mais buscas no processo de tradução e implica em aumento do seu tempo de completude.

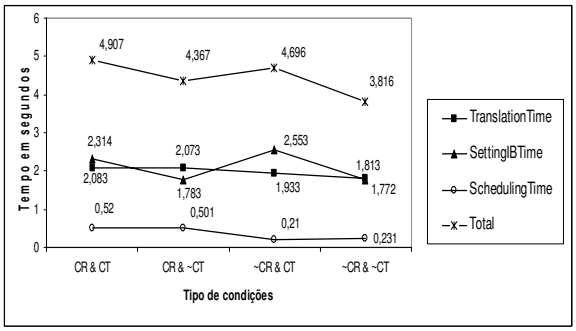

Figura 8 – Medições realizadas no PDP

Para o processo de escalonamento, as condições de tempo deveriam apresentar maior influência, já que implicam na verificação das datas e horas iniciais e finais do prazo de validade da política, enquanto que a ausência faz com que a política seja considerada sempre válida e nenhuma outra verificação adicional precisa ser feita. Nos testes, no entanto, a inclusão de tempo não alterou o tempo de tradução.

Além dos tempos medidos para cada bloco, foi contabilizado um tempo total para execução do sistema (figura 9). A diferença se dá porque na medição do tempo das tarefas principais não foram contabilizados os tempos de execução de ações

intermediárias ou secundárias, como o tempo que o servidor de mensagens leva para entregar as notificações aos módulos e o tempo de rede (“Outros”–figura 10).

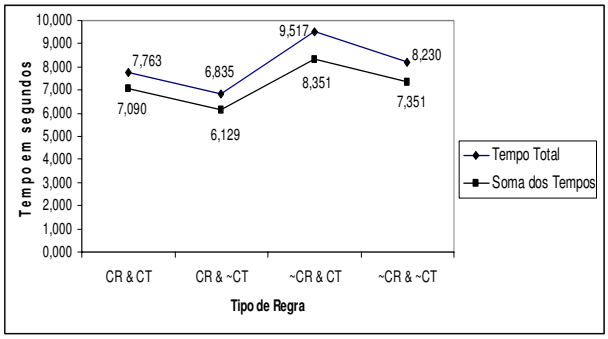

Figura 9 –Tempo total X soma dos módulos

A figura 10 apresenta um comparativo entre os tempos totais de cada módulo. Por ser o elemento de mais alta complexidade da arquitetura, envolvido na realização de um maior número de tarefas, o PDP consumiu mais da metade do tempo global.

Em seguida, por estar envolvido com o cadastro de políticas no repositório e pelo fato do LDAP ser reconhecidamente eficaz em requisições de buscas, mas de desempenho degradado face à requisições de adição, remoção ou alteração, o PMT foi o segundo bloco a consumir mais tempo no processo global.

O PEP, por ser um módulo simples, com poucas tarefas a serem executadas nessa versão do framework, consumiu menos tempo, como já era esperado.

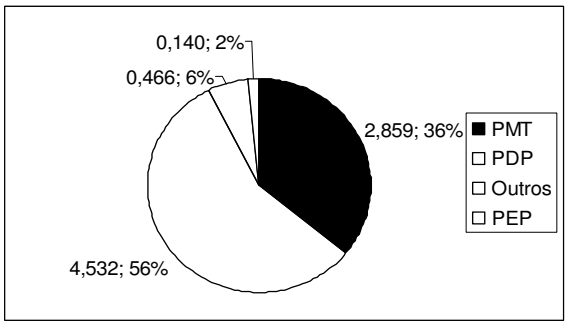

Figura10 –Tempos totais de cada módulo

8. Conclusão

Nesse artigo foi apresentado um framework para gerenciamento de QoS e segurança, em diversos tipos de rede, baseado em políticas. Foram implementados todos os módulos definidos pela arquitetura PBM do DMTF/IETF, sendo eles: PMT (*Policy Management Tool*), PR (*Policy Repository*), PDP (*Policy Decision Point*) e PEP (*Policy Enforcement Point*).

Toda a implementação foi feita em Java e como protocolo de comunicação entre os módulos foi utilizado principalmente o SNMP e em alguns casos de envio de notificações foi utilizado o JMS.

O modelo de informações trabalhado na implementação foi o PCIM, definido pelo DMTF/IETF. Foi elaborada toda a estruturação das informações no modelo

PCIM e o mapeamento das informações do PCIM para o modelo de informações do LDAP (repositório de políticas).

Como estudo de caso foi utilizada uma rede sem fio 802.11. O objetivo era prover QoS e segurança nessa rede através de políticas. Foi feito todo um estudo sobre a aplicação de QoS e segurança nesse tipo de rede. No entanto, o AP utilizado não suportava as definições para gerenciamento de qualidade de serviço do padrão 802.11e, o que limitou a configuração de parâmetros de QoS.

A importância desse trabalho está no fato de se ter um framework para gerenciamento de políticas que pode ser adaptado para gerenciar diversos tipos de redes, como por exemplo, redes óticas ou grades computacionais. Por se tratar de um framework, os módulos podem ser adaptados para cada tipo de rede.

9. Referências

[DRAFT_ LDAP] Policy Core Extension Lightweight Directory Access Protocol Schema. Network Working Group. INTERNET-DRAFT, Junho, 2004.

[DRAFT_PmMIB] Policy Based Management MIB Draft-IETF-SNMPCONF-PM-15, maio de 2004.

[JMS_SPECIF] Sun Microsystems, “JavaTM Message Service Specification”

[OpenLDAP] OpenLDAP. Página da World Wide Web. url: http://www.openldap.org, 30 de dezembro de 2004.

[RFC_LDAP] Strassner, J., Moore, B., Moats, R., Ellesson, E., “Policy Core Lightweight Directory Access Protocol (LDAP) Schema”, Request For Comments - RFC 3703, fevereiro de 2004.

[RFC_PCIM] Moore, B., Ellesson, E., Strassner, J., Westerinen, A., “Policy Core Information Model -- Version 1 Specification”. Request For Comments - RFC 3060, fevereiro de 2001.

[RFC_PCIM_EXT] Moore, B., “Policy Core Information Model (PCIM) Extensions”. Request For Comments - RFC 3460, Janeiro, 2003.

[WALDBUSSER] Waldbusser, S., Saperia, J. “Policy Based Management MIB”. draft-ietf-snmpconf-pm-15. Maio de 2004.