Um Sistema para Gerência de Redes Baseado em Políticas utilizando Tecnologia JINI

Helcio Wagner da Silva e Luís Geraldo P. Meloni

***Resumo*— Ao longo dos anos, iniciativas vêm sendo feitas no âmbito da IETF com vistas à flexibilização e distribuição das atividades de gerência. Este artigo apresenta um sistema para gerência de redes que incorpora algumas destas iniciativas e utiliza ainda uma tecnologia especificamente projetada para a construção de sistemas distribuídos. O resultado é um sistema ainda mais flexível e capaz de auto-resiliência nas suas operações, notadamente na ocorrência de falhas parciais.**

***Palavras-Chave*— Gerência de Redes, JINI, Sistemas Distribuídos, MIB de Evento, Gerência Baseada em Políticas.**

***Abstract*— For the past years, efforts, on the IETF, have been done to achieve higher flexibility and distributivity on management tasks. This article presents a network management system that incorporates some of those efforts and uses a technology specifically designed for distributed systems. The result is an even more flexible system able to perform self-resilience on the operations, especially at the occurrence of partial failures.**

***Keywords*— Network Management, JINI, Distributed Systems, Event MIB, Policy-based Network Management.**

I. Introdução

A maioria dos NMSs (*Network Management Systems*) desenvolvidos atualmente para a gerência de redes TCP/IP é baseada no modelo tradicional definido pela IETF. Este modelo é composto pela especificação de um protocolo (SNMP - *Simple Network Management System*) e de uma sintaxe para a formatação de MOs (*Managed Objects*) em MIBs (*Management Information Bases*). Esta última especificação é conhecida como SMI (*Structure of Management Information*).

Ao longo dos anos, novas funcionalidades têm sido incorporadas a esse modelo. A mais recente versão do SNMP, o SNMPv3, e o projeto de novas MIBs, como a MIB de Evento por exemplo, refletem esta evolução dentro do modelo. Além disso, novos modelos vêm sendo propostos, como a Gerência Baseada em Políticas por exemplo. Independentemente de quais sejam suas naturezas, todas elas visam prover flexibilidade, distribuição e segurança nas operações de gerência.

Apesar de todas as alternativas propostas pela IETF, na construção de NMSs alguns pontos permanecem em aberto, notadamente aqueles relacionados à configuração e manutenção destes sistemas.

Nesse artigo, é exposto um NMS construído com base na tecnologia JINI que, além de absorver as funcionalidades surgidas ao longo dos anos para a gerência de redes, ainda usufrui dos benefícios providos por JINI para a construção de sistemas distribuídos flexíveis e auto-resilientes.

Helcio Wagner da Silva e Luís Geraldo P. Meloni¸ Departamento de Comunicações, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, Brasil, E-mails: helcio@decom.fee.unicamp.br, meloni@decom.fee.unicamp.br.

A fim de melhor descrevê-lo, este artigo é estruturado da seguinte forma: na Seção II são citadas algumas iniciativas surgidas no âmbito da IETF para a gerência de redes. Em seguida, a Seção III resume os fundamentos da tecnologia JINI. A Seção IV aborda as considerações gerais que devem ser levadas em conta para o projeto de NMSs utilizando JINI. A Seção V é a responsável pela apresentação do NMS construído, enquanto a Seção VI efetua as considerações finais acerca do trabalho realizado.

II. Iniciativas da IETF para a Gerência de Redes

Conforme comentado na Seção anterior, o modelo tradicional para a gerência de redes TCP/IP é composto por um protocolo e uma sintaxe para formatação e armazenamento de informações bem definidos.

Ao longo dos anos, este modelo tem incorporado uma série de melhorias, que podem ser divididas em melhorias na estrutura do protocolo propriamente dito e no projeto de novas MIBs. Além disso, novos paradigmas têm surgido com o intuito de melhorar a flexibilidade, a distribuição e a segurança das atividades de gerência de redes, quer seja incorporando o modelo atualmente em vigência, quer mesmo substituindo-o.

Ao longo de sua existência, o SNMP tem recebido revisões que proporcionaram uma relativa capacidade de descentralização e um satisfatório grau de segurança em suas operações. A versão atual do protocolo, denominada SNMPv3 [1], permite comunicação entre Gerentes e é capaz de prover autenticação, autorização e privacidade nas comunicações.

Por outro lado, iniciativas relacionadas ao projeto de novas MIBs foram deflagradas com o intuito de descentralizar ainda mais a atividade de gerência. Uma destas iniciativas é a MIB de Evento [2], desenvolvida no âmbito do *DISMAN (DIStributed MANagement) WG*. Esta MIB possibilita a monitoração de MOs e a definição de eventos que ocorrem quando gatilhos são disparados com base em testes realizados naqueles MOs. Estes eventos podem consistir da realização de *Sets* e/ou o envio de Notificações SNMP.

Uma das metodologias alternativas criadas no âmbito da IETF para a gerência de redes é a denominada Gerência Baseada em Políticas, especificada pelo *Policy Framework WG*, e cujos fundamentos encontram-se descritos em [3]. Apesar de surgida num contexto da gerência do controle de admissão aos recursos de rede, essa metodologia pode seguramente ser aplicada em outros problemas de gerência mais gerais.

Nesta metodologia, Políticas podem ser representadas por meio do PCIM (*Policy Core Information Model*), especificado

em [4]. Neste modelo, o bloco de construção básico é a Regra de Política. Ela é a ligação de um conjunto de Ações de Política a um conjunto de Condições de Política. As Condições são avaliadas para determinar se as Ações são ou não realizadas.

Os principais componentes que fazem parte de um PNMS (*Policy-based Network Management System*) são denominados de PMA (*Policy Management Application*), PDP (*Policy Decision Point*) e PEP (*Policy Enforcement Point*).

A PMA provê a interface para um administrador de rede criar e empregar Regras de Política, armazenando-as num Repositório de Políticas. O PDP é uma entidade lógica que extrai as Regras de Política do Repositório e toma decisões por ele próprio ou por outros elementos de rede que solicitam tais decisões. Essas decisões são representadas pela avaliação do conjunto de Condições das Regras de Política. O PEP é uma entidade lógica que reforça decisões de Políticas. Este reforço é representado pela execução do conjunto de Ações das Regras de Política.

Uma sugestão para protocolo de acesso ao Repositório de Políticas que vem tendo uma razoável aceitação é o LDAP (*Lightweight Directory Access Protocol*), especificado em [5]. De fato, a IETF especifica em [6] um mapeamento do PCIM para um *schema* LDAP, denominado PCLS (*Policy Core LDAP Schema*).

Uma sugestão para protocolo de transferência de decisões de Políticas entre PDPs e PEPs é o COPS (*Common Open Policy Service*). No entanto, outros protocolos podem ser utilizados para esta finalidade, e dentre eles está o SNMP.

Um detalhe importante nesse caso, porém, é que nenhum protocolo é sugerido para a comunicação entre PMAs e PDPs, ou entre PDPs cooperativos. Em geral, nenhum detalhe de implementação, tal como distribuição, plataforma, protocolo ou linguagem, é prescrito.

III. Tecnologia JINI

JINI é uma tecnologia desenvolvida pela Sun Microsystems para a construção de sistemas distribuídos auto-resilientes, isto é, sistemas que necessitem o mínimo possível de intervenção humana para funcionarem a contento, principalmente na ocorrência de falhas parciais.

Descrita arquiteturalmente em [7], JINI é fortemente baseada no uso da tecnologia $Java^{tm}$ e, em termos práticos, pode ser vista como uma infra-estrutura e um modelo de programação que suprem os requisitos fundamentais acerca de como Clientes e Serviços descobrem e conectam-se uns aos outros para formar uma Comunidade.

Uma Comunidade é constituída por Serviços JINI, e Serviços carregam consigo *proxies* que provêem todo o código necessário para que Clientes interajam com eles. Estes *proxies* são obtidos dinamicamente por Clientes mediante comunicação com um Serviço denominado LUS (*LookUp Service*).

O registro de *proxies* junto ao LUS é mantido segundo uma disciplina de *leasing*. Isso tem por fim minimizar os efeitos de falhas parciais e possibilitar a auto-configuração da Comunidade, pois quando a execução de um Serviço é abrupta ou suavemente interrompida o *lease* associado ao registro de seu *proxy* não é mais renovado, e assim o *proxy* é removido do LUS, não estando mais disponível a qualquer Cliente em potencial e eliminando a necessidade de administração humana para remover registros antigos.

Um dos aspectos importantes da tecnologia JINI é a sua neutralidade de protocolo. De fato, não há nenhuma definição estrita do protocolo a ser utilizado na comunicação entre Clientes e Serviços JINI. Isso permite que seja utilizado o protocolo mais adequado em cada cenário de aplicação, o que provê maior flexibilidade às aplicações envolvidas.

Essa flexibilidade na utilização de protocolos é importante, haja visto que o projeto de um protocolo é frequentemente um *trade-off* entre generalidade e eficiência. Em sistemas centralizados em torno de um único protocolo, as decisões tendem a favorecer a generalidade. Assim, por mais que se projete um protocolo para atender a todos os cenários atualmente concebíveis, esse protocolo ainda é passível de ter um desempenho inferior a um protocolo específico a um daqueles possíveis cenários, ou até mesmo ser incapaz de atender a um cenário futuro.

Mecanismos de geração e manipulação de eventos remotos também são provido pela tecnologia JINI. A partir da definição de um tipo de evento remoto, aplicações podem ser projetadas de forma a gerarem eventos daquele tipo e/ou serem notificadas acerca dos mesmos. O LUS, por exemplo, permite às aplicações interessadas serem notificadas acerca de eventos tais como o registro de um novo Serviço que satisfaça certos requisitos.

O mecanismo de manipulação de eventos se dá mediante o registro de Ouvidores de Eventos junto às aplicações que os geram. E, de forma a não desperdiçar recursos registrando-se Ouvidores para aplicações que outrora experimentaram uma interrupção abrupta ou suave em sua execução, o mecanismo de notificação de eventos remotos providos por JINI também faz uso de *leasing*.

Versões mais recentes de JINI possuem um Modelo de Segurança que pode ser empregado nas interações entre Clientes e Serviços. Em particular, Serviços podem impor um conjunto extensível de restrições em seus *proxies*, e URLs HTTPMD (*HTTP Message Digest*) podem ser utilizadas de forma a prover garantias acerca da integridade do código do próprio *proxy*, descarregado dinamicamente da rede.

Comparações entre JINI e outras tecnologias são realizadas em [8] e [9], ambas apontando JINI como tecnologia preferencial na construção de aplicações distribuídas.

IV. Considerações gerais sobre a utilização de JINI na construção de NMSs

As considerações gerais sobre a utilização de JINI na construção de NMSs foram apresentadas inicialmente em [10]. De forma resumida, estas considerações são:

- a associação de Serviços JINI a Sistemas Gerenciados;
- o agrupamento de Sistemas Gerenciados em Domínios Gerenciados;
- a incorporação de Interfaces com o Usuário para os Sistemas Gerenciados em seus respectivos *proxies*;

- o provisionamento de interações seguras entre Aplicações de Gerência e Sistemas Gerenciados.

Na associação de Serviços JINI a Sistemas Gerenciados, vários níveis de granularidade podem ser utilizados, indo da associação de um único Serviço JINI a um Sistema Gerenciado até a associação de múltiplos Serviços JINI a esse mesmo Sistema. Um benefício imediato dessa abordagem é que, independentemente de qual seja o nível utilizado na modelagem, virtualmente qualquer protocolo pode ser empregado na comunicação entre a Aplicação de Gerência e o Sistema Gerenciado, ao contrário da abordagem tradicionalmente utilizada.

No agrupamento de vários Sistemas Gerenciados (modelados como Serviços JINI) em torno de um Domínio Gerenciado, uma Aplicação de Gerência pode usufruir do mecanismo de auto-resiliência proporcionado pela tecnologia JINI para manter uma visão consistente dos Sistemas ativos sob a sua responsabilidade.

Na incorporação de Interfaces com o Usuário nos *proxies* para os Sistemas Gerenciados, uma Aplicação de Gerência pode prescindir do código necessário para a gerência humanamente amigável do Sistema, já que esse código pode ser descarregado dinamicamente a partir do seu *proxy*. Inclusive, ao contrário das abordagens tradicionais, uma multiplicidade de Interfaces com o Usuário pode ser empregada (interfaces gráficas diversas, interfaces de voz, etc). Esta é uma vantagem observada até mesmo frente a outras propostas alternativas, como a de embutir-se Servidores Web em Sistemas Gerenciados [11].

Na provisão de interações seguras entre Aplicações de Gerência e Sistemas Gerenciados, mecanismos de segurança plugáveis podem ser empregados prevendo-se vários cenários distintos sob o ponto de vista de segurança. Basta que restrições relacionadas à autenticação e privacidade sejam impostas no *proxy* para a o Sistema Gerenciado e URLs HTTPMD sejam empregadas.

V. O PNMS DESENVOLVIDO

Conforme foi visto na Seção II, não há nenhum protocolo estritamente definido para a comunicação entre PMAs e PDPs no modelo de Gerência Baseada em Políticas.

Por outro lado, foi comentado na Seção III que sistemas distribuídos construídos com base em JINI não possuem nenhuma necessidade de ater-se programaticamente a nenhum protocolo específico.

Além disso, viu-se também que esses sistemas beneficiam-se das caracteristicas-chave da tecnologia para maximizar a automação de suas operações, minimizando assim a possibilidade de intervenção humana na ocorrência de falhas parciais.

A combinação desses dois fatores serve como motivação para o estudo de quais são os efeitos observáveis quando JINI é utilizada na construção de PNMSs. Para tal, desenvolveu-se o protótipo mostrado nas Subseções seguintes.

A. Visão geral

A Figura 1 ilustra uma visão geral do protótipo de PNMS desenvolvido. Nele, foram seguidas as considerações enumeradas na Seção anterior e o PDP foi modelado como um

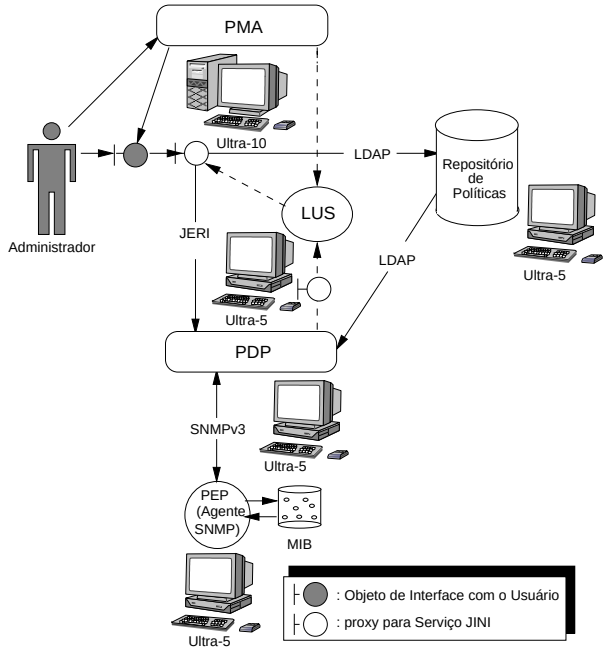

Fig. 1. Visão geral do PNMS desenvolvido.

Serviço JINI. Conforme abordado anteriormente, esta medida de projeto garante a flexibilidade na adoção do protocolo utilizado nas comunicações entre a PMA e o PDP, assim como contribui para a auto-resiliência do PNMS como um todo.

Também é notável na Figura 1 a incorporação de um Objeto de Interface com o Usuário ao *proxy* para o PDP, a partir do qual o usuário da PMA pode realizar a gerência daquele componente.

As interações entre a PMA e o PDP são realizadas utilizando-se JERI, uma nova implementação do modelo de programação RMI. São aplicadas restrições relativas à obrigatoriedade de autenticação e privacidade nas comunicações entre a PMA e o PDP. Ademais, são utilizadas URLs HTTPMD de forma a garantir a integridade do código descarregado dinamicamente através da infra-estrutura de rede.

O Repositório de Políticas foi representado pelo *daemon* `slapd` incluso na versão 2.1.25 do *OpenLDAP*, uma implementação de código aberto do LDAP.

O PEP foi representado pelo *daemon* `snmpd` incluso na versão 5.2.1 do *NET-SNMP*, um conjunto de ferramentas desenvolvido para o uso e implementação do SNMP em IPv4 e IPv6. Neste protótipo, as interações entre o PDP e o PEP ocorrem através da troca de Mensagens SNMPv3, o PDP criando gatilhos na MIB de Evento suportada pelo `snmpd` e eventualmente recebendo PDUs `Trap` quando estes gatilhos são disparados. Esta é uma abordagem diferente de outras que utilizam COPS/COPS-PR para aquele tipo de interação, como ocorre em [12] e [13], por exemplo. A escolha do SNMP e da MIB de Evento para este protótipo deu-se, na verdade, pela maior disponibilidade destes componentes em relação às demais opções.

As implementações do PDP, PEP e do Repositório de Políticas são executadas a partir de estações Sun Ultra-5, equipadas com processadores operando a 270 MHz e possuindo

128 MBytes de RAM. Já a PMA é executada a partir de uma estação Ultra 10 equipada com um processador operando a 300 MHz e possuindo 256 MBytes de RAM. Estas plataformas são equipadas com a versão 9 do Sistema Operacional Solaris.

B. O PDP desenvolvido

O PDP foi desenvolvido utilizando a versão 1.4.2_04 da J2SE ($Java^{tm}$ *2 Platform, Standard Edition*) e a versão 2.0_002 do JTSK (*JINI Technology Starter Kit*). Na construção deste componente, fez-se uso da API JAAS ($Java^{tm}$ *Authentication and Authorization Service*) de forma que apenas um usuário devidamente autenticado pudesse ter acesso à sua configuração.

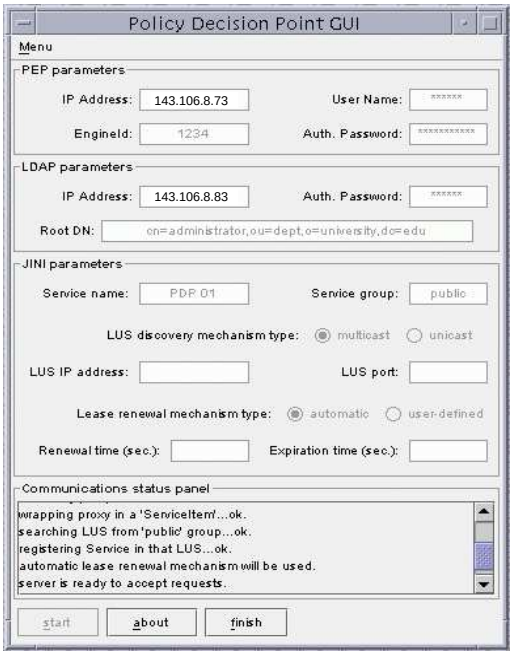

Fig. 2. GUI para o PDP desenvolvido.

A Figura 2 ilustra a GUI para o PDP desenvolvido, particularmente num momento no qual ele conclui sua adesão a um determinado Domínio Gerenciado. Conforme pode ser observado, para a sua inicialização é necessário configurar-se parâmetros alusivos ao PEP (Agente SNMPv3) com o qual irá comunicar-se, ao Repositório de Políticas (Servidor LDAP) a partir do qual coletará Regras de Política colocadas previamente pela PMA, e, finalmente, ao LUS responsável pelo Domínio Gerenciado ao qual irá pertencer.

Para a comunicação com o Agente SNMPv3, são necessários o endereço IP do Agente, o *User Name* a ser utilizado nas Mensagens trocadas, uma senha de autenticação e o *Engine ID* a ser utilizado pelo PDP para a recepção de eventuais notificações acerca de gatilhos disparados no Agente.

Para a comunicação com o Servidor LDAP, são necessários o endereço IP do Servidor, uma senha de autenticação e um DN (*Distinguished Name*) a partir do qual as Regras de Política serão buscadas no Servidor.

Para a comunicação como LUS, são necessários o nome administrativo do PDP e o nome do Domínio Gerenciado ao qual deseja-se que ele pertença. São também disponibilizadas as opções de escolha para o método a ser utilizado para encontrar-se o LUS (*multicast* ou *unicast*) e o método de renovação do *lease* junto àquele Serviço (renovação automática ou com parâmetros definidos pelo usuário).

O PDP, como qualquer Serviço JINI, implementa uma determinada interface. Em particular, a interface implementada no protótipo é ilustrada abaixo:

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface PDPService extends Remote {
  public void activate(String ruleName)
    throws RemoteException;
  public void deactivate(String ruleName)
    throws RemoteException;
  public String[] list() throws RemoteException;
}
```

Conforme observado a partir desta interface, o PDP provê suporte para a ativação ou a desativação de uma determinada Regra de Política cujo nome administrativo é passado como parâmetro, bem como para a obtenção de uma listagem das Regras de Política nele atualmente ativas (isto é, Regras para as quais existem entradas nas tabelas da MIB de Evento no Agente subordinado ao PDP).

C. A PMA desenvolvida

À semelhança do que foi comentado com relação ao PDP, a PMA também foi desenvolvida utilizando a versão 1.4.2_04 da J2SE e a versão 2.0_002 do JTSK. Além disso, também foi utilizada a API JAAS para a garantia de que apenas um usuário devidamente autenticado pudesse ter acesso à sua configuração.

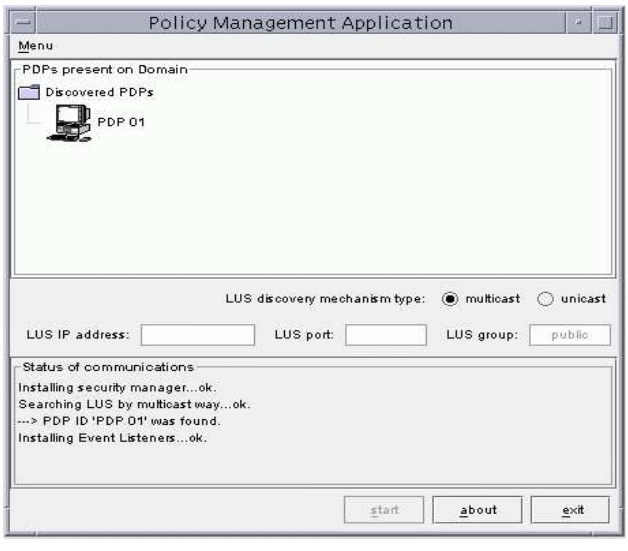

Fig. 3. GUI para a PMA desenvolvida.

A Figura 3 ilustra a GUI para a PMA desenvolvida, particularmente num momento no qual ela conclui sua inicialização. Para que isso ocorra, é necessário o fornecimento do nome de um Domínio Gerenciado e a escolha do método a ser utilizado para efetuar-se a busca pelo LUS responsável por esse Domínio.

A inicialização da PMA compreende a busca do LUS responsável pelo Domínio Gerenciado fornecido, a busca imediatas de PDPs que já pertencem a este Domínio e a instalação de Ouvidores de Evento junto ao LUS que denunciam a entrada e/ou a saída de PDPs do Domínio.

Na GUI mostrada na Figura 3, o Domínio buscado já contém um PDP associado. Este PDP é representado na tela por um ícone localizado numa estrutura em árvore da GUI, esta estrutura destinada a abrigar todos os os PDPs presentes naquele Domínio.

D. Exemplo de Operação

As operações de gerência neste PNMS compreendem fundamentalmente a invocação remota dos métodos pertencentes à interface disponibilizada pelo PDP, a qual foi descrita na Subseção V-B. Para que isto ocorra, o usuário da PMA deve selecionar um dos PDPs presentes no Domínio e obter dinamicamente a interface (gráfica) para gerenciá-lo.

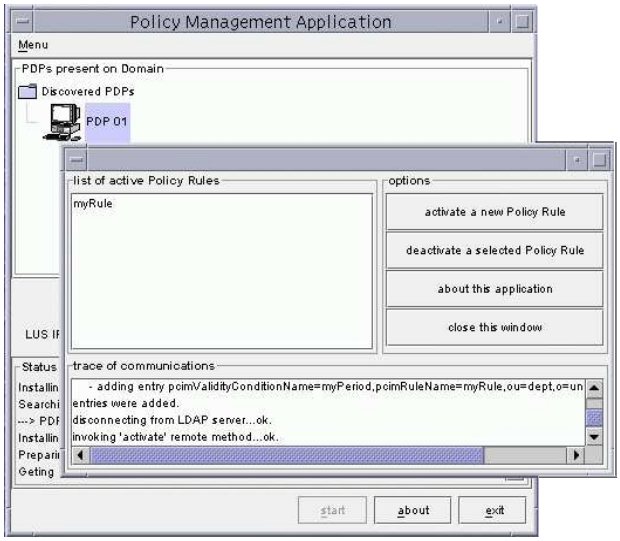

Fig. 4. GUI descarregada dinamicamente para a gerência do PDP.

A Figura 4 ilustra a GUI que é descarregada dinamicamente para a gerência do PDP, particularmente num momento no qual o método `activate` pertencente à interface do PDP é invocado.

No PNMS desenvolvido, uma Regra de Política é inicialmente expressa utilizando-se a XML (*eXtensible Markup Language*) e seguindo-se uma determinada DTD (*Document Type Definition*). Esta Regra de Política pode ser criada a partir de um editor de texto qualquer, sendo armazenada num arquivo no disco da máquina na qual a PMA reside.

Como exemplo, considere-se a seguinte Regra de Política:

Durante o período que vai das 08:00:00 do dia 15 de abril de 2005 até às 18:00:00 do dia 15 de maio de 2005, checar o valor do MO 'ifInOctets' para cada interface do Sistema Gerenciado a cada 1 (um) minuto. Caso a diferença entre dois valores consecutivos ultrapasse 30, enviar uma notificação para o 'host' 143.106.8.78 contendo o valor do MO 'sysUpTime' do Sistema Gerenciado.

Esta Regra de Política é assim inicialmente expressa:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE policy SYSTEM "policy.dtd">

<!-- definition of a Policy example -->
<policy name="myRule" status="enabled"
   conditionListType="dnf" priority="1">
  <!-- definition of a Policy condition -->
  <condition name="myCondition" groupNumber="1"
    negated="false">
    <objectTarget wildcarding="true">
      <objectTargetID>ifInOctets</objectTargetID>
      <objectTargetType>Counter</objectTargetType>
    </objectTarget>
    <test frequency="60" delta="true">
      <threshold type="rising">
        <thresholdValue>30</thresholdValue>
      </threshold>
    </test>
  </condition>
  <!-- definition of a Policy action -->
  <action name="myAction" order="1">
    <notification destination="143.106.8.78">
      <objectNotif wildcarding="false">
        <objectNotifID>sysUpTime</objectNotifID>
        <objectNotifType>TimeTicks</objectNotifType>
      </objectNotif>
    </notification>
  </action>
  <!-- definition of a time Policy condition -->
  <timePeriod name="myPeriod">
    <initial>
      <year>2005</year>
      <month>04</month>
      <day>15</day>
      <hours>08</hours>
      <minutes>00</minutes>
      <seconds>00</seconds>
    </initial>
    <final>
      <year>2005</year>
      <month>05</month>
      <day>15</day>
      <hours>18</hours>
      <minutes>00</minutes>
      <seconds>00</seconds>
    </final>
  </timePeriod>
</policy>
```

Para ativar uma Regra de Política, procede-se ao carregamento do seu respectivo arquivo XML e à sua conversão em Objetos PCLS. A Regra de Política mostrada acima, após a conversão, é mostrada abaixo no formato LDIF (*LDAP Data Interchange Format*).

```
dn: pcimRuleName=myRule,ou=dept,o=university,dc=edu
objectclass: top
objectclass: dlm1ManagedElement
objectclass: pcimPolicy
objectclass: pcimRule
objectclass: pcimRuleInstance
pcimRuleName: myRule
pcimRuleEnabled: 1
pcimRuleConditionListType: 1
pcimRulePriority: 1
pcimRuleConditionList: pcimConditionName=myCondition,
pcimRuleName=myRule,ou=dept,o=university,dc=edu
pcimRuleValidityPeriodList: pcimValidityConditionName=
myPeriod,pcimRuleName=myRule,ou=dept,o=university,dc=edu
pcimRuleActionList: pcimActionName=myAction,pcimRuleName=
myRule,ou=dept,o=university,dc=edu

dn: pcimConditionName=myCondition,pcimRuleName=myRule,
ou=dept,o=university,dc=edu
objectclass: top
objectclass: dlm1ManagedElement
objectclass: pcimPolicy
objectclass: pcimRuleConditionAssociation
objectclass: pcimConditionAuxClass
objectclass: myConditionAuxClass
pcimConditionName: myCondition
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE
objectID: ifInOctets
objectType: Counter
wildcarding: TRUE
testType: 3
```

```
samplingType: 2
samplingFreq: 60
targetValue: 30
thresholdType: 1

dn: pcimValidityConditionName=myPeriod,pcimRuleName=myRule,
ou=dept,o=university,dc=edu
objectclass: top
objectclass: dlm1ManagedElement
objectclass: pcimPolicy
objectclass: pcimRuleValidityAssociation
objectclass: pcimTPCAuxClass
pcimValidityConditionName: myPeriod
pcimTPCTime: 20050415T080000/20050515T180000

dn: pcimActionName=myAction,pcimRuleName=myRule,ou=dept,
o=university,dc=edu
objectclass: top
objectclass: dlm1ManagedElement
objectclass: pcimPolicy
objectclass: pcimRuleActionAssociation
objectclass: pcimActionAuxClass
objectclass: myActionAuxClass
pcimActionOrder: 1
pcimActionName: myAction
eventType: 2
destination: 143.106.8.78
objectNotifID: sysUpTime
objectNotifType: TimeTicks
```

Uma vez carregada e convertida em Objetos PCLS, a Regra de Política é escrita num Repositório de Políticas. Em seguida, uma chamada remota ao método `activate` é realizada automaticamente junto ao *proxy* para o PDP.

Quando o PDP recebe esta invocação, ele procede imediatamente à busca da Regra de Política (cujo nome é passado como parâmetro do método) junto ao Repositório de Políticas.

Conforme comentado na Subseção V-A, este PNMS faz uso do protocolo SNMP e da MIB de Evento para o reforço da Regra de Política junto a um PEP. Por isso, na implementação do método `activate` no PDP, os Objetos PCLS coletados no Repositório de Políticas são convertidos em MOs conformantes com a SMI e em informações alusivas aos períodos segundo os quais a Regra deve ser reforçada.

A ativação da Regra de Política tomada como exemplo corresponde inicialmente à criação de um gatilho, adicionando uma nova entrada na tabela *mteTriggerTable*. Em seguida, é criada uma nova entrada na tabela *mteEventTable* para o tipo de ação a ser realizada quando o gatilho é disparado, que neste caso é o envio de uma PDU *Trap*. Assim, é adicionada uma nova entrada na tabela *mteEventNotificationTable*, que passa a conter uma lista de MOs correspondente à entrada na tabela *mteEventTable*.

Uma vez definidos os parâmetros gerais do gatilho e configuradas as ações a serem levadas a cabo na ocorrência de um evento pelo disparo daquele gatilho, são definidos então seus parâmetros específicos. Dependendo do tipo de teste a ser realizado com o MO monitorado, o gatilho recebe a denominação de gatilho de existência, de valor-limite ou gatilho *booleano*. No caso da Regra de Política tomada como exemplo, trata-se de um gatilho de valor-limite. Assim, a próxima (e última) tabela que recebe uma nova entrada é a tabela *mteTriggerThresholdTable*.

No protótipo desenvolvido, o *schema* PCLS foi incorporado ao conjunto de *schemas* suportados pelo `slapd`. Além disso, um novo *schema* foi desenvolvido para dar suporte ao mapeamento entre Objetos conformantes com o PCLS e MOs conformantes com a SMI, de forma que o PDP pudesse estabelecer qualquer um dos tipos de testes previstos na especificação da MIB de Evento.

A desativação de uma determinada Regra de Política, iniciada a partir da invocação do método `deactivate`, é implementada com a eliminação de suas respectivas entradas nas tabelas da MIB de Evento. Já a listagem de Regras de Política atualmente ativas no PDP, iniciada a partir da invocação do método `list`, é implementada a partir da consulta aos registros internos daquela aplicação.

VI. Conclusões

Este artigo apresentou um NMS desenvolvido com base em técnicas inovadoras. Uma destas técnicas é a combinação de iniciativas surgidas paralelamente no âmbito da IETF para flexibilizar e descentralizar a gerência de redes.

Em particular, foram combinadas uma das iniciativas do *DISMAN WG* (a MIB de Evento) e a iniciativa do *Policy Framework WG* (o paradigma de Gerência Baseada em Políticas) para a construção de um PNMS cujo reforço de Políticas junto ao PEP é viabilizada segundo a monitoração de MOs e a definição de eventos que ocorrem quando gatilhos são disparados com base em testes realizados naqueles MOs.

Outra técnica inovadora empregada na construção do NMS foi a utilização da tecnologia JINI nas interações entre PMA e PDP. O efeito da aplicação desta técnica é um PNMS flexível e auto-resiliente, no qual a PMA mantém uma visão consistente sobre seus PDPs subordinados e é capaz de comunicar-se com estes componentes utilizando diversos tipos de Interface com o Usuário, empregando virtualmente qualquer protocolo.

Referências

[1] D. Harrington, R. Presuhn, and B. Wijnen. *An Architecture for Describing SNMP Management Frameworks - rfc 2271*, Janeiro de 1998.

[2] Ramanathan Kavasseri and Bob Stewart. *Event MIB - rfc2981*, Outubro de 2000.

[3] R. Yavatkar, D Pendarakis, and R Guerin. *A Framework for Policy-based Admission Control*, Janeiro de 2000.

[4] B. Moore, E. Ellesson, J Strassner, and A. Westerinen. *Policy Core Information Model – Version 1 Specification (rfc 3060)*, Fevereiro de 2001.

[5] M. Wahl, T. Howes, and S. Kille. *Lightweight Directory Access Protocol (v3)*, Dezembro de 1997.

[6] John Strassner, Bob Moore, Ryan Moats, and Ed Ellesson. *Policy Core Lightweight Directory Access Protocol (LDAP) Schema - rfc 3703*, Fevereiro de 2004.

[7] Sun Microsystems. $Jini^{tm}$ *Architecture Specification - Version 2.0*, Junho de 2003.

[8] Lai Olstad, Javier Ramirez, Clint Brady, and Bruce McHollan. Jini technology: Impromptu networking and its impact on telecommunications. In *Proceedings of Capstone 1999, University of Colorado at Boulder*, 1999.

[9] S. Raza, B. Pagurek, and T. White. Distributed computing for plug-and-play network service configuration. In *Proceedings of the 2000 IEEE/IFIP Network Operations and Management Symposium (NOMS'2000), Honolulu, Hawaii, EUA*, 10-14 Abril de 2000.

[10] H. Silva and L. Meloni. Using jini technology for building of network management systems. In *Proceedings of the IADIS International Conference Applied Computing 2004, Lisboa, Portugal*, Março de 2004.

[11] Barry Bruins. Some experiences with emerging management technologies. *The Simple Times - The Quarterly Newsletter of SNMP Technology, Comment, and Events*, 4(3), Julho de 1996.

[12] Juha Majalainen. Implementation of policy management tool for bandwidth provisioning. Master's thesis, Tampere University of Technology - Department of Information Technology, Abril de 2002.

[13] S. Jha and M. Hassan. Java implementation of policy-based badwidth management. *International Journal of Network Management*, 13(4):249–258, Julho de 2003.