Um Protocolo Atômico para Aplicação de Políticas de Gerenciamento em Redes com QoS

Lisandro Zambenedetti Granville, Rodrigo Sanger Alves, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{granville,sanger,janilce,liane}@inf.ufrgs.br

Abstract. *This paper introduces a protocol for the atomic deployment of management policies for computer networks with quality of services (QoS) support. Atomic protocols are needed when the deployment of a management policy partially works, configuring just a subset of the intended target devices, which is not enough to provide proper QoS for network users. Besides, policy partial deployment leads to the waste of network resources that could be better used. In addition to the protocol definition, this paper presents the protocol implementation in a policy-based network management architecture used to manage differentiated services (DiffServ) networks.*

Resumo. *Este artigo apresenta um protocolo para aplicação atômica de políticas de gerenciamento em redes com qualidade de serviço (QoS). A necessidade de um protocolo atômico vem do fato de que aplicações de políticas que funcionam parcialmente não são suficientes para o fornecimento adequado de QoS aos usuários. Além disso, aplicações parciais acabam por desperdiçar recursos que poderiam ser melhor utilizados. Além da definição do protocolo, este artigo apresenta a implementação do mesmo em uma arquitetura de gerenciamento baseada em políticas voltada para o gerenciamento de rede com serviços diferenciados (DiffServ).*

1. Introdução

O gerenciamento baseado em políticas (PBNM) (Westerinen et al. 2001) é uma abordagem efetiva para o controle de redes com suporte à qualidade de serviço (QoS) [Flegkas et al. 2002]. Ainda que diversas arquiteturas PBNM tenham sido propostas nos anos recentes [Tsarouchis et al.2003] [Nikolakis et al.2004] [Guo et al. 2003], a arquitetura de políticas do IETF (*Internet Engineering Task Force*) é, provavelmente, uma das mais largamente conhecidas. Na verdade, tal arquitetura não é formalmente definida nos documentos do IETF, mas tem uma larga aceitação de seus elementos básicos, cujas funcionalidades são bem conhecidas. Uma *ferramenta de políticas* é utilizada pelo gerente da rede para definir políticas de QoS que são armazenadas em um *repositório de políticas* para futuro reuso e/ou aplicação. Os *Policy Decision Points* (PDPs), distribuídos pela rede gerenciada, são responsáveis por receber tais políticas e traduzi-las em ações de configuração relacionadas a QoS, que são aplicadas nos *Policy Enforcement Points* (PEPs), encontrados nos dispositivos de rede gerenciados.

A comunicação entre os PDPs e os PEPs é crítica para prover uma adequada aplicação de políticas. O IETF tem trabalhado ativamente nesta área definindo protocolos para a comunicação PDP/PEP, tais como COPS (*Common Open Policy Service*) [Durham et al. 2000] e COPS-PR (COPS *Usage for Policy Provisioning*) [Chan et al. 2001]. Além disso, o IETF tem trabalhado na definição de modelos de informação para políticas, tais como PCIM (*Policy Core Information Model*) [Moore et al. 2001], PCIMe (PCIM *extensions*) [Moore 2003], e QPIM (*Quality of Service Policy Information Model*) [Snir et al. 2003], com o objetivo de padronizar o conjunto de informações utilizados na especificação de uma política.

Embora modelos de informação para políticas e a comunicação PDP/PEP sejam áreas onde se pode perceber claramente evoluções de pesquisa e desenvolvimento, o mesmo não ocorre com respeito à comunicação entre a ferramenta de políticas e os PDPs. Por exemplo, não há um protocolo padronizado, ou em processo de padronização, para contemplar essa comunicação. A falta de padronização gera um cenário onde cada sistema PBNM define e implementa seus próprios protocolos para comunicação ferramenta/PDP. Tais protocolos podem ser suscetíveis a erros, o que possibilita a presença de situações de falhas no processo de aplicação de uma política.

Neste artigo é apresentada uma pesquisa a respeito dos requisitos e da implementação de um procedimento atômico de aplicação de políticas, o qual capacita o sistema PBNM a garantir que uma política está aplicada em todos os PEPs de interesse, ou não está aplicada em nenhum PEP. Ou seja, se um PEP a ser configurado falhar ao aplicar uma política, todos os outros PEPs onde a mesma política já foi aplicada devem realizar *rollback* para o estado exatamente anterior à aplicação da política.

Para prover tal procedimento atômico de aplicação de políticas é definido um novo protocolo para suportar a comunicação entre a ferramenta de políticas e PDPs. O protocolo proposto foi implementado em um protótipo de sistema de gerenciamento e sua aplicabilidade foi avaliada em uma rede de testes DiffServ. Além disso, a comunicação entre a ferramentas de políticas e os PDPs foi implementada através do uso da tecnologia de Web Services, a qual possui importantes características requeridas pelo protocolo proposto, tal como entrega confiável de dados.

O restante do artigo está organizado como segue. A seção 2 contextualiza o problema investigado, apresentando um cenário de teste com falhas. A seção 3 introduz a solução proposta. A seção 4 apresenta o protocolo proposto e implementado utilizando tecnologia Web Services. A seção 5 apresenta a análise da solução e, finalmente, a seção 6 apresenta considerações finais e trabalhos futuros.

2. Aplicação de Políticas com Falhas

A aplicação de uma política envolve, considerando a arquitetura PBNM do IETF, no mínimo duas fases de comunicação entre os elementos da arquitetura. Primeiro, a ferramenta de políticas deve contatar os PDPs apropriados de modo a transferir uma política a partir do repositório de políticas. A transferência da política, nesta fase, pode ser realizada por uma abordagem *push* ou *pull*. Na abordagem *push*, a ferramenta de políticas recupera a política do repositório e a envia para os PDPs realizando um *upload*. Na abordagem *pull*, a ferramenta de políticas ordena aos PDPs que esses acessem o repositório de políticas e realizem o *download* da política.

Na segunda fase, cada PDP interage com seus PEPs para garantir a aplicação da política. Na abordagem *provisioning* o PDP configura os PEPs associados usando um protocolo de *provisioning* (ex.: COPS-PR), enquanto na abordagem *outsourcing* cada PEP contata o PDP associado a si toda vez que uma decisão precisa ser tomada, por exemplo, para determinar se uma nova requisição RSVP deve ser aceita.

Por simplicidade, neste texto será considerada somente a aplicação de políticas através da abordagem *provisioning*. Esta restrição será realizada também porque essa abordagem é mais adequada para redes DiffServ. Além disso, nos exemplos são usadas políticas definidas no formato condição-ação, porque este é o modelo de política geralmente utilizado pelo IETF, ainda que o modelo evento-condição-ação seja intensamente investigado pela comunidade de pesquisa em políticas.

2.1. Aplicação da Política

Dentro de um PDP, quando as condições da política forem verdadeiras, o PDP deve, na abordagem *provisioning*, traduzir as ações da política para ações de configuração dos PEPs alvo. Neste ponto, a política pode falhar devido à falta de recursos no PEP. Por exemplo, se uma ação de política realiza alocação de banda de 6 Mbps, mas o enlace disponível é de *backup*, limitado a 2 Mbps, então a aplicação da política falhará.

A recuperação desta aplicação de política pode ser realizada de diferentes maneiras. Pode ser utilizada uma estratégia de substituição de política baseada em PoP (*Policy of Policies*) [Granville et al. 2002] ou um sistema de raciocínio baseado em casos (CBR) [Samaan e Karmouch 2004]. Contudo, a questão a ser investigada neste artigo é o fato de que a falha de uma política em um único PEP, independentemente da abordagem de recuperação utilizada, não afeta somente a política naquele PEP, mas também em todos os PEPs onde a mesma política deveria ser aplicada.

2.2 Exemplo de Falha na Aplicação Distribuída de uma Política

No gerenciamento de QoS, a aplicação de uma política é essencialmente uma ação distribuída. Por exemplo, considere o cenário da Figura 1, onde dois PDPs recebem uma política a ser aplicada em quatro roteadores, mas a aplicação falha no roteador 2.

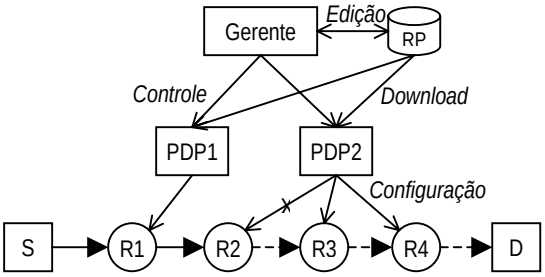

Figura 1. Aplicação de Política com Falha

Para realizar a reserva de banda na rede, a ferramenta de políticas precisa transferir uma política (via *push* ou *pull*) do repositório de políticas (RP) para os PDPs que controlam os PEPs dentro dos roteadores (R1, R2, R3 e R4) entre a origem (S) o destino (D). Para todos os PEPs dentro dos dispositivos onde a aplicação da política obtém sucesso (R1, R3 e R4) a banda requerida é alocada. Contudo, se uma aplicação de política falha (R2), os demais PEPs ainda proverão alocação de banda, mas isto se torna inútil, visto que um

dispositivo sem banda alocada é suficiente para comprometer todo o caminho. Logo, neste caso, apesar da aplicação da política ter sucesso na maioria dos PEPs, a falha em um único PEP afeta a aplicação global da política.

2.3. *Rollback* de Aplicação de uma Política

Uma vez que a aplicação de uma política falha no cenário anterior, deve-se garantir que as outras aplicações que obtiveram sucesso sejam desfeitas (*rollback*), de modo a liberar a banda alocada nos outros PEPs. Senão, banda será desperdiçada servindo a um tráfego que provavelmente não utilizará toda banda devido à falha em um PEP isolado.

Realizar o *rollback* de uma aplicação de política envolve, a partir da ferramenta de políticas, a indicação explícita de que todos PDPs devem realizar *rollback*. Nos PDPs que já aplicaram a política, devem ser executadas ações para desfazer a configuração aplicada. O mais importante, contudo, é notar que este procedimento de *rollback* distribuído é obtido somente com intervenção manual por parte do operador.

Neste cenário, pode-se imaginar que para algumas aplicações (ex.: gerenciamento de QoS), a aplicação de políticas deve funcionar para todos os PEPs alvo, ou para nenhum deles, isto é, a aplicação de uma política deve ser uma operação atômica. Uma das dificuldades em definir tal aplicação atômica é que a comunicação entre a ferramenta de políticas e os PDPs não é padronizada, como apresentado na introdução. Para lidar com este problema, na próxima seção é apresentada uma nova solução para aplicação de políticas e seu protocolo associado.

3. Um Protocolo para Aplicação Atômica de Políticas

Nesta seção é apresentado o protocolo proposto para suportar a aplicação atômica de políticas. São definidas as informações requeridas pelo protocolo, além das funções a serem exercidas por cada elemento de uma arquitetura PBNM que suporte aplicação atômica. Por fim, um novo elemento, chamado *Policy Deployment Coordinator* (PDC) é introduzido e as operações da solução proposta são definidas. O protocolo atômico proposto neste trabalho é uma versão modificada do protocolo 2PC (*two-phase commit*) [Lamport et al. 1993] adaptado às necessidades do gerenciamento baseado em políticas. Na literatura, existe uma variedade de trabalhos relacionados a protocolos que utilizam 2PC para implementar sistemas distribuídos robustos [Yu et al. 2004] [Tang 1996].

3.1. Informações Relativas à Aplicação da Política

Um operador de rede que deseja aplicar uma política em uma rede deve, primeiramente, escolher uma política *p* do repositório de políticas. Além disso, ele precisa escolher, na ferramenta de políticas, aqueles PEPs onde a política será aplicada (*tPeps*). A ferramenta de políticas deve então prover uma chave (*deplId*) que identifica a aplicação de *p* em *tPeps*. Dados os PEPs de *tPeps*, a ferramenta de políticas deve descobrir os PDPs (*tPdps*) responsáveis por controlá-los. Em resumo, tem-se:

- *p* é a política a ser aplicada;
- $tPeps=\{pep_1, pep_2, ..., pep_n\}$ é o conjunto de PEPs alvo.
- $tPdps=\{pdp_1, pdp_2, ..., pdp_n\}$ é o conjunto de PDPs alvo.
- *deplId* é o identificador da aplicação de *p* em *tPeps*.

Como exemplo, considere novamente a rede da Figura 1.

- *tPeps*={R1, R2, R3, R4}
- PDP1 está associado com {R1}, PDP2 está associado com {R2, R3, R4}
- *tPdps*={PDP1, PDP2}

Após ter-se a definição de *p*, *tPeps*, *tPdps*, *deplId*, e das associações entre PDPs e PEPs, a ferramenta de políticas contata os PDPs de *tPdps* e transfere a cada um deles: (a) a política *p*, (b) o identificador de aplicação *deplId*, (c) os PEPs onde o PDP deve aplicar *p*. Por exemplo, ao contatar o PDP2, a ferramenta de políticas transferiria *p*, *deplId*, e {R2, R3, R4}. É importante notar que *p* e *deplId* são valores globais compartilhados entre todos os PDPs envolvidos no procedimento de aplicação.

3.2. PDPs e os estados de uma Política

Uma política *p* dentro de um PDP possui um estado: *inválido*, *inativo* ou *ativo*. Para exemplificar esses diferentes estados, considere a política de exemplo da Figura 2.

```
se  (dia >= 02/01/2004) e (dia < 24/12/2004) e
    ((diaSemana = Sexta) ou (diaMes = Último)) e
    (hora >= 8pm) e (hora <= 9pm) e (PortaOrigem = 80)
então
    banda = 500 kbps
fim
```

Figura 2. Exemplo de Política

Esta definição de política declara que a data de início é 2 de Janeiro de 2004. Contudo, é provável que o operador da rede, junto à ferramenta de políticas, transfira a política para os PDPs alvo antes de 2 de Janeiro. Neste caso, imediatamente após a política ser transferida para um PDP, o estado da política é *inválido*, já que a data de início ainda não foi atigida. A política também define que a data de encerramento é 24 de Dezembro de 2004. Depois dessa data, a política torna-se *inválida*, é deve ser removida dos PDPs.

Quando a data de início é atingida, o estado da política muda para *inativa*, isto é, a política agora está válida, mas ainda não está ativa. Uma vez que a política está inativa, ela precisa passar para o estado *ativo* para efetivamente configurar os PEPs alvo. A política de exemplo passa para *ativo* – e então aloca 500 kbps de banda para tráfego HTTP – somente nas sextas-feiras e no último dia de cada mês, às 8 pm. Às 9 pm, a política não estará mais ativa, retornando para o estado *inativo*. A Figura 3 resume a seqüência de mudança de estados de uma política em um PDP.

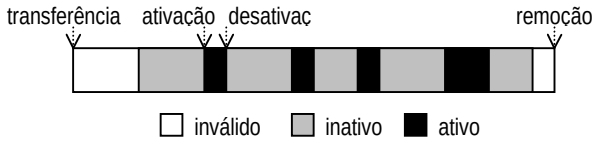

Figura 3. Estados de uma política dentro de um PDP

3.3 Policy Deployment Coordinator

A aplicação atômica de políticas é obtida através da ativação atômica. Cada vez que uma política é ativada em um PDP, ela deve ser aplicada também nos outros PDPs. Quando uma política for desativada, ela ser desativada também nos outros PDPs.

Um novo elemento na arquitetura PBNM, chamado *Policy Deployment Coordinator* (PDC), é definido. Sua responsabilidade é a de orquestrar as sucessivas ativações/desativações de políticas atômicas.

O PDC coordena as ativações de uma política usando uma estrutura cujas entradas são indexadas por *deplId*. Tal estrutura (registro de aplicação) inclui um inteiro que indica o estado corrente de uma aplicação, e um vetor de PDPs que guarda o estado de ativação da política nos PDPs envolvidos. A verificação deste vetor possibilita que o PDP identifique o estado global de ativação de uma política. Em resumo:

- *DeplKey*: a identificação única de uma aplicação de política (*deplId*)
- *DeplState*: [0:*inativa* 1:*ativando* 2:*ativo*]
- *DeplVector PDPs*: o conjunto de PDPs para esta aplicação de política (*tPdps*)
- *DeplVector state*: o estado de cada PDP ([0:*desconhecido* 1:*sucesso* 2:*falha*])

Analisando novamente a rede de exemplo da Figura 1, o registro de aplicação de *p* no PDC (supondo que o identificador de aplicação seja 4361) seria (Figura 4):

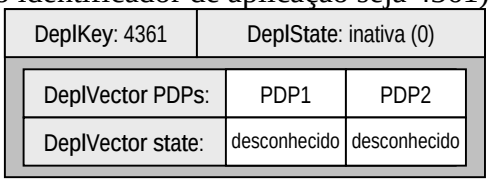

Figura 4. Exemplo de registro de aplicação

Ainda que o PDC tenha sido definido como um novo elemento na arquitetura PBNM do IETF, ele pode ser fisicamente implementado junto à ferramenta de políticas ou internamente a um PDP. Dado o PDC recém introduzido, e as informações de política definidas, as próximas subseções apresentam as operações do protocolo proposto.

3.4. Transferência da Política e Operações de Ativação

Depois que operador seleciona a política e os PEPs onde a política deve ser aplicada, a ferramenta de políticas deve obter *deplId* e *tPdps*. Então, a operação *RegisterPolicyDeployment* do PDC deve ser chamada de forma a criar um registro de aplicação para *p* no PDC. Esta operação recebe como argumentos a identificação da aplicação (*deplId*) a ser associada com a chave da ativação (*DeplKey*), e *tPdps* para instanciar *DeplVector PDPs*. O estado da aplicação (*DeplState*) é iniciado *inativo* (0), e o estado de cada elemento de *DeplVector state* é iniciado com *desconhecido* (0).

Após registrar a aplicação da política no PDC, a ferramenta de políticas transfere a política *p* para todos os PDPs em *tPdps* utilizando a operação *TransferPolicy*, disponibilizada por cada PDP. Esta operação recebe como argumento a política *p*, a lista de PEPs que este PDP deve configurar (que é um subconjunto de *tPeps*), e *deplId*. Assim, a transferência das informações de aplicação da política terá sido concluída.

Internamente, cada PDP deve levar em consideração as condições temporais de *p* de forma a ativar tal política quando for a ocasião. Visto que os relógios internos dos PDPs não estão necessariamente totalmente sincronizados, a ativação de uma política *p* pode ser disparada em diferentes momentos em diferentes PDPs. É assumido que o relógio dos PDPs podem estar ligeiramente fora de sincronia, por exemplo, porque eles

usam *Network Time Protocol* (NTP) para ajustar seus relógios. O primeiro PDP a ativar *p* deve notificar o PDC chamando sua operação *InitiateActivation*, e então proceder com a execução das ações da política.

Quando notificado, o PDC muda *DeplState* para *ativando* (1) e notifica todos os outros PDPs acessando a operação *InitiateActivation* disponibilizada por cada um deles. Esta notificação ordena que os PDPs restantes iniciem a ativação de *p*. Os PDPs que ativam uma política devido à notificação via *InitiateActivation* não precisam mais notificar o PDC (Figura 5).

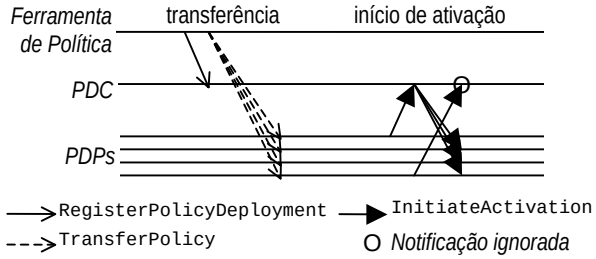

Figura 5. Transferência da Política e Início de sua Ativação

3.5. *Commit* e *Rollback* de Ativações de Políticas

Depois que a ativação de uma política é concluída, todos os PDPs configuram os PEPs. Neste momento, falhas podem ocorrer, e o PDC precisa ser notificado. Enquanto está no estado *ativando*, o PDC espera por avisos sobre o resultado de ativação dos PDPs.

Assim que um PDP configura com sucesso todos os PEPs associados, ele deve chamar a operação *ActivationSuccess* do PDC. Contudo, quando um PDP percebe que a configuração de um PEP falhou, ele deve desfazer (*rollback*) a configuração em todos os PEPs associados e notificar a falha chamando a operação *ActivationFail* do PDC.

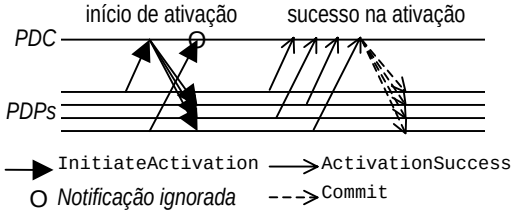

Figura 6. Início da ativação e *Commit*

Para cada chamada *ActivationSuccess*, o PDC ajusta o estado do PDP que invocou a chamada para *success* (1) em *DeplVector state*. Quando todas as posições de *DeplVector state* estão iguais a 1, o PDC chama a operação *Commit* de cada PDP, indicando que todo o procedimento de ativação global obteve sucesso (Figura 6).

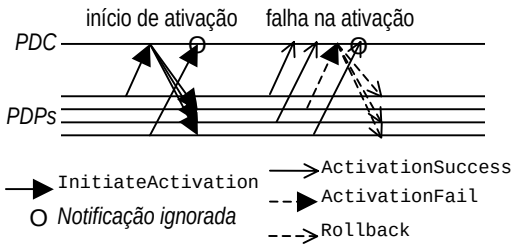

Figura 7. Início da ativação e *Rollback*

Se, contudo, um PDP notifica o PDC através de *ActivationFail*, então o PDC imediatamente notifica todos os outros PDPs, exceto o PDP que fez a notificação, chamando a operação *Rollback* (Figura 7). Cada PDP, por sua vez, contata os PEPs associados a política para remover a configuração aplicada previamente.

Visto que algumas falhas podem ocorrer não somente nos PEPs, mas também nos PDPs e no PDC, alguns *timeouts* foram definidos. O primeiro está localizado no PDC. O PDC espera por avisos de ativação dos PDPs durante um determinado tempo. Se o tempo relativo ao *timeout* expira, e existe algum estado *desconhecido* (0) no vetor *DeplVector state*, o PDC assume que a falta de notificações é devida a PDPs que falharam e então dispara a notificação *Rollback* em todos os PDPs (Figura 8).

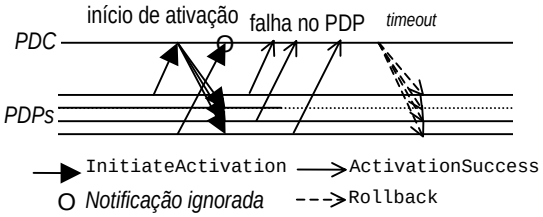

Figura 8. *Rollback* devido à falha no PDP e *timeout* no PDC

O segundo *timeout* está localizado nos PDPs. Depois de chamar a operação *ActivationSuccess* do PDC, o PDP espera por uma chamada *Commit* ou *Rollback*. Se nenhuma das duas é chamada em um determinado limite de tempo, então o PDP realiza *rollback* de configuração, assumindo que o PDC falhou (Figura 9). É importante notar que o *timeout* no PDP deve ser maior que o *timeout* no PDC, do contrário os PDPs realizarão rollback antes que o PDC possa invocar *Commit*.

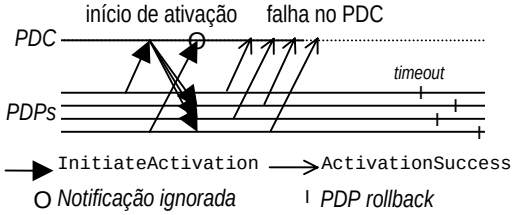

Figura 9. *Rollback* devido à falha no PDC e *timeout* no PDP

3.6. Desativação e Remoção de uma Política

A desativação de uma política é uma operação relativamente simples. O primeiro PDP que detecta que a política ativa deve ser desativada notifica o PDC chamando a operação *Deactive*. O PDC repassa esse aviso para todos os PDPs restantes chamando outra operação *Deactive* disponibilizada pelos PDPs. Além disso, o PDP ajusta *DeplStatus* para *inativa* (0) e ajusta todas as posições de *DeplVector state* para o estado *desconhecido* (0), de forma a preparar-se para coordenar uma nova ativação de política.

Ainda que a política seja desativada de tempos em tempos, ela permanece válida dentro dos PDPs até que a data de encerramento seja atingida. Neste caso, o primeiro PDP que percebe que uma política está *inválida* notifica o PDC através da operação *RemovePolicy*, a qual é bastante similiar a operação *Deactive* (Figura 10), exceto pelo fato de que *RemovePolicy* remove todas as instâncias da política de cada PDP associado, e libera a memória que guarda o registro de aplicação da política no PDC.

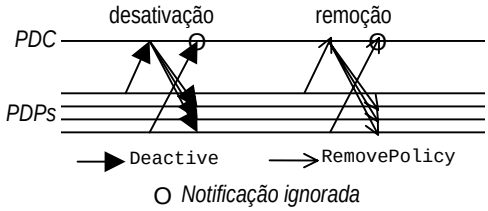

Figura 10. Desativação e Remoção de uma Política

4. Implementação do Protocolo

O protocolo proposto foi implementado em um protótipo QAME (*QoS-Aware Management Environment*) [Granville et al. 2001]. O QAME é um sistema que já possuía suporte ao gerenciamento baseado em políticas para redes DiffServ. Os PDPs do sistema foram expandidos de modo a incluir a aplicação, além de ter sido integrado um novo PDC à ferramenta de políticas disponibilizada pelo QAME.

4.1. O Protocolo como um conjunto de operações Web Services

O suporte ao protocolo proposto foi implementado como um conjunto de operações Web Services disponibilizados pelos elementos da arquitetura PBNM. Utilizar Web Services como base para o protocolo é interessante devido aos seguintes motivos:

- O tráfego de Web Services é usualmente transportado por SOAP sobre mensagens HTTP, as quais normalmente são aceitas por *firewalls*. Assim, há a possibilidade de uma distribuição mais fácil dos PDPs por diferentes domínios administrativos tendo a Internet como sua rede de comunicação.
- Visto que HTTP utiliza TCP – que é um protocolo de transporte confiável e orientado a conexão – a implementação do protocolo via Web Services é interessante porque essa tecnologia provê base confiável para a entrega das mensagens do protocolo.
- APIs para Web Services são largamente disponíveis em diferentes linguagens de programação, o que possibilita o uso direto do protocolo proposto por outros clientes e servidores implementados em diferentes ambientes.
- As operações do protocolo são implementadas como funções de uma linguagem de programação, e a troca de mensagens do protocolo ocorre quando um elemento (ex.: PDC) dispara uma chamada remota de procedimento (RPC) para acessar uma função localizada em outro elemento (ex.: PDP).
- Ainda que RPC seja geralmente responsável por uma queda no tempo de resposta de operação, lidar com RPC em Web Services é mais fácil e mais genérico que definir PDUs e codificação de dados para um protocolo.

O protocolo implementado utiliza a API PHP para Web Services chamada nuSOAP [Ayala 2004]. Esta API é largamente utilizada pela comunidade de desenvolvedores Web Service e é relativamente rápida se comparada com outras APIs PHP para Web Services. PHP, por sua vez, foi escolhido porque o sistema QAME é desenvolvido em PHP, e porque esta linguagem possui facilidades para desenvolvimento Web reconhecidas pela comunidade de software livre.

4.2. Operações Web Services nos elementos PBNM

A Figura 11 apresenta a arquitetura final do sistema QAME. É importante destacar que o PDC foi implementado junto à ferramenta de políticas. Desta forma, o atraso de comunicação entre a ferramenta de políticas e o PDC é mínimo.

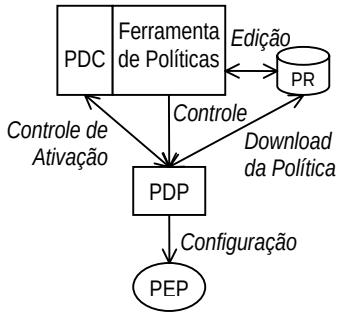

Figura 11. Arquitetura PBNM com PDC

As operações disponibilizadas pelos elementos PDC e PDP são listadas na Figura 12.

Operações	PDC	PDP
Aplicação atômica	- int RegisterPolicyDeployment (int deplId=0, array tPdps); - InitiateActivation(int deplId); - ActivationSucess(int deplId); - ActivationFail(int deplId); - RemovePolicy(int deplId);	- int PolicyTransfer (int deplId, string policy); - InitiateActivation (int deplId); - Commit(int deplId); - Rollback(int deplId); - RemovePolicy(int deplId);
Estatísticas	- int GetState(int deplId); - array GetHistory(int deplId); - SetHistoryTimeout (int deplId, int timeout); - array GetFaultyPDPs();	- array GetState(); - array GetHistory(); - SetHistorySize ();

Figura 12. Operações disponibilizadas pelo PDC e pelo PDP

Além das operações do protocolo, foram definidas operações para a obtenção de estatísticas da aplicação de políticas. Essas estatísticas permitem à ferramenta de políticas monitorar o desempenho do PDC e dos PDPs.

No PDC, a operação *RegisterPolicyDeployment* recebe *deplId* e uma lista de PDPs, e retorna *deplId*. Se nenhum *deplId* é passado como parâmetro, então o valor *default* 0 (zero) é passado informando que o próprio PDC deve gerar um identificador, o qual será retornado para a ferramenta de políticas. Desta forma, o *deplId* pode ser gerado tanto pela ferramenta de políticas quanto pelo PDC. Todas as outras operações do PDC (apresentadas anteriormente) recebem *deplId* como parâmetro.

No PDP, a operação *PolicyTransfer* recebe como entrada o *deplId* e a própria política via o argumento *policy*. Se tal argumento for uma URL, então o PDP realiza o *download* da política a partir da URL fornecida. Por outro lado, se o argumento é a política codificada em um documento XML, então esta será armazenada dentro do PDP. Isso significa que o modelo de transferência de políticas (*pull* ou *push*) é selecionado pelo conteúdo do argumento *policy*: *pull* se for envianda uma URL, e *push* se for enviando um documento XML. Todas as outras operações recebem apenas *deplId*.

As operações de estatísticas funcionam como segue. No PDC, *GetState* retorna o estado da aplicação de uma política, isto é, *inativa* (0), *ativando* (1), e *ativa* (2). Isto permite que a ferramenta de políticas monitore o PDC de forma a determinar mudanças

de estado em uma aplicação de políticas. O PDC pode também armazenar um histórico de informações que inclua as mudanças de estado e os *timestamps* das mudanças.

A ferramenta de políticas pode recuperar o histórico de aplicações de políticas através de *GetHistory*. Visto que informações de histórico requerem capacidade de armazenamento que pode ser limitada, a operação *SetHistoryTimeout* permite que a ferramenta de políticas defina quanto tempo um histórico de aplicação será armazenado dentro do PDP depois que uma política não é mais válida. Por exemplo, um *timeout* igual a zero indica que as informações de histórico estarão disponíveis somente enquanto a política for válida. Finalmente, a operação *GetFaultyPDPs* lista, sem importar a política que está sendo considerada, os PDPs que já falharam ao ativar alguma política, junto com o número de falhas de ativação.

Os PDPs que apresentaram falhas podem ser acessados. A operação *GetState* retorna um *array* listando o estado atual das políticas válidas. A operação *GetHistory* retorna um *array* de histórico informando o número de falhas de ativação ocorridos em um PDP. Finalmente, a operação *SetHistorySize* permite que a ferramenta de políticas ajuste a quantidade de informações de histórico a ser armazenada dentro de um PDP.

4.3. Consumo dos recursos de rede

Na tecnologia de Web Services, RPC normalmente é implementado usando SOAP (*Simple Object Access Protocol*) [Mitra 2003]. Ainda que SOAP possa utilizar protocolos diferentes para transportar suas mensagens (ex.: SMTP, FTP, etc), HTTP é o mais freqüentemente utilizado. No protótipo do sistema, SOAP sobre HTTP provê a base para a entrega confiável de mensagens. Contudo, na visão do modelo de referência OSI, dois protocolos de nível de aplicação estão sendo utilizados para carregar a chamada RPC. Isso, obviamente, aumenta o *overhead* total.

Ainda que sejam consideradas mensagens SOAP geradas a partir do uso da API nuSOAP (e mensagens SOAP podem variar de APIs para API), outras APIs SOAP presumidamente não devem gerar mensagens com tamanhos muito diferente das mensagens nuSOAP. Para facilitar a análise que segue, é considerado somente o tráfego SOAP, excluindo o *overhead* HTTP, TCP, IP e do nível de enlace.

Foram considerados dois tipos de operações do protocolo: operações com número variável de argumentos e operações com número fixo de argumentos. Operações com um número variável de argumentos, tais como *RegisterPolicyDeployment* e *PolicyTransfer*, geram um maior *overhead* na medida que seu número de argumentos aumenta. Se for considerado que inteiros são limitados a 5 dígitos, e que os *hostnames* dos PDPs e identificadores de PEPs são limitados a 20 bytes, a operação *RegisterPolicyDeployment* gera mensagens SOAP de requisição de acordo com:

```
bytes = 610 + 41 + n.55
```

41 bytes são utilizados para enviar o inteiro de 5 dígitos *deplId*, *n* é o número de PDPs (identificados em 20 caracteres) que são codificados em 55 bytes, e 610 são os bytes fixos restantes da codificação da requisição. Visto que uma aplicação de política envolve apenas uma chamada a operação *RegisterPolicyDeployment*, a quantidade de bytes desta requisição é proporcional ao número de PDPs. A Figura 13 apresenta o tráfego gerado quando ocorre uma requisição à operação *RegisterPolicyDeployment*.

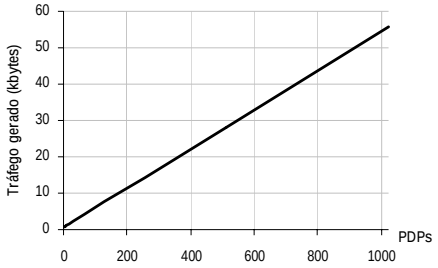

Figura 13. Tráfego gerado pela operação *RegisterPolicyDeployment*

Para *PolicyTransfer*, que também possui um número variável de argumentos, temos:

`bytes= (590 + 41 + 54 +` *m*`.55).`*n*

590 é a parte fixa. Novamente, 41 bytes são necessários para incluir *deplId*, 54 bytes são necessários para incluir uma URL de 15 caracteres, e *m* é o número médio de PEPs (codificados em 55 bytes cada) que devem ser contatados por cada um dos *n* PDPs. É importante notar que neste caso não existe apenas uma requisição simples, mas *n* requisições. A Figura 14 apresenta o tráfego gerado por uma operação *PolicyTransfer*.

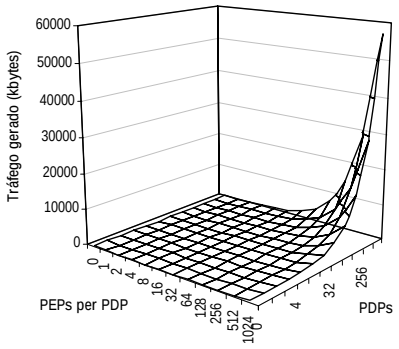

Figura 14. Tráfego gerado pela operação *PolicyTransfer*

Para as mensagens SOAP restantes que requerem apenas um inteiro (*deplId*), temos:

`bytes = (512 + 41).`*n*

Novamente, 41 bytes são codificam um inteiro de 5 dígitos e 512 bytes correspondem à parte fixa. Visto que o comprimento do nome das operações varia de uma para outra, foram considerados nomes de operações com 13 caracteres de comprimento (tamanho médio do nome das operações com apenas um parâmetro). *n* é o número de PDPs. A Figura 15 mostra o tráfego gerado por uma operação contendo *deplId* como parâmetro.

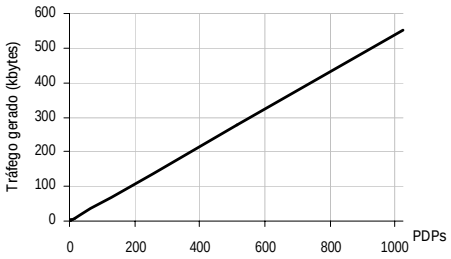

Figura 15. Tráfego gerado por uma operação contendo parâmetro *deplId*

A quantidade de bytes que trafegam em todas as operações de uma aplicação de política depende do número de PDPs utilizados. Por exemplo, se for considerado um número médio de 16 PDPs, o que, para as redes atuais, parece ser um número razoável, tem-se cerca de 1,49 kbytes para *RegisterPolicyDeployment*, 17,57 kbytes para *PolicyTransfer* (considerando 8 PEPs por PDP), e 8,64 kbytes para cada operação restante.

5. Conclusões e Trabalhos Futuros

Neste artigo foram apresentados a motivação, a proposta, a implementação e a análise de um protocolo para aplicação atômica de políticas em redes com QoS. A necessidade de um protocolo desse tipo vem do fato de que em certas situações uma política aplicada nos dispositivos (mais precisamente nos PEPs) deve ter sucesso em todos os pontos finais, sob o risco de desperdiçar recursos caso a aplicação falhe em algum PEP.

O protocolo proposto foi implementado usando a tecnologia de Web Services, que é baseada no protocolo SOAP. Nas implementações realizadas, SOAP foi encapsulado sobre HTTP, já que esse é o uso mais comum do protocolo, e que possui um suporte mais maduro nos ambientes de desenvolvimento. Pela forma como foi definido, o uso dos recursos de rede pelo protocolo (largura de banda, especificamente) depende do número de PDPs e, com uma importância menor, do número de PEPs. Se o número de PDPs for elevado, maior será o consumo de banda porque as mensagens ou serão maiores (ex.: `RegisterPolicyDeployment`) ou terão várias cópias (ex.: `TransferPolicy`). O número de PEPs influencia apenas no tamanho das mensagens de solicitação da operação `TranferPolicy`, já que os PEPs onde uma política deve ser aplicada precisam ser enviados como parâmetro. Entretanto, como cada PDP controla apenas sub-conjuntos de PEPs, e não todos eles, um grande número de PEPs não obrigatoriamente significa que cada PDP controlará muitos PEPs.

Cálculos rápidos mostraram que para um número de PDPs razoavelmente elevado (16), poucos kbytes foram necessários para registrar uma aplicação de políticas (1,49 kbytes), transferir tal política via *pull* (17,57 kbytes) e executar cada notificação do protocolo (8,64 kbytes). Assim, para redes atuais, o protocolo possui um consumo de recursos de rede relativamente baixo, principalmente se levarmos em conta o volume de dados necessário, por exemplo, numa configuração em linha de comando via TELNET.

Apesar de já existirem outros serviços de estatísticas definidos e implementados no protocolo, facilidades complementares precisam ser incorporadas. Em relação à segurança, atualmente está sendo definido o suporte para autenticação e controle de acesso, de forma que apenas gerentes, PDCs e PDPs autorizados possam trocar informações entre si. O uso de um protocolo de aplicação mais seguro se faz necessário e está sendo investigado, como é o caso do HTTPS. Por fim, a avaliação do consumo de recursos internos aos elementos (processamento e memória) ainda será executada.

Referências

Ayala, D. (2004) “nuSOAP - web services toolkit for PHP,” Novembro, http://dietrich.ganx4.com/nusoap/.

Chan, K. et al. (2001) “RFC 2084: COPS usage for policy provisioning”, IETF.

Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R. e Sastry, A. (2000) “RFC 2748: The COPS (Common Open Policy Service) protocol”, IETF.

Flegkas, P., Trimintzios, P., e Pavlou, G. (2002) “A policy-based quality of service management system for IP differentiated services networks” *IEEE Network*, v.16 n. 2

Granville, L., Ceccon, M., Tarouco, L., Almeida, M. e Carissimi, A. (2001) “An approach for integrated management of networks with quality of service support using QAME” in *IEEE/IFIP 12th International Workshop on Distributed Systems: Operations and Management, DSOM 2001*, Outubro.

Granville, L., Coelho, G., Almeida, M. e Tarouco, L. (2002) “Pop - an automated policy replacement architecture for PBNM,” in *3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, Junho, p. 140–145.

Guo, X., Yang, K., Galis, A., Cheng, X., Yang, B., e Liu, D. (2003) “A policy-based network management system for IP VPN.” in *ICCT*, vol. 2, Abril, p. 1630–1633.

Lampson, B. W., e Lomet, D. B., (1993) “A new presumed commit optimization for two phase commit,” in *Proceedings of the 19th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., pp. 630–640.

Mitra, N. (2003) “SOAP Version 1.2 Part 0: Primer”**,** World Wide Web Consortium, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, Setembro.

Moore, B. (2003) “RFC 3460: Policy Core Information Model extensions,” Janeiro.

Moore, B., Ellesson, E., Strassner, J. e Westerinen, A. (2001) “RFC 3060: Policy core information model – version 1 specification”, Fevereiro.

Nikolakis, Y., Magaña, E., Solarski, M., Tan, A., Salamanca, E., Serrat, J., Brou, C. e Galis, A. (2004) “A policy-based management architecture for flexible service deployment in active networks.” In *IWAN 2003*, ser. Lecture Notes in Computer Science, vol. 2982. Springer, p. 240–251.

Samaan, N. e Karmouch, A. (2004) “A case-based reasoning approach for automated management in policy-based networks.” In *15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2004*, Lecture Notes in Computer Science, vol. 3278. Springer, p. 76–87.

Snir, Y., Ramberg, Y., Strassner, J., Cohen, R. e Moore, B. (2003) “RFC 3644: Policy quality of service (QoS) information model,” Novembro.

Tang, L., (1996) “Verifiable transaction atomicity for electronic payment protocols.” in *16th International Conference on DistributedComputing Systems (ICDCS’96)*. IEEE Computer Society, 1996, p. 261.

Tsarouchis, C., Denazis, S., Kitahara, C., Vivero, J., Salamanca, E., Magana, E., Galis, A., Manas, J., Carlinet, L., Mathieu, B. e Koufopavlou, O. (2003) “A policy-based management architecture for active and programmable networks.” *IEEE Network*, vol. 17, no. 3, p. 22–28, Maio/Junho, issn: 0890-8044.

Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J. e Waldbusser, S. (2001) “RFC 3198: Terminology for policy-based management,” Novembro.

Yu, W., Wang, Y., e Pu, C., (2004) “A Dynamic Two-Phase Commit Protocol for Self-Adapting Services.” in *SCC ’04: Proceedings of the Services Computing, 2004 IEEE International Conference on (SCC’04)*. IEEE Computer Society, pp. 7–15.