UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO SANGER ALVES

Operações Atômicas para Gerenciamento Baseado em Políticas

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Maria Janilce B. Almeida
Orientadora

Porto Alegre, abril de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Alves, Rodrigo Sanger

Operações Atômicas para Gerenciamento Baseado em Políticas / Rodrigo Sanger Alves. – Porto Alegre: PPGC da UFRGS, 2007.

76 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientadora: Maria Janilce B. Almeida.

1. Gerenciamento de Configuração. 2. Gerenciamento de Redes Baseado em Políticas. 3. Operações Atômicas. I. Almeida, Maria Janilce B. II. Título.

“Não tá morto quem peleia.”
— SABEDORIA POPULAR GAÚCHA

AGRADECIMENTOS

Agradeço a toda minha família, mas principalmente aos meus pais, pelo lar e pela estrutura familiar que me proporcionaram, sem a qual tudo teria sido muito mais difícil. Muito obrigado, pai e mãe.

Graças aos afilhados, aos amigos e a minha namorada, os intervalos de lazer entre um artigo e outro foram suficientemente divertidos para que eu não desistisse desta empreitada. Muito obrigado a todos que me ajudaram a relaxar de vez em quando, evitando que eu virasse uma simples máquina de produzir textos e fazendo com que eu aproveitasse as coisas boas e simples da vida.

Gostaria de agradecer também aos colegas de mestrado Ricardo Vianna, Clarissa Marquezan, Tiago Fioreze, Weldson Lima, Rafael Huff, Karina Girardi, Daniel Lazzarotto, Gabriela Jacques, entre outros, que muito me ajudaram, seja na escrita de artigos, no dia-a-dia do laboratório de pesquisa ou simplesmente com a presença no Culto da 228 ou nos tradicionais Festivais da Bergamota. Vocês fizeram isto tudo valer muito a pena!

Agradeço ao pessoal do cluster LabTeC, que sempre foi muito atencioso e disponibilizou o cluster para os testes da minha dissertação.

Agradeço aos professores que me ajudaram desde meu ingresso na universidade, passando pela bolsa de iniciação científica e culminando no mestrado. Entre eles, cito nominalmente os professores Tiarajú Diverio, Lisandro Granville e Maria Janilce Almeida.

Agradeço a empresa onde eu trabalho, Datacom, pela compreensão e apoio nesta reta final do meu mestrado quando nem sempre eu podia estar presente em um horário totalmente convencional. Agradeço aos colegas de trabalho que torceram por minha vitória.

Em especial agradeço aos amigos Diego Contessa, Everton Polina e Rodrigo Machado pela amizade verdadeira e incondicional.

Agradeço a fisioterapeuta Priscila Gomes pela dedicação na difícil tarefa de combater minha tendinite e me manter o mais inteiro possível para dissertar durante o ano de 2006. Agradeço ao meu afilhado Raphael pelo carinho demonstrado pelo dindo, especialmente quando eu não conseguia digitar e ele assumia o teclado e ficava fazendo as alterações que eu ditava. Acreditem, para uma criança de 11 anos isto é uma tarefa muito, muito entediante.

Por fim, e não menos especial (muito pelo contrário), gostaria de agradecer a minha namorada Jamile, que foi um exemplo perfeito do que significa a palavra companheirismo. Além de amor, carinho e amizade, sobre tudo, ela me mostrou que é uma companheira para todas as horas, abdicando de momentos de lazer para ficar transformando as palavras que eu ditava em orações, parágrafos, figuras e tags LaTeX. Jamile, tu não foste somente as minhas mãos, meus punhos e meus braços durante o final desta dissertação. Tu foste a palavra de incentivo durante a desmotivação e a palavra de alívio durante a irritação. Em resumo, és muito mais do que eu poderia sonhar. Muito obrigado!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 7

LISTA DE FIGURAS 9

LISTA DE TABELAS 11

RESUMO 12

ABSTRACT 13

1 INTRODUÇÃO 14

2 CONFIGURAÇÕES ATÔMICAS E GERENCIAMENTO DE REDES BASEADO EM POLíTICAS 17

2.1 Gerenciamento de Redes Baseado em Políticas 19

2.2 PBNM e Configurações Atômicas 22

2.3 Exemplo de Necessidade de Atomicidade Utilizando PBNM 23

3 SOLUÇÃO PROPOSTA 25

3.1 Requisitos para a Obtenção de Aplicações Atômicas 25

3.1.1 Requisitos Relativos ao Dispositivo de Rede que Contém PEPs 25

3.1.2 Requisitos Relativos a um PDP Isoladamente 26

3.1.3 Requisitos Relativos à Coordenação de Múltiplos PDPs 27

3.2 Arquitetura PBNM com Suporte a Operações Atômicas 29

3.3 Protocolo de Consenso para Aplicação Atômica de Políticas 31

3.3.1 Informações Relativas à Aplicação de Políticas 33

3.3.2 PDPs e os Estados de uma Política 34

3.3.3 Policy Deployment Coordinator 34

3.3.4 Transferência da Política e Operações de Ativação 35

3.3.5 Commit e Rollback de Ativações de Políticas 37

3.3.6 Desativação e Remoção de uma Política 40

3.3.7 Alternativas de Operação para o Protocolo de Consenso 41

4 IMPLEMENTAÇÃO 44

4.1 Operações Web Services nos Elementos PBNM 45

4.2 Policy Deployment Coordinator (PDC) 47

4.3 Suporte ao Protocolo de Consenso nos PDPs 48

4.3.1 Processamento das Mensagens do Protocolo 50

4.3.2 PDP Cisco 51

4.3.3 PDP AltQ 51
4.4 Interface com o Usuário no QAME 51

5 AVALIAÇÃO DE DESEMPENHO DO PROTOCOLO 59
5.1 Análise Teórica do Tamanho das Mensagens SOAP 59
5.2 Avaliação do Protocolo em um Cenário de Testes 66
5.2.1 Ativações com Sucesso 67
5.2.2 Ativações sem Sucesso 69
5.2.3 Desativações 70

6 CONCLUSÕES E TRABALHOS FUTUROS 72

REFERÊNCIAS 74

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CLI	Command Line Interface
COPS	Common Open Policy Service
COPS-PR	COPS Usage for Policy Provisioning
DIFFSERV	Differentiated Services
DSCP	Differentiated Services Code Point
FCAPS	Fault, Configuration, Accounting, Performance, Security
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IETF	Internet Engeneering Task Force
IP	Internet Protocol
MIB	Management Information Base
MIB-II	MIB-padrão versão 2
OSI	Open Systems Interconnection
PBNM	Policy-based Network Management
PCIM	Policy Core Information Model
PDC	Policy Deployment Coordinator
PDP	Policy Decision Point
PDU	Protocol Data Unit
PEP	Policy Enforcement Point
PHP	PHP: Hypertext Preprocessor
PM-MIB	Policy Management MIB
QAME	QoS-Aware Management Environment
QoS	Quality of Service

RFC	Request For Comments
RSVP	Resource reSerVation Protocol
SMI	Structure of Management Information
SMIv2	Structure of Management Information versão 2
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNMPv1	SNMP versão 1
SNMPv2	SNMP versão 2
SNMPv3	SNMP versão 3
SOAP	Simple Object Access Protocol
SSH	Secure SHell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locators
VPN	Virtual Private Network
XML	eXtensible Markup Language

LISTA DE FIGURAS

Figura 1.1: Arquitetura PBNM 15

Figura 2.1: Exemplo de caminho de roteadores com reserva de banda comprometida. 19
Figura 2.2: Exemplo de política 20
Figura 2.3: Arquitetura PBNM com transferência da política seguindo a abordagem *pull* 21
Figura 2.4: Aplicação de Política com Falha 23

Figura 3.1: Divulgação de resultados parciais de todos para todos 28
Figura 3.2: Divulgação de resultados parciais através de anel lógico 28
Figura 3.3: Divulgação de resultados parciais através de um centralizador 29
Figura 3.4: Modelo adotado: comunicação entre PDPs incluindo intermediador (PDC) 29
Figura 3.5: Modelo não adotado: comunicação direta entre PDPs 30
Figura 3.6: Arquitetura PBNM com a inclusão do PDC 30
Figura 3.7: Comportamento do protocolo 2PC com sucesso de todos participantes 31
Figura 3.8: Comportamento do protocolo 2PC com falha em algum participante . 31
Figura 3.9: Diferença entre relógios internos dos PDPs faz com que a política esteja ativa em momentos diferentes. 32
Figura 3.10: Exemplo de uma rede com PDPs associados aos PEPs, formando um caminho passível de aplicação de políticas entre dois *hosts* O e D . . 33
Figura 3.11: Estados de uma política dentro de um PDP 34
Figura 3.12: Transferência da Política e Início de sua Ativação 36
Figura 3.13: Exemplo de situação em que um PDP pode receber uma mensagem InitiateActivation antes de ter recebido a transferência da política . . 36
Figura 3.14: Início da ativação e Commit 37
Figura 3.15: Falha na divulgação de Commit e conseqüente Abort 37
Figura 3.16: Início da ativação e Rollback 38
Figura 3.17: PDPs em um ciclo: iniciando ativações e percebendo insucessos. . . . 38
Figura 3.18: Exemplo de comportamento do protocolo com número de novas tentativas maior ou igual a 1, contendo intervalo entre tentativas. 39
Figura 3.19: Rollback devido à falha no PDP e *timeout* no PDC 40
Figura 3.20: Rollback devido à falha no PDC e *timeout* no PDP 40
Figura 3.21: Desativação e Remoção de uma Política 41
Figura 3.22: Comportamento do protocolo sem otimização de envio de Rollback logo após mensagem não ser entregue. 42

Figura 3.23: Comportamento do protocolo com otimização de envio de Rollback logo após mensagem não ser entregue. 43

Figura 4.1: Arquitetura PBNM com PDC 46
Figura 4.2: Arquitetura interna do PDP e interações com Ferramenta de Políticas, PDC e PEPs 49
Figura 4.3: Tela de *login* do QAME 52
Figura 4.4: Tela inicial do QAME contendo o mapa da rede 52
Figura 4.5: Tela de criação e edição de políticas 53
Figura 4.6: Tela para criação, alteração e remoção de fluxos 54
Figura 4.7: Tela para criação, alteração e remoção de agendamentos 54
Figura 4.8: Tela para criação, alteração e remoção de ações 55
Figura 4.9: Exemplo de rede contendo dois PDPs e três PEPs 56
Figura 4.10: Aplicação simultânea de política em três dispositivos 56
Figura 4.11: Exemplo de tela de aplicação atômica de políticas em três dispositivos 57
Figura 4.12: Tela de acompanhamento de aplicações atômicas e remoção das mesmas 58

Figura 5.1: Documento XML de requisição referente a uma operação RegisterPolicyDeployment envolvendo 2 PDPs. 60
Figura 5.2: Documento XML de resposta para uma operação RegisterPolicyDeployment. 61
Figura 5.3: Tráfego gerado pela operação RegisterPolicyDeployment 61
Figura 5.4: Documento XML de requisição referente a uma operação TransferPolicy envolvendo 2 associações política-PEP. 62
Figura 5.5: Documento XML de resposta referente a uma operação TransferPolicy. 62
Figura 5.6: Tráfego gerado pela operação TransferPolicy 63
Figura 5.7: Documento XML de requisição referente a uma operação ActivationSuccess. 64
Figura 5.8: Documento XML de resposta referente a uma operação ActivationSuccess. 64
Figura 5.9: Tráfego gerado por operação que contenha como parâmetro deplId e pdpId 64
Figura 5.10: Documento XML de requisição referente a uma operação Commit. . 65
Figura 5.11: Documento XML de resposta referente a uma operação Commit. . . . 65
Figura 5.12: Tráfego gerado por uma operação que contenha deplId como parâmetro único 66
Figura 5.13: Tráfego gerado por ativações com sucesso 68
Figura 5.14: Tempo decorrido durante ativações com sucesso 68
Figura 5.15: Tráfego gerado por ativações sem sucesso 69
Figura 5.16: Tempo decorrido durante ativações sem sucesso 70
Figura 5.17: Tráfego gerado por desativações 70
Figura 5.18: Tempo decorrido durante desativações 71

LISTA DE TABELAS

Tabela 5.1: Combinações de testes realizados envolvendo falhas 69

RESUMO

Nesta dissertação é apresentada uma avaliação da viabilidade de obtenção de operações atômicas para a configuração de múltiplos dispositivos em uma rede de computadores. Em especial, é tratada a configuração de dispositivos a partir do gerenciamento baseado em políticas. Nestes casos, a necessidade de operações atômicas vem do fato de que uma aplicação de política que falha em um determinado dispositivo pode levar a um estado inconsistente em uma rede da qual se deseja um comportamento uniforme e global. Assim, este trabalho analisa requisitos, limitações, implicações e alternativas de implementação para a obtenção de atomicidade na aplicação de políticas.

Como uma proposta de solução para este problema é definido um protocolo de consenso a ser utilizado entre os participantes da aplicação da política na rede. Tal protocolo é implementado utilizando Web Services e integrado junto ao sistema de gerenciamento baseado em políticas denominado QAME. Por fim, a solução é avaliada através de testes em um cenário de testes composto por um cluster de computadores.

Palavras-chave: Gerenciamento de Configuração, Gerenciamento de Redes Baseado em Políticas, Operações Atômicas.

Atomic Operations for Policy-Based Network Management

ABSTRACT

This work presents an investigation of the viability of performing atomic operations during the configuration of multiple devices in a computer network. Specially, the configuration of devices using policy-based network management is addressed. In these cases, the need of atomic operations comes from the fact that a policy application that fails in a certain device can lead to an inconsistent state in a network from which an uniform and global behavior is desired. Thus, this work analyses requirements, limitations, implications, and alternatives for obtaining atomicity in policy deployment.

As a solution for this problem, this work defines a consensus protocol to be used by the participants in the policy deployment process. Such protocol was implemented using Web Services and integrated to a policy-based network management system named QAME. At last, the proposed solution is evaluated through a set of tests performed over a testing scenario composed of a high-performance cluster.

Keywords: Configuration Management, Policy-Based Network Management, Atomic Operations.

1 INTRODUÇÃO

A complexidade das redes de computadores atuais aumenta não somente em razão do número cada vez maior de dispositivos, mas também no que se refere a variedade dos mesmos. Com cada dispositivo possuindo funcionalidades e comportamentos particulares, o gerenciamento de redes torna-se uma tarefa indispensável para manter o funcionamento correto de uma rede (STALLINGS, 1998). O gerenciamento de redes é bastante abrangente e pode ser dividido em uma série de áreas funcionais, dentre as quais pode-se citar: gerenciamento de falhas, de configuração, de contabilização, de desempenho e de segurança (RAMAN, 1998). Destas, em especial, o gerenciamento de configurações é apontado atualmente na literatura como um dos tópicos de pesquisa de gerenciamento dos quais mais se requer melhorias (SCHöNWäLDER; PRAS; MARTIN-FLATIN, 2003). O gerenciamento de configuração é responsável por alterações no estado de um dispositivo, provendo-lhe dados com o propósito de mantê-lo funcionando corretamente e continuamente. Como, cada vez mais, se exige um funcionamento adequado e ininterrupto de uma rede, evidencia-se a relevância desta categoria de gerenciamento.

Além da necessidade de um gerenciamento de configuração adequado, existe o apelo por um gerenciamento de rede que trate o estado global da rede, e não apenas lide com instâncias isoladas de dispositivos. O tradicional gerenciamento orientado a dispositivo (SANCHEZ; MCCLOGHRIE; SAPERIA, 2001) muitas vezes é suficiente na configuração de dispositivos em uma rede de pequeno porte. Porém, em outros casos, para que um objetivo da rede seja atingido, é necessário que vários dispositivos sejam configurados de forma harmônica. Na automatização deste procedimento, o gerenciamento de configuração orientado a rede mostra-se como uma opção extremamente desejável. Nesse tipo de gerenciamento ocorre uma mudança de abordagem, de uma visão direcionada a instâncias de dispositivos, para uma visão do estado da rede como um todo, através de descrições de como um conjunto de dispositivos que compartilham uma mesma característica deve funcionar.

Uma alternativa de implementação de gerenciamento de configuração orientado a rede é a utilização de gerenciamento de redes baseado em políticas (PBNM - *Policy-based Network Management*) (WESTERINEN et al., 2001). O PBNM permite trabalhar com o conceito de políticas de gerenciamento, com o qual é possível definir agendamentos de configurações e é possível criar filtros e ações de gerenciamento baseadas em classes de dispositivos. Por isso, pode-se dizer que PBNM é uma abordagem efetiva para o gerenciamento de configuração orientado a rede.

Ainda que diversas arquiteturas PBNM tenham sido propostas pela comunidade científica nos últimos anos (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 relevantes. Na verdade, tal arquitetura não é formalmente de-

finida nos documentos do IETF, mas tem uma larga aceitação de seus elementos básicos, cujas funcionalidades são bem conhecidas. Nesta arquitetura, apresentada na Figura 1.1, uma ferramenta de políticas é utilizada pelo gerente da rede para definir políticas de alto nível (independentes de dispositivo) que são armazenadas em um repositório de políticas (RP) 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 específicas de cada dispositivo, que são aplicadas nos *Policy Enforcement Points* (PEPs), encontrados nos dispositivos de rede gerenciados.

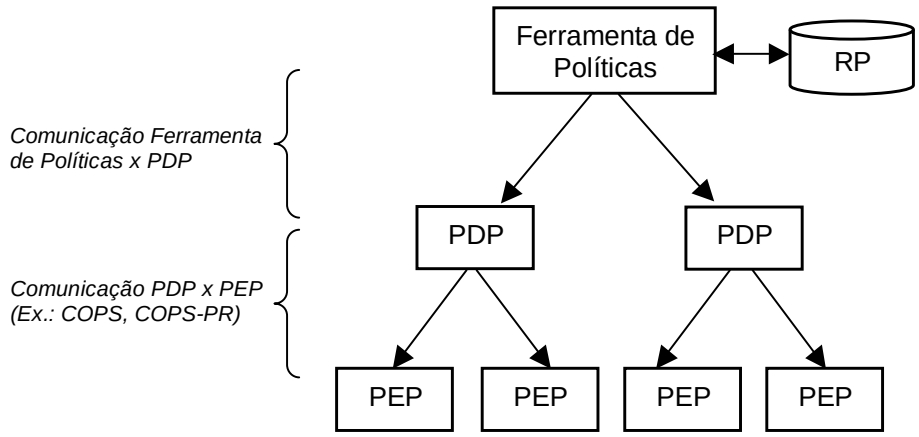

Figura 1.1: Arquitetura PBNM

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.

Falhas na aplicação de uma política podem afetar o gerenciamento orientado a rede quando este requer algumas propriedades especiais de funcionamento, dentre as quais ressaltamos neste trabalho a necessidade de configuração atômica de múltiplos dispositivos (SCHöNWäLDER; PRAS; MARTIN-FLATIN, 2003) (SANCHEZ; MCCLOGHRIE; SAPERIA, 2001). A configuração atômica determina que ou a totalidade de um determinado número de dispositivos seja devidamente configurada ou nenhum desses dispositivos seja. Algumas aplicações como reserva de banda com vistas a QoS (*Quality of Service*) e configuração de VPNs (*Virtual Private Networks*) requerem configuração atômica (ENNS, 2003). Para estas aplicações, o sucesso da configuração é completo apenas

quando todos os dispositivos de rede envolvidos são configurados adequadamente. Caso contrário, o funcionamento da aplicação é comprometido. Portanto, para estas aplicações, em um cenário onde pelo menos uma das configurações falhou, é preciso remover as configurações que obtiveram sucesso para que as mesmas não se tornem inúteis.

Atualmente, pode-se perceber que os protocolos de gerenciamento disponíveis (MACFADEN et al., 2003) (ENNS, 2003) lidam razoavelmente com garantias de atomicidade em configurações isoladas em um dispositivo. Porém, esses protocolos requerem uma capacidade adicional de gerenciamento para garantir atomicidade em procedimentos de configuração distribuídos entre diversos dispositivos. Esta necessidade também existe, independentemente dos protocolos utilizados, quando se considera o funcionamento da arquitetura PBNM. Por exemplo, para obter configurações atômicas utilizando PBNM seria necessário reunir os resultados da aplicação de uma política em cada um dos PEPs (*Policy Enforcement Points*) a que a ela se destina. De posse de tais resultados a ferramenta PBNM dispararia uma confirmação da transação ou uma ordem para que as configurações sejam desfeitas. Porém, até o momento, não são encontrados trabalhos lidando com garantias de atomicidade na configuração de múltiplos dispositivo via PBNM.

Assim, o objetivo desta dissertação é propor uma solução para a obtenção de configurações atômicas no contexto do gerenciamento de redes baseado em políticas (PBNM), apresentando requisitos, limitações, implicações e alternativas de implementação. Tal configuração atômica é importante para capacitar o sistema PBNM a garantir que uma política ou está aplicada em todos os PEPs de interesse, ou não está aplicada em nenhum PEP. Ou seja, se a aplicação de uma política falhar em um determinado PEP, todos os outros PEPs onde a mesma política já foi aplicada devem realizar *rollback* para o estado imediatamente anterior à aplicação da política.

Para prover aplicação atômica de políticas, nesta dissertação é definido um protocolo de consenso para suportar a comunicação entre a ferramenta de políticas e os PDPs. Tal comunicação foi implementada através do uso da tecnologia de Web Services (CURBERA et al., 2002), a qual provê, como será discutido no decorrer deste trabalho, importantes características necessárias para a implementação do protocolo proposto. Esse protocolo foi então integrado ao sistema de gerenciamento baseado em políticas para Serviços Diferenciados (DiffServ) denominado QAME (*QoS-Aware Management Environment*) (GRANVILLE et al., 2001) e avaliado através de medições de desempenho em um cenário de testes composto por um cluster de computadores.

O conteúdo desta dissertação está organizado como segue. O capítulo 2 contextualiza o problema investigado, descrevendo o gerenciamento de configuração, gerenciamento baseado em políticas e apresentando um exemplo de necessidade de atomicidade utilizando PBNM. O capítulo 3 apresenta a solução proposta através da descrição detalhada do protocolo de consenso definido. Os detalhes de implementação deste protocolo são descritos no capítulo 4. No capítulo 5 é apresentada a análise da solução e, finalmente, o capítulo 6 apresenta considerações finais e trabalhos futuros.

2 CONFIGURAÇÕES ATÔMICAS E GERENCIAMENTO DE REDES BASEADO EM POLÍTICAS

O gerenciamento de redes é, cada vez mais, uma atividade importante para manter as mesmas operando corretamente. Atualmente, para realizar tarefas de gerenciamento de redes, o uso de ferramentas de *software* é uma necessidade, dado o substancial aumento do número de dispositivos a serem gerenciados (que impossibilita um tratamento individual para cada dispositivo por parte do administrador) e dada a necessidade de procedimentos automatizados de configuração, monitoração e contabilização, entre outros.

Os objetivos e questões que o gerenciamento de redes propõe-se a resolver são diversos. Uma classificação das áreas funcionais de gerenciamento bastante aceita é a denominada FCAPS (*Fault*, *Configuration*, *Accounting*, *Performance* e *Security*) (RAMAN, 1998). Abaixo segue um breve descrição de cada uma das áreas funcionais FCAPS.

- *Fault* (Gerenciamento de Falhas): possibilita a detecção, o isolamento e a correção de procedimentos anormais nos dispositivos gerenciados.

- *Configuration* (Gerenciamento de Configuração): permite o controle do dispositivo, além de prover dados ao dispositivo com o propósito de dar manutenção para seu funcionamento correto e contínuo.

- *Accounting* (Gerenciamento de Contabilização): tem o objetivo de obter estatísticas a respeito do uso de um dado dispositivo através do acúmulo de informações.

- *Performance* (Gerenciamento de Desempenho): proporciona facilidades para a avaliação do comportamento de dispositivos gerenciados, possibilitando a identificação de funcionamento com qualidade abaixo da desejada.

- *Security* (Gerenciamento de Segurança): possibilita o gerenciamento de componentes ligados à proteção dos dispositivos gerenciados.

Todas as áreas funcionais FCAPS envolvem procedimentos de monitoração e de controle. Contudo, a ênfase nas áreas de gerenciamento de desempenho, de falhas e de contabilização é na monitoração, enquanto nas áreas de gerenciamento de configuração e de segurança a ênfase é no controle.

Por monitoração entende-se o ato de consultar dados a respeito dos dispositivos para a partir destes obter estatísticas e perceber comportamentos anormais. Por controle entende-se o ato de modificar parâmetros e disparar ações pré-definidas em dispositivos, alterando seu estado de configuração.

O gerenciamento de configuração, como um tópico principal deste trabalho, é descrito mais detalhadamente a seguir. Uma classificação possível para as funções do gerenciamento de configuração é apresentada em (STALLINGS, 1998). Os principais itens dessa classificação são apresentados abaixo de forma breve.

- Definição das informações de configuração: Esta funcionalidade é a responsável por descrever a natureza e o estado dos recursos a serem configurados. As informações de configuração incluem uma descrição dos recursos e dos atributos que podem ser configurados. Os recursos de rede podem ser físicos (roteadores, switches, sistemas finais, modems, serviços de comunicação, etc.) e lógicos (contadores, temporizadores e circuitos virtuais). Os atributos incluem, por exemplo, nome, endereço, número de identificação, estado, características operacionais, número de versão de *software*, etc. Esta funcionalidade deve proporcionar ao usuário a capacidade de especificar os intervalos e tipos de valores que podem ser ajustados nos atributos dos recursos de rede.

- Ajuste e modificação de valores de atributos: Esta funcionalidade deve habilitar uma estação remota a ajustar/modificar valores de atributos em agentes. Tais alterações, por questões de segurança, devem ser liberadas apenas para gerentes que estejam autorizados a tanto. A modificação de um atributo obviamente altera o estado de configuração de um agente.

- Definição e modificação de relacionamentos: Um relacionamento descreve uma associação, conexão ou condição que existe entre recursos de rede ou componentes de rede. Exemplos de relacionamentos são uma topologia, uma hierarquia, uma conexão física ou lógica, ou um domínio de gerenciamento. O gerenciamento de configuração deve permitir modificação de recursos sem interromper e/ou comprometer a rede ou parte dela. O usuário deve poder adicionar, remover e modificar os relacionamentos entre recursos de rede.

- Distribuição de *software*: O gerenciamento de distribuição deve prover a capacidade de distribuir *software* através da configuração de sistemas finais (*hosts*, servidores) e sistemas intermediários (*switches*, roteadores, *gateways* nível 7). Além disso, deve disponibilizar facilidades para permitir requisições de carga de *software*, transmissões de uma dada versão de *software* e atualizações das informações de versões.

Portanto, como visto nas particularidades descritas acima, pode-se considerar que o gerenciamento de configuração, principalmente por envolver alterações nos dispositivos (ao contrário da simples monitoração), é um tipo de gerenciamento que requisita atenção especial por parte do administrador da rede.

A configuração isolada de um dispositivo resolveu, e resolve até os dias de hoje, boa parte das necessidades dos usuários no gerenciamento de dispositivos, serviços e protocolos de rede. Porém, esta solução não tem se mostrado totalmente suficiente em alguns casos, tais como aqueles onde o administrador da rede necessita gerenciar o comportamento global da rede, exigindo a configuração coordenada dos dispositivos. O gerenciamento orientado à rede tem por objetivo lidar com este tipo de requisito, além de desempenhar importante papel na automatização dos procedimentos de gerenciamento, como será visto na próxima seção deste texto.

Quando se utiliza o conceito de gerenciamento orientado a rede, o foco principal das atividades de gerenciamento deixa de ser um dispositivo em particular, e passa a ser um

conjunto de dispositivos dos quais se quer um determinado comportamento global. Atualmente, sem a utilização de gerenciamento orientado à rede, para que um objetivo global da rede seja atingido é necessário que uma série de dispositivos sejam configurados individualmente. Para isto, os administradores da rede são forçados a interagir com as particularidades de cada um dos dispositivos relacionados, ou seja, a configuração de uma rede inteira nada mais é que uma série de configurações simples de cada um dos dispositivos.

Um exemplo que ilustra bem esta necessidade de configuração da rede para atingir um objetivo global é a definição de garantias de qualidade de serviço (QoS - *Quality of Service*) em uma rede (Figura 2.1). Se um dispositivo de rede O quer comunicar-se com outro dispositivo D, e os mesmos estão interligados por um caminho formado por uma série de roteadores R1, R2, R3 e R4, é necessário que cada roteador deste caminho seja configurado de modo a implementar tais garantias de QoS. A configuração de reserva de banda apenas nos roteadores R1, R3 e R4 pouco adiantará, visto que o roteador R2 não está configurado e se tornará um gargalo para a comunicação entre os dispositivos O e D. Neste exemplo, o objetivo global não foi atingido. Por isto, idealmente, as configurações dos roteadores do exemplo não deveriam ser tratadas pelo sistema de gerenciamento como quatro ações independentes, e sim como um procedimento único. Assim, o administrador da rede definiria o comportamento desejado para uma classe de dispositivos, e deixaria para o sistema de gerenciamento a tarefa de obter um resultado completo através da configuração da rede como um todo.

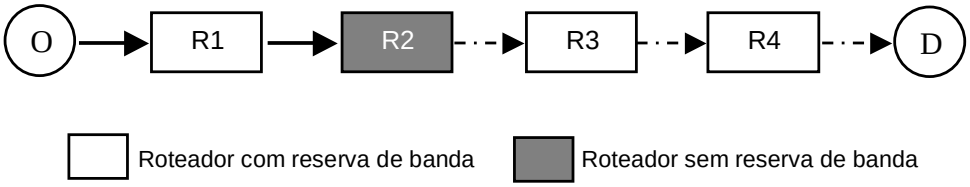

Figura 2.1: Exemplo de caminho de roteadores com reserva de banda comprometida.

O gerenciamento de redes baseado em políticas ou, em inglês, *Policy-Based Network Management* (PBNM) (YAVATKAR; PENDARAKIS; GUERIN, 2000) tem se mostrado uma técnica efetiva para implementar tanto gerenciamento de configuração e quanto gerenciamento orientado à rede. Em razão da crescente relevância destes dois tópicos, a solução para obtenção de configurações atômicas proposta nesta dissertação considera a utilização de gerenciamento baseado em políticas, descrito a seguir.

2.1 Gerenciamento de Redes Baseado em Políticas

Cada vez mais o gerenciamento de redes baseado em políticas vem sendo estudado, utilizado e adotado por fabricantes de dispositivos de rede. Muitos trabalhos relacionados vêm sendo realizados sobre este tema, sendo este tópico presença freqüente em simpósios que abordam pesquisas na área de gerenciamento de redes (CHADHA et al., 2004) (EDGAR, 2004) (SHERIDAN-SMITH, 2003).

Segundo definição encontrada na literatura (WALDBUSSER; SAPERIA; HONGAL, 2004), o gerenciamento baseado em políticas é a prática de aplicar operações de gerenciamento em diversos elementos gerenciados que compartilham certos atributos.

O PBNM (*Policy-Based Network Management*) facilita o gerenciamento de redes na medida que automatiza tarefas dos administradores. Além disso, permite um maior poder de expressão nas ações de gerenciamento, porque as políticas são descrições genéricas,

independentes de implementação e de fabricante, formatadas no estilo *SE-ENTÃO*, contendo condições (*SE*) e ações (*ENTÃO*). Nos exemplos desta dissertação são utilizadas 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 também investigado pela comunidade de pesquisa em políticas. Portanto, considerando o formato condição-ação, políticas são como segue:

```
SE (condição)
ENTÃO (ação)

SE (um elemento tem dada característica)
ENTÃO (aplicar uma configuração neste elemento)

SE (um pacote tem como destino determinada rede)
ENTÃO (priorizar este pacote)
```

A automatização das tarefas dos administradores é obtida principalmente pelo poder de expressão das condições que podem ser definidas, onde, por exemplo, pode-se expressar o comportamento desejado para uma dada situação ou para toda uma classe de dispositivos com alguma característica em comum. Além da definição de tradicionais itens de fluxos de dados (protocolos, endereços de origem e destino, portas de origem e destino, entre outros) pode-se criar agendamentos que funcionam como condições temporais para o disparo de uma política. A Figura 2.2 apresenta um exemplo prático de uma política de reserva de banda de 10 Kbits/s para um fluxo HTTP oriundo da máquina '143.54.47.240' e destinado à máquina '176.16.40.175' durante as sextas-feiras de 2006 entre as 20 horas e as 21 horas:

```
se  ((DIA >= 01/01/2006) e (DIA <= 31/12/2006) e
    (DIA_SEMANA == Sexta-feira) e
    (HORA >= 20:00) e (HORA <= 21:00) e
    (IP_ORIGEM == 143.54.47.240) e
    (IP_DESTINO == 176.16.40.175) e
    (PROTOCOLO == HTTP))
então
    Banda = 10 Kbits/s
```

Figura 2.2: Exemplo de política

Mais formalmente, o termo política pode ser definido de duas maneiras diferentes, mas não contradizentes (WESTERINEN et al., 2001):

1. Um objetivo ou método de ação para guiar e determinar decisões presentes e futuras. Políticas são implementadas ou executadas dentro de um contexto (escopo) próprio;

2. Políticas são conjuntos de regras para administrar/gerenciar recursos de rede, além de controlar acesso aos mesmos;

As políticas descrevem ocasiões onde determinadas ações devem ser tomadas. Porém, esta descrição por si só não é suficiente. Na arquitetura do gerenciamento de redes baseado em políticas (PBNM) do IETF, outros dois conceitos importantes são o de PDP

(*Policy Decision Point*) e de PEP (*Policy Enforcement Point*). A Figura 2.3 apresenta os componentes da arquitetura PBNM do IETF e as interações entre os mesmos.

A ferramenta de políticas possibilita ao administrador da rede a criação, alteração e exclusão de políticas, as quais são armazenadas em um repositório de políticas (RP).

O PDP é o *software* responsável por constantemente avaliar as políticas para verificar se suas condições (*SE*) estão satisfeitas em um dado momento. Quando as condições de uma política são satisfeitas, o PDP dispara um procedimento de ativação da política nos PEPs alvo, através da tradução das ações descritas em alto nível (*ENTÃO*) para ações específicas de configuração a serem enviadas aos PEPs alvo. Por outro lado, quando as condições de uma política deixam se estar satisfeitas, o PDP procede uma desativação (remoção) das configurações anteriormente enviadas para os PEPs alvo.

O PEP é um ponto passível de aplicação de políticas, podendo ser qualquer funcionalidade configurável em um dispositivo. Não é raro que seja chamado PEP o dispositivo a ser configurado, porém a denominação que parece mais criteriosa e que certamente é mais encontrada na literatura é a que afirma, como dito anteriormente, que um PEP é uma funcionalidade configurável de um dispositivo.

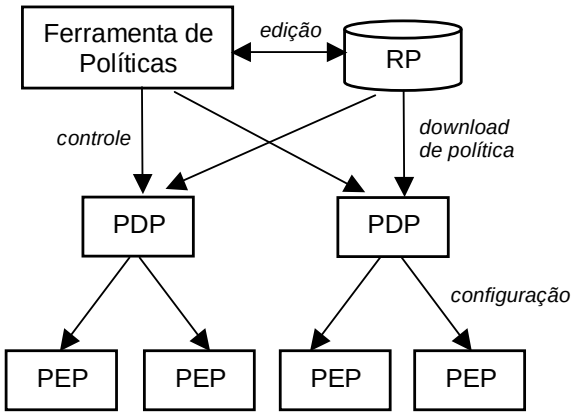

Figura 2.3: Arquitetura PBNM com transferência da política seguindo a abordagem *pull*

A aplicação de uma política envolve, considerando a arquitetura PBNM do IETF, no mínimo duas fases de comunicação entre os elementos da arquitetura: a comunicação entre a ferramentas de políticas e o PDP, e a comunicação entre o PDP e o PEP.

Na primeira fase, 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 (Figura 2.3). A comunicação entre a ferramenta de políticas e os PDPs é dependente de implementação. Exemplos de alternativas para realizar esta comunicação são SNMP (CASE et al., 1990) e Web Services (CURBERA et al., 2002).

A segunda fase de comunicação ocorre quando um PDP precisa interagir com seus PEPs para realizar a ativação ou a desativação de uma 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 (*Resource reSerVation Protocol*) (BRADEN et al., 1997) deve ser

aceita. A comunicação entre um PDP e um PEP pode ser realizada, por exemplo, via SNMP, COPS (DURHAM et al., 2000) ou COPS-PR (CHAN et al., 2001).

Por simplicidade, neste texto será considerada somente a aplicação de políticas através da abordagem *provisioning*. Esta restrição será realizada também porque essa abordagem é a mais adequada para redes que implementam Serviços Diferenciados (DiffServ), e as políticas suportas pelo sistema QAME, base para este trabalho, destinam-se justamente à configuração de DiffServ. Além disso, a comunicação entre a ferramenta de políticas e os PDPs seguirá a abordagem *pull*, que é o tipo de transferência de política suportado pelo QAME.

2.2 PBNM e Configurações Atômicas

Ainda que o gerenciamento baseado em políticas proporcione funcionalidades adequadas para gerenciamento orientado a rede, este não contempla outro requisito do gerenciamento de configuração que é bastante desejado pelos administradores de rede: atomicidade. Pode-se dizer que existem dois tipos principais de operações realizadas em múltiplos dispositivos:

- Aquelas em que o sucesso na realização da operação em um dispositivo independe do resultado da operação nos outros dispositivos;

- Aquelas em que o sucesso na realização da operação em um dispositivo depende diretamente do resultado global da operação, ou seja, do resultado em cada um dos demais dispositivos envolvidos. A estas dá-se o nome operações atômicas.

O primeiro tipo de operações é tratado diretamente pelo gerenciamento de redes baseado em políticas. Porém, o segundo tipo de operações não possui suporte direto no PBNM. Entre os exemplos de aplicações para operações atômicas pode-se citar:

- Reserva de banda: um procedimento de reserva de banda tem por objetivo garantir um tratamento especial para um determinado fluxo de dados em um caminho que liga dois pontos distintos de uma rede. Assumindo que tal caminho pode conter mais de um dispositivo com funcionalidade de roteador, tem-se que uma reserva de banda só será completamente efetiva se todos os dispositivos intermediários estiverem devidamente configurados para suportá-la. No caso de um dispositivo não estar configurado, este se tornará um gargalo e comprometerá o resultado desejado, como visto anteriormente no exemplo da Figura 2.1.

- Configuração de VPN (*Virtual Private Network*): quando deseja-se realizar a configuração de uma VPN pode ser preciso configurar as máquinas envolvidas para prover este serviço. Assim, como no caso da reserva de banda, basta que uma máquina integrante da VPN não esteja devidamente configurada para que as comunicações que trafegariam por esta rede sejam comprometidas.

Nestes exemplos, uma simples falha na configuração de um dos dispositivos envolvidos no processo de aplicação determina o insucesso global da configuração. Existe uma diversidade de falhas que podem ocorrer durante a aplicação de uma política em um determinado PEP. A falha mais geral, e comum a todas as aplicações que se utilizam de PBNM, é a indisponibilidade de acesso ao PEP. Este problema pode ocorrer ou pelo fato

da comunicação estar interrompida entre o PDP e o PEP ou pelo fato do PEP não estar operando corretamente.

Além destas falhas gerais existem falhas específicas da aplicação na qual se está utilizando PBNM. Se a política a ser aplicada é, por exemplo, uma atualização de *software* pode-se imaginar um cenário no qual o PEP que deve receber a política não possui memória suficiente ou para processar a atualização ou para armazenar a nova versão de *software* recebida. Outro exemplo ocorre com a utilização de PBNM para configuração de reserva de banda de rede. Neste caso, é possível imaginar um cenário onde uma reserva de banda requerida de um PEP não possa ser atendida pelo fato de não haver banda disponível, seja pelo fato do PEP não suportar tal quantidade de banda ou por outras alocações de banda terem diminuído a capacidade momentânea de alocação do PEP.

Portanto, falhas não são raras quando se está configurando dispositivos em uma rede. Este cenário onde falhas naturalmente não podem ser evitadas leva a reafirmar a necessidade de um sistema de gerenciamento suportar garantias de atomicidade.

A seguir é apresentado um exemplo de uma aplicação de política na qual ocorre uma falha de ativação em um dos dispositivos envolvidos no processo. Tal exemplo ilustra a necessidade de atomicidade durante a configuração de múltiplos dispositivos utilizando PBNM.

2.3 Exemplo de Necessidade de Atomicidade Utilizando PBNM

No gerenciamento de QoS, a aplicação de uma política é essencialmente uma ação distribuída, como pode ser percebido no cenário da Figura 2.4, onde dois PDPs recebem uma política a ser aplicada em quatro roteadores.

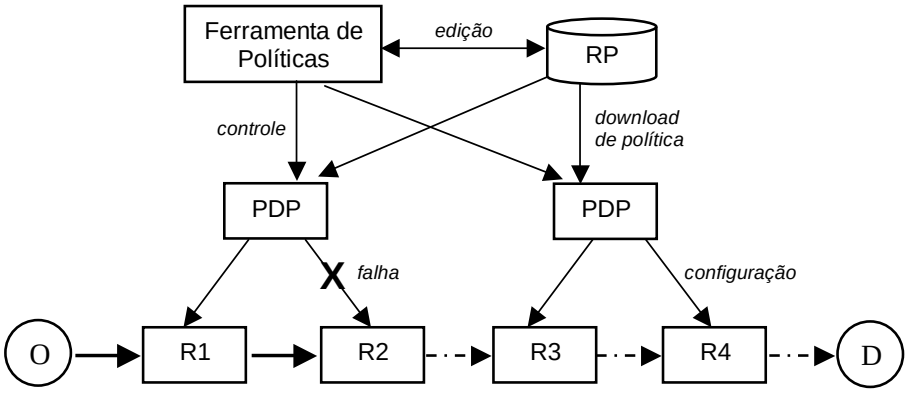

Figura 2.4: Aplicação de Política com Falha

Para realizar a reserva de banda entre os *hosts* de origem (O) e de destino (D), a ferramenta de políticas precisa transferir uma política do repositório de políticas (RP) para os PDPs que controlam os PEPs dos roteadores R1, R2, R3 e R4. A partir deste momento as condições da política passam a ser constantemente avaliadas pelos PDPs. Quando as condições da política são satisfeitas, os PDPs traduzem a política e aplicam a respectiva configuração em cada um dos roteadores.

No exemplo da Figura 2.4, a ativação da política é realizada com sucesso nos roteadores R1, R3 e R4, nos quais, portanto, a banda requerida é alocada. Contudo, a ativação da política falha no roteador R2, comprometendo globalmente a reserva de banda desejada para o caminho em questão. Além disso, há o problema de que os roteadores R1, R3 e R4 agora estão configurados com uma reserva de banda inútil, visto que o objetivo global

da política não pôde ser atingido. Para resolver esta situação os roteadores R1, R3 e R4 deveriam desfazer as configurações recém ativadas (efetuar *rollback*), de modo a liberar a banda alocada.

Realizar o *rollback* de uma aplicação de política envolve, a partir da ferramenta de políticas, a indicação explícita de que todos os PDPs devem realizar *rollback*. O mais importante, contudo, é notar que atualmente este procedimento de *rollback* distribuído é obtido somente com intervenção manual por parte do operador. Assim, evidencia-se a necessidade de que um sistema de gerenciamento baseado em políticas com suporte a atomicidade automatize não somente a aplicação da política (o que já é obtido utilizando agendamentos), mas também a tarefa de efetuar *rollback* caso a ativação da política falhe em algum dos PEPs alvo.

Como foi visto durante o decorrer deste capítulo, o gerenciamento baseado em políticas preenche lacunas importantes nas funcionalidades desejadas por um administrador de redes. Porém, as pesquisas efetuadas até o momento não trataram da necessidade de atomicidade em procedimentos de ativação/desativação distribuída de uma política. Assim, este trabalho tem por finalidade investigar a viabilidade e discutir soluções para a obtenção de atomicidade na aplicações de políticas seguindo a arquitetura PBNM do IETF.

3 SOLUÇÃO PROPOSTA

Como visto, o gerenciamento de configuração orientado a rede requer cuidados para evitar a criação de estados inconsistentes na rede quando falhas de configuração ocorrem. Assim, é desejável um mecanismo que, durante uma configuração, garanta a aplicação desta em todos os dispositivos ou, em caso de problemas, garanta a remoção da configuração dos dispositivos que já haviam sido configurados. Tem-se, neste caso, que o processo de configuração baseada na rede deve ser uma operação atômica.

A garantia contra falhas a ser tratada neste trabalho tem por objetivo assegurar que uma certa configuração seja aplicada corretamente em um conjunto de elementos de rede. Após a confirmação de que a configuração desejada foi aplicada com sucesso em todos os dispositivos, não serão considerados quaisquer tratamentos de possíveis erros que venham a surgir no sistema como um todo. O que se quer garantir é a correta ativação da política de acordo com os recursos disponíveis em um dado momento, e não a manutenção deste estado de configuração, nem uma possível adaptação a alterações na disponibilidade de recursos por parte dos dispositivos. Apesar destas características também serem extremamente desejáveis, estas não fazem parte do escopo deste trabalho e não serão abordadas. Assim, com esta dissertação, pretende-se dar suporte ao funcionamento coordenado de ativações e de desativações de políticas em múltiplos dispositivos.

3.1 Requisitos para a Obtenção de Aplicações Atômicas

Para a obtenção de atomicidade na aplicação de políticas é preciso que alguns pré-requisitos sejam atendidos pelos elementos envolvidos em um cenário que segue a arquitetura PBNM. De uma forma geral, tais requisitos podem ser divididos em três partes: questões relativas ao dispositivo de rede que contém PEPs, questões relativas a um PDP isoladamente e questões relativas à coordenação de vários PDPs.

3.1.1 Requisitos Relativos ao Dispositivo de Rede que Contém PEPs

Para a obtenção de aplicações atômicas dois conceitos fundamentais são o de perpetuar configuração (*commit*) e de desfazer configuração (*rollback*).

Por perpetuação de configuração entenda-se o ato de confirmar/ratificar uma certa configuração que foi aplicada em um PEP de um dispositivo de rede. A implementação de um *commit* de configuração em um dispositivo de rede pode variar bastante. Alguns dispositivos suportam o conceito de configuração candidata, onde as configurações podem ser efetuadas em uma memória temporária e posteriormente, quando se possuir maior segurança a respeito da corretude das alterações, podem ser copiadas para a memória em execução no dispositivo. Este é o caso, por exemplo, de dispositivos que suportam o pro-

tocolo NETCONF (ENNS, 2003). Porém, para outra boa parte dos dispositivos existentes, esta característica de funcionamento pode não existir, havendo apenas duas memórias disponíveis: a que está em execução e a memória não-volátil. Para estes dispositivos, pode-se aplicar a configuração diretamente em sua memória de execução. Nestes casos, a configuração funcionaria como um *commit* antecipado, onde caso uma falha aconteça, dispara-se um *rollback*, caso contrário, nenhuma ação é executada. Opcionalmente, pode-se definir que o *commit* perpetua a configuração salvando-a em uma memória não-volátil.

Já a capacidade de *rollback*, aquela através da qual um dispositivo de rede recupera seu estado anterior de configuração, descartando as alterações realizadas, nem sempre é suportada por um dispositivo. Novamente, os dispositivos que suportam NETCONF seriam um exemplo de capacidade de *rollback*, visto que nestes, quando for preciso, basta descartar a configuração candidata em andamento. Porém, para os dispositivos/protocolos em geral este conceito de *rollback* embutido pode não existir. Nestes casos, a responsabilidade pela capacidade de *rollback* pode ser transferida para o PDP, como será discutido durante a próxima seção.

3.1.2 Requisitos Relativos a um PDP Isoladamente

Um importante passo para a obtenção de um procedimento consistente de configuração é verificar se tal configuração realmente foi realizada. Cada configuração de um dispositivo deve, obrigatoriamente, ter seu resultado divulgado para o PDP que a desencadeou. A necessidade de obtenção do resultado de uma configuração não é exclusividade de operações atômicas. Porém, devido a sua importância, neste contexto, decidiu-se abordar brevemente sobre este tema neste documento.

Toda ação de configuração deve ser seguida por uma verificação de sucesso, para certificar-se de que a configuração ocorreu como esperado. Tal verificação é, muitas vezes, dependente do procedimento de configuração utilizado, podendo ser, no caso de uma configuração através de SNMP, o retorno da operação SNMP Set (*SNMP Response*). Alternativamente, o resultado de uma configuração poderia ser obtido, por exemplo, através de consultas a objetos de uma MIB que indiquem o estado atual de um elemento da rede. De uma forma geral, independentemente do protocolo utilizado para configurar um dispositivo, comandos de configuração retornam um resultado que indica o sucesso (ou não) do procedimento. Assim, em muitos casos, a tendência natural é a utilização deste resultado.

Uma vez que uma falha de configuração tenha sido detectada após a ativação de uma política em um PEP, cabe ao PDP desfazer as alterações que tenham sido produzidas com sucesso em outros PEPs, para manter assim a coerência exigida pela configuração atômica. A forma como o procedimento de *rollback* é efetuado depende do dispositivo e do protocolo utilizado para configurá-lo. Se o par dispositivo-protocolo possuir capacidade própria de *rollback* (como citado na seção anterior), basta o PDP invocar esta facilidade. No restante dos casos é preciso que o PDP assuma a responsabilidade por emular um procedimento de *rollback* que não existe de forma nativa no dispositivo em questão. Para isto, em geral, o PDP deve possuir os dados da política que cuja aplicação falhou. Com estas informações, o PDP pode ser capaz de montar um conjunto de instruções que anulem a configuração aplicada anteriormente, removendo-a. Com esta abordagem, sob o ponto de vista do PEP, uma configuração seguida de um *rollback* são, simplesmente, duas configurações em seqüência.

No caso de roteadores CISCO, por exemplo, basta incluir a *string* "no" antes de cada uma das instruções de configuração enviadas previamente para proceder a remoção das mesmas. No caso de roteadores AltQ, as diferenças entre as instruções para aplicação e

desaplicação não são tão diretas. Ainda assim, as instruções necessárias para uma desaplicação são deriváveis das informações originais da política. Outra alternativa, também dependente do protocolo e do dispositivo envolvido, é a existência de políticas padrão para utilização em situações de falha. Estas políticas poderiam estar armazenadas tanto no repositórios de políticas quanto em objetos do tipo *template* (MACFADEN et al., 2003) na MIB do PEP.

Os requisitos citados nesta seção podem tratar procedimentos de aplicações atômicas envolvendo, inclusive, múltiplos PEPs, porém sob o comando de apenas um PDP. Na próxima seção são analisados os requisitos para a coordenação de vários PDPs em uma mesma aplicação de políticas.

3.1.3 Requisitos Relativos à Coordenação de Múltiplos PDPs

Cenários onde uma rede é gerenciada por múltiplos PDPs não são difíceis de se obter na prática. Por exemplo, uma empresa com diversas filiais espalhadas geograficamente pode necessitar de comunicação privilegiada entre tais filiais. Para isto, pode haver em cada uma destas filiais, e, conseqüentemente, em cada domínio de rede, um PDP distinto.

Outro exemplo factível é a necessidade de divisão da carga de gerenciamento de vários dispositivos entre mais de um PDP. Isto porque há uma tendência de que quanto maior o número de PEPs controlados por um único PDP, maior a quantidade de políticas a serem avaliadas por este PDP, sobrecarregando-o. Assim, para reduzir a carga sobre um PDP pode-se introduzir mais PDPs na rede, dividindo a responsabilidade de gerenciamento dos PEPs entre estes PDPs.

Considerando-se que uma política tenha sido distribuída para uma série de PDPs, e estes tenham sido informados de que a política requer configuração atômica, percebe-se a necessidade de que tais PDPs entrem em acordo a respeito do sucesso global da ativação da política na totalidade dos PEPs. Assim, caberia a cada PDP, após realizar a configuração de seus PEPs, divulgar seus resultados locais de configuração aos outros PDPs. De posse destas informações, a obtenção do resultado global é direta.

Recuperando o exemplo da configuração de QoS em uma série de roteadores, e considerando a utilização de gerenciamento baseado em políticas neste cenário, pode-se considerar um caso em que cada roteador é controlado por um PDP diferente, responsável por aplicar a política de QoS em si. Neste contexto, o esperado é que cada PDP, após disparar uma ação de configuração em um roteador, espere pelo resultado desta operação. Este retorno é de obtenção relativamente simples, pois envolve apenas o dispositivo sendo configurado. Porém, não basta a um PDP saber se a sua ação de configuração obteve sucesso. Um PDP deve saber se todas as operações de configuração disparadas por outros PDPs também obtiveram sucesso. Quando o sucesso global ocorre, o processo de configuração como um todo entra em um estado global "*commited*". Caso contrário, os PDPs que obtiveram sucesso em suas configurações devem realizar um procedimento de *rollback*, retornando a rede a um estado consistente.

A divulgação dos resultados parciais de cada PDP pode se tornar um procedimento demasiadamente oneroso para a rede em termos de mensagens trocadas. Isto acontece, principalmente, se considerarmos uma solução de comunicação simples que seria o envio, por parte de cada PDP, de uma mensagem para cada um dos outros PDPs, totalizando o envio de (N - 1) mensagens de notificação de resultado local por parte de cada PDP (sendo N o número de PDPs envolvidos no processo). Certamente esta abordagem não apresentaria uma boa escalabilidade, visto que seriam necessárias (N - 1).N mensagens para divulgar todos os resultados locais dos PDPs. A Figura 3.1 ilustra esta situação.

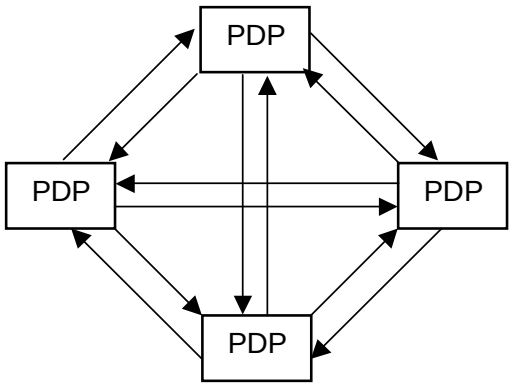

Figura 3.1: Divulgação de resultados parciais de todos para todos

Uma alternativa de implementação é a formação de um anel lógico entre os PDPs, onde um *token* percorre o anel e cada PDP altera o *token* indicando o sucesso de sua configuração local (Figura 3.2). Neste esquema, a topologia física da rede pouco importa, visto que o funcionamento em anel é totalmente lógico, pois cada elemento envolvido conhece seu vizinho imediato no anel e repassa o *token* para o mesmo. Esta solução envolve problemas clássicos de anéis lógicos como, por exemplo, a perda e a duplicação do *token* (TANENBAUM, 1995).

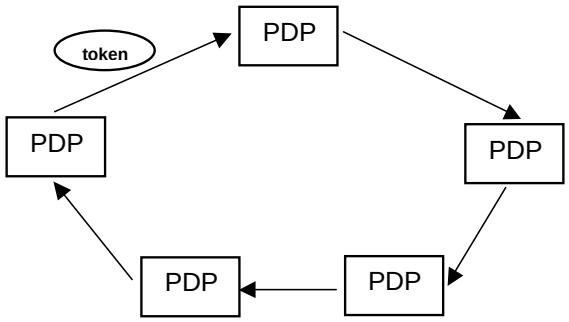

Figura 3.2: Divulgação de resultados parciais através de anel lógico

Nesta abordagem, mesmo após a ocorrência de uma falha, é necessário que o *token* percorra um ciclo completo pelo anel para que todos os PDPs tomem consciência da falha. Assim, após a falha, são necessárias N mensagens. No pior caso, ou seja, no caso mais oneroso, não existe falha e são necessárias N mensagens para todos PDPs confirmarem seu sucesso e mais N mensagens para todos PDPs descobrirem o sucesso alheio, totalizando 2.N mensagens.

Outra possibilidade seria a definição de um centralizador de mensagens, responsável por reunir as mensagens de resultado local de cada PDP e divulgar um resultado global para todos os PDPs (Figura 3.3). Nesta abordagem, são necessárias N mensagens para divulgação dos resultados locais mais N mensagens para divulgação do resultado global, totalizando 2.N mensagens.

Estas foram as principais alternativas de modelo de comunicação inter-PDPs estudadas durante o curso de mestrado. A escolha por um destes modelos deve ser realizada considerando não apenas aspectos relativos à redução do tráfego de rede gerado, mas também aspectos práticos de implementação, como será discutido na próxima seção.

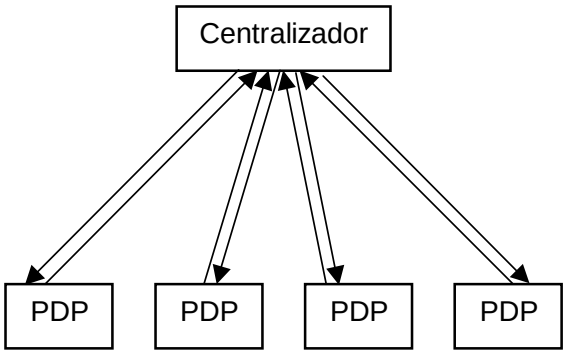

Figura 3.3: Divulgação de resultados parciais através de um centralizador

Resumindo os requisitos apresentados anteriormente, pode-se dizer que a principal questão em aberto com relação à obtenção de configuração atômica é a comunicação entre os PDPs em busca de um resultado global da aplicação de uma política. Esta interação é uma aplicação típica de algoritmos para resolução de problemas de consenso, clássicos de sistemas distribuídos.

3.2 Arquitetura PBNM com Suporte a Operações Atômicas

A partir das possibilidades de modelo de comunicação discutidas na seção anterior partiu-se para a definição de uma solução para a aplicação atômica de políticas envolvendo múltiplos PDPs.

O modelo de comunicação proposto nesta dissertação envolve a inclusão de um novo elemento na arquitetura PBNM. Este elemento é denominado *Policy Deployment Coordinator* (PDC) e seu objetivo é coordenar as sucessivas ativações/desativações atômicas de políticas, desempenhando o papel de um intermediador responsável por centralizar o recebimento do resultado local de cada PDP e divulgar um resultado global.

Neste modelo, cada PDP requer capacidade de comunicação apenas com o PDC, e não diretamente com outros PDPs. Logo, um PDP até mesmo desconhece os outros PDPs envolvidos em uma aplicação atômica de políticas.

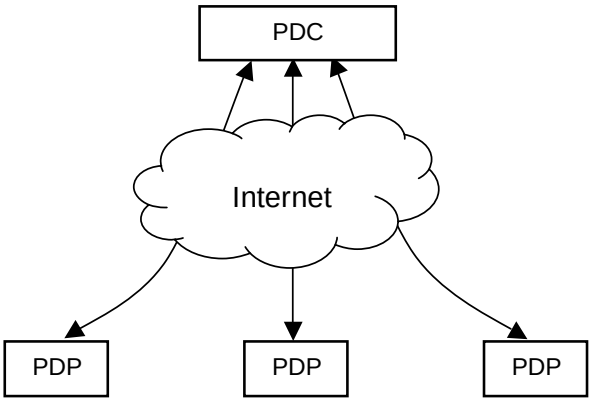

Figura 3.4: Modelo adotado: comunicação entre PDPs incluindo intermediador (PDC)

Como pode ser visto na Figura 3.4, pode-se imaginar PDPs espalhados por domínios administrativos distintos se comunicando através da Internet com o intermediador PDC. Por exemplo, quando um determinado PDP quer enviar uma mensagem para os outros PDPs, este envia a mensagem para o PDC, que por sua vez repassa tal mensagem para o restante dos PDPs.

Este modelo foi escolhido porque obrigar que cada PDP conheça o restante dos PDPs e, além disso, possua comunicação direta com cada um destes PDPs é um requisito muito forte. Por exemplo, quando se considera a possibilidade de haver PDPs espalhados por domínios administrativos distintos (ver Figura 3.5), sem a presença de um intermediador, teria-se a necessidade de que o acesso entre os PDPs fosse liberado nos *firewalls* de cada rede. Já com a presença do PDC, basta que cada PDP saiba comunicar-se com apenas um destino (PDC), simplificando o modelo.

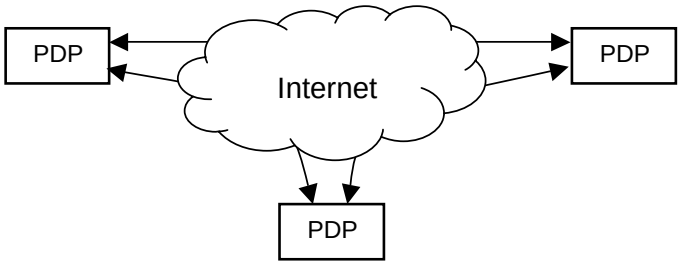

Figura 3.5: Modelo não adotado: comunicação direta entre PDPs

Assim, propõe-se um novo arranjo para a tradicional arquitetura PBNM através da inclusão do elemento PDC (ver Figura 3.6). O PDC comunica-se tanto com a ferramenta de políticas quanto com os PDPs. A ferramenta de políticas interage com o PDC para realizar o cadastro da aplicação atômica, informando quais os PDPs que participarão da aplicação. Os PDPs e o PDC interagem para que ocorra a coordenação dos PDPs em busca de um resultado global a respeito da ativação (ou desativação) de uma política.

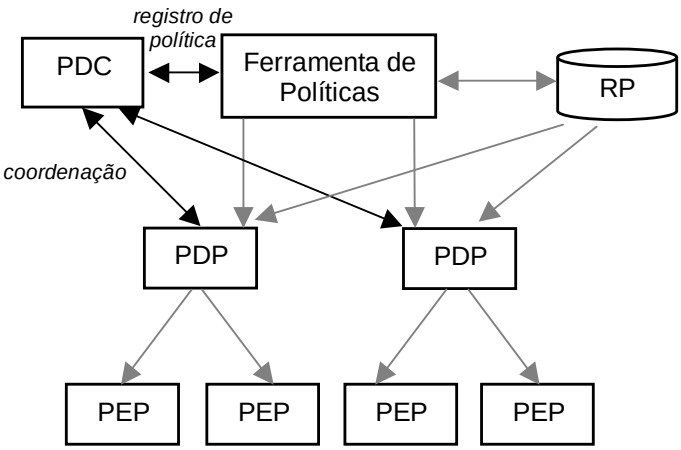

Figura 3.6: Arquitetura PBNM com a inclusão do PDC

Porém, cabe ressaltar que a definição de um modelo de comunicação, através da nova arquitetura PBNM proposta, não é suficiente. É preciso definir também o protocolo que será utilizado na interação entre o PDC e os PDPs, descrevendo suas mensagens. Na seção seguinte é apresentado em detalhes o protocolo de consenso utilizado nesta dissertação.

3.3 Protocolo de Consenso para Aplicação Atômica de Políticas

Um protocolo de consenso muito estudado e bastante conhecido da área de sistemas distribuídos é o protocolo de *commit* em duas fases (2PC - *Two Phase Commit Protocol* (GRAY, 1978)). Este protocolo engloba uma primeira fase de votação, onde cada integrante do sistema distribuído divulga sua capacidade (ou incapacidade) de realizar *commit*; e envolve uma segunda fase, onde cada integrante do sistema distribuído recebe uma ordem para efetivamente realizar *commit* ou então descartar as alterações. As Figuras 3.7 e 3.8 apresentam o modelo básico de funcionamento do protocolo 2PC quando, respectivamente, todos os participantes obtém sucesso e quando um dos participantes obtém falha.

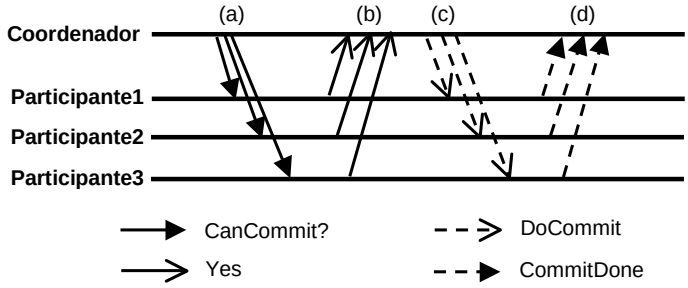

Figura 3.7: Comportamento do protocolo 2PC com sucesso de todos participantes

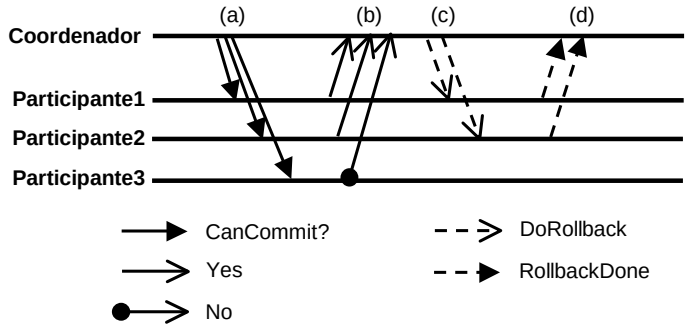

Figura 3.8: Comportamento do protocolo 2PC com falha em algum participante

O funcionamento do protocolo de consenso definido nesta dissertação basicamente segue a definição do 2PC. Porém, algumas alterações foram realizadas para adaptar seu funcionamento às necessidades específicas da aplicação atômica de políticas.

Como já foi citado nesta dissertação, políticas de gerenciamento são compostas por agendamentos, fluxos e ações. No gerenciamento baseado em políticas, para que uma política seja ativada em um dispositivo, é preciso que suas restrições de agendamentos sejam satisfeitas. Tal verificação é realizada pelos PDPs, os quais constantemente avaliam as políticas sob sua responsabilidade. O PDC definido neste trabalho não possui a funcionalidade de avaliar políticas, inclusive nem possui acesso às mesmas. A única função do PDC é coordenar a ativação das políticas. Por esse motivo, no protocolo definido nesta dissertação, o início do processo de ativação é sinalizado pelos PDPs, ao contrário do 2PC tradicional, onde o coordenador inicia o processo de votação.

Pelo fato de ser papel dos PDPs iniciar as ativações/desativações de políticas, evidencia-se a possibilidade de que uma política, durante um certo período de tempo, esteja ativa nos PEPs de um PDP e não esteja ativa nos PEPs controlados por outros PDPs, devido a

uma falta de sincronia entre os relógios internos dos PDPs. Recuperando o exemplo de política da Figura 2.2, onde a política deve estar ativa entre as 20 horas e as 21 horas, é trivial perceber que, com diferenças entre os relógios dos PDPs, a política pode estar ativa em momentos distintos se cada PDP utilizar somente seu relógio interno para definir o momento de realizar ativações/desativações (Figura 3.9).

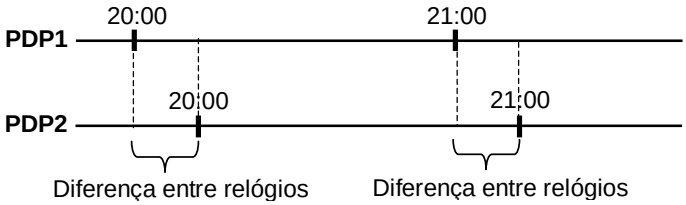

Figura 3.9: Diferença entre relógios internos dos PDPs faz com que a política esteja ativa em momentos diferentes.

Com o objetivo de fazer com que a ativação da política aconteça em um tempo mais próximo possível em cada um dos PDPs, decidiu-se por inserir um passo anterior à divulgação do resultado local de cada PDP: o envio de uma mensagem para o PDC que sinalize um início de ativação de política. O primeiro PDP que perceber que foi atingido o momento de ativação envia uma mensagem para o PDC e este a repassa para o restante dos PDPs. Deu-se a esta mensagem o nome de *InitiateActivation*, como será apresentado em detalhes a seguir. Assim, foi inserida uma etapa anterior ao passo (a) do 2PC, necessária porque no modelo proposto nesta dissertação o coordenador (PDC) não conhece o momento no qual ativações e desativações devem ser disparadas.

Em um ambiente onde se possui garantias de relógio que determinem uma proximidade significativa entre os horários dos diversos PDPs, é possível dispensar este passo do protocolo. Porém, quando tal proximidade não é garantida, o procedimento de *Initiate-Activation* é bastante útil, pois reduz possíveis desvios de tempo onde políticas poderiam ficar ativas apenas em alguns PDPs e não em outros.

Outra alteração no 2PC diz respeito ao passo (d) do protocolo. Na abordagem utilizada nesta dissertação, preferiu-se por ignorar o passo final de confirmação de sucesso na realização do *commit* e do *rollback*. Assim, assume-se que se a mensagem em questão foi entregue adequadamente, a operação foi realizada com sucesso. Tal decisão foi tomada por dois motivos. Uma operação de *commit* possui uma complexidade baixa, visto que a configuração enviada já está funcionando e que esta mensagem tem sua utilidade ligada praticamente apenas a uma confirmação de que as configurações tiveram sucesso no restante dos participantes. Já a operação de *rollback* é um pouco mais complexa e mais passível de não ser concluída com sucesso. Porém, no caso do protocolo apresentado nesta dissertação, é perfeitamente possível assumir que mesmo após um insucesso ao efetuar uma tentativa de *rollback* ordenada pelo Coordenador, o participante, por iniciativa própria, faça uma nova tentativa de *rollback*, como será discutido nas próximas seções deste capítulo.

Até o momento o protocolo de consenso definido nesta dissertação foi discutido em alto nível, traçando um paralelo com o 2PC. A seguir, na próxima seção, o protocolo é detalhado. São definidas as informações requeridas pelo protocolo, além das funções a serem exercidas por cada elemento de uma arquitetura PBNM que suporte aplicação atômica. Além disso, o funcionamento do PDC é descrito e as mensagens que integram a solução proposta são definidas.

3.3.1 Informações Relativas à Aplicação de Políticas

Um operador de rede que deseja aplicar uma política em uma rede deve, primeiramente, escolher uma política *p* de um 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. Tipicamente, em uma ferramenta de gerenciamento baseada em políticas, esta associação entre PDPs e PEPs é realizada pelo gerente da rede durante a fase de inserção e configuração dos dispositivos no sistema de gerenciamento. Em resumo, as informações relativas a uma aplicação de política são:

- *p* é a política a ser aplicada;
- *tPeps*={pep1, pep2, ..., pepn} é o conjunto de PEPs alvo.
- *tPdps*={pdp1, pdp2, ..., pdpm} é o conjunto de PDPs alvo.
- *deplId* é o identificador da aplicação de *p* em *tPeps*.

Como exemplo, considere a rede da Figura 3.10, onde se deseja realizar uma reserva de banda entre o dispositivo de origem (O) e o dispositivo de destino (D).

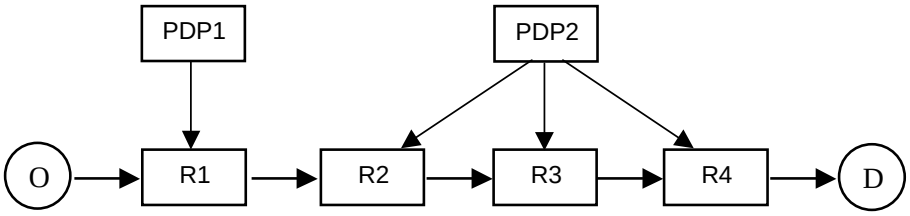

Figura 3.10: Exemplo de uma rede com PDPs associados aos PEPs, formando um caminho passível de aplicação de políticas entre dois *hosts* O e D

Para a rede da Figura 3.10, as informações relativas à aplicação seriam:

- *p* pode ser, por exemplo, a política apresentada na Figura 2.2, considerando que O possui endereço IP 143.54.47.240 e D possui endereço IP 176.16.40.175;
- *tPeps*=R1, R2, R3, R4;
- PDP1 está associado com R1, PDP2 está associado com R2, R3, R4;
- *tPdps*=PDP1, PDP2.

Dada a definição de *p*, *tPeps*, *tPdps*, *deplId*, e das associações entre PDPs e PEPs, a ferramenta de políticas deve contatar os PDPs de *tPdps* e transferir 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.

3.3.2 PDPs e os Estados de uma Política

Uma política *p* dentro de um PDP possui um estado: inválida, inativa ou ativa. Para exemplificar esses diferentes estados, considere novamente a política de exemplo da Figura 2.2.

Esta definição de política declara que sua data de validade tem início no dia 1 de janeiro de 2006. Contudo, é possível que o operador da rede, junto à ferramenta de políticas, transfira a política para os PDPs alvo antes de 1 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 atingida. A política também define que a data de encerramento é 31 de dezembro de 2006. Depois dessa data, a política torna-se inválida, e 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. Estando válida, a política passa a ser avaliada também por suas outras restrições, tais como, neste exemplo, hora, dia da semana, IP de origem, IP de destino e protocolo. Apenas quando a política passar para o estado ativa é que efetivamente haverá configuração dos PEPs alvo. A política de exemplo (Figura 2.2) passa para o estado ativa - e então aloca 10 Kbits/s de banda para tráfego HTTP oriundo da máquina 143.54.47.240 com destino para a máquina 176.16.40.175 - somente nas sextas-feiras, às 20:00. Às 21:00, a política não estará mais ativa, retornando para o estado inativa e, conseqüentemente, disparando um procedimento de remoção (desativação) dos comandos enviados ao PEP durante a ativação. É importante não confundir a desativação de uma política com a remoção da mesma. Por desativação entende-se remover os comandos de configuração enviados, sendo que a política volta a ter suas condições (fluxos) avaliados novamente. Por remoção da política entende-se, além da desativação da política, a exclusão da política da base de dados do PDP, deixando de existir. A Figura 3.11 apresenta um exemplo de seqüência de mudança de estados de uma política em um PDP.

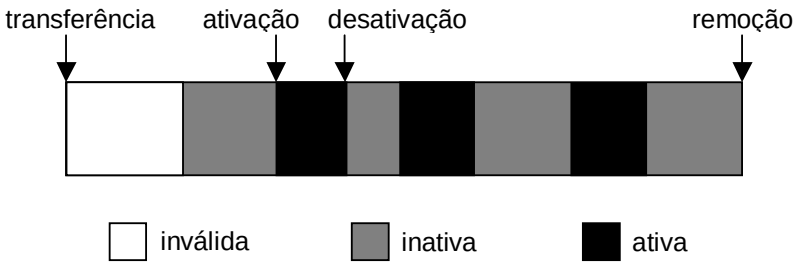

Figura 3.11: Estados de uma política dentro de um PDP

3.3.3 Policy Deployment Coordinator

A aplicação atômica de políticas é obtida através da ativação/desativação atômica. Cada vez que uma política é ativada em um PDP, ela deve ser ativada também nos outros PDPs. Quando uma política for desativada, ela deve ser desativada também nos outros PDPs. É na intermediação dos resultados locais de cada PDP e na divulgação de um resultado global que o PDC tem seu papel principal.

O PDC coordena as ativações de uma política usando uma estrutura cujas entradas são indexadas pelo identificador de aplicação (*deplId*). Tal estrutura (registro de aplicação) inclui um inteiro que indica o estado corrente de uma aplicação (*deplState*), e um vetor de

PDPs que guarda o estado de ativação da política nos PDPs envolvidos (*deplVector state*). A verificação deste vetor possibilita que o PDC identifique o estado global de ativação de uma política. Em resumo:

- *DeplKey*: a identificação única de uma aplicação de política (*deplId*);
- *DeplState*: [0: Inactive, 1: Activating, 2: Active]
- *DeplVector PDPs*: o conjunto de PDPs para esta aplicação de política (*tPdps*)
- *DeplVector state*: o estado de cada PDP [0: Unknown, 1: Success, 2: Failure]

Dado o PDC recém introduzido, e as informações de política definidas, as próximas subseções apresentam as operações do protocolo proposto.

3.3.4 Transferência da Política e Operações de Ativação

Depois que o operador da rede seleciona a política e os PEPs onde a política deve ser aplicada, a ferramenta de políticas deve obter *deplId* e *tPdps*. Então, é enviada uma mensagem RegisterPolicyDeployment ao PDC de forma a criar um registro de aplicação para *p* no PDC. Esta mensagem carrega como parâmetros 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 com o valor 0 (Inactive), e o estado de cada elemento de *DeplVector state* é iniciado com o valor 0 (Unknown).

Após registrar a aplicação da política no PDC, a ferramenta de políticas transfere a política *p* para todos os PDPs contidos em *tPdps* através da mensagem TransferPolicy. Esta mensagem carrega como parâmetros 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, como pode ser visto na Figura 3.12.

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 sincronizados, a ativação de uma política *p* pode ser disparada em diferentes momentos em diferentes PDPs. O primeiro PDP a ativar *p* deve notificar o PDC enviando a mensagem InitiateActivation, e então proceder com a execução das ações da política. O PDC trata somente a primeira mensagem InitiateActivation que ele recebe. Após o recebimento desta mensagem, o PDC passa a ignorar novos recebimentos de InitiateActivation.

Quando notificado, o PDC altera *DeplState* para "Activating" e notifica todos os outros PDPs enviando a mensagem InitiateActivation para cada um deles. Esta mensagem ordena que os PDPs restantes iniciem a ativação de *p*.

A Figura 3.12 apresenta um exemplo de interações entre PDC e PDPs desde a transferência da política até o início da ativação da política.

Uma vez que um PDP tenha recebido uma mensagem InitiateActivation, ele não mais envia esta mensagem para o PDC, mesmo que seja atingido um horário de início de ativação. Isto se justifica pelo fato de que a aplicação atômica já teve início (indicado pelo recebimento de InitiateActivation) e, portanto, não há razão para o PDP comunicar ao PDC que a política tornou-se ativa.

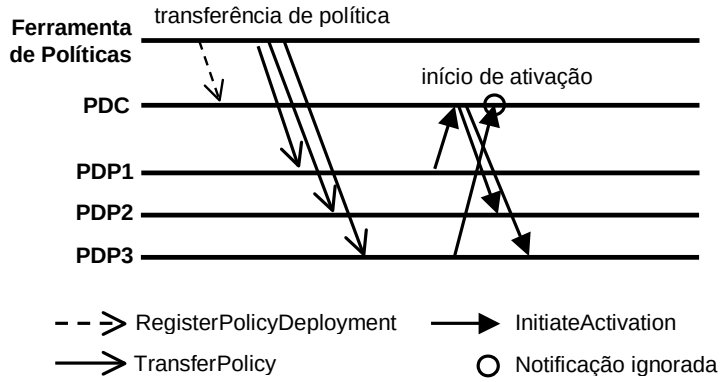

Figura 3.12: Transferência da Política e Início de sua Ativação

Ainda com relação à transferência da política e sua ativação, é preciso notar que uma política pode já estar no estado "Active" no momento em que for transferida para um PDP. Assim, o PDP imediatamente enviaria uma mensagem InitiateActivation para o PDC, desencadeando o envio de uma mensagem do PDC para cada PDP envolvido na aplicação atômica (Figura 3.13).

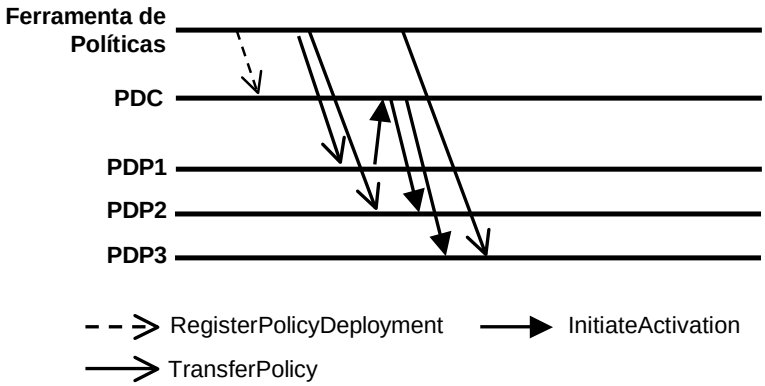

Figura 3.13: Exemplo de situação em que um PDP pode receber uma mensagem InitiateActivation antes de ter recebido a transferência da política

Portanto, como a transferência da política para cada um dos PDPs é um procedimento independente, sujeito a atrasos locais no PDP ou até mesmo atrasos na rede, é possível que o PDC envie uma mensagem InitiateActivation para um PDP (que no caso do exemplo da Figura 3.13 é o PDP3) que ainda nem recebeu a política e que ainda desconhece tal aplicação atômica. Neste caso, a mensagem será descartada pelo PDP3, visto que este nem mesmo possui a aplicação atômica em questão em sua base de dados. Por causa deste descarte, a ativação como um todo não terá sucesso, visto que o PDP3 não participará corretamente dos próximos passos do protocolo (apresentados a seguir). Para reduzir os casos em que isto pode ocorrer é possível configurar em cada PDP um tempo pelo qual ele esperará antes de divulgar um início de ativação para uma política recém cadastrada em sua base de dados. Assim, ao receber uma política, o PDP espera um intervalo de tempo antes de enviar InitiateActivation, diminuindo a probabilidade de que o PDC consiga enviar InitiateActivation para um PDP que ainda não tenha recebido a política. Este intervalo de tempo tem o objetivo de minimizar a ocorrência de casos em que a fase de início de ativação inicie antes da conclusão da fase de transferência da política.

3.3.5 Commit e Rollback de Ativações de Políticas

Depois da fase de comunicação responsável por avisar que é chegado o momento de iniciar uma ativação (mensagens InitiateActivation), os PDPs partem para a configuração dos PEPs. Neste momento, falhas podem ocorrer, e o PDC precisa ser notificado. Enquanto está no estado "Activating", o PDC aguarda pelos resultados das ativações que serão enviadas pelos PDPs.

Assim que um PDP configura com sucesso todos os PEPs associados, ele deve enviar a mensagem ActivationSuccess para o PDC. Caso o PDP não consiga enviar esta mensagem ao PDC, então o PDP assume que a ativação fracassou e desfaz as configurações realizadas localmente. Para cada chamada ActivationSuccess, o PDC altera o estado do PDP que enviou esta mensagem para "Success" em *DeplVector state*. Quando todas as posições de *DeplVector state* estão iguais a "Success", o PDC envia uma mensagem Commit para cada PDP, indicando que todo o procedimento de ativação global obteve sucesso (Figura 3.14). Neste ponto, a ativação da política se encerra com sucesso e o PDC altera *DeplState* para 2 (Active).

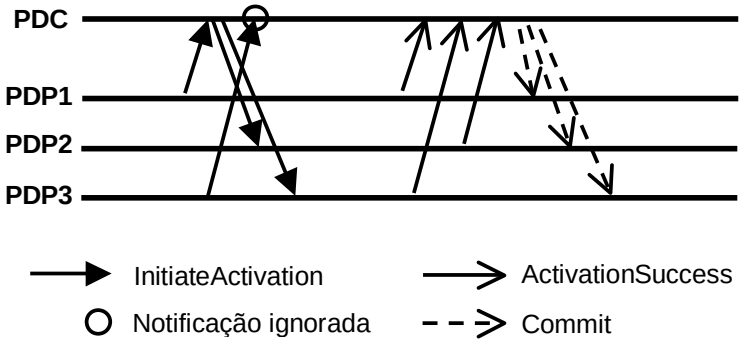

Figura 3.14: Início da ativação e Commit

Por outro lado, caso o PDC não consiga enviar a mensagem de Commit para algum PDP, e outros PDPs já tenham recebido esta mensagem, cabe ao PDC enviar a mensagem Abort para cada PDP que anteriormente tenha recebido a mensagem Commit, como pode ser visto na Figura 3.15. Para os demais PDPs, exceto para o PDP que estava incomunicável, o PDC envia a mensagem Rollback. Neste caso, o PDC altera *DeplState* para 1 (Inactive) e *DeplVector State* para 0 (Unknown).

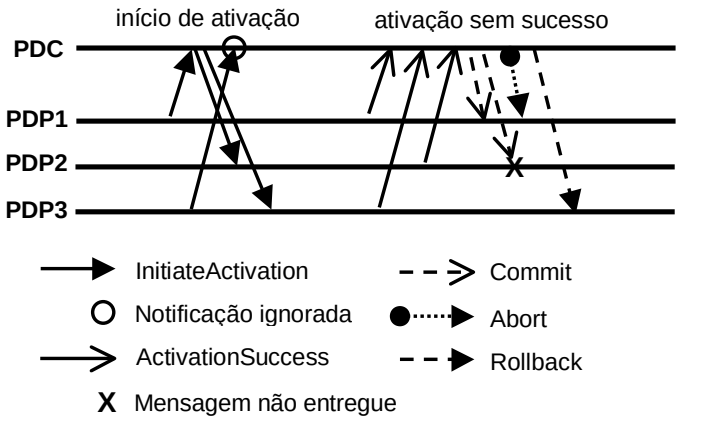

Figura 3.15: Falha na divulgação de Commit e conseqüente Abort

Se, contudo, algum PDP envia uma mensagem ActivationFail para o PDC, então o PDC imediatamente comunica o insucesso da ativação a todos os outros PDPs, exceto o PDP que enviou a mensagem de falha através da mensagem Rollback (Figura 3.16). O PDC pode receber uma mensagem ActivationFail antes mesmo de ter acabado de enviar InitiateActivation para todos os PDPs. Neste caso, o PDC interrompe o envio dessas mensagens e envia Rollback para todos os PDPs, exceto o que enviou ActivationFail.

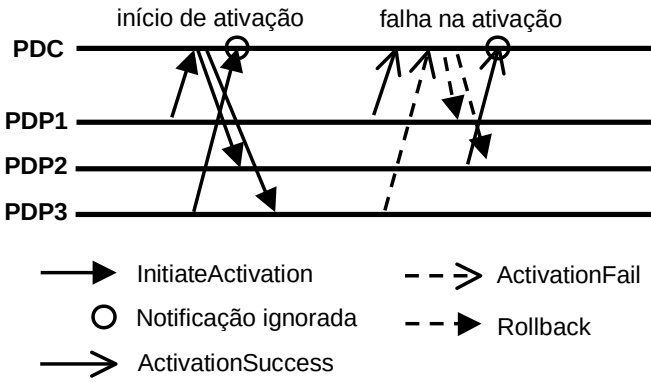

Figura 3.16: Início da ativação e Rollback

Quando um PDP recebe uma mensagem Rollback, cabe-lhe contatar os PEPs associados à política para remover a configuração aplicada previamente. Caso o PDC não consiga enviar Rollback para algum PDP (por exemplo, pelo fato deste não estar acessível via rede), nenhuma medida adicional é tomada pelo PDC. Nestes casos, espera-se que o PDP desista da ativação quando for atingido um *timeout* pré-definido, descrito a seguir.

Após uma ativação sem sucesso, o PDC passa a ignorar novos pedidos de início de ativação para a aplicação atômica em questão durante um determinado tempo configurável. Esta medida pode ser útil para evitar tráfego indesejado referente a mensagens InitiateActivation supostamente atrasadas na rede. Já no PDP, ainda considerando o caso de uma ativação atômica não concluída com sucesso, foram disponibilizadas duas configurações para evitar tráfego desnecessário na rede. Visto que uma dada política pode continuar em seu estado "Active" (ou seja, com seus agendamentos satisfeitos) mesmo após uma ativação sem sucesso, seria possível que o PDP entrasse em um ciclo: iniciando uma ativação (enviando InitiateActivation) e percebendo o insucesso (enviando ActivationFail ou recebendo Rollback), durante todo o período em que política está ativa (Figura 3.17).

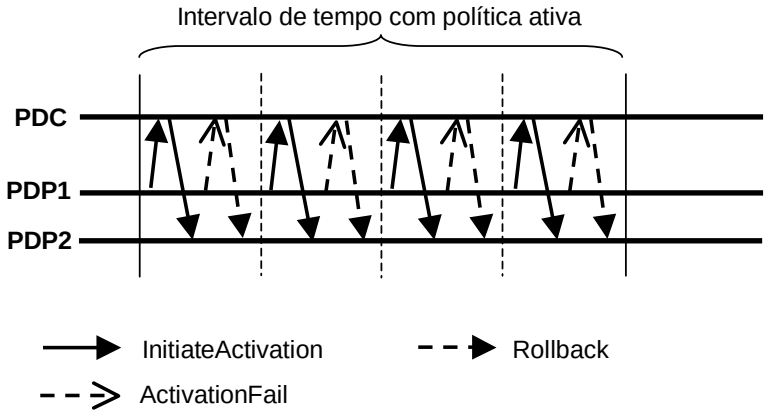

Figura 3.17: PDPs em um ciclo: iniciando ativações e percebendo insucessos.

Para controlar este comportamento, o protocolo pode ser configurado para admitir um número de novas tentativas possíveis, ou seja, o número de novos InitiateActivation que poderão ser enviados após o primeiro insucesso dentro deste período em que a política está ativa. Caso este número seja igual a zero, nenhuma nova tentativa é disparada até que a política passe por um período de inatividade e, após, torne-se ativa novamente. A Figura 3.18 apresenta um exemplo de ativação de política onde o protocolo foi configurado para realizar apenas uma nova tentativa de início de ativação.

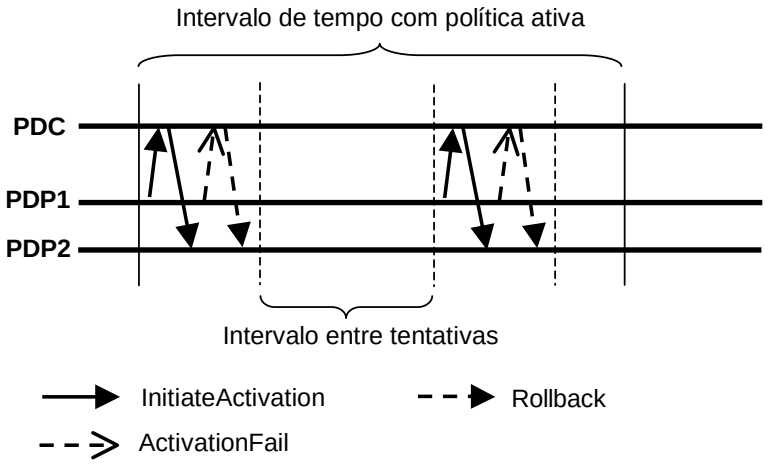

Figura 3.18: Exemplo de comportamento do protocolo com número de novas tentativas maior ou igual a 1, contendo intervalo entre tentativas.

Ainda na Figura 3.18, é apresentado o conceito de "Intervalo entre tentativas". Este é o outro parâmetro configurável que o protocolo disponibiliza com o intuito de reduzir tráfego desnecessário na rede. Trata-se de um tempo que um PDP deve aguardar antes de enviar uma nova tentativa de InitiateActivation, dentro um mesmo período em que a política está ativa. Idealmente este tempo de espera no PDP deve ser maior que o tempo de espera no PDC (descrito anteriormente), sob pena de que novas tentativas de início de ativação enviadas pelos PDPs sejam ignoradas pelo PDC.

O insucesso da ativação de uma política não ocorre exclusivamente devido a problemas relativos aos PEPs. Problemas podem ocorrer tanto com o PDC quanto com o PDP. Por exemplo, pode haver perda de comunicação com a rede, sobrecarga de tarefas ou até mesmo o software do PDC e/ou do PDP pode parar de ser executado devido a algum caso inesperado. Por este motivo, dois *timeouts* foram definidos no protocolo. O primeiro está localizado no PDC. O PDC espera por avisos de ativação (ActivationSuccess ou ActivationFail) oriundos dos PDPs, durante um determinado tempo, a partir do momento que se encerra a fase de divulgação da mensagem de InitiateActivation. Se o tempo relativo ao *timeout* expira, e existe algum estado "Unknown" no vetor *DeplVector state*, o PDC assume que a falta de mensagens é devida a PDPs que falharam. Assim, o PDC envia a mensagem Rollback para todos os PDPs (Figura 3.19).

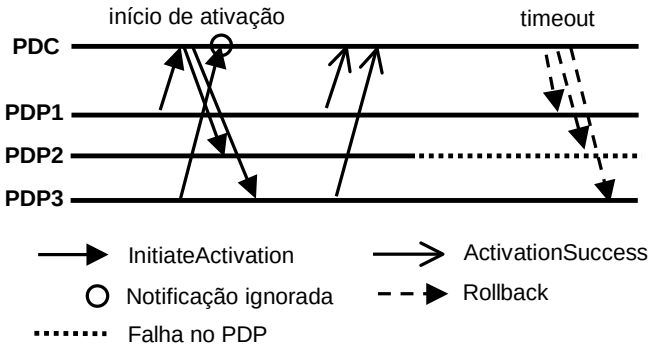

Figura 3.19: Rollback devido à falha no PDP e *timeout* no PDC

O segundo *timeout* está localizado nos PDPs. Depois de enviar a mensagem ActivationSuccess para o PDC, o PDP espera por uma mensagem Commit ou Rollback. Se nenhuma das duas for recebida em um determinado limite de tempo, então o PDP realiza *rollback* de configuração, assumindo que o PDC falhou (Figura 3.20). 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.

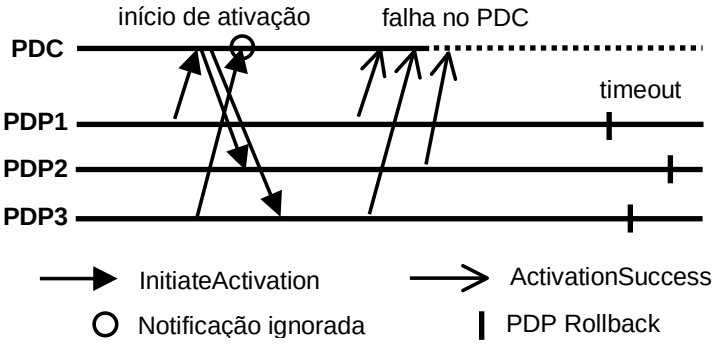

Figura 3.20: Rollback devido à falha no PDC e *timeout* no PDP

3.3.6 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 enviando a mensagem Deactive. O PDC repassa esse aviso para todos os PDPs restantes enviando-lhes uma mensagem Deactive. Além disso, o PDC ajusta *DeplState* para 0 (Inactive) e ajusta todas as posições de *DeplVector state* para o estado 0 (Unknown), alterando assim o registro de aplicação desta política para seus valores iniciais (pré-ativação) e, portanto, preparando-se para uma possível nova ativação desta política.

Porém, quando o PDC envia a mensagem Deactive para cada PDP, pode acontecer de algum dos PDPs estar incomunicável via rede, não recebendo tal mensagem. Neste caso, fatalmente a política continuará ativa no PEP, visto que não há meios para entrar em contato com o mesmo. Mesmo assim, há a possibilidade deste PDP desativar a política por si só, devido a avaliação de seus agendamentos. Por exemplo, considerando a política da Figure 2.2 é possível que o PDC desative a política ao perceber que o dia da semana é uma sexta-feira e que são 21 horas e 1 minuto (portanto, o intervalo de tempo de ativação expirou), mesmo nas situações onde o PDC não tenha conseguido lhe enviar a mensagem Deactive.

Mesmo que a política seja desativada de tempos em tempos, ela permanece válida dentro dos PDPs até que a data de encerramento seja atingida. Ainda considerando a política de exemplo da Figura 2.2, isto quer dizer que mesmo que a cada sexta-feira as 21 horas e 1 minuto a política seja desativada, ela permanece sendo avaliada para detectar novas necessidades de ativação/desativação até o dia 31/12/2006. Após esta data, o primeiro PDP que percebe que a política está inválida notifica o PDC através da mensagem RemovePolicy, a qual é bastante similar à mensagem Deactive (Figura 3.21), exceto pelo fato de que RemovePolicy remove todas as instâncias da política em cada PDP associado, e apaga os registros que guardam dados sobre a aplicação da política no PDC. A partir deste momento esta aplicação de política não existirá mais na base de dados do PDC e do PDP, e portanto não será mais nem mesmo avaliada.

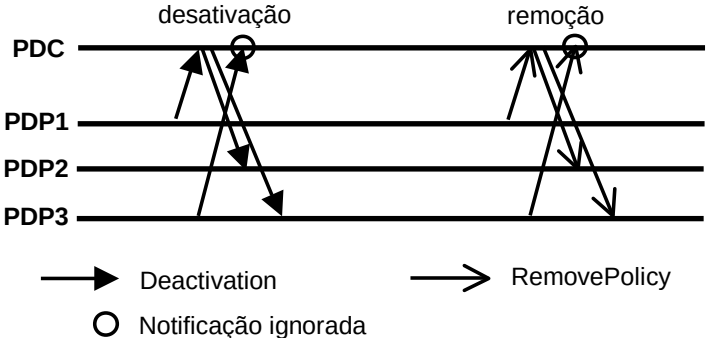

Figura 3.21: Desativação e Remoção de uma Política

3.3.7 Alternativas de Operação para o Protocolo de Consenso

Recapitulando o que foi citado nas seções anteriores, as variáveis do protocolo proposto passíveis de configuração e implementadas neste trabalho de mestrado são:

- *Timeout* do PDC: tempo, em segundos, que o PDC espera pela resposta de um PDP antes de decretar que este falhou.
- *Timeout* do PDP: tempo, em segundos, que o PDP espera pela resposta do PDC antes de decretar que este falhou;
- Número de novas tentativas de iniciar ativação que o PDP disparará após uma ativação sem sucesso, dentro de um mesmo período em que a política estiver ativa;
- Tempo a partir do qual um PDP realizará uma nova tentativa de ativação após uma ativação sem sucesso, dentro de um mesmo período em que a política estiver ativa;
- Tempo a partir do qual o PDC aceita novas mensagens InitiateActivation, para uma dada aplicação atômica, após a última ativação sem sucesso;
- Tempo pós transferência da política a partir do qual um PDP pode enviar InitiateActivation. Este tempo tem o objetivo de reduzir a ocorrência de casos em que mensagens InitiateActivation são trocadas antes mesmo da política ter sido transferida para todos os PDPs.

Além destas variáveis, algumas possíveis variações foram analisadas durante a definição do protocolo proposto nesta dissertação e serão apresentadas a seguir.

Primeiramente, como já foi citado neste capítulo, poderiam ser retiradas do protocolo as mensagens InitiateActivation, caso fosse considerado um ambiente com garantia de que os relógios dos PDPs estivessem sincronizados, por exemplo, através da utilização de NTP (*Network Time Protocol*) (MILLS, 1992).

Mesmo que as mensagens InitiateActivation fossem mantidas, ainda assim outra variação poderia ser realizada. Quando um PDP percebe que é chegado o momento de iniciar uma ativação e não consegue enviar a mensagem de InitiateActivation para o PDC, este PDP pode reagir de acordo com duas abordagens: uma otimista e outra pessimista. Na abordagem pessimista, a qual foi utilizada nesta dissertação, o PDP não prossegue com a ativação de uma política quando percebe que o PDC está indisponível (ao tentar se conectar e não conseguir estabelecer conexão TCP), encerrando a tentativa de aplicação atômica. Já em uma abordagem otimista poderia-se assumir que, mesmo o PDC estando indisponível, cada PDP realizaria sua tarefa de ativar a política em seus PEPs com sucesso, ou seja, em caso de falha na comunicação com o PDC o sistema desistiria de coordenar uma aplicação atômica e partiria para o esquema tradicional de configuração (sem garantias de atomicidade).

Como uma possível otimização do protocolo de consenso poderia-se definir que o PDC, ao tentar divulgar InitiateActivation e não conseguir enviar esta mensagem para um determinado PDP, enviasse imediatamente uma mensagem de Rollback para cada um dos PDPs que conseguiram receber a mensagem InitiateActivation. Esta medida faria com que uma tentativa de ativação sem sucesso fosse encerrada de forma mais eficiente.

A Figura 3.22 apresenta o comportamento do protocolo sem esta otimização, onde o PDC espera por mensagem ActivationSuccess ou ActivationFail por parte do PDP3 até o limite definido para o *timeout* no PDC, e só depois decreta que a ativação não teve sucesso (enviando Rollback).

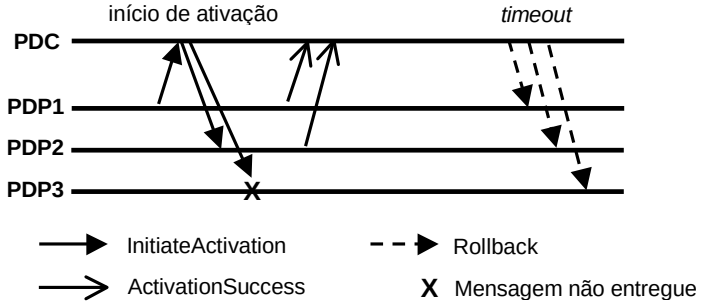

Figura 3.22: Comportamento do protocolo sem otimização de envio de Rollback logo após mensagem não ser entregue.

A Figura 3.23 apresenta o comportamento do protocolo quando esta otimização é utilizada. Assim que é percebido que não foi possível enviar a mensagem InitiateActivation, o PDC envia Rollback para todos os PDPs envolvidos. É possível, inclusive, enviar Rollback para o PDP3, para a eventualidade deste PDP já estar acessível novamente.

Como uma quarta alternativa pode-se pensar no uso do passo inicial do protocolo (InitiateActivation) em uma configuração de PDPs onde apenas um subconjunto dos PDPs possui o controle de relógio e o direito de disparar procedimentos de aplicação atômica (enviar InitiateActivation). Assim, alguns PDPs participantes ficariam apenas aguardando por mensagens de início de ativação, ficando dependentes de outros PDPs.

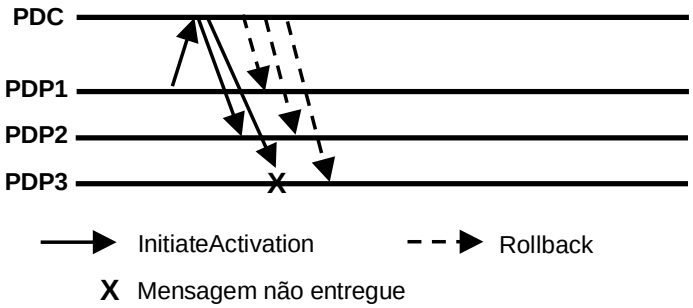

Figura 3.23: Comportamento do protocolo com otimização de envio de Rollback logo após mensagem não ser entregue.

Por fim, cabe ressaltar que o funcionamento do protocolo proposto deve poder assumir variações de acordo com as necessidades e as preferências do administrador da rede. Idealmente, uma ferramentas de políticas que provesse aplicações atômica de políticas de gerenciamento poderia disponibilizar estas alternativas de comportamento via configuração da própria ferramenta, flexibilizando ao máximo sua operação.

4 IMPLEMENTAÇÃO

O protocolo apresentado nesta dissertação foi implementado junto ao QAME (QoS-Aware Management Environment) (GRANVILLE et al., 2001). O QAME é um sistema para gerenciamento de redes implementado em PHP (PHP, 2001) que já possuía suporte ao gerenciamento baseado em políticas para redes que suportam Serviços Diferenciados (DiffServ).

O suporte ao protocolo proposto foi implementado como um conjunto de operações Web Services (CURBERA et al., 2002) disponibilizados pelos elementos da arquitetura PBNM. Utilizar Web Services como base para o protocolo traz os seguintes benefícios:

- 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 comunicação mais fácil entre PDPs de 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.

Além destas vantagens, deve-se destacar a uniformidade com o funcionamento prévio do sistema QAME, que já utilizava Web Services para comunicar-se com os PDPs. Exemplos de onde Web Services já eram utilizados no QAME são:

- Transferência de uma política: um identificador de uma política é passado para o PDP via Web Services. De posse deste identificador, o PDP acessa um repositório de políticas LDAP e efetua o seu *download*.

- Descoberta de interfaces de rede de um PEP: a descoberta das interfaces de rede de um PEP é realizada através de uma mensagem Web Service enviada para o PDP responsável pelo PEP. O PDP recebe a mensagem, consulta o PEP via SNMP e retorna a lista de interfaces do PEP.

- Remoção de uma política: quando o usuário remove uma política na interface gráfica do QAME, o PDP é avisado perante o envio de uma mensagem Web Service.

- Pré-visualização da tradução de uma política: antes de realmente aplicar uma política, um usuário do QAME pode prever a tradução de tal política em comandos específicos de configuração, que são gerados por um dado PDP. Assim, a política que terá sua tradução pré-visualizada é passada para o PDP via uma mensagem Web Service, e o PDP responde com os comandos que seriam executados caso a política fosse aplicada.

Os Web Services implementados no QAME utilizam a API PHP chamada nuSOAP (AYALA, 2004). Esta API é largamente utilizada pela comunidade de desenvolvedores de Web Services e apresenta um bom desempenho se comparada com outras APIs PHP para Web Services, como PEAR::SOAP (PEAR, 2005) e ezSOAP (EZ SYSTEMS, 2005). A linguagem PHP, por sua vez, foi escolhida por questões de compatibilidade com a implementação já existente do sistema QAME.

Até o início desta dissertação o QAME suportava, através de PDPs específicos para este fim, a configuração de roteadores Cisco e de disciplinadores de fila AltQ, sem qualquer suporte a operações atômicas. Com a realização desta dissertação, uma série de funcionalidades foram agregadas. São elas:

- Implementação do protocolo de consenso utilizando Web Services;

- Implementação do PDC;

- Implementação do suporte ao protocolo de consenso nos PDPs;

- Implementação do suporte a operações atômicas junto à interface gráfica do QAME;

A seguir, será dedicada uma seção para detalhar cada contribuição citada acima.

4.1 Operações Web Services nos Elementos PBNM

A Figura 4.1 apresenta a arquitetura final do sistema QAME incluindo suporte a gerenciamento baseado em políticas e ativação/desativação coordenada de políticas. É importante destacar que o PDC foi implementado junto à ferramenta de políticas, como será melhor discutido na seção a seguir que trata especialmente da implementação do PDC.

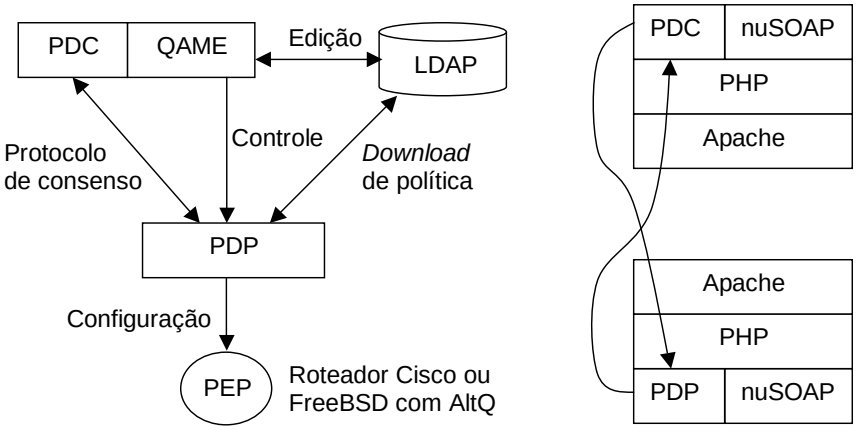

Figura 4.1: Arquitetura PBNM com PDC

As operações disponibilizadas pelos elementos PDC são as seguintes:

```
RegisterPolicyDeployment(int deplId, String deplName, array int
pdpId, array String pdpUrl);

InitiateActivation(int deplId, int pdpId);

ActivationSuccess(int deplId, int pdpId);

ActivationFail(int deplId, int pdpId);

Deactive(int deplId, int pdpId);

RemovePolicy(int deplId, int pdpId);

GetAllDeploymentsInformation();

GetDeploymentInformation(int deplId);

GetDeploymentsNames();
```

A operação `RegisterPolicyDeployment` recebe um identificador de aplicação (deplId), uma *string* deplName que dá nome à aplicação, uma lista de identificadores de PDPs (pdpId) e uma lista de URLs de PDPs (pdpUrl), 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 identificador de aplicação pode ser gerado tanto pela ferramenta de políticas quanto pelo PDC. A operação `GetAllDeploymentsInformation` retorna as informações relativas a todas aplicações atômicas registradas naquele PDC. As informações retornadas para cada aplicação são: identificador, nome, estado e seus PDPs integrantes. A operação `GetDeploymentInformation` recebe como parâmetro um identificador de aplicação (deplId) e retorna informações sobre esta aplicação. A operação `GetDeploymentsNames` retorna o identificador e o nome de cada aplicação atômica registrada no PDC. As demais operações do PDC (apresentadas acima) recebem como

parâmetro deplId e pdpId, este último que identifica o PDP que realizou a chamada Web Service.

As operações disponibilizadas pelos elementos PDP são apresentadas abaixo:

```
TransferPolicy(int deplId, array String policy, array String
device, array String interface, array char direction);

InitiateActivation(int deplId);

Commit(int deplId);

Rollback(int deplId);

Abort(int deplId);

Deactive(int deplId);

RemovePolicy(int deplId);

ListAppliedPolicies(int deplId);
```

A operação `TransferPolicy`, a qual já existia antes do suporte a operações atômicas, passou a receber adicionalmente como parâmetro de entrada o identificador deplId. Além desse parâmetro, foram mantidos os outros parâmetros necessários para a transferência da política em si. É importante destacar que através de uma única mensagem é possível associar várias políticas a vários PEPs controlados por um mesmo PDP. Assim, os demais parâmetros da mensagem são: vetor com identificadores de política, vetor com identificadores de dispositivos (endereço IP), vetor de identificadores de interfaces de rede e vetor de direções nas quais cada política deve ser aplicada (entrada ou saída). A política é obtida através do identificador passado, o qual permite o acesso a um servidor LDAP (OPENLDAP, 2005) para a realização do *download* da mesma. A operação `ListAppliedPolicies` recebe como parâmetro o identificador de uma aplicação (deplId) e retorna um vetor com as informações referentes às políticas desse deplId para cada PEP que esse PDP controla. Estas informações são: o identificador da política, o identificador do PEP e a interface de rede e a direção (entrada ou saída) nas quais a política foi aplicada. Todas as outras operações disponibilizadas pelos PDPs recebem apenas deplId como parâmetro de entrada.

4.2 Policy Deployment Coordinator (PDC)

O PDC foi implementado como um conjunto de Web Services (descrito na seção anterior), um *daemon* e uma base de dados. Assim como nos PDPs, a base de dados utilizada no PDC é MySQL ((MySQL, 2006)) e o *daemon* foi implementado como um *script* PHP. Cada mensagem Web Service que o PDC recebe é gravada na base de dados para futuro processamento do *daemon* do PDP. Portanto, o *daemon* é implementado como um laço infinito, responsável por constantemente varrer a base de dados em busca de mensagens do protocolo esperando para serem tratadas. Ao encontrar mensagens, o PDC realiza o processamento adequado e invoca operações Web Services nos PDPs envolvidos.

Como exemplo de algumas ações tomadas pelo PDC ao receber mensagens, pode-se citar:

- Ao receber InitiateActivation: o PDC busca na base de dados a URL de todos os PDPs envolvidos na aplicação atômica e chama InitiateActivation em cada um destes PDPs. Além disso, o PDC registra o *timestamp* de recebimento desta mensagem. A cada execução do laço, o PDC verifica se o tempo de *timeout* (fixo) já foi atingido. Em caso positivo, o PDC chama Rollback em todos os PDPs.

- Ao receber ActivationSuccess: o PDC percorre sua estrutura de dados para verificar se já recebeu o resultado de todos os PDPs. Em caso positivo, chama Commit em todos os PDPs. Caso contrário, continua aguardando.

- Ao receber ActivationFail: o PDC imediatamente chama Rollback em todos os PDPs.

No capítulo anterior foram comentadas possíveis alternativas de modelos de comunicação otimizados para a implementação do protocolo de consenso. Porém, além da redução do tráfego gerado na rede, existe uma outra importante questão a ser discutida quando se pretende definir como o protocolo de consenso será implementado: a possibilidade de que os PDPs envolvidos na configuração pertençam a domínios administrativos diferentes.

No protótipo desenvolvido nesta dissertação, o PDC foi implementado junto à ferramenta de políticas. Esta decisão é justificada pelo fato de que a ferramenta de políticas (QAME), naturalmente, conhece todos os PDPs que controla, além de possuir capacidade de comunicação com os mesmos. Assim, evita-se o problema de que, caso o PDC fosse implementado junto a algum dos PDPs, tal PDP precisaria ter conhecimento de todos os PDPs envolvidos na aplicação atômica, além de precisar possuir comunicação direta com cada um destes PDPs. Isto poderia envolver, por exemplo, a necessidade de configuração de regras de *firewall* adicionais para permitir tal comunicação direta, além das regras responsáveis pela liberação do tráfego oriundo da máquina que hospeda a ferramenta de políticas.

Por fim, ainda descrevendo o comportamento do PDC, pode-se afirmar que a ordem de chegada das mensagens originadas a partir de um mesmo PDP é garantida, apesar destas mensagens pertencerem a diferentes conexões TCP. Isto é garantido porque um PDP só envia uma nova mensagem Web Service para o PDC após ter recebido o resultado positivo do processamento da mensagem enviada anteriormente.

4.3 Suporte ao Protocolo de Consenso nos PDPs

Durante a realização deste trabalho foram utilizados dois tipos de PDPs: um PDP para roteadores Cisco (CISCO, 2006) e um PDP para roteadores AltQ (CHO, 1999). Cada PDP utilizado é composto basicamente por um *daemon* (implementado em PHP) e uma base de dados (MySQL). Visto que o protocolo de consenso definido é independente do tipo de PDP utilizado (suas mensagens são genéricas), o suporte a este protocolo foi implementado junto a camada genérica do PDP, aquela responsável por efetuar rotinas comuns a qualquer PDP. Assim, o funcionamento do protocolo de consenso é idêntico tanto no PDP Cisco quanto no PDP AltQ.

Com a adição da funcionalidade de operações atômicas, a complexidade do PDP cresceu consideravelmente. Por exemplo, no modo de operação normal de um PDP (sem

atomicidade), este constantemente avalia as condições de suas políticas e dispara um processo de configuração apenas quando tais condições forem completamente satisfeitas. Já no caso de operações atômicas, o PDP pode disparar uma configuração mesmo que as condições da política não estejam satisfeitas. Isto ocorre quando um PDP recebe uma mensagem do tipo InitiateActivation, indicando que as condições da política foram satisfeitas em algum dos outros PDPs que fazem parte da configuração atômica.

A Figura 4.2 apresenta a arquitetura interna do PDP com suporte à aplicação atômica de políticas. É importante notar que o módulo de "Avaliação de condições de políticas e de mensagens do protocolo" recebe dados tanto das políticas quanto das mensagens Web Services recebidas pelo PDP. Caso a política torne-se ativa neste PDP devido às suas próprias avaliações das condições da política, então este módulo requisita junto ao módulo "Emissão de mensagens do protocolo de consenso" o envio de uma mensagem InitiateActivation. Após isso, o procedimento adotado é o mesmo que ocorre quando é recebida uma mensagem InitiateActivation: a política é repassada para o módulo de tradução das políticas e posteriormente para o módulo de disparo de configurações. Por fim, é importante ressaltar que o módulo "Disparo de configurações e obtenção dos resultados" é responsável não só por enviar as configurações para os PEPs, mas também por concentrar os resultados da configuração em cada PEP e requisitar junto ao módulo de emissão de mensagens Web Services o envio, para o PDC, de ActivationSuccess ou ActivationFail.

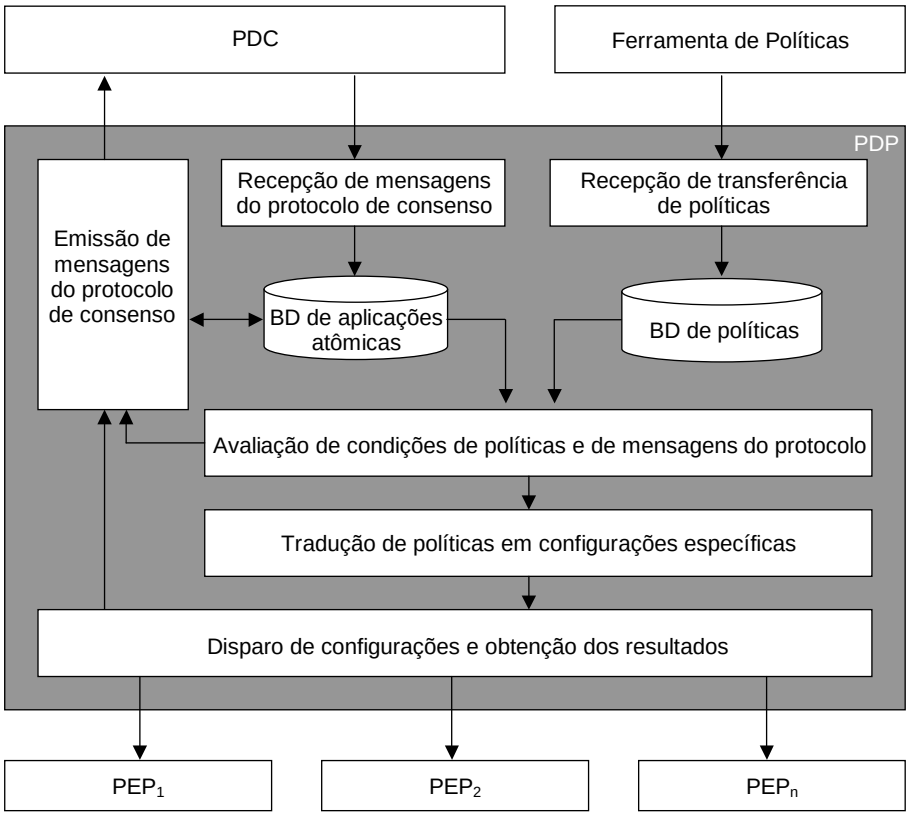

Figura 4.2: Arquitetura interna do PDP e interações com Ferramenta de Políticas, PDC e PEPs

A maioria das novas funcionalidades que um PDP com suporte a operações atômicas deve implementar estão diretamente relacionadas ao recebimento e ao envio das mensagens do protocolo de consenso. Entre as tarefas de um PDP com suporte a operações atômicas pode-se citar:

- Chamar InitiateActivation quando as condições de uma política tornarem-se verdadeiras.
- Iniciar a aplicação de uma política quando receber uma mensagem do tipo InitiateActivation.
- Perpetuar uma configuração quando receber uma mensagem do tipo Commit.
- Desfazer uma configuração quando receber uma mensagem do tipo Deactive.
- Remover uma política quando receber uma mensagem do tipo RemovePolicy.

Outra nova funcionalidade que um PDP deve implementar não diz respeito diretamente às trocas de mensagens do protocolo de consenso. No funcionamento normal de um PDP, sem atomicidade, cabe-lhe a responsabilidade apenas de ativar as políticas e efetuar *rollback* nos PEPs onde a aplicação da política falhar. Porém, implementando operações atômicas, o PDP precisa agir localmente como um PDC, reunindo os resultados de cada PEP e decidindo se é necessário realizar *rollback* nos PEPs. Obviamente, após esta decisão, o PDP deve emitir a mensagem adequada para o PDC (*commit* em caso de sucesso de ativação em todos PEPs e *rollback* em caso de falha de ativação em algum PEP).

4.3.1 Processamento das Mensagens do Protocolo

Ao *daemon* do PDP cabe constantemente ler as políticas da base de dados, avaliá-las e traduzi-las em configurações quando suas condições forem satisfeitas. À base de dados do PDP cabe armazenar as políticas sendo analisadas e, com o advento da implementação das operações atômicas, armazenar as mensagens do protocolo de consenso que chegam ao PDP.

Portanto, considerando-se a inclusão do suporte a operações atômicas, o PDP além de varrer a base de dados em busca de políticas para serem avaliadas, deve constantemente efetuar uma busca por novas mensagens do protocolo de consenso a serem tratadas.

O *daemon* do PDP é um programa PHP que roda como um *script* independente do *browser*, executando um *loop* infinito. O tratamento das mensagens do protocolo é realizado a medida que as mensagens do protocolo são lidas da base de dados. Cabe ressaltar que a ordem de chegadas das mesmas é garantida pelos Web Services que armazenam as mensagens na base de dados. Quando é necessário efetuar *broadcast* de uma mensagem como, por exemplo, InitiateActivation, o PDC realiza um *fork* e dispara uma série de processos, cada um com o objetivo de enviar uma mensagem para um PDP. Antes de criar cada novo processo o PDC acessa a base de dados para verificar se chegou alguma mensagem ActivationFail.

Quando o envio de uma mensagem do protocolo de consenso exige uma resposta e, portanto, há a possibilidade desta situação desencadear um *timeout*, o PDP envia a mensagem Web Service, registra na base de dados o momento do envio e prossegue seu processamento normalmente. Futuramente, em uma nova varredura da base de dados em busca de mensagens, o PDP pode descobrir a ausência da resposta requerida anteriormente, avaliando a data/hora atual e o tempo de *timeout* configurado na ferramenta. Neste caso, o PDP toma a devida providência, que depende da mensagem em questão (como foi descrito no capítulo anterior).

4.3.2 PDP Cisco

O PDP Cisco utilizado nesta dissertação teve sua primeira versão implementada pelo Grupo de Redes da Universidade Federal do Rio Grande do Sul (UFRGS) em um projeto conjunto com a RNP - Rede de Nacional de Ensino e Pesquisa (RNP, 2006) chamado Grupo de Trabalho de Configuração de Redes (GT-Config).

Este PDP já possuía a capacidade de configurar em roteadores Cisco parâmetros de qualidade de serviço (QoS) como: reserva de banda, definição de prioridade de envio e de descarte de pacotes, além de classificação de pacotes via DSCP (*Differentiated Services Code Point*). O DSCP é um campo no cabeçalho do protocolo IP no qual pode-se marcar (e posteriormente selecionar) pacotes com o intuito de dar diferentes tratamentos para cada pacote. Cabe ressaltar que esta versão do PDP, desenvolvida pelo GT-Config, não possuía qualquer suporte a aplicação atômica de políticas.

Para realizar a configuração do roteador Cisco, o PDP primeiramente cria o arquivo de configuração Cisco referente às políticas a serem aplicadas e o disponibiliza para *download* em um diretório via TFTP (SOLLINS, 1992). Posteriormente, o PDP configura, via SNMP, um objeto da MIB do roteador indicando o endereço do servidor TFTP (PDP) e o nome do arquivo de configuração que deve ser recuperado remotamente. Assim, o roteador Cisco realiza o *download* do arquivo de configuração e efetua um *merge* com suas configurações atuais. Quando é necessário remover uma política que está aplicada no roteador Cisco, um novo arquivo de configuração é criado incluindo uma negação (*string* "*no*") antes de cada comando que deve ser removido.

4.3.3 PDP AltQ

O PDP AltQ utilizado nesta dissertação foi baseado em duas outras implementações. Os módulos genéricos foram baseados no PDP Cisco descrito anteriormente. Já o módulo específico, responsável pela aplicação das políticas no AltQ, foi baseado em uma implementação prévia utilizando a linguagem Java, produzida pelo Grupo de Redes da UFRGS em um projeto conjunto com a Fundação CPqD (CPQD, 2006) chamado SCQoS (Sistema para Configuração de QoS em Redes IP).

O disciplinador de filas AltQ (CHO, 1999) foi instalado em uma máquina contendo o sistema operacional FreeBSD (FREEBSD, 1995). A comunicação entre o PDP e esta máquina (PEP) se dá através da execução de comandos remotos. Através de chamadas SSH é disparado um programa, implementado em C, que é responsável por, a partir dos comandos de configuração AltQ recebidos como parâmetro, comunicar-se com o *daemon* AltQ via *sockets* UNIX e assim configurar este disciplinador de filas. Nesta versão implementada foi disponibilizado apenas a capacidade de configuração de reserva de banda.

4.4 Interface com o Usuário no QAME

Nesta seção são apresentadas as janelas de interface através das quais o usuário do QAME efetua *login*, define políticas, monta o cenário de rede contendo PDPs e PEPs, e, por fim, realiza a aplicação atômica de políticas.

A Figura 4.3 apresenta a tela de *login* do QAME. Nesta janela entra-se com o nome de usuário e senha. É possível também escolher o idioma no qual se deseja utilizar o *software* (atualmente Português ou Inglês), além do tema de interface gráfica (*skin*) a ser utilizado.

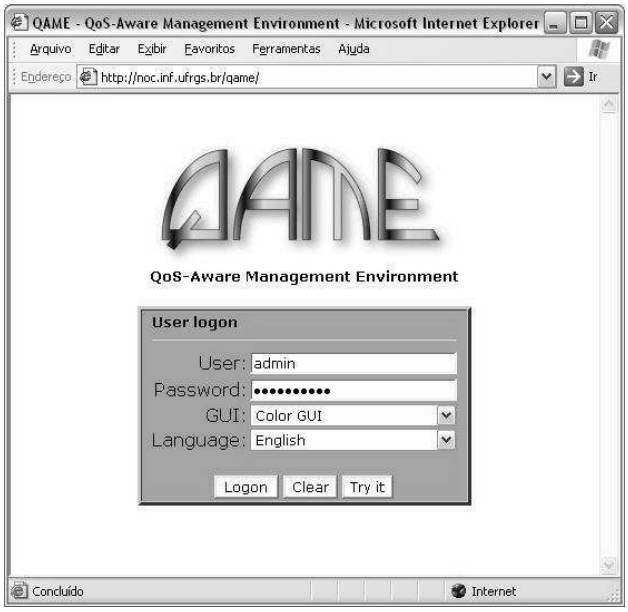

Figura 4.3: Tela de *login* do QAME

Efetuado o *login*, o usuário acessa o mapa da rede (Figura 4.4). À esquerda, na mesma figura, existe um menu com opções de gerenciamento, entre elas, o grupo "*QoS Policies*".

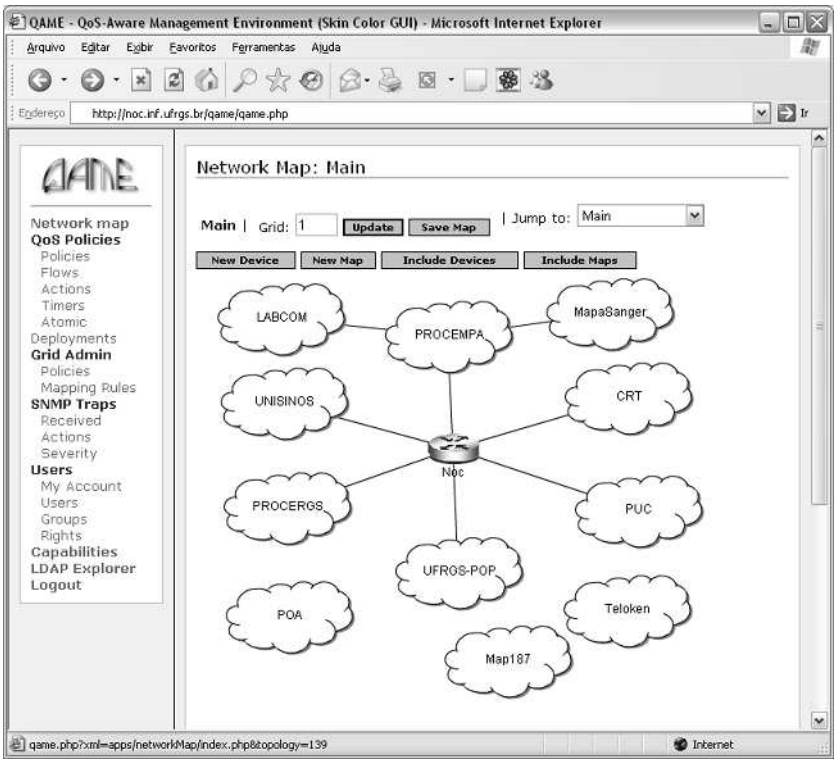

Figura 4.4: Tela inicial do QAME contendo o mapa da rede

No grupo "*Qos Policies*" o usuário tem acesso às políticas, fluxos, ações e agendamentos (respectivamente "*Policies*", "*Flows*", "*Actions*", "*Timers*") onde é possível efetuar a criação, alteração e exclusão de políticas e uma última opção para controlar aplicações atômicas de políticas ("*Atomic Deployment*"), a qual será detalhada posteriormente.

Como descrito anteriormente nesta dissertação, um modelo de política bastante aceito é o formado por fluxos, agendamentos e ações. No QAME foi dada a liberdade para o usuário definir cada um destes componentes independentemente, de modo a permitir o reuso dos componentes em diferentes políticas. Se os componentes necessários para uma determinada política já estão criados, o usuário pode acessar a opção "*Policies*" e montar a política que deseja, escolhendo um ou mais fluxos, um ou mais agendamentos e uma ação de gerenciamento, como pode ser visto na Figura 4.5.

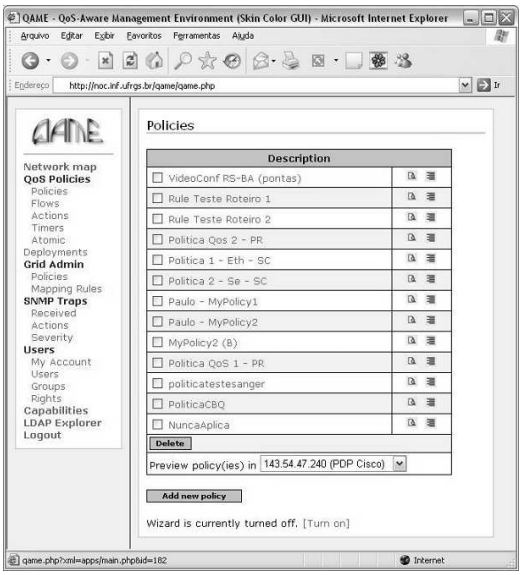

Figura 4.5: Tela de criação e edição de políticas

Caso o usuário necessite criar ou editar um ou mais componentes, basta acessar o componente de interesse e realizar a operação desejada. Após concluir este passo, o usuário pode acessar a opção "*Policies*" e finalmente montar sua política.

A Figura 4.6 ilustra a interface gráfica apresentada ao acessar a opção "*Flows*". A partir desta interface o usuário do QAME pode criar, alterar e excluir fluxos. Os fluxos são os componentes das políticas responsáveis por criar restrições que indiquem em quais condições uma política deve ser aplicada. Um exemplo de uma restrição é filtrar para o recebimento das ações das políticas apenas os pacotes que apresentarem porta destino igual a "HTTP". Nos fluxos suportados pelo QAME é possível filtrar pacotes por endereço IP de origem e destino, porta de origem e destino, protocolo (TCP, UDP, ICMP) e valor de DSCP.

Os agendamentos ("*Timers*") são as regras que indicam em que momento de tempo (data e hora) uma política deve ser ativada e desativada em um PEP. A Figura 4.7 apresenta a interface de definição e alteração de agendamentos no QAME. O QAME implementa um modelo de informação de agendamento que segue o proposto pelo IETF no modelo PCIM. Neste modelo, primeiramente, o usuário deve definir o intervalo de tempo onde a política ficará válida, ou seja, o período de tempo no qual o PDP constantemente avaliará a política. Posteriormente, o usuário define os momentos em que a política estará ativa, escolhendo entre meses do ano, dias do mês, dias da semana e faixa horária do dia.

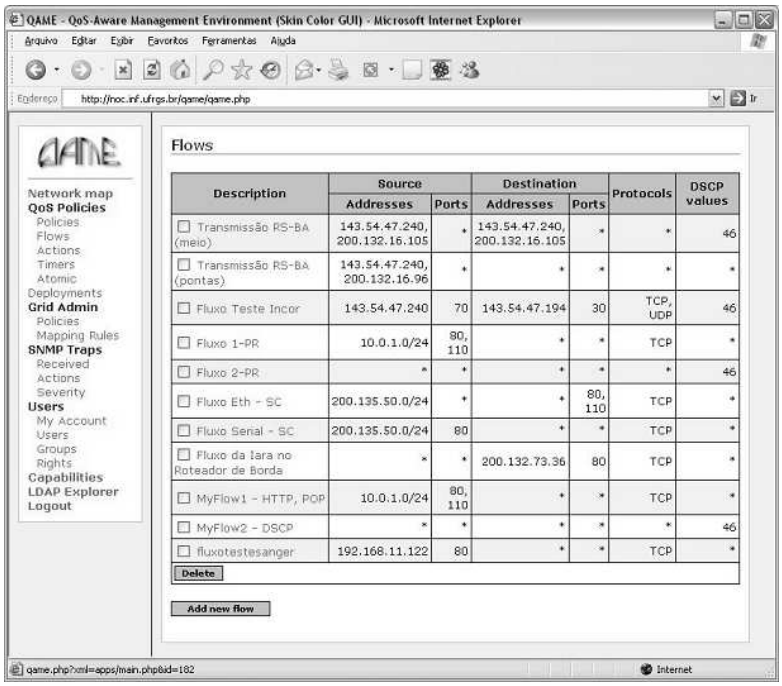

Figura 4.6: Tela para criação, alteração e remoção de fluxos

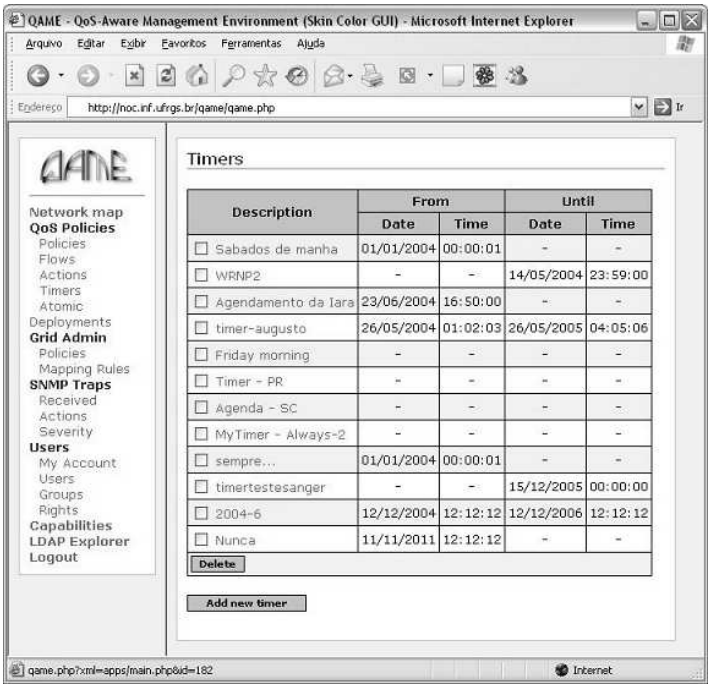

Figura 4.7: Tela para criação, alteração e remoção de agendamentos

Além de lidar com fluxos e agendamentos, o usuário define ações de gerenciamento, que são as diretivas de configuração responsáveis por alterar, de fato, o equipamento de acordo com o objetivo do usuário. Como um exemplo simples de ação pode-se citar uma reserva de banda de rede. A Figura 4.8 apresenta a interface de criação, alteração e exclusão de ações no QAME.

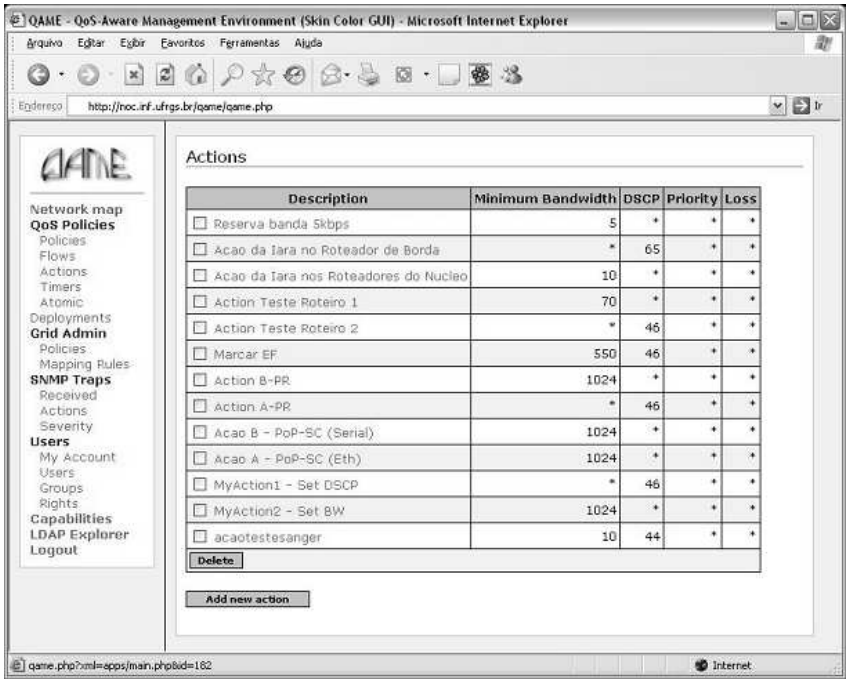

Figura 4.8: Tela para criação, alteração e remoção de ações

Uma vez que o usuário possua as políticas que deseja aplicar criadas, o próximo passo é criar o cenário de rede onde a política será aplicada. Voltando à Figura 4.4, pode-se visualizar o mapa da rede do QAME. No QAME cada segmento de rede é representado por uma nuvem. Quando o usuário clica em uma nuvem, abre-se um novo mapa representando outro segmento da rede. Assim, é possível organizar a rede hierarquicamente.

No mapa é possível adicionar dispositivos de rede. Para cada dispositivo adicionado, o usuário indica se o equipamento se trata de um *host*, um roteador ou um *switch* (tal configuração define o ícone que será utilizado para representar o dispositivo). Ao adicionar o dispositivo, o usuário deve informar qual o endereço IP do dispositivo, além de indicar quais capacidades de gerenciamento são suportadas pelo mesmo. A capacidade mais comum é o suporte a MIB-II (MCCLOGHRIE; ROSE, 1991), através da qual o QAME descobre informações básicas sobre o dispositivo, como, por exemplo, a descrição do sistema. Outras capacidades diretamente ligadas ao PBNM são a de PDP e de PEP. Como pode-se imaginar, a primeira indica que o dispositivo roda um *software* PDP. Já a segunda indica que o dispositivo possui pontos passíveis de aplicação de políticas.

Para ilustrar a montagem do cenário onde uma política pode ser aplicada considere a Figura 4.9. Nesta figura, "Host1" e "Host2" representam dois computadores pessoais que desejam comunicar-se por intermédio de um caminho formado pelos equipamentos AltQ1, Cisco e AltQ2, os quais agirão como pontos de aplicação de políticas (PEPs) e poderão ser configurados para fornecer as características que se deseja na comunicação entre os dois *hosts*. Ao adicionar estes três equipamentos, deve-se informar que os mesmos têm a capacidade "PEP". Para configurar estes equipamentos foi adicionado neste segmento de rede dois PDPs (PDP Cisco e PDP AltQ), os quais devem ser associados com a capacidade "PDP". Adicionalmente, pode-se inserir *links* entre os dispositivos, de modo a organizar visualmente o segmento de rede. Porém, cabe ressaltar que no QAME os *links* entre os equipamentos têm funcionalidade apenas visual.

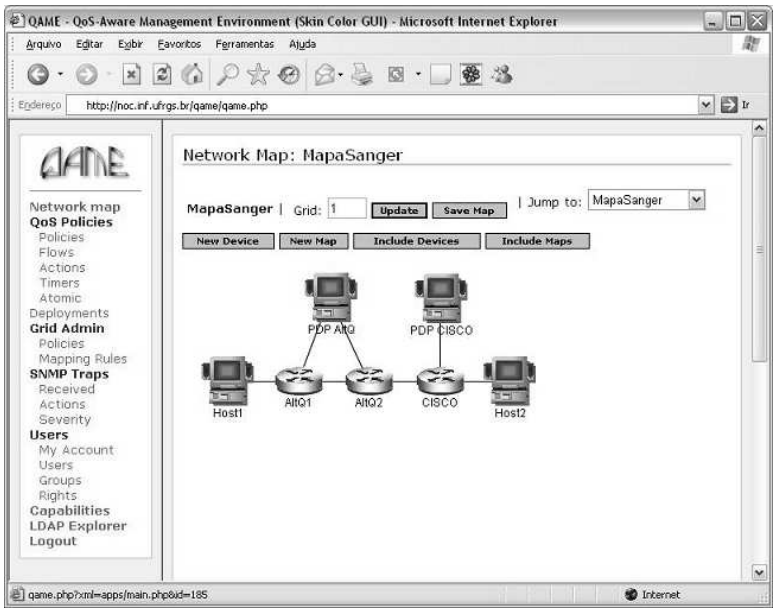

Figura 4.9: Exemplo de rede contendo dois PDPs e três PEPs

Se o usuário já possui a(s) política(s) a ser(em) aplicada(s) e também já possui todos os equipamentos envolvidos na aplicação da política cadastrados e devidamente configurados, basta partir para a aplicação da política em si. Anteriormente às implementações produzidas nesta dissertação, o usuário aplicava políticas individualmente, dispositivo por dispositivo. Com o advento das aplicações atômicas possibilitou-se a seleção de diversos equipamentos e a execução de uma tarefa de gerenciamento em todos estes dispositivos. No caso desta dissertação, a tarefa a ser executada em diversos dispositivos é a aplicação de políticas. A Figura 4.10 apresenta os três roteadores configurados anteriormente selecionados e o menu de opções com a opção "*Apply Policy*" selecionada.

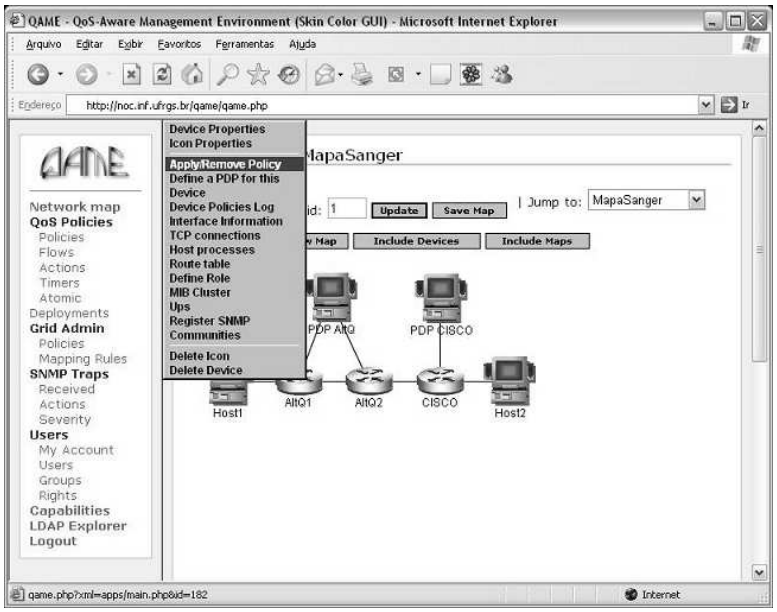

Figura 4.10: Aplicação simultânea de política em três dispositivos

Quando o usuário clica na opção "*Apply Policy*" é apresentada a interface da Figura 4.11. Para cada dispositivo selecionado no mapa é apresentado um painel contendo primeiramente a lista de interfaces de rede deste dispositivo. Ao lado de cada interface de rede existe uma lista das políticas que estão atualmente aplicada nesta interface. Ao lado do nome da política é apresentado, entre parênteses, o sentido no qual a política foi aplicada (*input* ou *output*). Ao lado do sentido é apresentado, entre colchetes, a *string* "*atomic*" para indicar que aquela política, naquela interface, faz parte de uma aplicação atômica de políticas. Além disso, após "[*atomic*]", é apresentado o nome que o usuário atribuiu à aplicação atômica durante sua criação.

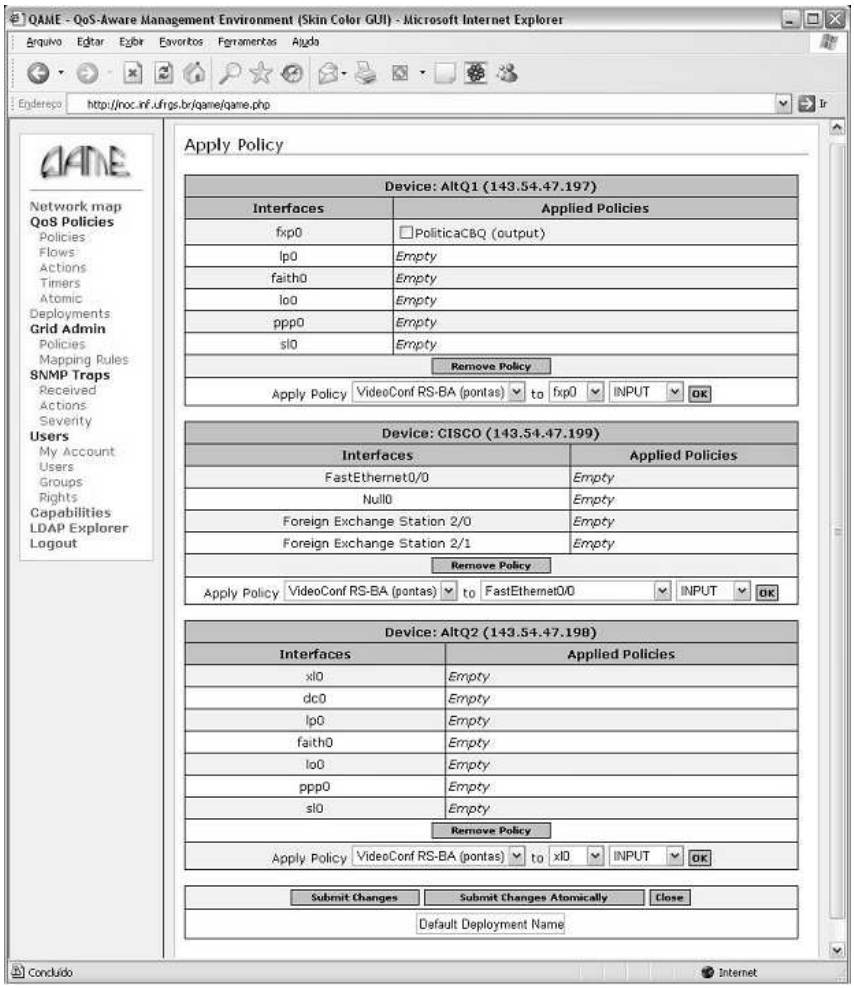

Figura 4.11: Exemplo de tela de aplicação atômica de políticas em três dispositivos

As políticas marcadas como "[*atomic*]", ao contrário das que não apresentam essa *string*, não podem ser removidas nesta tela. Para remover a aplicação de uma política que não faz parte de uma aplicação atômica basta selecionar o *checkbox* referente a política e clicar no botão "*Remove*". A remoção de políticas pertencentes a aplicações atômicas será vista adiante.

Abaixo da lista de interfaces de rede de cada equipamento e abaixo do botão "*Remove*" são apresentadas as opções para a aplicação de políticas. O usuário deve escolher a política, a interface e a direção na qual ocorrerá a aplicação.

Por fim, quando as políticas a serem aplicadas já foram escolhidas, o usuário deve selecionar um dos dois botões disponíveis no rodapé da tela, abaixo de todos os painéis

de dispositivos. O primeiro, chamado "*Apply*", realiza a aplicação das políticas nos dispositivos sem qualquer preocupação com atomicidade. Por outro lado, se o usuário optar pelo botão "*Apply Atomically*", então a política será transferida para todos os dispositivos e as informações sobre a política serão registradas no PDC. Atualmente, cada instância do *software* QAME possui apenas um PDC configurado, o qual deve ser definido via arquivo de configuração.

Para acompanhar o andamento das aplicações atômicas e remover aplicações atomicamente, o usuário deve utilizar a opção "*Atomic Deployments*", disponibilizada no grupo "*QoS Policies*" do menu esquerdo do QAME. A Figura 4.12 apresenta a tela de gerenciamento de aplicações atômicas no QAME. Nesta tela, cada aplicação atômica possui um painel de informações próprio, o qual apresenta o estado corrente da aplicação (obtido consultando o PDC) e a lista de políticas aplicadas em cada um dos PEPs (obtida em cada PDP que compõe a aplicação atômica). A lista de PDPs que fazem parte da aplicação atômica é obtida junto ao PDC.

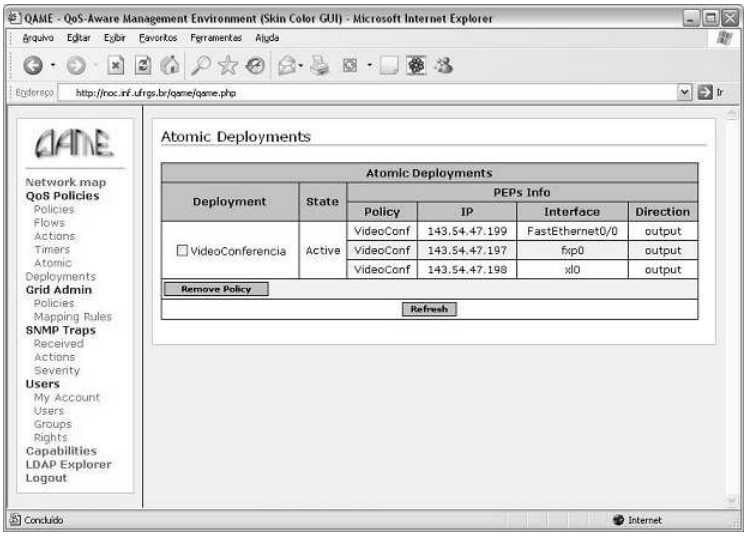

Figura 4.12: Tela de acompanhamento de aplicações atômicas e remoção das mesmas

Para efetuar a remoção de uma aplicação atômica, basta o usuário clicar no *checkbox* correspondente e clicar no botão "*Remove*".

5 AVALIAÇÃO DE DESEMPENHO DO PROTOCOLO

Neste capítulo é apresentada a avaliação de desempenho do protocolo proposto sob dois aspectos. Primeiramente, é realizada uma análise teórica considerando o tamanho das mensagens SOAP geradas por cada mensagem do protocolo. Posteriormente, em um cenário real de testes formado por um cluster de computadores, é avaliado o tráfego gerado pelo conjunto de mensagens que são trocadas durante uma ativação ou uma desativação. Nas duas abordagens, para análise de escalabilidade, foi variado o número de PDPs envolvidos na aplicação atômica.

5.1 Análise Teórica do Tamanho das Mensagens SOAP

Na tecnologia de Web Services, a chamada remota a procedimentos (RPC) normalmente é implementada 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 das mensagens. Isto evidencia que, na visão do modelo de referência OSI (*Open Systems Interconnection*), dois protocolos de nível de aplicação estão sendo utilizados para carregar as chamadas RPC. Isso, obviamente, aumenta o *overhead* total.

Embora sejam consideradas mensagens SOAP geradas a partir do uso da API (*Application Programming Interface*) nuSOAP (e mensagens SOAP podem variar de APIs para API), presume-se que outras APIs SOAP não devem gerar mensagens com tamanhos significativamente diferentes das mensagens nuSOAP. Para facilitar a análise teórica que segue, é considerado somente o tráfego SOAP, excluindo o *overhead* HTTP, TCP, IP e o nível de enlace. Por outro lado, no final deste capítulo, na análise prática do protocolo obtida a partir da execução de testes, os dados apresentados contemplarão todos os *bytes* trafegados na rede, ou seja, serão considerados todos os cabeçalhos da pilha de protocolos utilizada para transportar os dados dos Web Services.

Foram considerados dois tipos de operações do protocolo:

- Operações de registro da aplicação atômica nos elementos da arquitetura PBNM proposta neste trabalho. Fazem parte desse grupo RegisterPolicyDeployment e TransferPolicy, as quais são enviadas apenas durante a configuração inicial da aplicação atômica. Estas são as mensagens do protocolo que incluem parâmetros variáveis, visto que dependem do número de PDPs e PEPs envolvidos.

- Operações de coordenação da aplicação atômica. Fazem parte desse grupo as mensagens InitiateActivation, ActivationSuccess, ActivationFail, Commit, Rollback,

Abort e RemovePolicy, as quais são enviadas durante o período de coordenação de ativações, desativações e remoções. Estas mensagens possuem número de parâmetros fixos, porém podem se repetir diversas vezes durante uma ativação/desativação.

Inicialmente, então, serão analisadas as operações de registro da aplicação atômica.

Se for considerado valores de deplId contendo 3 dígitos, nome de aplicação atômica contendo 15 caracteres, valores de pdpId contendo 3 dígitos e URLs dos Web Services contendo 25 caracteres cada, a operação RegisterPolicyDeployment gera mensagens SOAP de requisição de acordo com a seguinte fórmula:

```
bytes = 1082 + 39 + 58 + 153 + N.(35 + 60), ou
bytes = 1332 + N.95
```

onde 39 *bytes* são utilizados para enviar o inteiro de 3 dígitos deplId e 58 *bytes* são ocupados para enviar o nome da aplicação atômica (15 caracteres). Como nesta operação são passados como parâmetro dois vetores (identificadores de PDPs e URLs de PDPs) existe ainda uma parcela fixa de 153 *bytes* correspondente à estrutura de armazenamento de dois vetores, independentemente do tamanho dos vetores. Para cada um dos N PDPs envolvidos na aplicação da política são utilizados 35 *bytes* para transmitir pdpId e 60 *bytes* para transmitir a URL do Web Service do PDP (identificada em 25 caracteres). Além disso, 1082 são os *bytes* fixos restantes para a codificação da requisição e da resposta. A Figura 5.1 apresenta um exemplo de requisição para uma operação RegisterPolicyDeployment envolvendo 2 PDPs. Os trechos destacados em negrito indicam as partes da requisição que variam de acordo com o número de PDPs envolvidos.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<ns1:RegisterPolicyDeployment xmlns:ns1="http://testuri.org">.
  <deplId xsi:type="xsd:int">123</deplId>
  <deplName xsi:type="xsd:string">nomedaaplicacao</deplName>
  <pdpId xsi:type="SOAP-ENC:Array"..SOAP-ENC:arrayType="xsd:int[2]">
     <item xsi:type="xsd:int">111</item>
     <item xsi:type="xsd:int">112</item>
  </pdpId>
  <pdpUrl xsi:type="SOAP-ENC:Array"..SOAP-ENC:arrayType="xsd:string[2]">
     <item xsi:type="xsd:string">http://www.example.com.br</item>
     <item xsi:type="xsd:string">http://www.pdp.com.br/112</item>
  </pdpUrl>
</ns1:RegisterPolicyDeployment>.
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.1: Documento XML de requisição referente a uma operação RegisterPolicyDeployment envolvendo 2 PDPs.

A Figura 5.2 apresenta um exemplo de resposta para uma operação RegisterPolicyDeployment.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<RegisterPolicyDeploymentResponse>
  <soapVal xsi:type="xsd:int">1</soapVal>
</RegisterPolicyDeploymentResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.2: Documento XML de resposta para uma operação RegisterPolicyDeployment.

A Figura 5.3 apresenta o tráfego gerado quando ocorre uma requisição à operação RegisterPolicyDeployment.

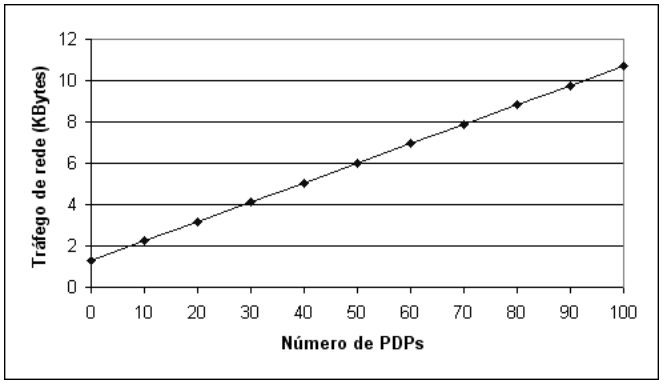

Figura 5.3: Tráfego gerado pela operação RegisterPolicyDeployment

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

```
bytes = 1042 + 39 + 328 + M.(70 + 50 + 49 + 36), ou
bytes = 1409 + M.205
```

sendo 1042 *bytes* a parte fixa da chamada Web Service e, novamente, 39 *bytes* necessários para incluir deplId. Como nesta operação são passados como parâmetro quatro vetores (identificadores de políticas, de IPs de dispositivos, de interfaces de rede e de direção) existe ainda uma parcela fixa de 328 *bytes* correspondente à estrutura de armazenamento de quatro vetores, independentemente do tamanho dos vetores. A variável M representa o número de associações de políticas com PEPs que devem ser registradas no PDP. Primeiramente, o identificador da política requer 70 *bytes* para representar uma URL de 35 caracteres que identifica a política que sofrerá *download*. O PEP a receber a política é identificado por três parâmetros: IP do dispositivo, interface de rede e direção (entrada ou saída). Um IP consome 50 *bytes* para representar 15 caracteres. Cada identificador de interface de rede requer 49 *bytes* para representar um nome de interface com 14 caracteres. Por fim, cada identificador de direção requer 36 *bytes* para representar 1 caracter

que significa entrada ou saída. A Figura 5.4 apresenta um exemplo de requisição para uma operação TransferPolicy envolvendo 2 associações política-PEP. Os trechos destacados em negrito indicam as partes da requisição que variam de acordo com o número de associações política-PEP envolvidas.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<ns1:TransferPolicy xmlns:ns1="http://testuri.org">.
  <deplId xsi:type="xsd:int">123</deplId>
  <policy xsi:type="SOAP-ENC:Array"..SOAP-ENC:arrayType="xsd:string[2]">
     <item xsi:type="xsd:string">http://www.policyserver.com/policy1</item>
     <item xsi:type="xsd:string">http://www.policyserver.com/policy2</item>
  </policy>
  <device xsi:type="SOAP-ENC:Array"..SOAP-ENC:arrayType="xsd:string[2]">
     <item xsi:type="xsd:string">176.124.136.240</item>
     <item xsi:type="xsd:string">176.124.136.241</item>
  </device>
  <interface xsi:type="SOAP-ENC:Array"..SOAP-ENC:arrayType="xsd:string[2]">
     <item xsi:type="xsd:string">interface-eth0</item>
     <item xsi:type="xsd:string">interface-eth1</item>
  </interface>
  <direction xsi:type="SOAP-ENC:Array"..SOAP-ENC:arrayType="xsd:string[2]">
     <item xsi:type="xsd:string">i</item>
     <item xsi:type="xsd:string">o</item>
  </direction>
</ns1:TransferPolicy>.
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.4: Documento XML de requisição referente a uma operação TransferPolicy envolvendo 2 associações política-PEP.

A Figura 5.5 apresenta um exemplo de resposta para uma operação TransferPolicy.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<TransferPolicyResponse>
  <soapVal xsi:type="xsd:int">1</soapVal>
</TransferPolicyResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.5: Documento XML de resposta referente a uma operação TransferPolicy.

A Figura 5.6 apresenta o tráfego gerado por uma operação TransferPolicy para contactar um único PDP. É importante notar que, em um caso real de aplicação, não existiria apenas uma requisição simples, mas sim tantas requisições quantos forem os PDPs envolvidos na aplicação atômica.

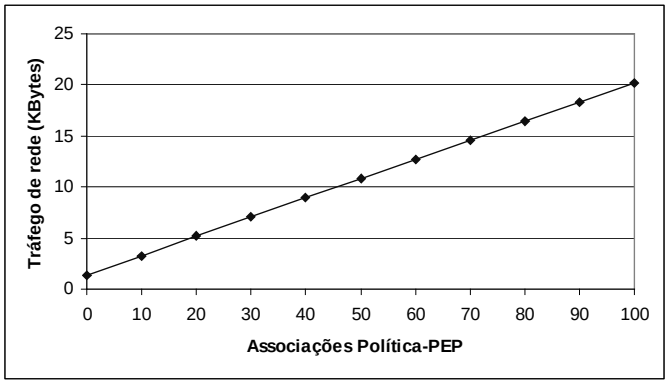

Figura 5.6: Tráfego gerado pela operação TransferPolicy

As mensagens que realizam a coordenação das ativações e desativações atômicas ocorrem diversas vezes durante uma aplicação atômica. Além disso, diferentemente das mensagens analisadas anteriormente, ocorrem em tempos e quantidades não controladas diretamente pelo usuário. Este comportamento é diferente das mensagens RegisterPolicyDeployment e TransferPolicy, que ocorrem apenas quando o gerente da rede efetua a aplicação atômica. Assim, a análise das mensagens de coordenação é ainda mais importante. Para estas mensagens, os gráficos que serão apresentados levam em consideração sempre o número de PDPs envolvidos. Isto se justifica pela tendência é de que cada mensagem desse tipo ocorra em forma de *broadcast*, seja oriunda do PDC para cada PDP, seja de cada PDP para o PDC.

Para as mensagens de coordenação que um PDP envia para o PDC, as quais transportam como parâmetro apenas deplId e pdpId, tem-se a fórmula que segue abaixo. É importante notar que na fórmula são contabilizados tantos os *bytes* oriundos da invocação da operação Web Service quanto os *bytes* resultantes do retorno do resultado da operação. São exemplos deste tipo de mensagem: InitiateActivation, ActivationSuccess, ActivationFail, Deactive e RemovePolicy.

```
bytes = N.(1042 + 39 + 37), ou
bytes = N.1118
```

Novamente, 39 *bytes* codificam um deplId formado por 3 dígitos e 37 *bytes* codificam um pdpId formado por 3 dígitos. Por fim, os outros 1042 *bytes* correspondem à parte fixa das mensagens SOAP de requisição e de resposta (incluindo o nome da operação Web Service). Visto que o número de caracteres que formam o nome das operações varia, foi considerado o tamanho médio do nome das operações deste tipo, que é igual a 14 caracteres. Na fórmula, N representa o número de PDPs.

A Figura 5.7 apresenta um exemplo de requisição de operação ActivationSuccess, a qual contém deplId e pdpId como parâmetro.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<ns1:ActivationSuccess xmlns:ns1="http://testuri.org">.
  <deplId xsi:type="xsd:int">123</deplId>
  <pdpId xsi:type="xsd:int">456</pdpId>
</ns1:ActivationSuccess>.
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.7: Documento XML de requisição referente a uma operação ActivationSuccess.

A Figura 5.8 apresenta um exemplo de resposta para uma operação ActivationSuccess.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<ActivationSuccessResponse>
  <soapVal xsi:type="xsd:int">1</soapVal>
</ActivationSuccessResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.8: Documento XML de resposta referente a uma operação ActivationSuccess.

A Figura 5.9 apresenta o tráfego gerado por operações que contenham deplId e pdpId como parâmetro.

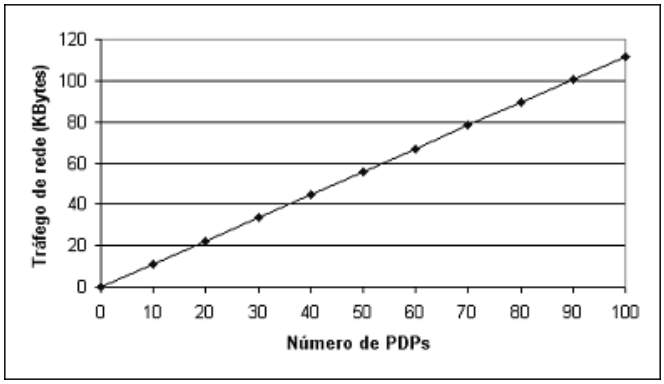

Figura 5.9: Tráfego gerado por operação que contenha como parâmetro deplId e pdpId

Para as mensagens SOAP que transportam como parâmetro apenas deplId, tem-se a fórmula que segue abaixo. Novamente foram contabilizados tantos os *bytes* oriundos da invocação da operação Web Service quanto os *bytes* resultantes do retorno do resultado da operação. São exemplos deste tipo de mensagem as seguintes operações que o PDC invoca em um PDP: InitiateActivation, Commit, RollBack, Abort, Deactive e RemovePolicy.

```
bytes = N.(1026 + 39), ou
bytes = N.1065
```

Mais uma vez, 39 *bytes* codificam um deplId formado por 3 dígitos e outros 1026 *bytes* correspondem à parte fixa das mensagens SOAP de requisição e de resposta somadas (incluindo o nome da operação Web Service). Visto que o número de caracteres que formam o nome das operações varia, foi considerado o tamanho médio do nome das operações deste tipo, que é igual a 10 caracteres. Na fórmula, N representa o número de PDPs. A Figura 5.10 apresenta um exemplo de requisição de uma operação Commit, a qual contém apenas deplId como parâmetro.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<ns1:Commit xmlns:ns1="http://testuri.org">.
  <deplId xsi:type="xsd:int">123</deplId>
</ns1:Commit>.
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.10: Documento XML de requisição referente a uma operação Commit.

A Figura 5.11 apresenta um exemplo de resposta para uma operação Commit.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:si="http://soapinterop.org/xsd">
<SOAP-ENV:Body>
<CommitResponse>
  <soapVal xsi:type="xsd:int">1</soapVal>
</CommitResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.11: Documento XML de resposta referente a uma operação Commit.

A Figura 5.12 mostra o tráfego gerado por operações que contenham apenas deplId como parâmetro.

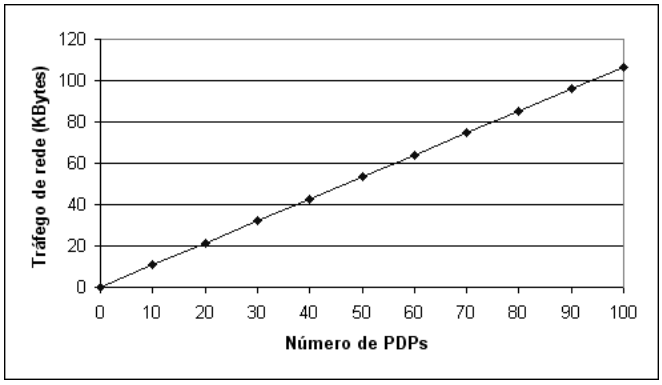

Figura 5.12: Tráfego gerado por uma operação que contenha deplId como parâmetro único

A quantidade total de *bytes* que trafega durante a ativação ou a desativação de uma política não depende somente do número de PDPs envolvidos. Fatores como: poder de processamento do PDP, tráfego no *link* entre o PDP e o PEP, e tempo de resposta de configuração do PEP tornam imprevisíveis a ordem de entrega das mensagens e, conseqüentemente, a quantidade de mensagens enviadas. Além disso, outro fator é determinante: o nível de sincronia de relógio dos PDPs. Caso o relógio interno dos PDPs estejam consideravelmente sincronizados, há a possibilidade de que várias mensagens redundantes sejam enviadas para o PDC. Por exemplo, vários PDPs poderiam enviar "simultaneamente" mensagens de InitiateActivation ou Deactive, apesar de apenas a primeira mensagem destes tipos ser levada em consideração pelo PDC.

Por haver esta série de fatores imponderáveis em uma avaliação teórica, partiu-se para uma análise do desempenho do protocolo através da execução do mesmo em um cenário de testes.

5.2 Avaliação do Protocolo em um Cenário de Testes

Os testes foram realizados no cluster LabTeC do Instituto de Informática da Universidade Federal do Rio Grande do Sul. Este é um cluster de alto desempenho atualmente formado por 20 nodos biprocessados interligados por uma rede FastEthernet de 100 Mbps, via um *switch*. A configuração de cada nodo é a seguinte:

- 2 Processadores Pentium III 1.2 Ghz;
- Memória RAM 1 GB;
- Sistema Operacional GNU/Linux Gentoo, kernel 2.6.14;
- Apache 2.0.54
- PHP 4.4.0
- MySQL

Nos testes desta dissertação foram utilizados 17 nodos do cluster. Destes, 16 ficaram responsáveis por executar PDPs, e o nodo restante abrigou o *software* PDC. Os testes foram executados para as seguintes quantidades de PDPs: 2, 3, 4, 6, 8, 12, 16, 24 e 32. Nos testes que variaram de 2 a 16 PDPs foi disparado um *software* PDP por nodo. Para 24

PDPs teve-se 8 nodos rodando apenas 1 PDP e os outros 8 nodos rodando 2 PDPs cada. Já para 32 PDPs teve-se os 16 nodos rodando 2 PDPs cada um. Nestes casos onde existem 2 PDPs rodando em um nodo, a mesma instalação MySQL foi compartilhada pelos dois PDPs de um mesmo nodo, porém utilizando bases de dados distintas.

Para a obtenção dos resultados dos testes foi utilizado o *software* TCPDUMP (TCPDUMP, 2002). O TCPDUMP foi executado apenas no nodo que hospedou o PDC. Isto foi suficiente para analisar o protocolo como um todo, pois qualquer mensagem do protocolo tem o PDC como origem ou como destino da comunicação.

O comportamento do protocolo de consenso foi avaliado mediante três situações: ativações com sucesso, ativações com falha e desativações de uma política. A avaliação separada de ativações com sucesso e de ativações sem sucesso é importante por se tratar de dois casos onde o número de mensagens que são trocadas poder ser diferente.

Quando uma ativação tem sucesso é preciso que cada PDP envie uma mensagem ActivationSuccess para o PDC, para que este último divulgue Commit para todos os PDPs. Por outro lado, quando uma ativação falha, basta a primeira mensagem ActivationFail ser recebida pelo PDC para o mesmo divulgar Rollback para todos os PDPs. Se o PDP receber uma mensagem Rollback antes de enviar seu resultado local (ActivationSuccess ou ActivationFail), este envio é evitado, pois já se conhece o resultado global da ativação. Portanto, há a tendência natural de que o tráfego gerado por ativações sem sucesso ser menor que o tráfego gerado por ativações com sucesso, visto que mensagens ActivationFail podem ser omitidas.

Para cada uma das três situações avaliadas foram medidos tempo de resposta e tráfego gerado. Um fator que afeta as avaliações destes dois parâmetros de desempenho é a diferença entre os relógios internos de cada nodo que hospeda os PDPs. Esta diferença afeta diretamente o número de mensagens InitiateActivation que são recebidas pelo PDC. O pior caso, onde o maior número dessas mensagens são enviadas, é quando os PDPs estão com os horários de seus relógios internos mais próximos. Conforme a distância entre os tempos destes relógios aumenta, existe a tendência de que o PDC consiga evitar o envio de várias mensagens InitiateActivation por parte dos PDPs. Isto porque o PDC recebe a primeira mensagem InitiateActivation e a envia para o restante dos PDPs envolvidos. Se a distância entre os relógios dos PDPs é suficientemente grande, o PDP acaba recebendo InitiateActivation antes mesmo de enviá-la, pois seu tempo de ativação ainda foi atingido. Levando esse fato em consideração, nos testes realizados nesta dissertação decidiu-se por utilizar os relógios internos o mais próximos possível, obtendo-se assim uma estimativa do pior caso. Para este fim de sincronização foi utilizado o protocolo NTP (*Network Time Protocol*) (NTP Project, 2006).

Durante os testes não foram realizadas reais configurações de PEPs. Cada PDP, ao chegar o momento de efetivamente ativar ou desativar uma política em seus PEPs, não executava qualquer procedimento de configuração. O PDP apenas respondia para o PDC "sucesso" (ActivationSuccess) ou "falha" (ActivationFail). Tal medida foi adotada para simplificar os requisitos do cenário de testes, pois pôde-se descartar a necessidade de PEPs reais a serem configurados. Tendo em vista que o objetivo dos testes era analisar apenas o comportamento do protocolo de consenso, tal simplificação foi bastante prática.

5.2.1 Ativações com Sucesso

O atraso inserido por uma ativação atômica de uma política foi contabilizado a partir do instante de tempo em que o PDC recebeu a primeira requisição de abertura de conexão TCP, enviada por um PDP, com o objetivo de enviar uma mensagem InitiateActivation.

O término de uma ativação foi considerado o instante de tempo em que o PDC recebeu o último pacote "ACK"de fechamento de conexão TCP relativa ao envio, por parte do PDC, da última mensagem de Commit ou de Rollback destinada aos PDPs. Com a utilização de PDPs que não efetuavam reais aplicações de política, pôde-se reduzir o tempo de aplicação de uma política (dependente de dispositivo de rede) ao mínimo possível (apenas o suficiente para o PDP responder ActivationSuccess ou ActivationFail). Assim, foram obtidos resultados que retratam apenas o atraso inserido pelo protocolo de consenso.

O tráfego gerado durante ativações com sucesso são apresentados na Figura 5.13.

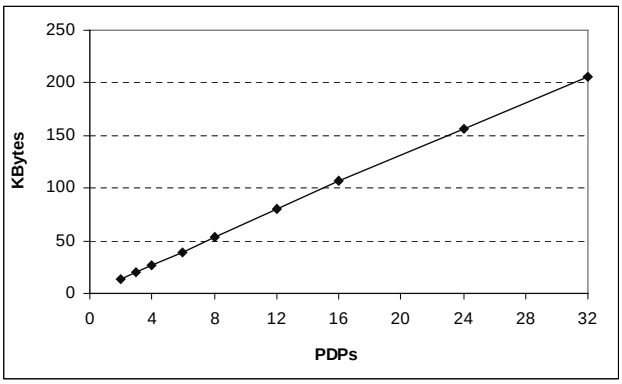

Figura 5.13: Tráfego gerado por ativações com sucesso

O gráfico apresenta uma tendência natural das ativações com sucesso no protocolo proposto: um comportamento previsível e derivável do número de PDPs utilizados. A única variação possível no número de mensagens trocadas nesta operação é o número de mensagens InitiateActivation enviadas para o PDC (conforme citado acima quando foi explicado a influência das diferenças de relógio dos PDPs). Excetuando-se isso, o número de mensagens enviadas é diretamente proporcional ao número de PDPs envolvidos.

O tempo decorrido durante ativações com sucesso são apresentados na Figura 5.14.

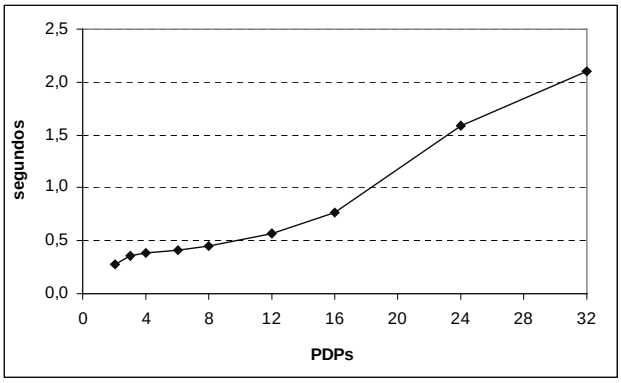

Figura 5.14: Tempo decorrido durante ativações com sucesso

Pode-se observar que o comportamento das ativações com sucesso varia de acordo com o número de PDPs envolvidos. Porém, nas execuções onde estavam envolvidos 24 e 32 PDPs pode-se perceber um acréscimo considerável no tempo decorrido durante a ativação. Tal diferença é creditada ao fato de que nestes testes existiam nodos executando dois *softwares* PDP. Portanto, o fato dos nodos serem biprocessados não foi suficiente para manter a tendência da reta obtida nas medições com menos de 16 PDPs, fazendo com que os resultados para 24 e 32 PDPs destoassem na reta obtida.

5.2.2 Ativações sem Sucesso

Com o objetivo de avaliar as diferenças de tráfego gerado e de tempo decorrido decorrentes de ativação sem sucesso partiu-se para um cenário onde PDPs foram configurados para responderem de modo a indicar falha na ativação. A cada execução de uma ativação contendo, por exemplo, N PDPs, foi escolhido um subconjunto desses N PDPs para responderem ActivationFail. O restante dos PDPs continuou respondendo ActivationSuccess.

Os testes de ativação com falha foram executados com diversas combinações de número de PDPs respondendo falha. A tabela abaixo apresenta, na coluna da esquerda, o número de PDPs utilizados em cada teste. Na coluna da direita são apresentadas as combinações de PDPs que responderam falha ao receber um pedido de início de ativação de política. Por exemplo, utilizando 8 PDPs foram realizadas 4 combinações de testes, envolvendo 8, 4, 2 e 1 PDPs respondendo falha na ativação.

Tabela 5.1: Combinações de testes realizados envolvendo falhas

	PDPs em falha					
Número de PDPs	100%	50%	25%	12,5%	6,75%	3,375%
2	2	1				
4	4	2	1			
8	8	4	2	1		
16	16	8	4	2	1	
32	32	16	8	4	2	1

O tráfego gerado durante ativações sem sucesso são apresentados na Figura 5.15.

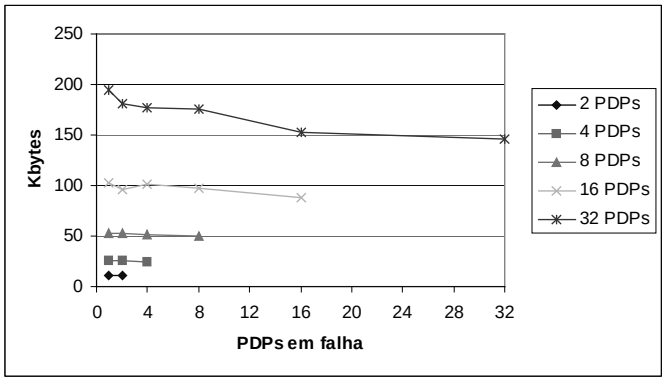

Figura 5.15: Tráfego gerado por ativações sem sucesso

Os resultados obtidos apresentaram pequenas reduções de tráfego para as ativações sem sucesso. A proximidade dos relógios dos PDPs determinou o envio de ActivationFail de forma praticamente simultânea por parte de cada PDP. Os testes onde ficou mais clara a redução de tráfego gerado conforme o número de PDPs com falha de ativação era aumentado foram os de 16 e de 32 PDPs.

As ativações sem sucesso poderiam apresentar melhores resultados de tráfego de rede gerado caso houvesse alguma real configuração de PEP por parte dos PDPs. Neste caso, o tempo que cada PDP levaria para responder ActivationSuccess ou ActivationFail poderia variar, fazendo com que tais mensagens não trafegassem tão simultaneamente na rede.

O tempo decorrido durante ativações sem sucesso são apresentados na Figura 5.16.

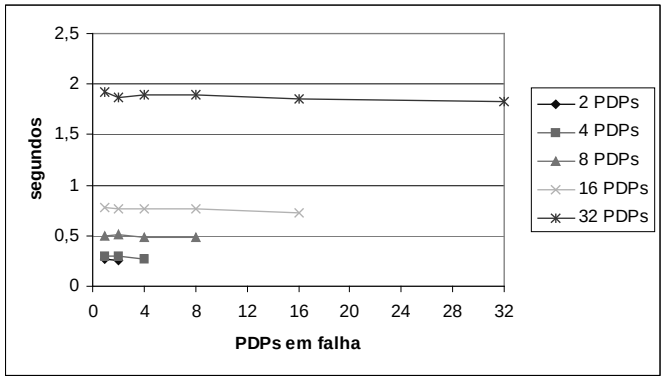

Figura 5.16: Tempo decorrido durante ativações sem sucesso

Avaliando os tempos apresentados no gráfico percebe-se que os resultados sofreram poucas alterações conforme o número de PDPs com falha de ativação variou. O fato de que o tráfego gerado variou mais acentuadamente do que o tempo de resposta se explica porque, na prática, os *bytes* trafegados a mais nos testes em que haviam poucos PDPs com falha ativação correspondem a troca de mensagens desnecessárias, disparadas quando o resultado da ativação já estava determinado. Assim, estas comunicações a mais não influenciaram o tempo gasto em cada passo do algoritmo, a não ser pelo tempo necessário para o PDC receber uma mensagem ActivationFail desnecessária e descartá-la.

5.2.3 Desativações

O atraso inserido por uma desativação atômica foi contabilizado a partir do instante de tempo em que o PDC recebeu a primeira requisição de abertura de conexão TCP, enviada por um PDP, com o objetivo de enviar uma mensagem Deactive. O término de uma desativação foi considerado o instante de tempo em que o PDC recebeu o último pacote "ACK"de fechamento de conexão TCP relativa ao envio, por parte do PDC, da última mensagem Deactive destinada aos PDPs. O intervalo de tempo de desativação foi definido excluindo o tempo em que a política seria efetivamente removida nos PEPs, com o objetivo de isolar o *overhead* inserido pelo protocolo.

O tráfego gerado durante desativações são apresentados na Figura 5.17.

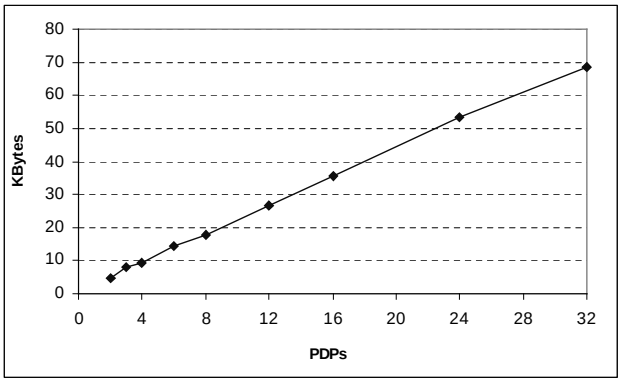

Figura 5.17: Tráfego gerado por desativações

Os resultados obtidos nos testes de desativação são similares aos obtidos nos testes de ativação com sucesso. A reta obtida no gráfico de tráfego gerado mostrou-se bastante previsível conforme o número de PDPs era aumentado. Assim como nas ativações com

sucesso, a única variação possível no número de mensagens trocadas é a mensagem que comunica a chegada do momento de disparar o processo desejado (seja ativação ou desativação). Nas ativações, esta mensagem é InitiateActivation. Nas desativações, esta mensagem é Deactive, enviada por cada PDP. Como na ativação, esta variação é decorrente da diferença entre os relógios internos de cada PDP.

Em um cenário onde a diferença entre os relógios dos PDPs não fosse tão reduzida, tem-se que os resultados de tráfego de rede obtidos poderiam reduzir bastante. Nos testes realizados nesta dissertação, com os relógios dos PDPs significativamente sincronizados, a tendência é de que em uma desativação cada PDP envie uma mensagem Deactive para o PDC (totalizando N para N PDPs). Com as invariáveis N mensagens Deactive enviadas pelo PDC (uma para cada PDP), tem-se um total de mensagens que tende a 2N. Com uma maior distância entre os relógios, poderia-se imaginar, por exemplo, N + 1 mensagens, sendo N as mensagens enviadas pelo PDC e 1 a mensagem necessária para que apenas um PDP comunique Deactive para o PDC.

O tempo decorrido durante desativações são apresentados na Figura 5.18.

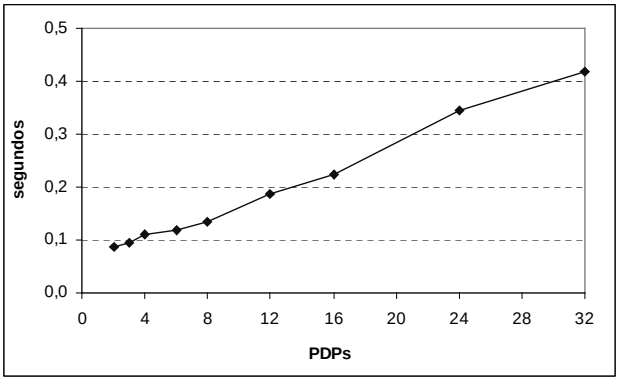

Figura 5.18: Tempo decorrido durante desativações

O comportamento da reta obtida para os resultados de tempo de desativação seguem os resultados obtidos para ativações com sucesso. Porém, obviamente, os tempos de desativação são significativamente menores, visto que o processo de desativação é mais simples que o de ativação, envolvendo menos passos.

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação foi definido um protocolo de consenso, baseado no protocolo 2PC (*Two-Phase Commit*), que trata as particularidades da aplicação atômica de políticas de gerenciamento de redes com suporte a QoS, auxiliando na coordenação de uma série de PDPs. Foram apresentadas algumas alternativas de funcionamento, sendo suas vantagens e cenários de adoção discutidos.

O modelo de comunicação proposto, envolvendo o PDC como um elemento centralizador das comunicações entre os PDPs e implementando o mesmo junto à ferramenta de políticas, apresenta as vantagens já discutidas durante esta dissertação. Entre elas pode-se citar a ausência de necessidade de cada PDP conhecer e poder se comunicar com os PDPs restantes. Porém, cabe ressaltar que o protocolo pode se tornar um tanto frágil na medida em que a existência de um ponto único de falha restringe sua robustez, como é inerente a qualquer solução onde um elemento centralizador é utilizado.

A implementação da aplicação atômica de políticas deu-se não somente a partir da codificação do protocolo de consenso utilizando a tecnologia de Web Services, mas também envolveu o desenvolvimento da entidade PDC (responsável por processar as mensagens vindas dos PDPs) e da codificação de um procedimento efetivo de *rollback*. A utilização de PHP nestas implementações deve-se a compatibilidade com o restante do ambiente QAME. Porém, é importante ressaltar que esta linguagem, em certos momentos, mostrou-se limitada, principalmente quando se desejava um suporte mais elaborado de programação com *threads*.

Considerando a análise de desempenho apresentada pelo protocolo, conclui-se que sua adoção é viável. A onerosa codificação textual do XML, típica de mensagens SOAP, não se caracterizou como um empecilho para a utilização do protocolo. Sendo assim, para esta implementação, o uso de Web Services mostrou-se bastante proveitoso, uma vez que obtiveram-se as vantagens inerentes à solução sem comprometer significativamente o consumo de banda na rede.

Nas avaliações práticas realizadas nesta dissertação o tempo de ativação da política junto ao PEP não foi considerado. Como descrito na seção anterior, esta decisão foi adotada com o objetivo de medir exclusivamente o tempo consumido pelo protocolo de consenso. Porém, quando se considera um cenário real de aplicação de políticas, é preciso levar em conta diferentes tempos que podem ser consumidos por diferentes PEPs para que uma política efetivamente entre em funcionamento. Estes diferentes tempos podem inclusive afetar o tráfego gerado na rede pelo protocolo, visto que as mensagens de ActivationSuccess e ActivationFail serão enviadas para o PDC de forma não tão simultânea quanto ocorreu nos testes realizados nesta dissertação.

Apesar da segurança não ser o ponto principal deste trabalho, esta deve ser uma preocupação de qualquer sistema computacional. Portanto, alguns pontos da implementação

realizada nesta dissertação devem ser abordados em trabalhos futuros. Por exemplo, no PDP AltQ, a comunicação entre o PDP e o PEP se dá através da execução de comandos remotos utilizando SSH (*Secure Shell*), sem a necessidade de fornecimento de senha. Além disso, cabe ressaltar que os comandos são executados com privilégios de superusuário (*root*), pois apenas este usuário tem permissão para configurar o *daemon* do AltQ.

Como trabalhos futuros pretende-se abordar o suporte a políticas com mais de uma ação associada. Nestas políticas haverá uma ação principal e uma secundária, onde, ao chegar o momento da ativação da política, um PDP tentaria aplicar a primeira ação. Em caso de insucesso, o protocolo determinaria um *rollback* e uma nova ativação seria disparada imediatamente, porém dessa vez com o objetivo de aplicar a segunda ação (secundária).

Este trabalho discutiu uma série de aspectos importantes para a adoção de operações atômicas no gerenciamento baseado em políticas. Ainda que alguns pontos tenham sido destinados a trabalhos futuros, pode-se afirmar que esta dissertação deve servir como uma base importante para futuras investigações sobre o assunto. Além disso, ainda que a implementação efetuada nesta dissertação seja apenas um protótipo, pode-se afirmar que a mesma está operacional e pode ser adotada prontamente para aplicações atômicas dentro do ambiente QAME.

REFERÊNCIAS

AYALA, D. **NuSOAP - Web Services Toolkit for PHP.** Disponível em: <http://dietrich.ganx4.com/nusoap>. Acesso em: 27 maio 2006.

BRADEN, R. et al. **Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification**: RFC 2205. [S.l.]: IETF, 1997.

CASE, J. et al. **A Simple Network Management Protocol (SNMP)**: RFC 1157. [S.l.]: IETF, 1990.

CHADHA, R. et al. Policy-based mobile ad hoc network management. In: INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS POLICY, 5., 2004. **Proceedings...** [S.l.]: IEEE Computer Society, 2004. p.35–44.

CHAN, K. et al. **COPS usage for policy provisioning (COPS-PR)**: RFC 3084. [S.l.]: IETF, 2001.

CHO, K. Managing Traffic with ALTQ. In: USENIX ANNUAL TECHNICAL CONFERENCE, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.121–128.

CISCO. 1992. Disponível em: <http://www.cisco.com>. Acesso em: 12 nov. 2005.

CPQD. **CPqD Telecom and IT Solutions**. Disponível em: <http://www.cpqd.com.br>. Acesso em: 22 maio 2006.

CURBERA, F. et al. Unraveling the Web Services Web: an Introduction to SOAP, WSDL, and UDDI. **IEEE Internet Computing**, New York, v.6, n.2, p.86-93, Mar./Apr. 2002.

DURHAM, D. et al. **The COPS (Common Open Policy Service) Protocol**: RFC 2748. [S.l.]: IETF, 2000.

EDGAR, E. T. **A XML Policy-Based Approach for RSVP**. [S.l.: s.n.], 2004. Disponível em: <http://citeseer.ist.psu.edu/721855.html>. Acesso em: 12 maio 2005.

ENNS, R. **NETCONF Configuration Protocol - draft-ietf-netconf-prot-04**. [S.l.]: IETF, 2003. DRAFT.

EZ SYSTEMS. **ezSOAP (Transcription Replacement Software**. Disponível em: <http://dietrich.ganx4.com/nusoap>. Acesso em: 15 out. 2005.

FREEBSD. **The FreeBSD Project**. [S.l.: s.n.], 1995. Disponível em: <http://www.freebsd.org/>. Acesso em: 18 dez. 2005.

GRANVILLE, L.; CECCON, M.; TAROUCO, L.; ALMEIDA, M.; CARISSIMI, A. An Approach for Integrated Management of Networks with Quality of Service Support Using QAME. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, DSOM, 12., 2001. **Proceedings**... [S.l.]: INRIA, 2001, p.167-178.

GRAY, J. **Notes on Database Systems**. [S.l.]: IBM, 1978. (Reserch Report RJ2188).

GUO, X. et al. A policy-based network management system for IP VPN. In: ICCT, 2003. **Proceedings...** [Piscataway, NJ]: IEEE, 2003. v.2, p.1630–1633.

MACFADEN, M. et al. **Configuring Networks and Devices with Simple Network Management Protocol (SNMP)**: RFC 3512. [S.l.]: IETF, 2003.

MCCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets: MIB-II**: RFC 1213. [S.l.]: IETF, 1991.

MILLS, D. **Network Time Protocol (Version 3) Specification, Implementation and Analysis**: RFC 1305. [S.l.]: IETF, 1992.

MITRA, N. **SOAP Version 1.2 Part 0: primer.** W3C Recommendation 24. Disponível em: <http://www.w3c.org>. Acesso em: 06 jun. 2005.

MOORE, B. **Policy Core Information Model (PCIM) Extensions**: RFC 3460. [S.l.]: IETF, 2003.

MOORE, B. et al. **Policy Core Information Model – Version 1 Specification**: RFC 3060. [S.l.]: IETF, 2001.

MySQL. **MySQL**: the world's most popular open source database. 1995. Disponível em: <http://www.mysql.com>. Acesso em: 22 maio 2006.

NIKOLAKIS, Y. et al. A Policy-Based Management Architecture for Flexible Service Deployment in Active Networks. In: IWAN, 5., 2003. **Proceedings...** New York: Springer, 2004. p.240–251. (Lecture Notes in Computer Science, v.2982).

NTP Project. **NTP**: The Network Time Protocol. Disponível em: <http://www.ntp.org/>. Acesso em: 6 fev. 2006.

OPENLDAP. **OpenLDAP**: community developed ldap software. 2005. Disponível em: <http://www.openldap.org>. Acesso em: 10 out. 2005.

PHP. **PHP**: Hypertext Preprocessor. 2001. Disponível em: <http://www.php.net>. Acesso em: 27 maio 2005.

PEAR. **The PHP extension and application repository**. Disponível em: <http://pear.php.net>. Acesso em: 16 out. 2005.

RAMAN, L. OSI Systems and Network Management. **IEEE Communications Magazine**, [S.l.], p.46–53, Mar. 1998.

RNP. **Rede Nacional de Ensino e Pesquisa**. [S.l.: s.n.], 2006. Disponível em: <http://www.rnp.br>. Acesso em: 22 maio 2006.

SANCHEZ, L.; MCCLOGHRIE, K.; SAPERIA, J. **Requirements for Configuration Management of IP-based Networks**: RFC 3139. [S.l.]: IETF, 2001.

SCHöNWäLDER, J.; PRAS, A.; MARTIN-FLATIN, J. On the Future of Internet Management Technologies. **IEEE Communications Magazine**, [S.l.], v.41, n.10, p.90–97, Oct. 2003.

SHERIDAN-SMITH, N. **A Distributed Policy-based Network Management (PBNM) System for Enriched Experience Networks (EENs)**. [S.l.: s.n.], 2003. Disponível em: <http://citeseer.ist.psu.edu/sheridan-smith03distributed.html>. Acesso em: 6 maio 2005.

SNIR, Y. et al. **Policy Quality of Service (QoS) Information Model**: RFC 3644. [S.l.]: IETF, 2003.

SOLLINS, K. **The TFTP Protocol**: RFC 1350. [S.l.]: IETF, 1992.

STALLINGS, W. **SNMP, SNMPv2,SNMPv3, and RMON 1 and 2**. 3rd ed. Boston: Addison-Wesley, 1998.

TANENBAUM, A. S. **Distributed Operating Systems**. Upper Saddle River: Prentice-Hall, 1995. 614p.

TCPDUMP. **The Tcpdump Project**. [S.l.: s.n.], 2002. Disponível em: <http://sourceforge.net/projects/tcpdump>. Acesso em: 12 fev. 2006.

TSAROUCHIS, C. et al. A policy-based management architecture for active and programmable networks. **IEEE Network**, [S.l.], v.17, n.3, p.22–28, May/June 2003.

WALDBUSSER, S.; SAPERIA, J.; HONGAL, T. **Policy Based Management MIB - draft-ietf-snmpconf-pm-15 (Work-in-progress)**. [S.l.]: IETF, 2004. DRAFT.

WESTERINEN, A. et al. **Terminology for Policy-Based Management**: RFC 3198. [S.l.]: IETF, 2001.

YAVATKAR, R.; PENDARAKIS, D.; GUERIN, R. **A Framework for Policy-based Admission Control**: RFC 2753. [S.l.]: IETF, 2000.