ANDRÉ GUSTAVO BELLER

UMA ARQUITETURA PARA GERENCIAMENTO DE QoS BASEADO EM POLÍTICAS

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

CURITIBA

2005

ANDRÉ GUSTAVO BELLER

UMA ARQUITETURA PARA GERENCIAMENTO DE QoS BASEADO EM POLÍTICAS

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

Área de Concentração: *QoS e Gerenciamento baseado em Políticas*

Orientador: Prof. Dr. Edgard Jamhour

CURITIBA

2005

Beller, André Gustavo
Uma Arquitetura para Gerenciamento de QoS Baseado em Políticas. Curitiba, 2005. 144p.

Dissertação – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. QoS 2. QPIM 3. DiffServ 4. Política.
I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada.

À minha esposa.

Agradecimentos

Primeiramente, ao Prof. Edgard Jamhour pela excelente orientação, pelas produtivas discussões e pelos conselhos que, inclusive, foram além do tema desta pesquisa.

Às contribuições dos professores Manoel Camillo Penna e Mauro Fonseca.

Ao Ricardo Nabhen que forneceu a base do código COPS implementado.

Ao Timothy Squair com quem trabalhei no desenvolvimento do código COPS-PR.

Ao Marcelo Zanetti sempre muito prestativo para auxiliar na configuração *DiffServ* dos roteadores Linux.

E a todos aqueles que de alguma forma contribuíram para realização deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AF	*Assured Forwarding*
ANS.1	*Abstract Syntax Notation One*
API	*Application Program Interface*
BA	*Behavior Aggregate*
BW	*Bandwidth*
CAT	*Client Accept*
CC	*Client Close*
CIM	*Common Information Model*
CNF	*Conjunctive Normal Form*
COPS	*Common Open Policy Service*
COPS-PR	*Common Open Policy Service for Policy Provisioning*
DEC	*Decision*
DEN	*Directory Enabled Networks*
DIT	*Directory Information Tree*
DMTF	*Distributed Management Task Force*
DNF	*Disjunctive Normal Form*
DRQ	*Delete Request State*
DS	*Differentiated Service*
DSCP	*Differentiated Service Code Point*
DTD	*Document Type Definitions*

ECN	*Explicit Congestion Notification*
EF	*Expedited Forwarding*
FTP	*File Transfer Protocol*
HTTP	*Hypertext Transfer Protocol*
IANA	*Internet Assigned Numbers Authority*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPSec	*Internet Protocol Security*
ISO	*International Standards Organization*
ITU-T	*International Telecommunication Union - Telecommunication Standardization Sector*
KA	*Keep Alive*
LDAP	*Lightweight Directory Access Protocol*
LPDP	*Local Policy Decision Point*
MF	*Multifield*
MIB	*Management Information Base*
MOF	*Management Object Format*
OPN	*Client Open*
PBN	*Policy Based Network*
PBNM	*Policy Based Network Management*
PCIM	*Policy Core Information Model*
PCIMe	*Policy Core Information Model Extensions*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PHB	*Per Hop Behavior*
PIB	*Policy Information Base*
QoS	*Quality of Service*
QPIM	*Quality of Service Policy Information Model*
RAP	*Resource Allocation Protocol*
REQ	*Requisition*
RFC	*Request for Comments*
RPT	*Report State*

RSVP	*Resource Reservation Protocol*
SLA	*Service Level Agreement*
SLS	*Service Level Specification*
SNMP	*Simple Network Management Protocol*
SSC	*Synchronize Complete*
SSQ	*Synchronize State Request*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
TOS	*Type of Service*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
VoD	*Video on Demand*
VoIP	*Voice over Internet Protocol*
XML	*Extensible Markup Language*
XSLT	*Extensible Stylesheet Language Transformations*
W3C	*World Wide Web Consortium*

Resumo

A convergência dos meios de comunicação (voz, vídeo, dados) está afetando diretamente o planejamento, gerenciamento e manutenção das infra-estruturas de redes, as quais devem transportar simultaneamente e de maneira eficaz dados, voz e vídeo. Dentro deste novo conceito de redes, um importante elemento a ser gerenciado é a qualidade de serviço (QoS). O gerenciamento de QoS é uma tarefa bastante complexa e envolve conceitos administrativos, tais como: distribuição dos recursos disponíveis, configuração de níveis de usuários, diferenciação das aplicações e contabilidade. O Gerenciamento de Redes Baseado em Políticas (PBNM) tem demonstrado ser uma estratégia eficiente para simplificar a administração de sistemas complexos, normalmente caracterizados pela heterogeneidade dos dispositivos e pela variedade de serviços oferecidos. Este trabalho propõe uma arquitetura de gerenciamento de redes baseado em políticas para automatizar os processos de geração e distribuição de configuração para dispositivos de rede em um ambiente *DiffServ*. A arquitetura é baseada nos padrões do IETF e introduz um novo modelo de política de alto nível (objetivos de negócio) para simplificar o processo de descrição das políticas de QoS. A arquitetura é definida em três camadas: modelo de política de alto nível, modelo de política independente de dispositivo e um modelo de política dependente de dispositivo. O trabalho explica os modelos propostos e descreve os processos utilizados para conversão das políticas de alto nível em configuração de dispositivos. Dentro da arquitetura, também são discutidas e avaliadas três diferentes estratégias para o servidor de políticas atualizar dinamicamente a configuração dos dispositivos de rede.

Palavras-Chave: 1. QoS 2. QPIM 3. DiffServ 4. Política

Abstract

The communication convergence (voice, video, data) is changing the way to plan, manage and maintain the network infrastructures. In this new concept of networking an important element to be managed is quality of service (QoS). The QoS management is a complex task and includes administrative concepts, such as: resource allocation, user levels assignment, differentiation of applications and billing. Policy-Based Network Management (PBNM) has proved to be an efficient strategy for simplifying the administration of complex systems in which, normally, there are heterogeneous devices and large variety of services. This work proposes a policy-based network management framework for automating the processes of generating and distributing *DiffServ* configuration to network devices. The framework is based on IETF standards and introduces a new high-level policy model for simplifying the process of defining QoS policies. The framework is defined in three layers: a high-level policy model, a device-independent policy model and a device-dependent policy model. The work explains the proposed models and describes the policy translation processes. Also, three different strategies are discussed and evaluated for the policy server to dynamically update the device configuration.

Keywords: 1. QoS 2. QPIM 3. DiffServ 4. Policy.

Capítulo 1

Introdução

1.1. Desafio

Atualmente, a maioria das redes de computadores ainda utiliza o modelo de serviço que trata todas as requisições de forma equivalente, seguindo a concepção do protocolo IP (Internet Protocol) [RFC791], o qual foi projetado para funcionar através do método do "melhor esforço" (*best-effort*). Este modelo é aplicado tanto no processamento dos servidores como no tráfego das redes.

O protocolo IP foi concebido visando a simplicidade da configuração e das infra-estruturas das redes, transferindo a complexidade para os pontos finais (*end-hosts*). Grande parte do sucesso da Internet deve-se a este conceito de projeto que permitiu um modelo eficiente para escalabilidade das redes, capacitando-as para absorção do crescimento na demanda. O desenvolvimento do protocolo IP teve suas origens na década de 1970 [LEI00], com o objetivo de atender às aplicações típicas da Internet como: correio eletrônico (e-mail) e transferência de arquivos (FTP), para as quais o modelo do melhor esforço é adequado.

Mas a crescente utilização das aplicações multimídia e a forte tendência para convergência dos meios de comunicação (telefonia, televisão, internet), tornaram o modelo do melhor esforço ineficaz e inadequado. Esta mudança de paradigma afeta diretamente o planejamento, gerenciamento e manutenção das infra-estruturas de redes, as quais devem transportar simultaneamente e de maneira eficaz dados, voz e vídeo. Dentro deste novo conceito de redes, um importante elemento a ser gerenciado é a qualidade de serviço (QoS).

QoS pode ser definida, genericamente, como o conjunto de requisitos necessários para que uma determinada funcionalidade seja executada de acordo com

suas especificações. A utilização de mecanismos de QoS torna-se essencial em aplicações como videoconferência, voz sobre IP (*VoIP*), vídeo sob demanda (*VoD*), *e-learning*, *e-commerce* etc., as quais têm seu funcionamento dependente do cumprimento de certos parâmetros, principalmente relacionados à largura de banda e atraso de entrega. O conceito de QoS pode ser implementado também na diferenciação da prestação de serviços, com níveis de priorização e preços diferenciados.

O gerenciamento de QoS é uma tarefa bastante complexa e envolve conceitos administrativos, tais como: distribuição dos recursos disponíveis, configuração de níveis de usuários, diferenciação das aplicações e contabilidade. Para que o comportamento da rede atenda às necessidades coorporativas definidas a partir das regras de negócio, é necessário que o sistema de gerenciamento utilizado seja capaz de traduzir os objetivos administrativos descritos em linguagem humana para comandos de configuração específicos dos dispositivos de rede. Além disso, a distribuição das configurações precisa ser realizada de forma coerente para que cada dispositivo receba informações de acordo com suas capacidades e função desempenhada na rede.

Devido à grande heterogeneidade dos sistemas e a diversidade nas estruturas de negócios, a definição de uma ferramenta genérica de gerenciamento capaz de facilitar a administração de QoS em diferentes cenários é o desafio deste trabalho.

1.2. Motivação

O Gerenciamento de Redes Baseado em Políticas (PBNM) tem demonstrado ser uma eficiente estratégia para simplificar a administração de sistemas complexos, normalmente caracterizados pela heterogeneidade dos dispositivos e pela variedade de serviços oferecidos [FLE01] [PON02]. Dentro do IETF[1], dois grupos de trabalho: *Resource Allocation Protocol* (RAP) *Working Group* (WG) e *Policy Framework WG* estão desenvolvendo e padronizando protocolos usados em PBNM.

Para representação de políticas em uma forma padronizada, o IETF especificou o modelo PCIM (*Policy Common Information Model*) [MOO01], que foi atualizado pelo PCIMe (*Policy Common Information Model Extensions*) [MOO03], e é derivado do CIM (*Common Information Model*). O PCIM/PCIMe é composto por classes genéricas que servem de base para especialização do modelo de acordo com as necessidades específicas de cada área de aplicação. Particularmente, para representação de políticas

[1] Internet Engineering Task Force

de QoS, o IETF define o modelo QPIM (*Policy Quality of Service Information Model*) [SNI03]. As classes QPIM descrevem o condicionamento de tráfego na rede, conforme as metodologias *IntServ* [BRA94] e *DiffServ* [BLA98]. Além de definir modelos para representação de políticas, o IETF também descreve a arquitetura básica para implementação do gerenciamento baseado em políticas. A [GUE00] especifica os dois principais componentes desta arquitetura:

- *Policy Decision Point* (PDP) é a entidade lógica que reside no servidor de políticas e é o responsável por tomar as decisões das políticas a serem e executadas pelos nós da rede.
- *Policy Enforcement Point* (PEP) é a entidade que reside em um nó de rede e atua como cliente do PDP. A responsabilidade do PEP é realizar os procedimentos necessários para que as decisões enviadas pelo PDP sejam cumpridas.

Para comunicação entre o servidor (PDP) e seus clientes (PEPs) foi desenvolvido o protocolo Common Open Policy Service Protocol (COPS) [DUR00]. O COPS suporta, fundamentalmente, dois modelos para interação entre PEP e PDP:

- Modelo *Outsourcing* – todas as decisões são tomadas em tempo real, ou seja, cada evento recebido pelo PEP gera uma requisição para o PDP. A decisão do PDP determina o comportamento do PEP naquele instante.
- Modelo *Provisioning* – também é chamado de configuração e todas as informações das políticas relevantes para o PEP são enviadas pelo PDP em uma transação atômica, respondendo a uma requisição inicial. Na estratégia de configuração, é usada a extensão COPS-PR [CHA01].

O objetivo das políticas é estabelecer um método de administração mais amigável à linguagem humana, promovendo o gerenciamento de todo o ambiente, ao invés de tratar da configuração individual de cada elemento da rede. As políticas são representadas por regras que combinam condições e ações, conforme a semântica: se <condição> então <ação>. O uso de políticas torna o sistema programável e flexível. A modificação ou cancelamento de antigas políticas e a inclusão de novas políticas, permite que o sistema seja facilmente adaptado para atender a mudanças de requisitos.

Atualmente, a grande maioria das redes utiliza o protocolo de gerenciamento SNMP (*Simple Network Management Protocol)*, que teve sua especificação inicial feita pelo IETF no final da década de 80 [CAS88]. O SNMP vem sendo constantemente atualizado e encontra-se na sua terceira versão (SNMPv3) [CAS02]. Na arquitetura de gerenciamento com SNMP, os dispositivos da rede normalmente são configurados

individualmente. Este processo despende tempo e torna o sistema propício a erros, principalmente quando o número de dispositivos cresce. Além disto a reconfiguração do sistema é infreqüente e executada manualmente.

A estrutura de gerenciamento baseado em políticas representa uma evolução conceitual em relação ao SNMP, pois a implementação de políticas possibilita que toda a rede seja gerenciada como uma entidade, ao invés do gerenciamento individual dos componentes. A utilização deste conceito possibilita que a rede adapte-se facilmente para atender novos objetivos, estabelecendo um processo adequado para a constante necessidade de atualização dos sistemas.

O modelo de gerenciamento baseado em políticas tem como foco principal a função ou papel (*role*) que cada elemento desempenha na rede e não os detalhes específicos para configuração de cada equipamento. Em alto nível, as políticas são representadas de forma declarativa e expressam os objetivos administrativos. Com isso, além de coordenarem as funcionalidades e configuração dos dispositivos, definem também o gerenciamento de usuários e aplicações.

1.3. Proposta

Este trabalho propõe uma arquitetura de Gerenciamento de Rede Baseado em Políticas (PBNM) para automatizar os processos de geração e distribuição de configuração para os dispositivos em um ambiente DiffServ. A arquitetura é baseada nos padrões do IETF e introduz um novo modelo de política de alto nível para simplificar o processo de descrição das políticas de QoS.

A arquitetura é definida em três camadas: modelo de política de alto nível (que estende o IETF PCIM/PCIMe), modelo de política de configuração independente de dispositivo (que estende o IETF QPIM) e um modelo de política dependente de dispositivo (baseado na estrutura IETF DiffServ PIB). As políticas de alto nível, inseridas no sistema pelo administrador, são convertidas para políticas de configuração através da execução de um processo de tradução. As políticas de configuração são armazenadas no repositório que é acessado pelo servidor de políticas (PDP) durante o processo de decisão, o qual é executado para responder à solicitação de configuração de um dispositivo de rede (PEP). O processo de decisão leva em consideração a função desempenhada pelas interfaces do dispositivo e suas capacidades para selecionar as políticas de configuração e convertê-las em instâncias das tabelas da DiffServ PIB. Para

comunicação entre PEP e PDP é implementado o protocolo COPS e sua extensão COPS-PR.

O modelo proposto para descrição das políticas de alto nível suporta a representação de especificações de níveis de serviço (SLSs) dependentes de horário, isto é, um SLS pode não ser permanentemente válido e, portanto, deve ser praticado apenas durante determinados intervalos de tempo. Com isso um importante aspecto a ser tratado é a atualização dinâmica das configurações de acordo com o período de validade das regras definido pelo administrador. No trabalho, são discutidas três diferentes estratégias para o servidor de políticas gerenciar a grade horária de configuração dos dispositivos de rede. Para comparação e avaliação destas estratégias, os principais aspectos verificados foram: o tráfego resultante das mensagens COPS-PR utilizadas para atualização das configurações dos dispositivos; o tempo médio para atualização das configurações dos dispositivos; e a escalabilidade do PDP, ou seja, o impacto resultante do processamento no servidor de políticas para atualização simultânea da configuração de vários dispositivos.

O conceito de reuso de informações é aplicado em todas as camadas da arquitetura, considerando dois contextos diferentes: reaproveitamento dos objetos CIM normalmente cadastrados nos sistemas e utilização de contêineres que armazenam informações reutilizáveis de políticas. Para implementação da proposta, os modelos são mapeados em esquemas XML e o reuso de informação é possível a partir de referências *XPointer* [W3C01].

1.4. Organização

O conteúdo desta dissertação está distribuído em oito capítulos, organizados da seguinte forma:

- Capítulo 1 – Introdução: apresenta o contexto geral do trabalho.
- Capítulo 2 – QoS em Redes IP: descreve os principais conceitos envolvidos nas metodologias de Serviços Integrados (IntServ) e Serviços Diferenciados (DiffServ).
- Capítulo 3 – Padrões IETF para Gerenciamento de Redes Baseado em Políticas (PBNM): descreve os modelos definidos para representação de políticas, os elementos centrais para implementação PBNM e o protocolo para transporte de informações de políticas.

- Capítulo 4 – Trabalhos Relacionados com Políticas de Negócio para QoS: apresenta e discute alguns trabalhos que também propõem modelos de alto nível para representação de políticas de QoS.
- Capítulo 5 – Arquitetura Proposta: detalha a arquitetura de gerenciamento de QoS proposta neste trabalho.
- Capítulo 6 – Implementação dos Modelos de Política Propostos e Processo de Tradução: apresenta o mapeamento dos modelos propostos em XML, define os contêineres de políticas reutilizáveis e descreve a conversão das informações das políticas de alto nível para as políticas de configuração.
- Capítulo 7 – Processo de Decisão e Distribuição das Políticas para os Dispositivos de Rede: descreve o processo de decisão executado pelo servidor de políticas (PDP), incluindo a seleção das políticas relevantes para um determinado dispositivo, a agenda das configurações, a conversão das políticas em instâncias da *DiffServ* PIB e a distribuição das instâncias da *DiffServ* PIB para os dispositivos utilizando o protocolo COPS/COPS-PR. O capítulo também discute três estratégias distintas para atualização dinâmica da configuração dos dispositivos.
- Capítulo 8 – Estudo de Caso e Avaliação da Proposta: demonstra a utilização da ferramenta de gerenciamento de QoS proposta em diferentes cenários e apresenta os tempos médios para atualização da configuração de um roteador Linux, considerando as três estratégias distintas. Os experimentos levam em consideração dois principais fatores: quantidade de dispositivos sendo atualizados simultaneamente e o efeito da restrição da largura de manda para o tráfego das mensagens COPS/COPS-PR
- Capítulo 9 – Conclusões e Trabalhos Futuros: descreve as conclusões obtidas com a realização deste trabalho e indica os principais aspectos para melhoramento da arquitetura.

Capítulo 2

Qualidade de Serviço em Redes IP

2.1 Introdução

No modelo do melhor esforço (*best-effort*) utilizado pelo protocolo IP (*Internet Protocol*), o encaminhamento dos pacotes na rede é realizado de forma equivalente, portanto não permite a priorização e a diferenciação dos tráfegos de diferentes usuários e aplicações. Considerando a atual necessidade das redes IP transportarem simultaneamente dados, voz e vídeo, o modelo do melhor esforço torna-se ineficaz e inadequado. Para que os requisitos das aplicações multimídia sejam atendidos e diferentes níveis de serviço possam ser definidos na rede, é necessário que mecanismos adicionas para controle de QoS sejam implementados.

Com o intuito de propor mecanismos de controle de QoS em redes IP, surgiram diferentes grupos de trabalho dentro do IETF (*Internet Engineering Task Force)*. As pesquisas dos grupos do IETF resultaram, fundamentalmente, em duas abordagens distintas: os Serviços Integrados (*IntServ*) [BRA94] e os Serviços Diferenciados (*DiffServ*) [BLA98].

2.2. Serviços Integrados (*IntServ*)

O intuito do *IntServ* é fornecer algo mais próximo possível da emulação de circuitos dedicados em redes IP. Esta proposta introduz uma grande mudança de paradigma em relação ao serviço de melhor esforço. A tentativa da arquitetura *IntServ* é prover o mais alto nível de QoS em termos de garantias de serviço, granularidade de alocação de recursos e detalhes de resposta. Para que estes objetivos possam ser alcançados, um protocolo de sinalização chamado RSVP (ReSource ReserVation Protocol) [BRA97] é utilizado para reserva de recursos. O RSVP é um mecanismo

complexo de QoS e atua tanto na camada de sistema como nos elementos de rede (roteadores). O *IntServ* é aplicado em fluxos individuais, utilizando o RSVP para reserva dos recursos necessários de fim-a-fim. O RSVP atende ambas sessões: *unicast* e *multicast*.

Uma reserva que utiliza o protocolo RSVP é caracterizada pela descrição do fluxo (*Flowspec*)[RFC 2210]. Esta descrição é composta por dois elementos:

- Especificação da reserva (*Rspec*) – indica a classe de serviço desejada;
- Especificação do tráfego (*Tspec*) – indica as características do que será transmitido;

  A definição dos parâmetros *Flowspec*, geralmente é baseada no modelo *Token Bucket*.

A partir do modelo *Token Bucke*t mostrado na figura 2.1, a especificação do tráfego (*Tspec*)[SHE97] é composta pelos seguintes parâmetros:

- *Token rate* (r) – taxa média em bytes/s;
- *Bucket depth* (b) – tamanho da rajada (*burst*) sustentável, em bytes;
- *Peak rate* (p) – taxa de pico;
- *Minimum policed size* (m) – pacotes menores que m são considerados com o tamanho de m bytes;
- *Maximum packet size* (M) – tamanho máximo de pacote (em bytes) que pode ser enviado pelo fluxo.

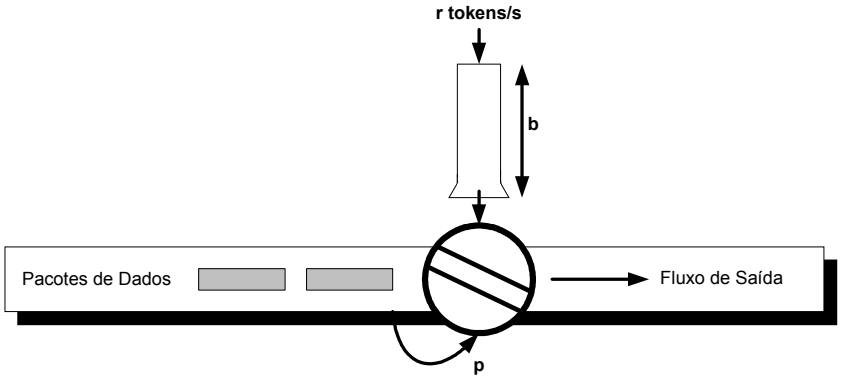

Figura 2.1: Modelo *Tocken Bucket*

A especificação de reserva (Rspec) também utiliza o modelo *Token Bucket* e é composta pelos seguintes parâmetros:

- *Rate* (R) – taxa média solicitada;

- *Slack term* (S) – saldo de retardo, é o valor de atraso que pode ser utilizado pelos nós intermediários. Corresponde a diferença entre o atraso garantido se a banda R for reservada e o atraso realmente necessário, especificado pela aplicação.

O procedimento de reserva de recursos através do protocolo RSVP é sempre unidirecional, conforme mostrado na figura 2.2. Assim, para uma mesma aplicação transmitir e receber dados com tratamento de QoS são necessárias duas reservas e cada uma delas utiliza os seguintes passos:

1) A fonte (servidor) do fluxo determina a largura de banda, atraso e *jitter* do fluxo, através do objeto *SENDER_TSPEC* (utilizando os parâmetros Tspec citados anteriormente) e envia esta descrição do fluxo para o receptor através de uma mensagem *PATH*.
2) Quando um roteador recebe a mensagem *PATH*, ele armazena a informação do estado do caminho "*path-state*" e guarda o endereço do nó anterior, a partir do qual a mensagem *PATH* foi enviada. Em cada nó do caminho, o objeto *ADSPEC*, que representa as informações do caminho, é atualizado. Através do objeto *ADSPEC* o cliente tem as informações de atrasos no caminho.
3) Quando o destino final (cliente) recebe a mensagem *PATH*, ele solicita a reserva de recursos que lhe é conveniente, enviando uma mensagem *RESV* no sentido contrário ao caminho estabelecido pela mensagem *PATH*. A mensagem *RESV* contém a informação dos recursos necessários (Flowspec) e identifica os pacotes que devem utilizar a reserva (Filterspec).
4) Um roteador ao receber a mensagem *RESV* possivelmente autentica a mensagem e executa o controle de admissão para verificar a disponibilidade dos recursos requisitados. Se houver a disponibilidade os recursos são alocados (reserva de banda e espaço em *buffer*) e a mensagem *RESV* é encaminhada para o próximo roteador. Caso contrário, o roteador rejeita a reserva e envia uma mensagem de erro ao receptor (cliente).
5) Se a fonte (servidor) do fluxo recebe a mensagem *RESV*, então a reserva dos recursos fim-a-fim foi feita e os pacotes começam a ser transmitidos.
6) Quando a reserva não for mais utilizada (fim da transmissão do fluxo), faz-se necessária uma notificação explícita para que os roteadores liberem os recursos reservados. Outro método de aplicação do RSVP utiliza um modo chamado "*soft-state*", no qual a reserva de recursos é associada a um limite de tempo

(*timeout*), exigindo que a reserva (mensagem *PATH + RESV*) seja executada periodicamente.

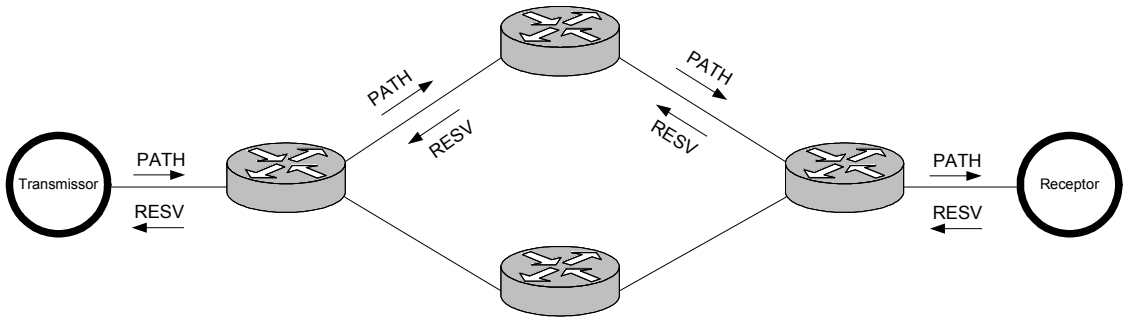

Figura 2.2: Mecanismo de Reserva com Protocolo RSVP

A metodologia *IntServ* é dividida em dois tipos de serviços: *Guaranteed Service*[GUE97] e *Contolled Load Service*[WRO97].

No Serviço Garantido são estabelecidos circuitos virtuais através da utilização de protocolos de reserva (o mais utilizado é o RSVP conforme descrito acima), desta forma este é o serviço que estabelece o maior nível de garantia no cumprimento dos parâmetros de QoS requisitados por um determinado fluxo. Para o Serviço Garantido, ambos os parâmetros Tspec e Rspec são utilizados para descrição do fluxo (*Flowspec*).

O Serviço de Carga Controlada funciona em um modo mais flexível, que ao invés de atender rígidas exigências, apenas fornece um nível de serviço equivalente ao do melhor esforço em uma rede descongestionada (carga controlada), sendo suficiente para aplicações adaptativas de áudio e vídeo (não-interativos) que utilizam mecanismos de compensação (técnicas de *bufferização*, diminuição da taxa de *frames*,etc.) para atrasos e variações de atraso na entrega de pacotes (*jitter*). No Serviço de Carga Controlada, a banda necessária é deduzida a partir dos parâmetros Tspec, não sendo utilizada a informação Rspec.

Para implementação dos Serviços Integrados é necessário que todos os roteadores envolvidos no encaminhamento dos pacotes suportem o protocolo RSVP. Além disso, os roteadores precisam manter a informação de estado por fluxo para implementação dos mecanismos de *scheduling* e gerenciamento de *buffers*. Devido a estas características o modelo de Serviços Integrados é pouco escalável, o que dificulta sua utilização em rotas com elevado número de fluxos (*backbones*).

2.3. Serviços Diferenciados (*DiffServ*)

O principal objetivo dos Serviços Diferenciados é estabelecer classes de tráfego que receberão tratamento diferenciado em redes IP. O tratamento diferenciado é determinado por contratos de serviços (*Service Level Agreements* - SLAs) entre clientes e seus provedores, nos quais são especificadas as priorizações de tráfego e, possivelmente, diferenciação no custo do serviço (*accouting*).

O *DiffServ* opera sobre a agregação e classificação de fluxos, com isso representam uma alternativa escalável ao modelo dos Serviços Integrados. Para realizar a classificação dos fluxos, o *DiffServ* definiu uma reutilização para o campo TOS (8 bits -*Type-of-Service*) do cabeçalho IPv4 e para o campo *Traffic Class* do IPv6 (também de 8 bits). Os campos TOS (IPv4) e *Traffic Class* (IPv6) foram renomeados como *byte* DS ou *DSField* [RFC 2474], mostrado na figura 2.3. O *byte* DS é dividido da seguinte maneira: DSCP (*Differentiated Service CodePoint*) – 6 primeiros bits que indicam a classe de tráfego para o pacote e os 2 últimos bits que, originalmente eram reservados, atualmente representam o campo ECN (Notificação Explícita de Congestionamento) [BLA01].

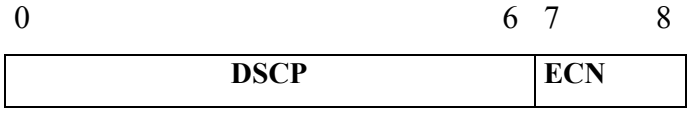

Figura 2.3: DS *Field* para IPv4 (alteração do TOS) e IPv6 (*Traffic Class*)

A marcação DSCP é adicionada ao pacote através do sistema operacional do *host* de origem ou no primeiro roteador (roteador de ingresso do domínio DS) pelo qual o pacote passa. Desta forma, o *DiffServ* é transparente para o aplicativo.

A partir da classificação DSCP dos pacotes, cada nó (roteador) dentro da rede de serviços diferenciados encaminha os pacotes de acordo com o mapeamento do campo DSCP para um comportamento específico de QoS. Por exemplo, se DSCP=X, use *token-bucket* r, b. A forma como a marcação DSPC é mapeada para especificações de QoS em cada nó (roteador) é denominada PHB (*Per-Hop Behavior*). O controle das políticas do mapeamento DSPC em PHB é uma função administrativa e normalmente está associada a um contrato de serviços (SLA).

Para que os níveis de QoS contratados sejam mantidos no tráfego entre Domínios DS (um Domínio DS normalmente é caracterizado como uma ou mais redes

de Serviços Diferenciados sob mesma administração) são utilizados os SLAs entre domínios e os componentes (roteador, *host*, *firewall*) de fronteira executam os mecanismos de condicionamento e policiamento de tráfego e efetuam as remarcações DSCP, se necessário, garantindo que o contrato seja obedecido.

A figura abaixo exibe os elementos que compõem a arquitetura de um nó *DiffServ*.

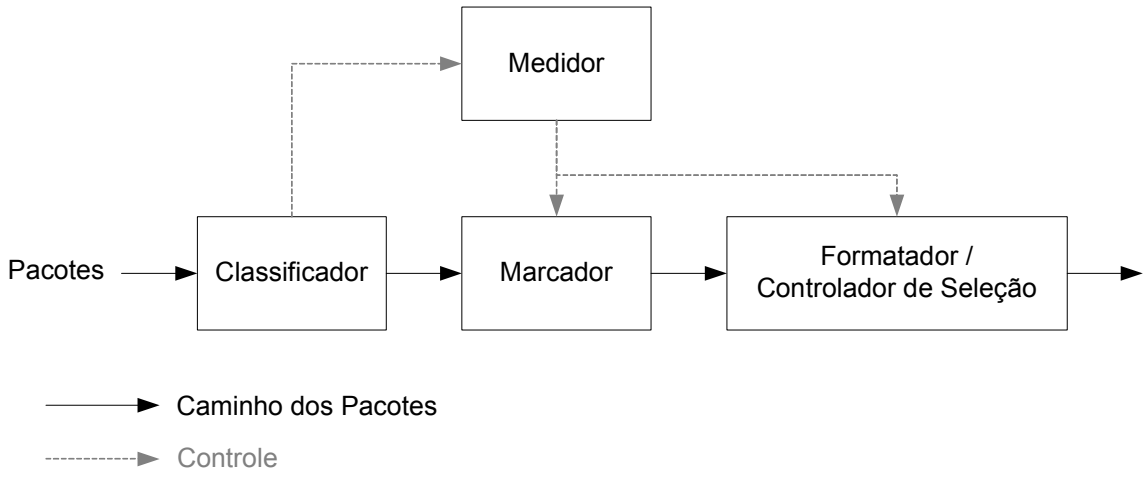

Figura 2.4: Arquitetura de um nó DS

Classificador – seleciona os pacotes através dos cabeçalhos e encaminha aqueles que correspondem às regras de classificação para processamento posterior. O modelo DiffServ especifica dois tipos de classificadores:

- Multicampos (MF): além do byte DS pode utilizar outras informações do cabeçalho IP, como endereço, porta, etc.
- Comportamento Agregado (BA): a classificação é baseada somente no byte DS.

Medidor – verifica se o encaminhamento dos pacotes que estão selecionados pelo classificador está de acordo com o perfil do tráfego especificado no contrato (SLA) estabelecido entre o cliente e o provedor de serviços e passa informações de estado para os outros elementos do nó.

Marcador – responsável pela marcação ou remarcação do byte DS dos pacotes. A marcação é feita nos pacotes emitidos sem marcação pelo cliente e a remarcação é executada quando o nó seguinte no encaminhamento possuir uma interpretação diferente para o byte DS, ou quando o tráfego exceder um certo nível pré-definido.

Formador / Controlador de Seleção – é responsável pela adaptação e descarte dos pacotes de acordo com as propriedades estabelecidas pelo PHB.

Atualmente, estão definidos dois tipos de serviços na metodologia *DiffServ*: *Expedited Forwarding PHB* [JAC99] e *Assured Forwarding PHB* [HEI99].

O EF PHB também é conhecido como *Premium Service* e é similar ao Serviço Garantido da arquitetura *IntServ*, ou seja, simula um enlace dedicado com banda fixa entre dois *hosts*, estabelecendo rígidos limites de atrasos e praticamente sem perdas de pacotes. Devido a estas características o EF PHB representa o mais alto nível de priorização de tráfego dentro do *DiffServ*, ou seja, o encaminhamento dos pacotes marcados com *Premium Service* é executado com taxa, no mínimo, igual à especificada pelo contrato deste serviço, independentemente da intensidade de qualquer outro tráfego tentando transitar pelo nó. Para limitar o dano que o tráfego EF PHB pode causar a outros fluxos, deve-se determinar parâmetros como taxa máxima e tamanho de rajadas, sendo que o tráfego que exceder estes parâmetros será descartado.

A principal diferença do EF PHB (*DiffServ*) em relação ao Serviço Garantido (*IntServ*) é que ele opera sobre fluxos agregados ao invés de manter o gerenciamento por fluxo. Além disso, o EF PHB não realiza reserva de recursos e também não mantém o estado para o caminho dos pacotes. A única informação verificada no encaminhamento *DiffServ* é a marcação DSCP do pacote. O DSCP recomendado [JAC99] para o EF PHB é 101110.

No modelo AF PHB são determinadas quatro classes de encaminhamento e para cada classe está alocada uma certa quantidade de recursos (espaço de *buffer* e largura de banda). Cada uma destas classes AF é dividida em três níveis de preferência de descarte. Em caso de congestionamento, o nível de descarte estabelece a importância relativa do pacote dentro da sua classe AF. Um nó *DiffServ* congestionado tenta evitar a perda de um pacote com menor nível de preferência de descarte, descartando primeiramente os pacotes com maiores níveis de preferência de descarte. Assim, a garantia de encaminhamento de um pacote IP dentro do modelo AF PHB, depende de três fatores:

1) Quantidade de recurso para encaminhamento alocado para a classe AF a qual o pacote pertence.
2) Qual é a carga momentânea da classe AF.
3) Em caso de congestionamento, qual é o nível de preferência de descarte do pacote.

Através destes mecanismos, o AF PHB oferece uma diferenciação relativa de serviços, por exemplo, usuários que assinam um perfil de banda maior, recebem uma qualidade de serviço superior ao dos assinantes com perfis de banda menores.

Na tabela 2.1, são descritos os DSCPs das classes AF:

Tabela 2.1: Valores DSCP para classes AF

	Prioridades de encaminhamento (Classe 1 tem a maior prioridade e a Classe 4 a menor)			
	Classe 1	Classe 2	Classe 3	Classe 4
Baixo nível de descarte	001010 (AF11)	010010 (AF21)	011010 (AF31)	100010 (AF41)
Médio nível de descarte	001100 (AF12)	010100 (AF22)	011100 (AF32)	100100 (AF42)
Alto nível de descarte	001110 (AF13)	010110 (AF23)	011110 (AF33)	100110 (AF43)

2.4. Conclusão

Devido às necessidades dos serviços de rede que requerem o cumprimento de determinados parâmetros de QoS, grupos de trabalho do IETF direcionaram suas pesquisas em busca de mecanismos para controlar o encaminhamento dos pacotes nas redes IP e , fundamentalmente, duas metodologias distintas foram propostas: *IntServ* e *DiffServ*. A principal diferença entre as duas metodologias é que o *IntServ* requer a implementação de um protocolo de sinalização, enquanto que o *DiffServ* utiliza a marcação dos pacotes IP. Além disto, o *IntServ* controla cada fluxo individualmente e o *DiffServ* utiliza o conceito de classes agregadas de fluxos.

Capítulo 3

Padrões IETF para Gerenciamento de Redes Baseado em Políticas (PBNM)

3.1. Introdução

Como a arquitetura proposta por este trabalho é baseada nos padrões do IETF (*Internet Engineering Task Force*), neste capítulo são apresentados os conceitos, modelos e protocolos definidos pelo IETF para o gerenciamento baseado em políticas.

3.2. Política

Os trabalhos científicos relacionados a gerenciamento de rede baseado em políticas (PBNM) não apresentam uma notação comum utilizada como definição formal para política. Deste modo, o que se encontra são definições variadas que não representam conceitos divergentes, ao invés disso expressam de forma mais adequada o contexto de política dentro de cada proposta.

A [RFC 3198] apresenta duas formas para definição de políticas:

- Um objetivo definido ou método de ação que determinam o modo como as decisões presentes e futuras são tomadas. Políticas são implementadas ou executadas dentro de um contexto particular.
- Políticas são um conjunto de regras para administrar, gerenciar e controlar o acesso dos recursos da rede.[RFC 3060]

Em [DAM01], política é definida como um conjunto de regras que expressam e executam o comportamento desejado dentro de um sistema.

No contexto da arquitetura proposta por este trabalho as políticas são um conjunto de regras que relacionam as informações dos usuários, servidores e aplicações

disponíveis no sistema para definir o tratamento de QoS recebido pelos vários fluxos de tráfego dentro do ambiente gerenciado.

3.3. O Uso de "Papéis" em PBNM

Na [RFC 3060] um papel (*role*) é definido como um atributo utilizado para selecionar, dentro do conjunto das políticas disponíveis no sistema, a(s) política(s) que será aplicada a um elemento gerenciado em particular. O uso do conceito de papel é fundamental para implementação do gerenciamento baseado em políticas. Em PBNM, o administrador define o papel ou conjunto de papéis que cada elemento (por exemplo, interface) desempenha na rede e determina as políticas de acordo com estes papéis, ao invés de configurar independentemente cada um dos dispositivos.

Como cenário de uso para ilustrar a aplicação de papéis, pode-se considerar uma arquitetura de rede *DiffServ* [RFC 2475] sendo gerenciada através de políticas. Basicamente, o administrador pode definir as interfaces dos roteadores que atuam como "borda" (*edge*) e as interfaces que representam a rede central (*core*) e, então, descrever apenas dois conjuntos (um para o papel "borda" e outro para o papel "rede central") de políticas para expressar o efeito desejado em todo o ambiente gerenciado.

3.4. Arquitetura para Implementação de PBNM

Os dois principais componentes de um sistema baseado em políticas [GUE00] [WES01] são o PDP (*Policy Decision Point*) e o PEP (*Policy Enforcement Point*).

- PDP é a entidade lógica que reside no servidor de políticas e é o responsável por tomar as decisões das políticas a serem e executadas pelos nós da rede.
- PEP é a entidade que reside em um nó de rede e atua como cliente do PDP. A responsabilidade do PEP é realizar os procedimentos necessários para que as decisões enviadas pelo PDP sejam cumpridas.

Normalmente, o servidor de políticas está conectado a um repositório onde são mantidas as regras, suas condições e ações, e demais dados relacionados a políticas. Durante o processo de decisão, o PDP consulta estas informações. Uma base de dados ou um serviço de diretórios seriam exemplos de implementação para repositórios de política. Além disto, o PDP pode conectar-se a outros servidores e bancos de dados para implementar funcionalidades adicionais como autenticação de usuário e contabilidade. Através de uma ferramenta de gerenciamento (*Policy Management Tool* – PMT), as

informações de políticas são inseridas pelo administrador no repositório. Como não é necessário um formato padronizado para garantir a interoperabilidade da ferramenta de gerenciamento, a estrutura da PMT não é definida pelo IETF. Assim, a complexidade de uma ferramenta varia de acordo com sua capacidade de validar, resolver conflitos e traduzir níveis de abstração antes de armazenar as políticas no repositório.

Para comunicação entre PDP e PEP é utilizado o protocolo COPS (*Common Open Policy Service*), que está descrito na seção 3.8.

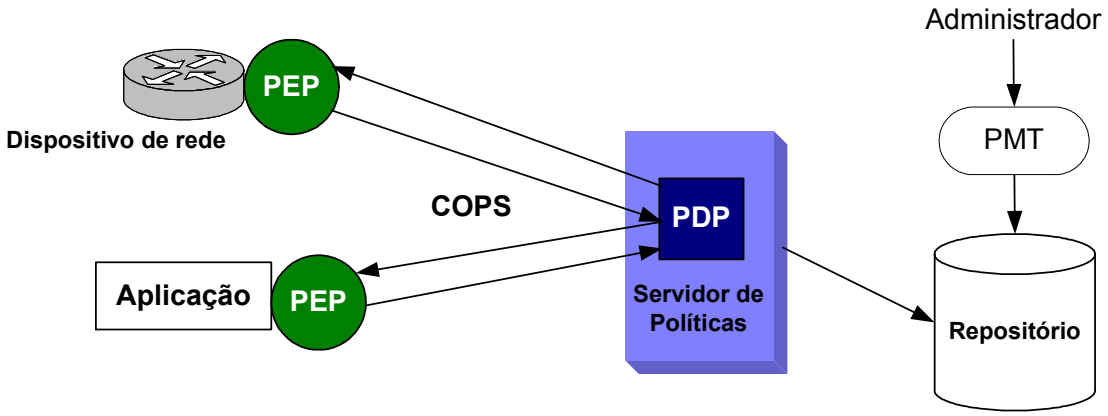

Figura 3.1: Elementos para implementação PBNM

3.5. PCIM

A especificação *Policy Core Information Model* (PCIM) [MOO01] é o resultado da junção de forças do IETF com o DMTF. O PCIM é uma extensão do *Common Information Model* (CIM) desenvolvido pelo DMTF e descreve o modelo de informação orientado a objetos para representação das informações de políticas. Seguindo o conceito de modelo de informação [WES01], o PCIM é independente de implementação específica de repositório, linguagem de programação, plataforma ou fabricante. A estrutura do modelo é formada por dois tipos de classes de objetos:

1) Classes estruturais, que definem a forma de representar e controlar a informação de política;
2) Classes associativas, que indicam a forma como as instâncias das classes estruturais estão relacionadas.

As classes PCIM são genéricas o suficiente para representar qualquer tipo de política, portanto a implementação concreta deste modelo requer que suas classes sejam especializadas de acordo com as áreas específicas de aplicação, tais como segurança e QoS. A [MOO01] sugere que as especializações necessárias sejam feitas diretamente

nas classes *PolicyGroup*, *PolicyRule* e *PolicyTimePeriodCondition*. A figura 3.2 mostra as classes centrais, utilizadas para representação de uma política dentro do modelo PCIM.

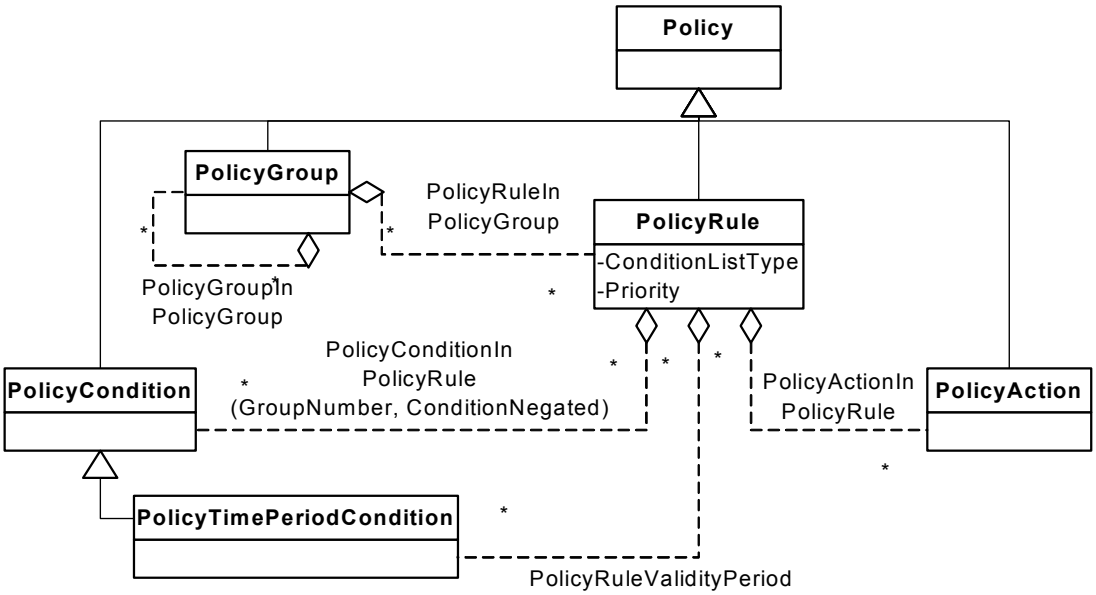

Figura 3.2: Principais classes do modelo PCIM

Através das classes modeladas no PCIM, uma política (classe *Policy*) é definida por um conjunto de regras (classe *PolicyRule*) que são agrupadas de forma coerente utilizando-se a classe *PolicyGroup*. Cada regra é composta por um conjunto de condições (classe *PolicyCondition*) e um conjunto de ações (classe *PolicyAction*), que estabelecem a semântica se <condição> então <ação>. Desta maneira, se o conjunto de condições de uma regra for satisfeito, então as ações relacionadas a esta regra serão executadas. As regras podem também conter condições de período de tempo, as quais são representadas pela classe *PolicyTimePeriodCondition*. A estrutura das classes permite o aninhamento de grupos (*PolicyGroupInPolicyGroup*) para formar uma hierarquia de políticas.

A prioridade de uma regra em relação a outras, dentro de um mesmo grupo, é determinada pelo atributo *PolicyRule.Priority*. O uso deste atributo fornece um mecanismo básico para resolução de conflitos entre políticas e possibilita a implementação de regras que possuem uma condição genérica e algumas exceções específicas. Um exemplo de política com regras conflitantes seria:

- Regra1: O tráfego originado por todos os funcionários do Departamento Comercial recebe a classe de serviço *"Bronze"*.
- Regra2: O tráfego originado pelo Diretor do Departamento Comercial recebe a classe de serviço *"*Ouro*"*.

Se o administrador estabelecer maior prioridade para a exceção (Regra2) em relação ao caso genérico (Regra1), o efeito desejado é obtido de forma consistente.

As condições associadas a uma regra de política podem ser agrupadas de duas maneiras:

1) DNF (*Disjunctive Normal Form*) – as condições que pertencem ao mesmo grupo são representadas através da operação lógica "E" e a união dos diferentes grupos de condições é realizada através da operação lógica "OU".
2) CNF (*Conjunctive Normal Form*) – as condições que pertencem ao mesmo grupo são representadas através da operação lógica "OU" e a união dos diferentes grupos de condições é realizada através da operação lógica "E".

A escolha do formato DNF ou CNF é representada pelo atributo *PolicyRule.ConditionListType*. A expressão lógica resultante do conjunto de condições de uma regra também depende de dois atributos da classe associativa *PolicyConditionInPolicyRule*: *GroupNumber* e *ConditionNegated*. Para demonstrar a modelagem do agrupamento das condições de uma regra, pode-se considerar a existência de um conjunto (C) com quatro condições (*PolicyConditions*) associadas a mesma regra (R) e representadas por C_i(*GroupNumber, ConditionNegated*): C={C_1(1,*false*), C_2(2,*true*), C_3(1,*true*), C_4(2,*false*)}. Então a expressão lógica resultante para avaliação de R será:

1) Se *ConditionListType* = DNF(1), a avaliação de (C) = (C_1 **E** !C_3) **OU** (!C_2 **E** C_4);
2) Se *ConditionListType* = CNF(2), a avaliação de (C) = (C_1 **OU** !C_3) **E** (!C_2 **OU** C_4);

Com relação às ações associadas a uma regra de política, é possível: determinar se a ordem de execução das ações é relevante; especificar uma ordem; e ainda indicar se esta ordem é obrigatória ou meramente recomendada. Para expressar a relevância na ordem de execução é utilizado o atributo *PolicyRule.SequencedAction*.

A ordem de execução de uma ação (*PolicyAction*) dentro de uma regra é indicada pelo atributo da classe associativa *PolicyActionInPolicyRule.ActionOrder*.

De acordo com a possibilidade de reutilização das informações, as condições e ações são divididas, conceitualmente, em duas categorias:

1) *Rule-Specific* – são condições e ações associadas a uma única regra.
2) *Reusable* – são condições e ações que podem estar associadas a mais de uma regra e, portanto, são armazenadas no contêiner *PolicyRepository* (no PCIMe [MOO03] esta classe é substituída por *ReusablePolicyContainer*).

No modelo PCIM, o conceito de papel é utilizado para seleção de regras através do atributo *PolicyRule.Role*, mas na sua extensão PCIMe, que será apresentada na próxima seção, o atributo "*Role"* é deslocado para uma classe mais genérica, possibilitando a seleção de um grupo inteiro de políticas.

3.6. PCIMe

A [MOO03] descreve o *Policy Informatio Model extensions* (PCIMe). O PCIMe estende o modelo PCIM de duas maneiras: novos elementos são introduzidos e alguns elementos são alterados ou substituídos, sendo que em ambos os casos é mantida a compatibilidade com as implementações do modelo original PCIM. O diagrama UML da figura 3.3 representa a estrutura geral do modelo, com as novas classes desenhadas com preenchimento. O detalhamento dos atributos e associações é apresentado em figuras posteriores, seguindo a explicação.

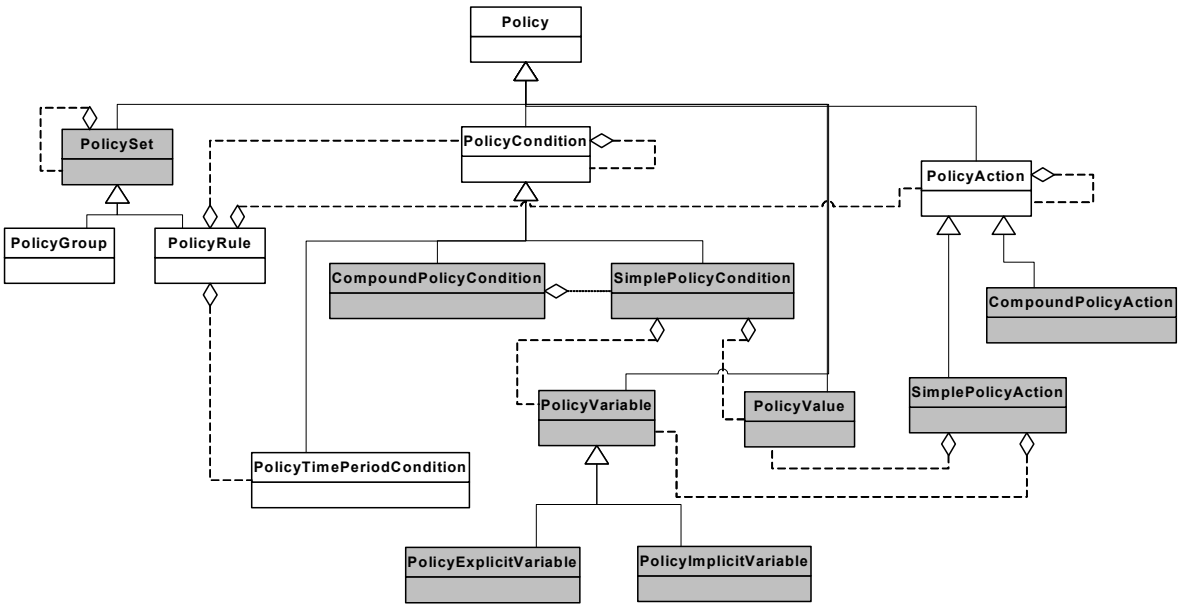

Figura 3.3: Estrutura geral do modelo PCIMe

A classe abstrata *PolicySet* derivada de *Policy* é inserida acima de *PolicyGroup* e *PolicyRule*, resultando em maior flexibilidade estrutural e semântica para ambas as classes. O atributo *PolicySet.PolicyDecisionStrategy* determina o método para avaliação das regras que pertencem a um mesmo conjunto. Duas estratégias são definidas:

1) *FirstMatching* (1) – somente a primeira regra (a de maior prioridade) satisfeita é executada.
2) *AllMatching* (2) – as ações de todas as regras satisfeitas são executadas.

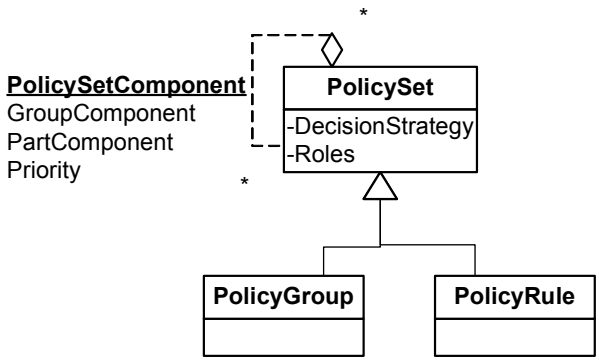

Figura 3.4: Classe PolicySet

A partir da classe *PolicySet* também foi criada a associação *PolicySetComponent*, a qual foi introduzida no modelo para generalizar a representação de quatro agregações: *PolicyRuleInPolicyGroup*, *PolicyGroupInPolicyGroup* (descritas no modelo original PCIM), *PolicyRuleInPolicyRule* e *PolicyGroupInPolicyRule* (adicionadas pelo PCIMe). A classe *PolicySetComponent* facilita a descrição de políticas hierárquicas através do aninhamento de regras, ou seja, representação de uma regra ou conjunto de regras dentro do escopo de outra regra. O atributo "*Priority*" foi deslocado de *PolicyRule* (PCIM) para *PolicySetComponent* no modelo PCIMe. Desta forma, é estabelecida a prioridade relativa entre as regras ou grupos aninhados.

O principal objetivo da criação das classes "compostas" (*CompoundPolicyCondition* e *CompoundPolicyAction*) é facilitar a reutilização das informações de política. As composições de condições ou ações podem ser nomeadas e armazenadas em um repositório (classe R*eusablePolicyConteiner*), permitindo que várias políticas compartilhem (através do uso de referências) as mesmas composições.

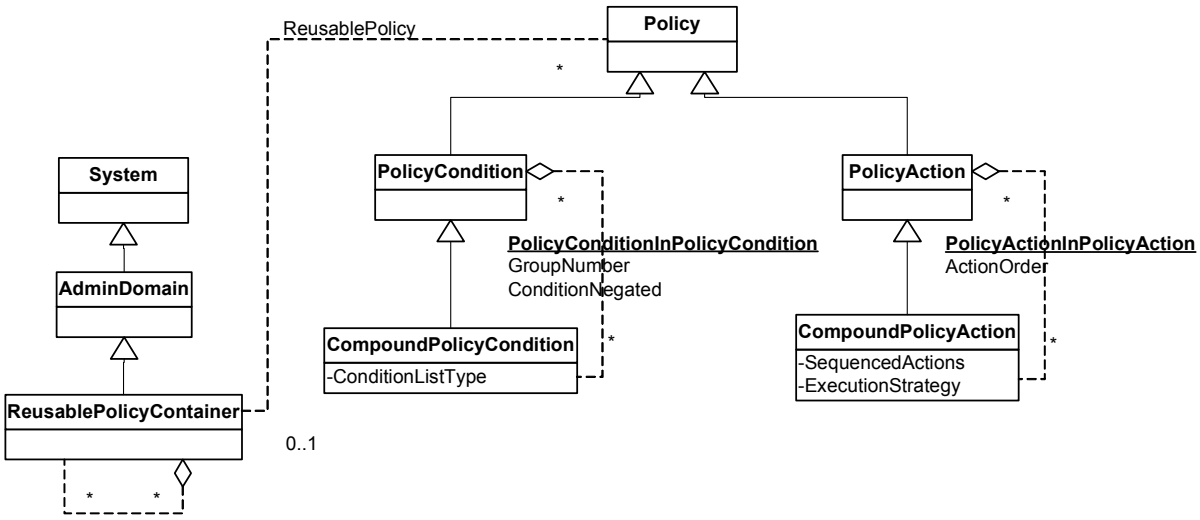

Figura 3.5: Condições e Ações Compostas

As composições de condições possuem o atributo *CompoundPolicyCondition.ConditionListType* para determinar a semântica DNF / CNF aplicada às condições que estão associadas em uma mesma composição. As composições de ações possuem dois atributos: *CompoundPolicyAction.SequencedActions* – que tem significado idêntico ao *PolicyRule.SequencedAction* definido no PCIM e *CompoundPolicyAction.ExecutionStrategy* – que define três estratégias:

1) *Do Until Success* (1) – executar as ações, seguindo a ordem definida, até que a execução de uma ação da composição seja bem sucedida.
2) *Do All* (2) – executar todas as ações que fazem parte da composição, seguindo a ordem definida. Continuar a execução mesmo havendo falha de uma ou mais ações da composição.
3) *Do Until Failure* (3) – executar as ações, seguindo a ordem definida, até que ocorra a primeira falha de execução em uma das ações da composição.

Com o acréscimo das classes "compostas", surgiram também novas classes associativas (*PolicyConditionInPolicyCondition* e *PolicyActionInPolicyAction*). *PolicyConditionInPolicyCondition* utiliza os mesmos atributos (*GroupNumber* e *ConditionNegated*) da classe *PolicyConditionInPolicyRule*. Seguindo o conceito de orientação a objetos, o modo mais adequado para representar classes que compartilham os mesmos objetos é a partir da herança de uma classe mais genérica. No PCIMe foram criadas as classes abstratas *PolicyConditionStructure* e *PolicyActionStructure*.

As estruturas (*SimplePolicyCondition, PolicyVariable, PolicyValue*) e (*SimplePolicyAction, PolicyVariable, PolicyValue*) são utilizadas para representar as condições e ações que são diretamente formadas pela combinação variável / valor.

Nas condições simples a semântica adotada é (VARIÁVEL "está de acordo com" VALOR) e nas ações simples a semântica adotada é (configurar VARIÁVEL com VALOR). O operador "está de acordo com" é implícito no modelo PCIMe e não está formalmente definido. A classe abstrata *PolicyVariable* é especializada por dois tipos de variáveis: explícitas (*PolicyExplicitVariable*) e implícitas (*PolicyImplicitVariable*). As variáveis explícitas fazem referência a objetos modelados pelas classes do CIM, um exemplo de condição simples com variável explícita é "*Person.BusinessCategory* = Engenheiro". O atributo *BusinessCategory* é definido na classe *Person* do CIM, assim *PolicyExplicitVariable.ModelClass* = "*Person*" e *PolicyExplicitVariable.Property = "BusinessCategory"* , neste caso a classe *PolicyValue* que também é abstrata é especializada por *PolicyStringValue.StringList* = "Engenheiro". As variáveis implícitas não referenciam objetos do modelo CIM e, normalmente, indicam informações contidas no cabeçalho de protocolos. Por exemplo, a condição "IP de origem = 10.0.2.2/24" é representada por *PolicySourceIPv4Variable* e *PolicyIPv4AddrValue. IPv4AddrList* = "10.0.2.2/24". A classe *PolicyValue* também possibilita a definição de condições com range ou conjunto de valores. Para uma condição "Porta de origem = [1024 a 65535]" é usado *PolicyIntegerValue.IntegerList = "1024..65535"*.

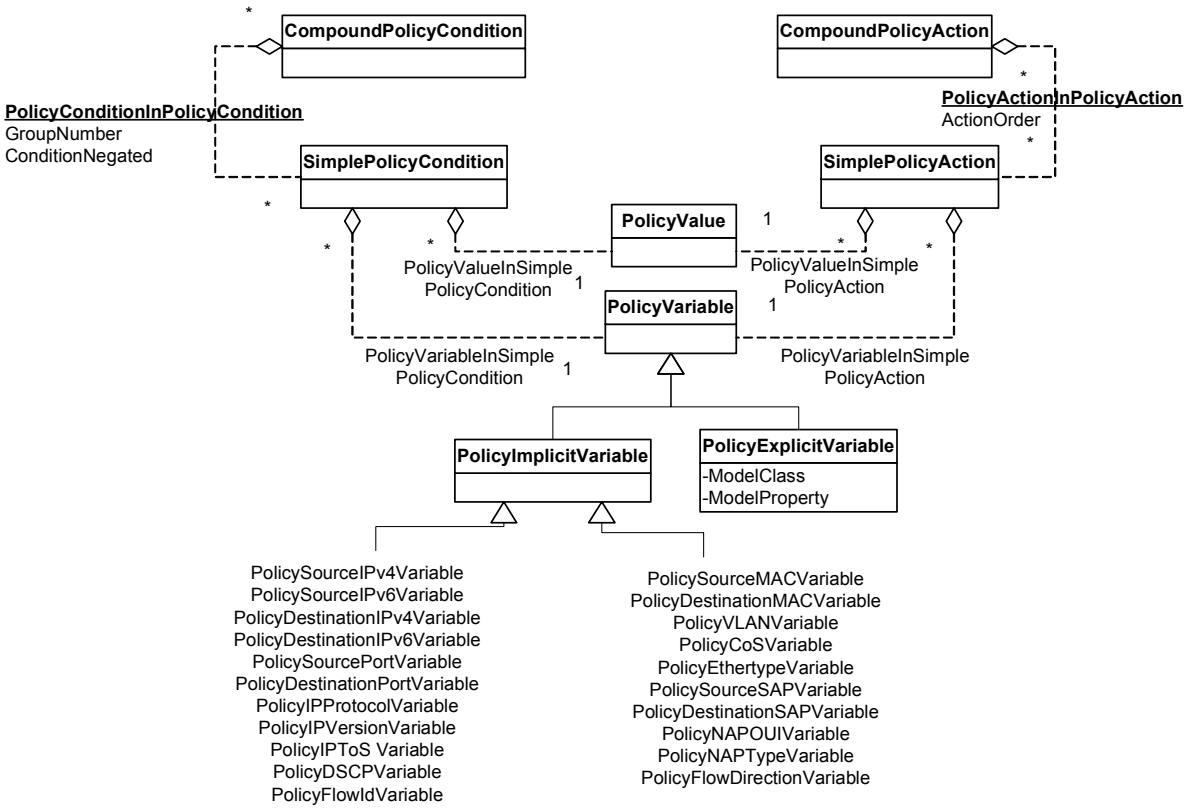

Figura 3.6: Estrutura Condição / Ação Simples, Variável, Valor

A classe *PolicyRoleCollection* foi criada para suprir a falta do PCIM na associação entre os recursos da rede e papéis. Por exemplo, o PCIM possibilita a criação de regras que dizem respeito ao papel "RedeCentral && Camada2", mas não define nenhum mecanismo capaz de relacionar as interfaces dos dispositivos com estes critérios. No PCIMe, *PolicyRoleCollection* representa a coleção de recursos associada a um determinado papel. Assim, as instâncias desta classe definem a ligação entre uma regra ou grupo e um conjunto de recursos. Por isto, valores equivalentes devem ser definidos em *PolicySet.PolicyRoles* e *PolicyRoleCollection.PolicyRole*.

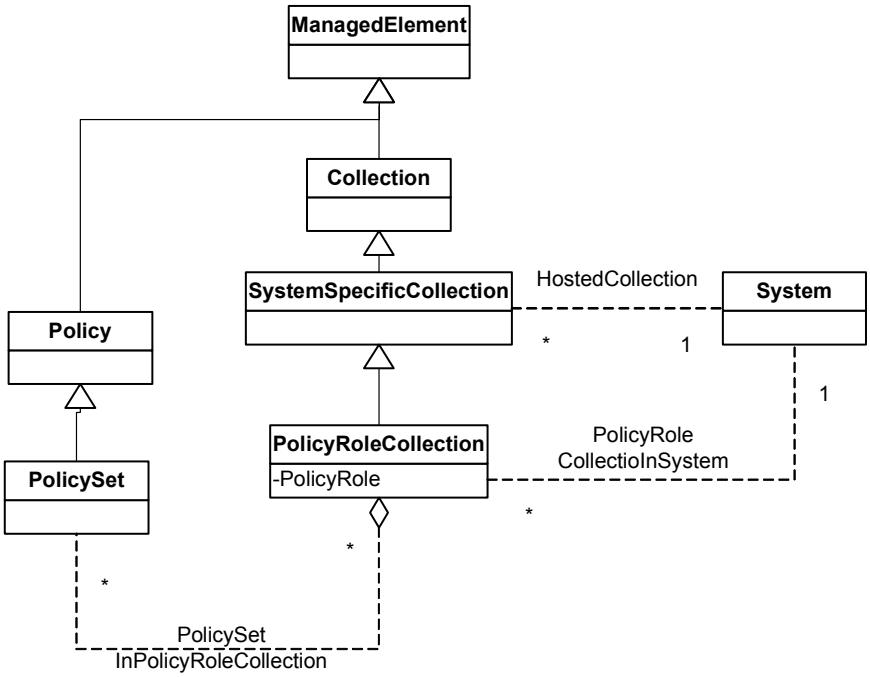

Figura 3.7: Associação de recursos da rede (sistemas) e papéis

Para evitar a possível confusão entre o repositório de políticas (*Policy Repository*) definido na arquitetura para implementação PBNM (fig.3.1) e a classe *PolicyRepository* do modelo PCIM, no PCIMe a classe *PolicyRepository* foi substituída pelo contêiner de políticas reutilizáveis *ReusablePolicyContainer*.

3.7. QPIM

O *Policy Quality of Service Information Model* (QPIM) [SNI03] especializa o modelo genérico PCIM no que diz respeito à representação de políticas para administração, gerenciamento e controle dos recursos de QoS utilizados nos Serviços Integrados e Diferenciados. Na verdade, a RFC 3644 não especifica um modelo completo, apenas as novas classes que descrevem as ações de QoS, são representadas. A RFC sugere que os desenvolvedores devem combinar os elementos QPIM com PCIM / PCIMe para modelar as políticas de QoS.

O processo para definição das políticas de QoS depende de três tipos de informação: regras de negócio, topologia da rede a ser gerenciada e mecanismo de QoS a ser utilizado [SNI03]. Mas as classes QPIM, mostradas na figura 3.8, modelam diretamente apenas as informações relacionadas ao controle dos mecanismos de QoS (*IntServ* e *DiffServ*), estabelecendo uma forma padronizada para representar ações de QoS e perfis de tráfego. A implementação do QPIM auxilia o mapeamento das regras de

negócio em um modo que define os requisitos para condicionamento dos diferentes tipos de tráfego na rede, seguindo a semântica se <condição> então <ação>. Por exemplo: se protocolo == HTTP então mínimo BW = 50%.

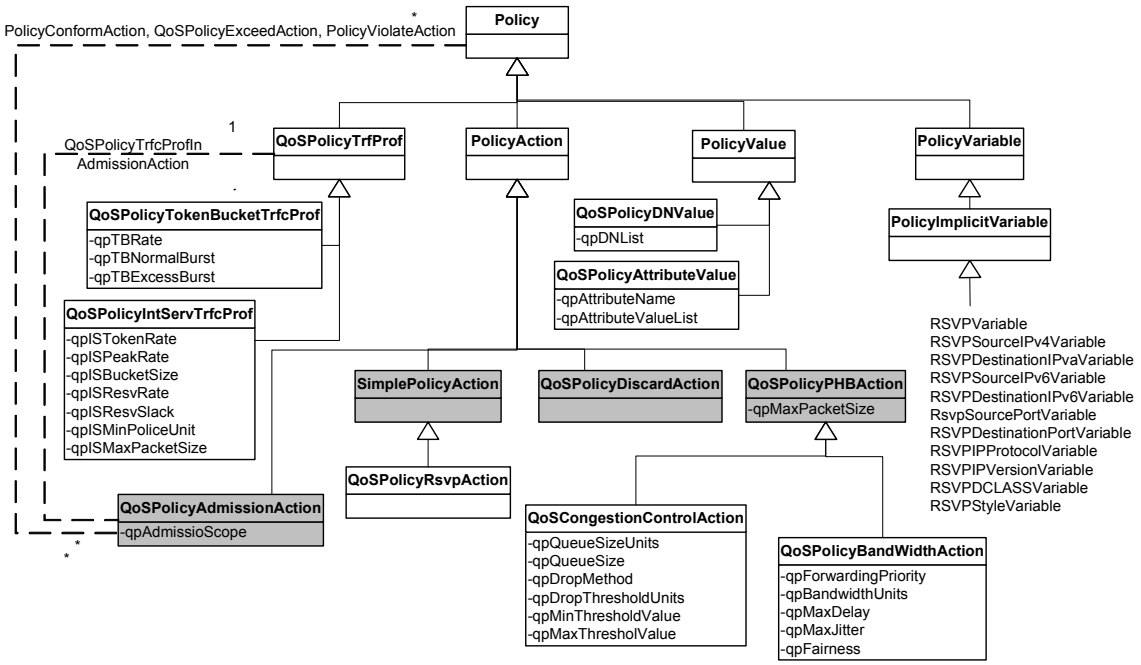

Figura 3.8: Classes QPIM

Como mostrado no diagrama UML da figura 3.8, o QPIM define quatro grupos de ação (classes com preenchimento) para QoS. O primeiro grupo (*SimplePolicyAction*) representa as ações simples que, através da utilização apropriada de variáveis e valores (definidas no PCIMe), modela indiretamente algumas ações de QoS como, por exemplo, as três marcações (DSCP, IPP e CoS). A especialização *QoSPolicyRSVPSimpleAction* descreve o controle do protocolo RSVP. No grupo formado por *QoSPolicyAdmissionAction* e suas subclasses, a associação *QoSPolicyTrfcProfInAdmissionAction* define o perfil de tráfego usado para: a) aceitar ou rejeitar uma requisição RSVP (*QoSPolicyRSVPAdmissionAction)*; b) modelar (atrasar) o encaminhamento dos pacotes (*QoSPolicyShapeAction*); c) classificar os pacotes em conformidade, excesso ou violação (*QoSPolicyPoliceAction*). Em conjunto com *QoSPolicyPoliceAction* são definidas também três associações (*QoSPolicyConformAction, QoSPolicyExceedAction, QosPolicyViolatedAction*), que determinam uma ação específica para cada caso. O atributo *QoSPolicyAdmissionAction.qpAdmissionScope* indica se o escopo da admissão é um

fluxo individual de tráfego ou uma classe agregada de tráfegos. As ações do terceiro grupo (*QoSPolicyPHBAction*) especificam o comportamento de encaminhamento por salto (PHB). As classes *QoSPolicyBandwidthAction* e *QoSPolicyCongestionControl* descrevem a diferenciação de alocação de banda e o controle de congestionamento respectivamente. No último grupo é representando o descarte incondicional de pacotes (*QoSPolicyDiscardAction*).

A classe *QoSPolicyDNValue,* derivada de *PolicyValue,* representa o *Distinguished name* (DN) usado para selecionar um objeto em um serviço de diretório. Embora também derivada de *PolicyValue*, a classe *QoSPolicyAttributeValue* pode ser utilizada para descrever o par variável explícita / valor. Com a definição das estruturas *SimplePolicyCondition* (ou *SimplePolicyAction*) *, PolicyVariable, PolicyValue* no PCIMe, a implementação de *QoSPolicyattributeValue* não faz mais sentido.

O modelo QPIM permite a representação de políticas de configuração independentes de dispositivo. A partir deste conceito, é possível o reuso de configurações de QoS, isto é, políticas de configuração destinadas a vários dispositivos que desempenham função similar na rede podem ser definidas uma única vez no sistema. Fica sob responsabilidade do PDP ou PEPs tratar dos detalhes específicos de configuração de cada dispositivo ou fabricante.

O intuito do QPIM é modelar as políticas para controle de QoS em um modo que reflita, o mais próximo possível, a maneira como os seres humanos tendem a pensar sobre políticas. Para tornar o processo de especificação das políticas mais intuitivo para as pessoas, as ações de QoS podem ser organizadas hierarquicamente, utilizando a associação *PolicySetComponent* (definida no PCIMe) para representar o aninhamento de uma regra ou grupo dentro de outra regra, possibilitando que políticas complexas sejam descritas a partir de um conjunto de regras simples. Na tabela 3.1, está um exemplo de alocação de banda hierárquica.

Tabela 3.1: Alocação de banda hierárquica

SE (Protocolo IP = UDP) ENTÃO (Garantir 30% da BW disponível)	1
SE (Protocolo = TFTP) ENTÃO (Garantir 10% da BW disponível)	1a
SE (Protocolo = NFS) ENTÃO (Garantir 40% da BW disponíviel)	1b
SE (Protocolo IP = TCP) ENTÃO (Garantir 40% da BW disponível)	2
SE (Protocolo = HTTP) ENTÃO (Garantir 20% da BW disponível)	2a
SE (Protocolo = FTP) ENTÃO (Garantir 30% da BW disponíviel)	2b

As regras 1a e 1b estão aninhadas na regra 1, desta forma o efeito resultante é que todas as aplicações UDP irão compartilhar 30% da largura de banda da interface. Além disto, a regra 1 é refinada para garantir que os tráfego TFTP e NFS recebam 10% e 40%, respectivamente, do total de recursos disponíveis para classe UDP, ou seja, 3% e 12% da largura de banda total da interface. Enquanto que outras aplicações UDP não recebem nenhuma garantia. De forma análoga ocorre no caso TCP (regra 2).

3.8. COPS

COPS (*Common Open Policy Service)* é o protocolo padronizado pelo IETF [DUR00] para transportar as mensagens, trocadas entre PEP e PDP, com informações de políticas. Suas principais características são:

- Funciona no modelo requisição / resposta;
- É transportado sobre o protocolo TCP, com isso estabelece uma comunicação confiável entre cliente e servidor;
- As mensagens são enviadas através de objetos independentes e flexíveis, possibilitando a criação de novos objetos;
- Para privacidade das mensagens pode-se utilizar IPSec ou TLS;
- Protocolo *stateful*, ou seja, as requisições do PEP são memorizadas pelo PDP até que sejam explicitamente apagadas pelo PEP. O PDP também "lembra" das decisões já instaladas no PEP e eventos anteriores podem influenciar na resposta de uma nova requisição;

Cada mensagem COPS consiste de um cabeçalho seguido por objetos. A tabela 3.2 descreve as pricipais mensagens do protocolo COPS.

Version	*Flags*	*Op Code*	*Client-type*
Message Length			

Figura 3.9: Cabeçalho COPS

***Version* (4 bits)** – a versão COPS atual é 1.

***Flags* (4 bits)** – apenas o valor 0x1 está definido como *"Solicited Message Flag Bit"*.

***Op Code* (8 bits)** – determina o tipo da mensagem (operação) COPS.

***Client-type* (16 bits)** – Identifica o tipo do cliente.

***Message Length* (32 bits)** – descreve o tamanho da mensagem em bytes, inclui o cabeçalho padrão e todos objetos encapsulados.

Tabela 3.2: Mensagens COPS

1 = *Request* (REQ)	Mensagem enviada do PEP ao PDP para solicitar as políticas correspondentes.
2 = *Decision* (DEC)	Mensagem enviada do PDP ao PEP com as políticas a serem implementadas.
3 = *Report State* (RPT)	Mensagem enviada pelo PEP para comunicar o sucesso ou falha no carregamento das decisões enviadas pelo PDP. Também é utilizada para contabilidade (*accouting)* e envio de relatórios de uso dos recursos do PEP para o PDP.
4 = *Delete Request State* (DRQ)	Mensagem enviada pelo PEP para indicar ao PDP que identificador de estado (*handle*) pode ser desconsiderado.
5 = *Synchronize State Req* (SSQ)	Mensagem enviada pelo PDP para solicitar ao PEP que reenvie seu estado atual.
6 = *Client-Open* (OPN)	Mensagem enviada pelo PEP para solicitar ao PDP a abertura de conexão a ser utilizada pelo tipo de cliente suportado. Deve ser enviada uma mensagem OPN para cada tipo de cliente suportado pelo PEP. E clientes do mesmo tipo

	também podem enviar múltiplas mensagens OPN.
7 = *Client-Accept* (CAT)	Mensagem enviada pelo PDP para responder positivamente a solicitação OPN.
8 = *Client-Close* (CC)	Mensagem que pode ser enviada por ambos (PEP ou PDP) para encerrar uma conexão.
9 = *Keep-Alive* (KA)	Mensagem trocada periodicamente entre PEP e PDP para sinalizar que a conexão está ativa.
10= *Synchronize Complete* (SSC)	Mensagem enviada pelo PEP para indicar ao PDP que a sincronização solicitada através da mensagem (SSQ) foi completada.

Todos os objetos COPS seguem o mesmo formato: 32 bits de cabeçalho e uma ou mais palavras de 32 bits.

Lenght (bytes)	*C-Num*	*C-Type*
Object Contents		

Figura 3.10: Objeto COPS

***Length* (16 bits)** – é o valor que descreve o número de bytes que compõem o objeto. Se o objeto for menor que 32 bits, deve ser adicionado enchimento para que o objeto fique alinhado ao limite de 32 bits.

***C-Num* (8bits)** – identifica a classe da informação contida no objeto.

Tabela 3.3:Valores C-NUM

1 = *Handle*	9 = *Client Specific Info*
2 = *Context*	10 = *Keep-Alive Timer*
3 = *In Interface*	11 = *PEP Identification*
4 = *Out Interface*	12 = *Report Type*
5 = *Reason code*	13 = *PDP Redirect Address*
6 = *Decision*	14 = *Last PDP Address*
7 = *LPDP Decision*	15 = *Accounting Timer*
8 = *Error*	16 = *Message Integrity*

***C-Type* (8 bits)** – depende do *C-Num* e identifica o subtipo ou versão da informação da informação contida no objeto.

A conexão TCP para envio das mensagens COPS é permanente e sempre iniciada pelo PEP, para isto o servidor de políticas (PDP) deve escutar o número de porta TCP determinado pelo IANA (COPS=3288). O processo de inicialização do PEP segue os seguintes passos:

1) O PEP estabelece a conexão com um PDP remoto e envia mensagens *Client-Open* (uma para cada tipo de cliente suportado pelo PEP). A negociação de mecanismos de segurança, quando necessário, ocorre também nesta etapa;
2) O PDP responde com uma mensagem *Client-Accept* em separado para cada cliente requisitado pelo PEP e também suportado pelo PDP. Na mensagem *Client-Accept* é especificado também o intervalo de tempo entre as mensagens *Keep-Alive*;
3) Se algum tipo de cliente requisitado pelo PEP não for suportado pelo PDP, este último responde com uma mensagem *Client-Close,* especificando que o tipo de cliente não é suportado e ainda pode sugerir ao PEP um endereço alternativo de outro PDP;
4) O PEP pode começar a enviar requisições ao PDP;

3.9. COPS-PR

A especificação COPS-PR (COPS *Usage for Policy Provisioning*) [CHA01] define novos objetos destinados a transportar informações para provisão de políticas (também definida como modelo de configuração).

Para identificar as informações de provisão de política, o COPS-PR baseia-se em uma estrutura de dados nomeados, conhecida como PIB - *Policy Information Base*. A estrutura nomeada é comum ao PEP e PDP, e as instâncias de dados dentro desta estrutura são únicas dentro do escopo de um determinado tipo de cliente (*Client-Type*) e estado de requisição (*Request-State*) em cada conexão TCP estabelecida entre um PEP e o PDP. Um módulo PIB é constituído por classes *Provisioning Classes*(PRCs) e suas instâncias *Provisioning Instances* (PRIs). O formato genérico dos objetos COPS-PR (mostrado na figura 3.11) possibilita a identificação de uma determinada instância (PRID) e a representação dos dados correspondentes a ela.

Length (bytes)	*S-Num*	*S-Type*
Object Contents		

Figura 3.11: Objeto COPS-PR

Os campos S-Num e S-Type são similares a C-Num e C-Type dos objetos COPS. A diferença é que estes objetos COPS-PR sempre estão encapsulados nos objetos *Named Client SI* ou *Named Decision Data*, existentes no COPS. *Named Client SI* é utilizado na mensagem *request* (REQ), enviada do PEP para o PDP e transporta as informações auto-descritivas do PEP. Estas informações auxiliam o PDP a selecionar as políticas pertinentes ao cliente e também a determinar quais informações de configuração devem ser enviadas, de acordo com as capacidades do PEP. *Named Decision Data* transporta os dados de configuração enviados do PDP para o PEP na mensagem *decision* (DEC). Uma decisão pode instalar e/ou remover dados nas instâncias (PRIs). No caso da instalação é obrigatória a identificação individual de cada instância (PRID), mas a operação de remoção pode ser otimizada através da identificação do prefixo da PRID (PPRID), desta maneira, todas as instâncias com o mesmo prefixo serão removidas de uma só vez.

***Length* (16 bits)** – é o valor que descreve o número de bytes que compõem o objeto. Se o objeto for menor que 32 bits, deve ser adicionado enchimento para que o objeto fique alinhado ao limite de 32 bits.

***S-Num* (8bits)** – identifica o propósito da informação contida no objeto.

Tabela 3.4 Valores S-NUM

1 = *Complete PRID*	4 = *Global Provisioning Error Object*
2 = *Prefix PRID*	5 = *Class Provisioning Error Object*
3 = *Enconded Provisioning Instance Data*	6 = *Error PRID*

***S-Type* (8bits)** – descreve o tipo de codificação utilizada no objeto. Atualmente, o protocolo define apenas S-Type = 1, que representa a codificação BER (*Basic Encoding Rules* – ITU-T Rec. X.690) e deixa em aberto a possibilidade para definição de outros tipos de codificação no futuro. Os elementos BER são formados pela estrutura TLV (*Type, Length, Value*).

Na arquitetura de gerenciamento proposta, o protocolo COPS / COPS-PR é implementado na comunicação entre PEP e PDP, transportando as informações de configuração *DiffServ* que preenchem as instâncias na *Framework* PIB e *DiffServ* PIB.

3.10. Modelos de Comunicação entre PEP e PDP

Fundamentalmente, são utilizados dois modelos de comunicação entre PEP e PDP: *Outsourcing* e *Provisioning*.

3.10.1. Modelo *Outsourcing*

Na estratégia *outsourcing* as decisões são tomadas em tempo real, ou seja, o PEP, ao receber um evento que requer uma nova decisão de política, monta uma mensagem de requisição (REQ) e a envia ao PDP. Quando o PDP recebe a requisição, ele acessa o repositório de políticas e executa os algoritmos de decisão. Após tomar a decisão o PDP envia a mensagem de resposta (DEC), determinando os procedimentos para que a decisão seja executada no PEP. Por fim, o PEP envia uma mensagem de notificação (RPT) para indicar o sucesso ou falha na instalação da decisão do PDP. A mensagem RPT também pode ser utilizada para contabilidade (*accounting*). O PEP é responsável por apagar qualquer estado que não seja mais aplicável e notificar o PDP quando o estado de alguma requisição for alterado.

Opcionalmente, o PEP pode ter a capacidade de executar decisões de políticas localmente através do seu LPDP (*Local Policy Decision Point*). Entretanto, o PDP possui a autoridade da decisão durante todo o tempo. Por isso as decisões locais devem ser informadas ao PDP através de um objeto de decisão LPDP. Na decisão final estabelecida pelo PDP, as decisões locais podem ser validadas ou modificadas. A utilização do LPDP é bastante útil para tornar o sistema tolerante a falhas, possibilitando que o PEP execute decisões locais na falta de comunicação com o PDP.

O modelo *outsourcing* é principalmente utilizado para controlar as políticas de redes que implementam Serviços Integrados (*IntServ*), assim quando o protocolo de sinalização RSVP solicita a reserva de recursos em um nó da rede (PEP), este consulta o servidor de políticas (PDP) que por sua vez decide se os parâmetros da reserva podem ser atendidos totalmente, parcialmente ou não podem ser atendidos. A extensão do COPS para utilização com RSVP é especificada em [HEZ00].

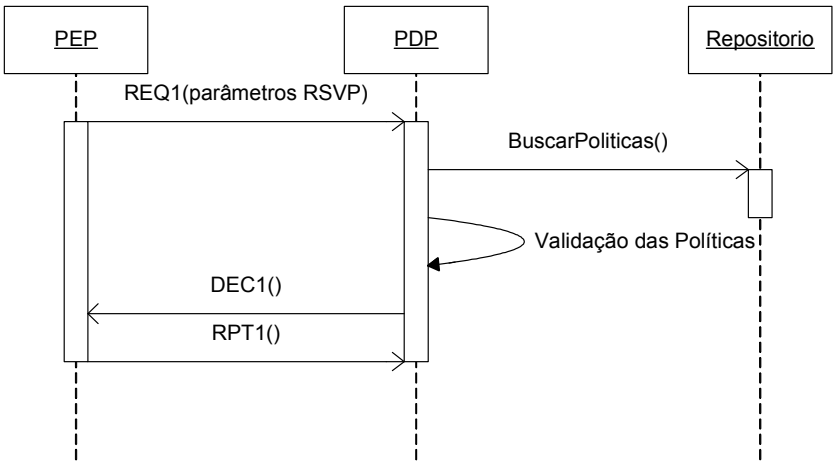

Figura 3.12: Estratégia *Outsourcing*

3.10.2. Modelo *Provisioning*

No modelo *provisioning* (também chamada de *configuration*) a troca de mensagens de requisição (REQ) e decisão (DEC) ocorre de forma assíncrona, ou seja, a relação 1:1 não é mantida. Neste modelo o PEP, após estabelecer conexão com o PDP, envia uma requisição de configuração contendo informações auto-descritivas (ao invés de representar um evento operacional), tais como tipo de *hardware*, versão do *software*, possíveis configurações e seus parâmetros. Como resposta para solicitação de configuração, o PDP envia todas as políticas relevantes para o dispositivo no momento atual. A partir de então o PDP envia as alterações de configuração (instalação de novas políticas ou atualização/remoção de políticas previamente instaladas) para o PEP de modo pró-ativo, reagindo às alterações de estado detectadas na rede.

Sempre que a configuração de um dispositivo cliente (PEP) é alterada (uma placa é acrescentada, novo *software* é instalado, etc.), uma nova mensagem de requisição de configuração (REQ), contendo as novas informações do PEP é enviada ao PDP.

Análogo ao que ocorre no modelo *outsourcing*, no *provisioning* toda mensagem de decisão (DEC) enviada pelo PDP é respondida com uma mensagem de notificação (RPT) do PEP para indicar o sucesso ou falha da instalação da decisão e também para fins de contabilidade.

Na área de QoS, o modelo *provisioning* é destinado ao controle de políticas em redes que não implementam protocolos de sinalização. Assim, os recursos da rede seguem os contratos (SLAs) que são previamente estabelecidos e relativamente estáticos. Esta estratégia atende aos requisitos da arquitetura de *DiffServ*, na qual os

dispositivos da rede (roteador / *switcher*) precisam ser previamente configurados com as políticas de encaminhamento de tráfego. Na implementação da estratégia *provisioning*, a extensão COPS-PR transporta as informações de configuração.

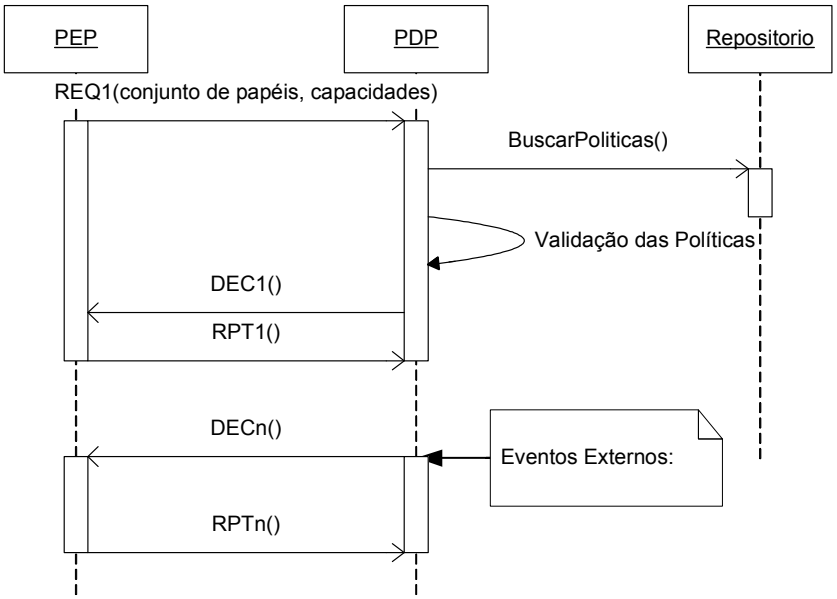

Figura 3.13: Estratégia *Provisioning*

3.11. *Policy Information Base* (PIB)

Um módulo PIB é representado por uma coleção de tabelas com o formato conceitual de árvore, onde são armazenadas as informações de políticas. Sua estrutura (galhos) é formada por classes *Provisioning Classes* (PRCs) que são unicamente identificadas através de um OID (*Object Identifier*) registrado, seguindo a recomendação ITU-T X.208. Estas classes (PRCs), por sua vez, podem possuir várias instâncias (folhas) *Provisioning Instances* (PRIs), como mostrado na figura 3.14.

Os módulos PIB são descritos conforme a estrutura especificada na SPPI (*Structure of Policy Provisioning Information*)[RFC 3159]. A descrição da SPPI é feita através da notação ANS.1 (*Abstract Syntax Notation One*) e seu formato é muito similar à SMI (*Structure Management Information*). Com isto, várias estruturas definidas para MIBs (*Management Information Bases*), podem ser importadas para PIBs.

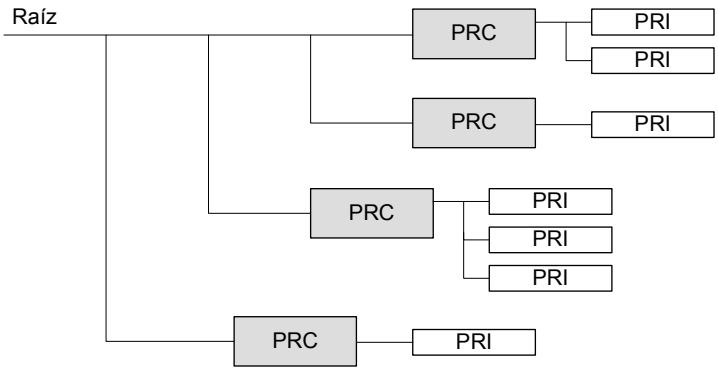

Figura 3.14: Estrutura de árvore da PIB

Na [SAH03], o IETF especifica um módulo PIB (*Framework Policy Information Base*) com convenções textuais e classes (PRCs) genéricas, que são comuns a todos os tipos de clientes que recebem a provisão de políticas através do protocolo COPS-PR. O módulo com classes especializadas para controle de QoS em Serviços Diferenciados (*Differentiated Services QoS Policy Information Base*) é descrito em [RFC 3317].

Quanto à forma de acesso, existem três tipos de classes:

1) *Notify* – os valores dos atributos destas classes são definidos pelo próprio dispositivo (PEP) e não podem ser alterados pelo PDP. Normalmente estas classes representam as características e capacidades inerentes a cada dispositivo. Estas informações são transmitidas ao PDP na mensagem de requisição de configuração enviada pelo PEP.
2) *Install* – os valores dos atributos destas classes são preenchidos de acordo com o resultado do algoritmo de decisão de políticas executado pelo PDP.
3) *Notify / Install* – as classes deste tipo combinam as características dos dois ítens acima, ou seja, o PEP pode pré-configurar os valores dos atributos destas classes e o PDP decide se estes valores devem ser mantidos ou alterados. Um exemplo de PRC deste tipo é *IfRoleComboTable*, que representa a combinação de papéis de uma interface, assim o PEP pode deixar os atributos desta classe em branco para que o PDP determine a combinação de papéis da interface, ou pré-configurar os valores para que o PDP aceite ou altere.

3.11.1. *Framework* PIB

A *Framework* PIB especifica quatro grupos de classes, como mostrado no diagrama UML da figura 3.15.

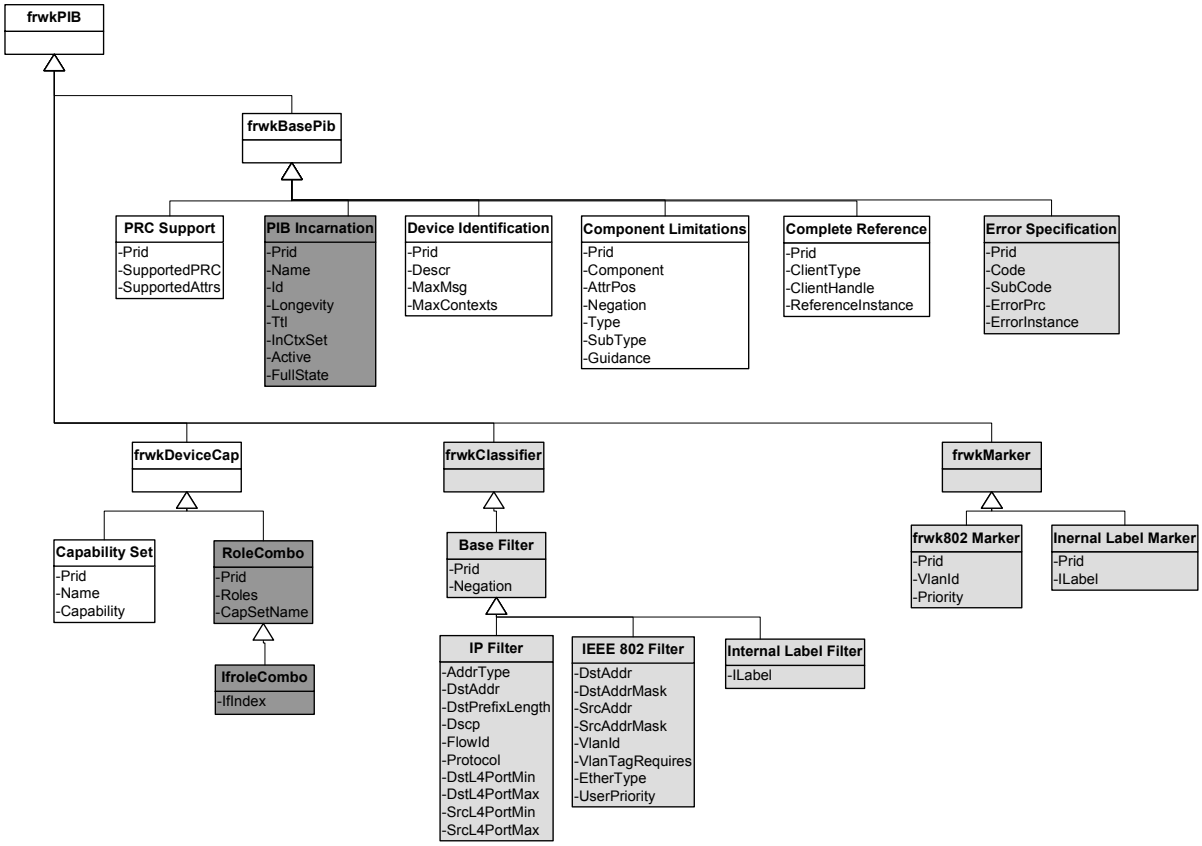

Figura 3.15 PRCs Framework PIB

(sem preenchimento = *notify;* preenchimento cinza escuro = *notify/install;* preenchimento cinza claro = *install*)

1) Grupo *Base PIB* – descreve as classes e os atributos suportados pelo PEP, suas limitações e a sua configuração atual.
2) Grupo *Device Capabilities* – neste grupo estão as classes que descrevem o conjunto de capacidades das interfaces do dispositivo e a combinação de papéis associada a estas interfaces.
3) Grupo *Classifier* – as classes deste grupo representam filtros IP, filtros IEEE 802 e rótulo interno (*Internal Label*).
4) Grupo *Marker* – As classes deste grupo representam o marcador 802 e o marcador *Internal Label*.

3.11.2. *DiffServ* PIB

O módulo *DiffServ* PIB [RFC3317] é projetado de acordo com o Modelo de Gerenciamento Informal de Serviços Diferenciados documentado em [BER02], o qual

descreve a forma como são modeladas as interfaces de ingresso e egresso de um roteador genérico com "n" portas. O modelo define a configuração e gerenciamento de uma interface *DiffServ* seguindo a estrutura *Traffic Conditioning Block* (TCB). Um TCB, por definição, é constituído por zero ou mais classificadores, medidores, ações, algoritmos de descarte, filas e escalonadores, portanto as PRCs do módulo DiffServ PIB modelam estes elementos.

As classes *DiffServ* PIB são divididas em dois grupos: *dsCapibilityClasses* e *dsPolicyClasses*.

As classes do grupo *dsCapability* são do tipo *notify* e descrevem as capacidades e limitações do dispositivo, utilizando a estrutura genérica e extensível *Framework* PIB. Através destas tabelas o PEP informa ao PDP quais classes (PRCs) são implementas ns sua PIB, indicando os elementos funcionais que podem ser configurados no dispositivo.

O grupo *dsPolicy* contém classes do tipo *install* que são utilizadas para definir a seqüência de tratamento a ser aplicada aos pacotes e os parâmetros destes tratamentos. Estes dois aspectos subdividem o grupo *dsPolicy* em classes que representam um elemento funcional dentro do caminho dos pacotes (classificadores, medidores, ações etc.) e as classes que especificam os parâmetros a serem aplicados em um determinado tipo de elemento funcional (medidor *Token Bucket*, marcador DSCP etc.). Os parâmetros são normalmente representados em classes separadas para possibilitar o reuso de informações, assim várias políticas podem utilizar o mesmo conjunto de parâmetros. A indicação do próximo elemento funcional no caminho dos dados é feita através da referência que identifica a instância (PRID) do elemento correspondente. Esta estratégia permite que os elementos sejam arranjados em qualquer forma, seguindo a política da QoS a ser praticada.

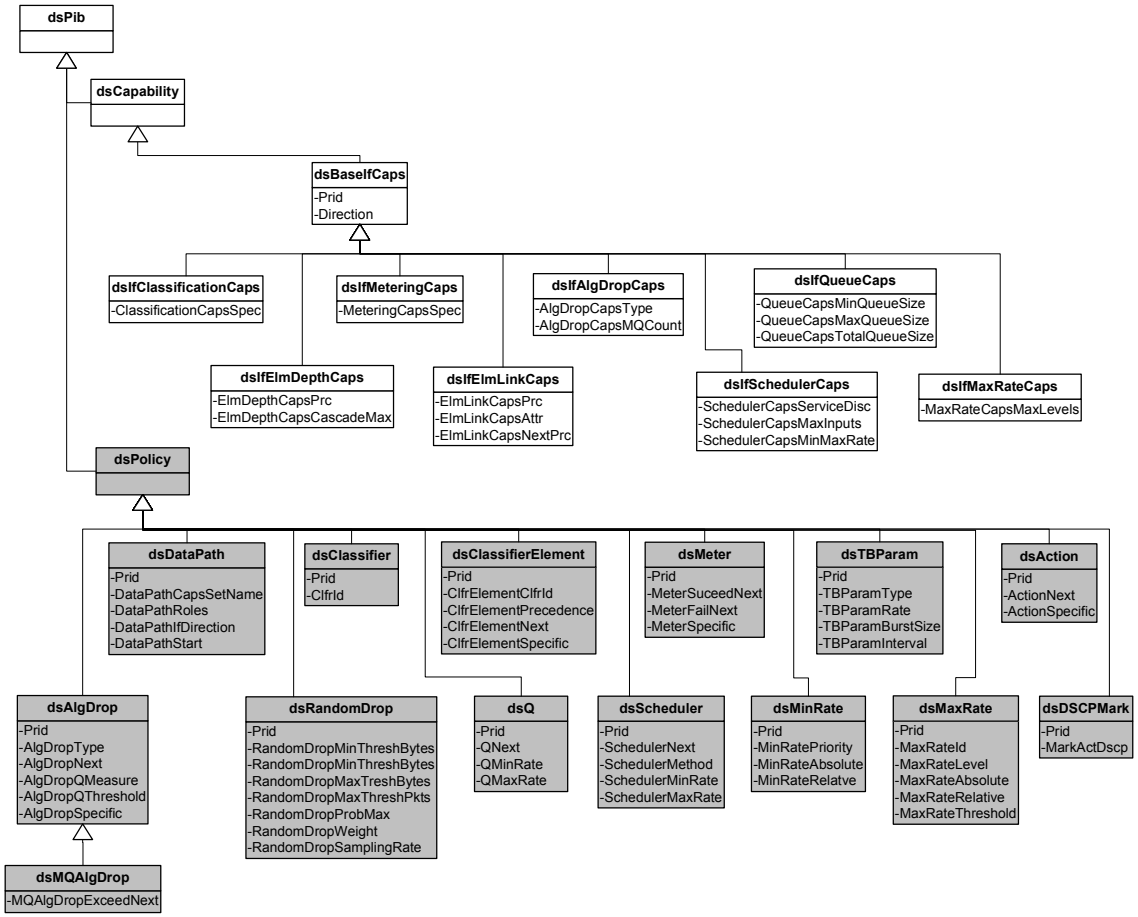

Figura 3.16: PRCs DiffServ PIB

3.12. Conclusão

Como apresentado neste capítulo, o IETF descreve modelos genéricos para representação de políticas, sugere uam arquitetura para implementação PBNM e especifica os protocolos adequados para transportar as informações de políticas. Portanto, é possível a implementação completa de uma arquitetura PBNM, baseada nos padrões do IETF. A proposta deste trabalho (detalhada nos capítulos 5 e 6) justamente segue esta estratégia, definindo uma arquitetura de gerenciamento baseado em políticas de alto nível para *DiffServ*.

Capítulo 4

Trabalhos Relacionados com Políticas de Negócio para QoS

4.1. Introdução

Como discutido no capítulo 3, políticas de QoS representam um conjunto de regras para administração, gerenciamento e controle de acesso dos recursos de rede [RFC 3060]. No caso de políticas de QoS destinadas a metodologia *DiffServ*, pode-se considerar dois níveis de abstração extremos:

1) Políticas de Alto Nível – expressam as regras de negócio e refletem o modo como os seres humanos definem o desempenho da rede. Normalmente, as políticas de alto nível são representadas através de Contratos de Níveis de Serviços (SLAs) e descrevem o efeito desejado sem abordar os detalhes técnicos da implementação. Por exemplo: Os integrantes do Departamento de Pesquisa, acessando aplicações multimídia recebem tratamento Ouro.
2) Configuração de Dispositivos – determina os parâmetros dos elementos que compõem as funções disponíveis nos equipamentos. Esta configuração é específica para cada dispositivo e leva em consideração suas capacidades e detalhes técnicos, por isso é de baixo nível.

Para que os objetivos definidos em alto nível sejam efetivamente realizados na rede, é necessária a implementação de mecanismos intermediários de mapeamento, capazes de traduzir as regras de negócio em configurações de dispositivos. Este processo de mapeamento é definido em [WES01] como *policy translation*.

O processo de tradução é um importante aspecto para implementação do gerenciamento baseado em políticas.

Neste capítulo são discutidas algumas propostas relacionadas ao tratamento de políticas de negócio e configuração de dispositivos.

4.2. Gerenciamento de SLA Baseado em Políticas para Redes Coorporativas

A estrutura do sistema baseado em políticas do IETF foi apresentada na figura 3.1 e é composta por quatro elementos: a ferramenta de gerenciamento de políticas, a entidade que toma as decisões de políticas (PDP), a entidade que aplica as políticas (PEP) e o repositório de políticas. [VER01] e [VER02] são trabalhos que têm como foco principal a ferramenta de gerenciamento de políticas. [VER01] propõe uma ferramenta de Gerenciamento de QoS para administrar Serviços Diferenciados em redes coorporativas e [VER02] torna a ferramenta genérica, acrescentando também um exemplo de gerenciamento de Extranets coorporativas utilizando IPsec .

Diferente do que ocorre em um *Internet Service Provider* (ISP), no ambiente de uma rede empresarial a questão da diferenciação da qualidade de serviço associada a preço é pouco importante. Na empresa, o fator preponderante para definição das classes de serviço é a importância que um determinado tráfego ou conjunto de tráfegos tem dentro do negócio da empresa. A ferramenta proposta por [VER01] consiste de cinco componentes principais como mostrado na figura 4.1:

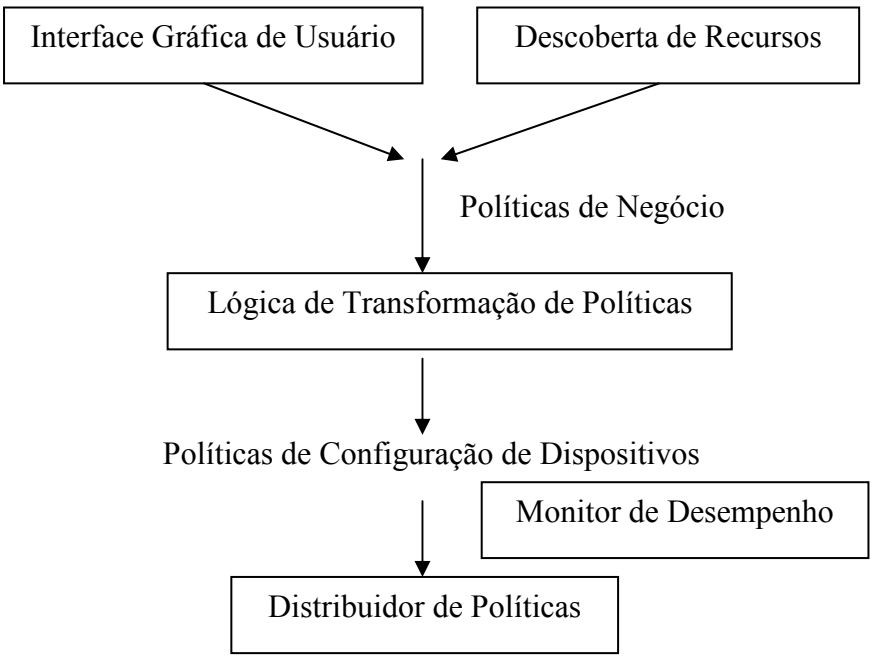

Figura 4.1: Componentes da Ferramenta de Gerenciamento de QoS

1) Interface gráfica para usuário – é o meio através do qual um administrador insere os objetivos para implementação de QoS na rede. Os objetivos representam as regras de negócio e definem a performance da rede para que as necessidades dos seus clientes sejam satisfeitas.
2) Componente para Descoberta de Recursos – responsável por determinar a topologia da rede, seus usuários e aplicações disponíveis. Este componente também identifica as capacidades de cada dispositivo dentro da rede.
3) Lógica de transformação – certifica se as regras de negócio que são especificadas pelo administrador da rede são consistentes, corretas e adequadas às capacidades e topologia da rede. Também traduz as regras de negócio em informações para configuração de dispositivos.
4) Distribuidor de Configuração – é responsável por distribuir a configuração para os diversos dispositivos.
5) Monitor de performance – compara a performance obtida pelos diferentes fluxos de tráfego na rede e determina se as regras especificadas pelo administrador estão sendo satisfeitas ou não.

Embora todos os componentes tenham sua importância dentro da estrutura, [VER02] admite que a lógica de transformação é o ponto vital no gerenciamento baseado em políticas, pois envolve o modo como as políticas são representadas e a forma como estas são gerenciadas.

4.2.1. Lógica de Transformação

O módulo que implementa a lógica de transformação é responsável por validar as informações das políticas descritas em alto nível e transformá-las em configuração dos dispositivos dentro da rede. O processo de validação incorpora verificações de sintaxe e de semântica. A validação semântica das políticas de alto nível consiste nas seguintes verificações:

- Verificação de limites – verifica se o valor de um determinado atributo que compõe a especificação da política está de acordo com os limites determinados pelo administrador da rede para aquele atributo.
- Verificação de consistência – verifica se existe conflito entre as políticas definidas por um administrador.

- Verificação de domínio – verifica se foram especificadas políticas que nunca se tornarão ativas dentro da rede. Este fato pode ocorrer devido a definição de políticas que se sobrepõem tornando outras inoperantes.
- Verificação de aplicabilidade das políticas - garante que o conjunto de regras descritas por um administrador podem ser praticadas dentro do ambiente operacional ao qual elas são destinadas, verificando se estão de acordo com as capacidades da rede.

Uma discussão mais detalhada sobre os algoritmos de validação das políticas é apresentada em [VER02].

O processo exato de tradução das políticas de alto nível para configuração de dispositivos é específico para cada disciplina de política[2] , entretanto, em [VER02] a ferramenta para gerenciamento de políticas utiliza uma estrutura comum, dentro da qual o procedimento de tradução para cada disciplina pode ser executado. Esta estrutura comum define regras que estabelecem o mapeamento entre as tabelas com as políticas de alto nível (regras de negócio) e as tabelas com políticas de baixo nível (configuração de dispositivos, levando em consideração os detalhes técnicos). Uma vez que estas regras são definidas para uma determinada disciplina, a ferramenta de gerenciamento pode utilizá-las para executar a tradução de um modo genérico.

Para exemplificar este processo, [VER02] utiliza tabelas XML tanto para representar as regras de negócio como para representar a configuração dos dispositivos e para cada um destes casos é utilizado um DTD diferente. As regras de tradução consistem em mapeamentos XSLT que são aplicados pela ferramenta de gerenciamento. Enquanto as regras XSLT precisam ser especificadas para cada disciplina específica, a ferramenta de gerenciamento de políticas executa a tradução de forma genérica.

[2] [VER02] define disciplina de política como a combinação de dois fatores: necessidade do negócio e a tecnologia que suporta esta necessidade.

4.2.2. Representação e Distribuição das Políticas

Para representação das políticas [VER01] e [VER02] propõem a utilização de tabelas. Estas tabelas são constituídas por múltiplos atributos e parte destes atributos refere-se às condições e a outra parte refere-se às ações, desta forma as tabelas implementam a notação de regras. Diferentes tabelas precisam ser especificadas, conforme a variação dos elementos que compõem as condições e ações das regras.

Em [VER01] uma política representa um determinado SLA de negócio e é descrita da seguinte maneira:

CONDIÇÃO → Um usuário (ou grupo de usuários) acessando uma aplicação (ou grupo de aplicações) em um servidor (ou grupo de servidores).

AÇÃO → o tráfego que atende as condições especificadas recebe o tratamento de uma classe de serviço específica.

Para cada classe de serviço é definido o tempo de resposta, ou seja, o atraso *round-trip* dos pacotes pertencentes aos fluxos de tráfego da classe.

A modelagem destas políticas pode ser representada pelos diagramas a seguir. O primeiro diagrama mostra os elementos que o administrador manipula para compor as regras de negócio e o segundo diagrama mostra as informações que são utilizadas para configuração dos dispositivos.

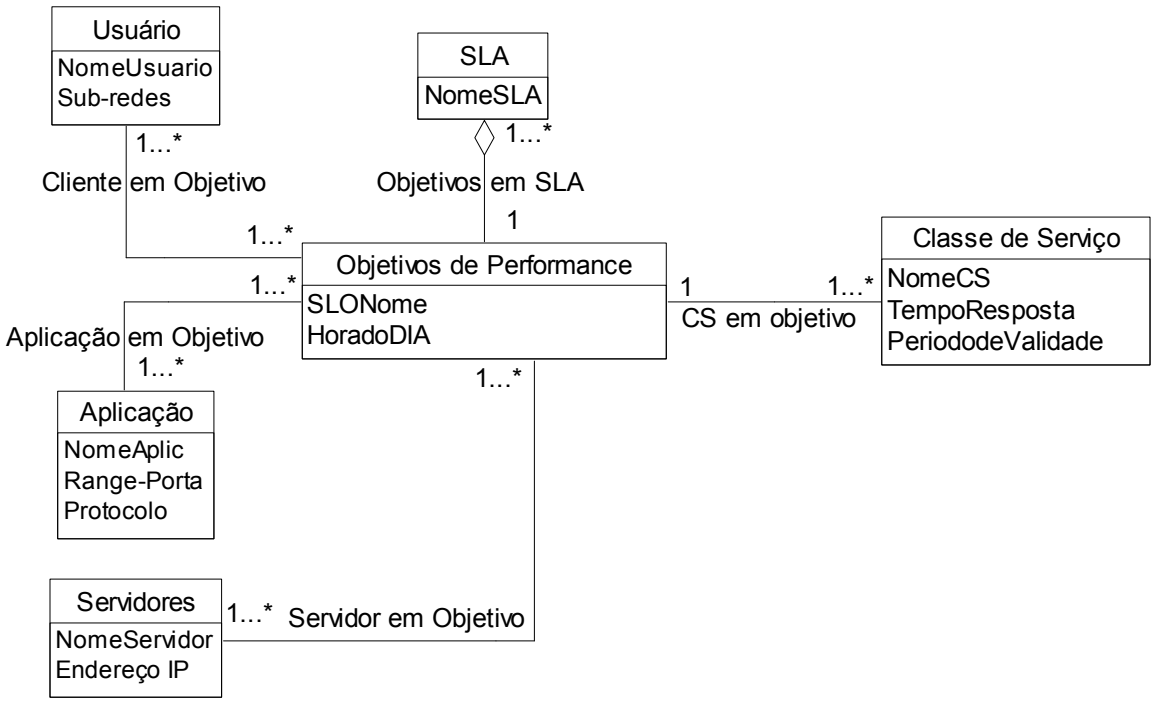

Figura 4.2: Modelo de Objetos para políticas de regras de negócio em [VER02]

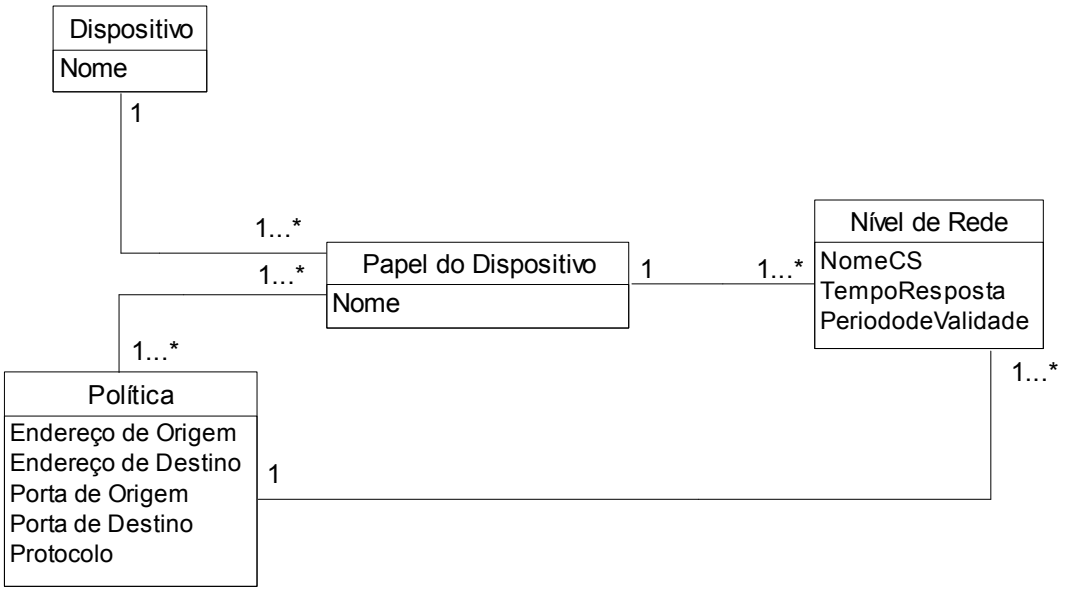

Figura 4.3: Modelo de Objetos para políticas de configuração de dispositivos em [VER02]

Antes de distribuir a configuração para os dispositivos, a ferramenta de gerenciamento verifica a relevância das políticas para cada dispositivo. Para os roteadores, uma regra de configuração de dispositivo é considerada relevante somente se o caminho com o menor número de saltos (*shortest-hop)* entre o IP de origem e o IP de destino passar através de um roteador de acesso. Para os servidores, uma regra é relevante se dentro do seu range de endereços IP de origem ou destino estiver o endereço IP do servidor.

Os dispositivos que possuem o mesmo conjunto de políticas relevantes são agrupados em uma classe com papel do dispositivo comum, esta classe é associada a um PHB específico e então as regras de configuração são distribuídas para cada dispositivo.

4.2.3. Discussão

A ferramenta para gerenciamento de SLA baseado em políticas proposta por [VER01] e [VER02] permite que o administrador manipule os elementos necessários para classificação refinada do tráfego em alto nível. A classificação possibilita a definição de um usuário acessando uma aplicação em um determinado servidor e em um período de tempo definido. Mas a especificação das classes de serviço as quais os tráfegos são associados é bastante restrita, determinando apenas o tempo de resposta para os fluxos de pacotes. A descrição das classes de serviço não contempla parâmetros

para descrição do perfil de tráfego, tratamento de excesso e limites para taxa de perda de pacotes.

Com relação ao modelo para representação das políticas, as propostas utilizam apenas o conceito e a semântica de regras definida no modelo PCIM, mas as classes do PCIM não são efetivamente implementadas. Como alternativa para modelagem das regras, são utilizadas tabelas simplificadas em XML. O modelo de representação das políticas afeta diretamente o processo de tradução de alto nível para baixo nível e por isso são utilizados DTDs específicos e mapeamentos XSLT.

Um fator que não está claro nestas propostas é o modo como as configurações são efetivamente distribuídas. Ao invés de definir o método utilizado, [VER01] enumera algumas alternativas para implementação do processo de distribuição. O protocolo COPS-PR está padronizado e possui os objetos necessários para a provisão de configuração de dispositivos, inclusive pode transportar informações sobre as capacidades e papéis dos dispositivos. Estas informações otimizariam a recuperação das regras relevantes e tornariam a ferramenta mais flexível.

4.3. Gerenciamento de QoS Baseado em Políticas Utilizando Solaris *Bandwidth Manager*

Os elementos utilizados para descrever as política de QoS em alto nível em [SUN03] são similares aos utilizados em [VER01] e [VER02], ou seja, as condições das políticas definem um usuário ou grupo de usuários acessando uma determinada aplicação. As políticas descritas em [SUN03] implementam a semântica de regra definida no padrão do IETF/DMF, assim os fluxos de pacotes que atendem as condições definidas são associados a ações que determinam o tratamento de QoS descrito para mesma política. O foco principal de [SUN03] também são as redes corporativas (*Intranets*) e quatro classes de serviço (Platina, Ouro, Prata e Bronze) são utilizadas para definir a porcentagem de largura de banda e a prioridade do tráfego.

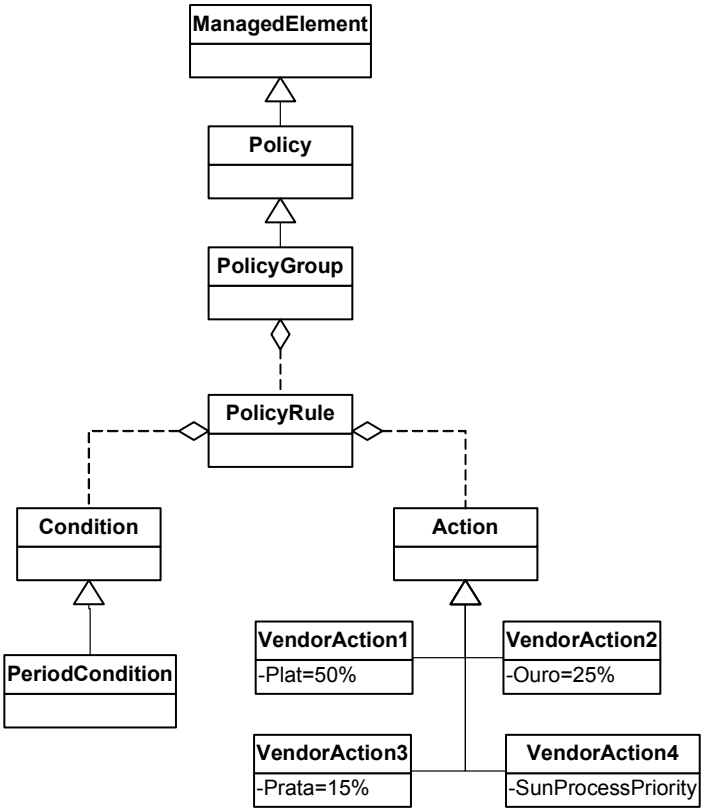

Figura 4.4: Objeto de política em [SUN03]

A principal contribuição de [SUN03] é a abordagem adotada na execução das políticas. Enquanto a maioria dos sistemas de gerenciamento de QoS baseado em políticas é centrado na rede, ou seja, as políticas são efetivamente executadas somente nos dispositivos que tratam do encaminhamento dos pacotes, normalmente roteadores ou *switches*, em [SUN03] a execução das políticas também é aplicada para priorização de processos nos servidores. O argumento que justifica esta abordagem é que muitas vezes a origem do gargalo do sistema é o processamento das requisições no servidor e não a banda que a rede dispõe para encaminhamento dos pacotes. Com isso o gerenciamento exclusivamente centrado na rede pode não contribuir no aumento da performance do sistema como um todo. A combinação das abordagens para execução das políticas, define os sistemas como centrado na rede, centrado nos sistemas ou integrado (rede + sistemas).

[SUN03] apresenta os resultados obtidos com um sistema de gerenciamento de QoS integrado, utilizando produtos da SUN. Em particular, foi usado o *Sun Management Center* 3.0 para implementar o PDP e *Solaris Bandwidth Manager 1.6* para implementar o PEP. No protótipo testado, ambos rodam no mesmo servidor dedicado, conforme mostrado na figura 4.5.

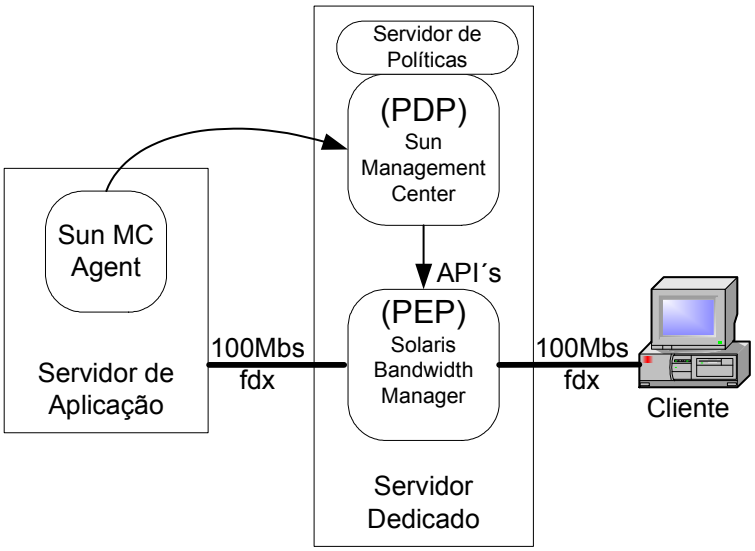

Figura 4.5: Sistema de Gerenciamento de QoS baseado em produtos SUN

Do lado do servidor (e.g. Servidor Web) é usado um agente que mede a carga de requisições no servidor e envia esta informação para o PDP (*Sun Management Center)*. A partir das informações enviadas pelo agente, o PDP toma as decisões necessárias para reconfiguração do PEP (*Solaris Bandwidth Manager*). Quando a carga está "leve", o PEP é configurado para a máxima largura de banda do dispositivo e à medida que a carga aumenta a largura é reduzida. Com a diminuição da banda, a carga injetada no servidor é reduzida Existem duas possibilidades para ajuste da largura de banda: reduzir a banda de forma generalizada ou seletivamente modificar as classes de serviço. Para reconfiguração dinâmica do PEP são utilizadas as APIs disponíveis no *Solaris Bandwidth Manager 1.6* e para coletar as informações da carga no servidor e atualizar o Servidor de Políticas são utilizadas as APIs disponíveis no *SunMC 3.0*.

4.3.1. Discussão

[SUN03] propõe um sistema de gerenciamento de QoS baseado em políticas que além de estabelecer a configuração dos dispositivos que controlam o encaminhamento dos pacotes na rede, também controla a carga das requisições nos servidores. Os resultados medidos comprovaram a efetiva diferenciação dos serviços de acordo com as políticas definidas. Apesar das políticas seguirem o modelo PCIM, a arquitetura para implementação do sistema é baseada em tecnologia proprietária e como o *SunMC 3.0* e

o *Solaris Bandwidth Manager 1.6* não implementam o protocolo COPS, a comunicação entre PDP e PEP é feita através das APIs disponíveis.

4.4. Propostas de Padronização para Especificação de Nível de Serviço (SLS)

As definições utilizadas para a especificação de níveis de serviço (SLS) variam de acordo com a literatura. A [WES01] que SLS é um subconjunto contido no contrato de níveis de serviços (SLA). Enquanto o SLA engloba os aspectos administrativos e legais do contrato de serviços, o SLS é a parte do SLA que descreve, essencialmente, os aspectos técnicos do contrato e representa o conjunto dos objetivos de performance através de parâmetros e valores de QoS. Este é o conceito de SLS adotado nesta dissertação.

A padronização da semântica dos SLSs é um fator bastante útil na negociação destes contratos e, principalmente, representa uma importante ferramenta para tornar possível o grande desejo de se estabelecer serviços com garantia de QoS utilizando a infra-estrutura pública da Internet.

Dentro do IETF existem duas proposta de padronização para SLS: [TEQ01] e [AQU00]. Diferente do que se possa imaginar, as duas propostas não são distintas, mas sim complementares. Em [TEQ01] são descritos os atributos para um SLS genérico e [AQU00] utiliza estes atributos para introduzir o conceito de SLSs pré-definidos.

4.4.1. Especificação de Níveis de Serviço em TEQUILA

Uma das propostas do projeto Tequila[3] (***T****raffic* ***E****ngineering for* ***QU****ality of* ***S****ervice in the* ***I****nternet, at* ***L****arge* ***S****cale*) é a padronização da semântica de SLSs. A seguir, são apresentados os atributos que compõem um SLS genérico especificado em [TEQ01]:

[3] http://www.ist-tequila.org

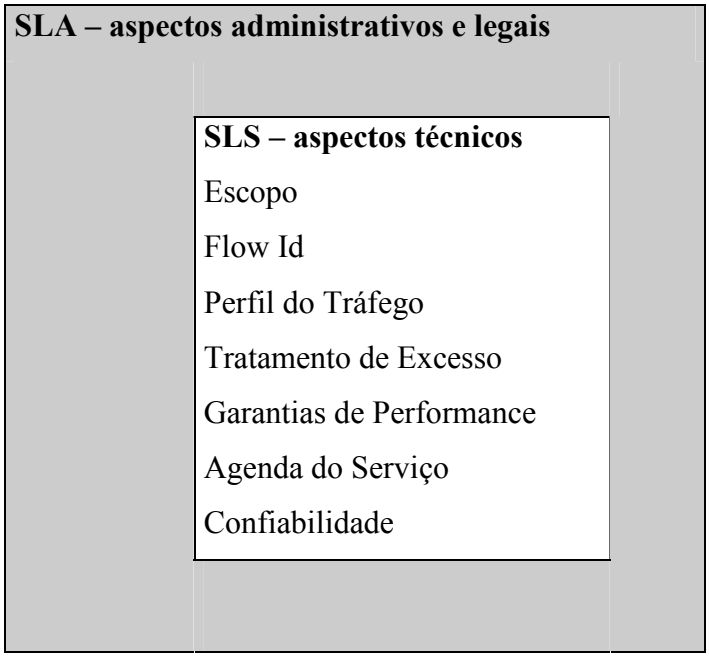

Figura 4.6: SLS em TEQUILA

- Escopo – identifica a região geográfica / topológica onde a política de QoS será executada para o serviço que está sendo especificado. O escopo de um SLS deve ser representado através da combinação de interfaces de ingresso e egresso que representam, respectivamente, os pontos de entrada e saída dos pacotes IP na região (rede) definida.

Escopo = (ingresso, egresso)

A combinação das interfaces de ingresso e egresso possibilita os seguintes tipos de comunicação:

(1,1) – ponto a ponto;

(1,N) – um para muitos;

(1,any) – um para qualquer;

(N,1) – muitos para um;

(any,1) – qualquer para um;

- Identificação de fluxo – (Flow Id) indica quais pacotes IP receberão o tratamento de QoS definido pelo serviço especificado. Um SLA contém somente um Flow Id, que pode ser especificado através de um ou mais termos descritos abaixo:

Flow Id = (informações de Serviços Diferenciados, informações de origem, informações de destino, informações de aplicação)

- Perfil do Tráfego – descreve as características do fluxo dos pacotes IP identificados pelo Flow Id. O perfil de tráfego é formado por um conjunto de parâmetros que definem a conformidade que um fluxo deve ter para receber as garantias estabelecidas pelos parâmetros de performance. A conformidade de um tráfego pode ser testada de forma binária ou multi-nível. No teste binário é determinado se os pacotes de um fluxo IP estão dentro do perfil de tráfego ou excedem os parâmetros definidos. No caso multi-nível é utilizado um algoritmo para determinar o valor inteiro (1...n) a ser marcado nos pacotes. Os pacotes marcados com valor n são chamados excesso. Para especificação do perfil de tráfego são utilizados os parâmetros *Token Bucket* (b,r), a taxa de pico (p), a unidade máxima de transporte (MTU) e o tamanho mínimo do pacote.

- Tratamento de Excesso – é o atributo do SLS que determina a ação a ser tomada para os pacotes de excesso. O tráfego excedente pode ser descartado, formatado ou remarcado. Se o tratamento de excesso não for indicado, então o tráfego em excesso é descartado. No caso de formatação os pacotes são atrasados até que estejam em conformidade com o perfil de tráfego especificado e para isto é utilizado um atributo extra que representa o tamanho do *buffer* do formatador. Para (re)marcação o atributo extra utilizado é o DSCP.

- Garantias de Performance – descreve as garantias do serviço oferecido pela rede para o fluxo de pacotes definido em Flow Id e dentro do espaço geográfico / topológico determinado pelo escopo. Quatro parâmetros podem ser utilizados para descrever as garantias de performance: atraso, *jitter*, perda de pacote e *throughput.* A garantia de performance pode ser definida de forma quantitativa ou qualitativa. Um parâmetro é dito quantitativo quando é especificado por um valor numérico e um SLS é classificado como quantitativo quando pelo menos um dos seus parâmetros é especificado de forma quantitativa. Se nenhum parâmetro do SLS for definido quantitativamente, existe a possibilidade das performances de atraso e perda de pacotes serem qualificadas, ou seja, definidas através de valores relativos e abstratos. Por exemplo, um serviço com baixo valor de atraso poderia ser

denominado de ouro e um serviço com baixa perda de pacotes poderia ser denominado verde. Estes serviços podem ser combinados, definindo serviços do tipo ouro verde. E a quantificação das diferenças relativas entre os níveis baixo, médio e alto é estabelecida pelo administrador de políticas. Por exemplo, alto = 2 x médio e médio = 2 X baixo.

- Agenda do serviço – indica a disponibilidade do serviço, definindo seu tempo de início e fim. Para expressar a agenda do serviço podem ser utilizados os seguintes parâmetros: range de horas do dia, range de dias da semana, range de meses do ano e range de anos.

- Confiabilidade – define o maior tempo médio de falhas do serviço no ano (MDT) e o máximo tempo para reparo dos serviços (TTR).

4.4.2. Utilização de Níveis de Serviços Pré-Definidos em AQUILA

O consórcio AQUILA[4] (***A**daptative Resourse Control for **Q**oS **U**sing an **I**P-based **L**ayered **A**rchitecture)* possui vários aspectos comuns ao TEQUILA. A principal diferença é que [AQU00] introduz o conceito de tipos de SLS pré-definidos, baseados na definição genérica de SLS.

A idéia central dos tipos de SLS pré-definidos é simplificar a tarefa de gerenciamento da rede, possibilitando uma eficiente agregação de fluxos. Do ponto de vista das aplicações, um tipo de SLS pré-definido suporta um range de aplicações que se comportam de modo similar e por isso também possuem requisitos de QoS similares.

Na proposta [AQU00] os tipos pré-definidos fixam valores (ou range de valores) para um subconjunto de parâmetros de um SLS genérico definido em [TEQ01]. Desta forma os tipos de SLS pré-definidos representam uma compressão do SLS genérico, com intuito de facilitar o processo de mapeamento para os mecanismos que concretamente implementam QoS.

Outro argumento apresentado em [AQU00] para justificar o uso de SLSs pré-definidos é que as aplicações de Internet podem ser basicamente reunidas em três importantes grupos com características de QoS semelhantes:

---

[4] www.ist-aquila.org

1) Aplicações multimídia interativas: os dados recebidos são exibidos imediatamente (por exemplo: Videoconferência, Voz sobre IP). Este tipo de aplicações depende principalmente de largura de banda (*throughput*), atraso e *jitter*.
2) Aplicações multimídia do tipo *stream*: os dados são armazenados parcialmente ou totalmente antes de serem exibidos (por exemplo: áudio e vídeo sob demanda, Internet TV). Este tipo de aplicações depende principalmente de largura de banda (*throughput*).
3) Aplicações de missão crítica: os dados precisam ser comunicados em tempo real e de forma confiável (por exemplo: B2B, transações bancárias, jogos em rede). Este tipo de aplicações depende principalmente de atraso e perda de pacotes, enquanto que a largura de banda necessária é relativamente baixa.

Os tipos de SLS pré-definidos em [AQU00] mantêm compatibilidade com os atributos definidos em [TEQ01]. Para diferenciação entre um SLS genérico (customizado) e um SLS pré-definido, [AQU00] acrescenta o atributo tipo SLS. Quando um tipo pré-definido é utilizado, apenas um subconjunto de parâmetros é especificado na instância do SLS e algumas restrições são aplicadas ao range de valores dos parâmetros e também na combinação de parâmetros permitida.

Quatro tipos de serviço são definidos em [AQU00]: *Premium* CBR, *Premium* VBR, *Premium* Multimídia, *Premium* Missão Crítica.

4.4.3. Discussão

As Especificações de Nível de Serviço (SLSs) são o ponto de partida para o administrador escrever as políticas que representam o efeito desejado na rede. Desta forma, a utilização de estruturas padronizadas facilita o mapeamento do SLS para o modelo de informação PCIM e suas extensões. O conceito de tipos de SLS pré-definidos proposto em [AQU00] reduz a complexidade do processo de tradução das políticas em configuração efetiva dos dispositivos da rede, por isso esta é a abordagem adotada neste trabalho.

4.5. Conclusão

Apesar da arquitetura de gerenciamento de QoS proposta nesta dissertação possuir várias similaridades com os trabalhos discutidos neste capítulo, ela difere destes

em aspectos importantes. Nenhum dos trabalhos apresentados implementa completamente os padrões IETF / DMTF no que diz respeito à representação e distribuição de políticas. Em [VER01] [VER02] alguns conceitos CIM e PCIM são mencionados, mas o trabalho define seu próprio modelo para representação de políticas, servidores, clientes e parâmetros de configuração de QoS. O modelo de políticas usado em [SUN03] segue a extensão PCIM (PCIMe), mas a distribuição e representação das políticas no cliente (PEP) seguem um modelo proprietário que não adota COPS nem PIB. O projeto [TEQ01] tenta definir uma forma padronizada para representação de SLS e [AQU00] propõe um conjunto de tipos de SLS pré-definidos com intuito de reduzir a alta complexidade resultante do mapeamento entre a definição de um SLS genérico e os mecanismos de QoS necessários para sua implementação. A arquitetura proposta segue a estratégia de [AQU00], adotando tipos pré-definidos de SLS. Entretanto, na proposta, os tipos SLS descrevem ações pré-definidas, as quais são modeladas através das classes QPIM.

Capítulo 5

Arquitetura Proposta

5.1. Introdução

Levando em consideração a forte tendência que tem sido demonstrada pelos fabricantes de equipamentos de rede em seguir os padrões do IETF, traduzir políticas de alto nível para um módulo *DiffServ* PIB e distribuir as informações de configuração através do protocolo COPS-PR certamente é uma estratégia coerente a ser adotada em uma arquitetura de gerenciamento de QoS. Neste capítulo é apresentada uma visão geral da proposta, incluindo a explicação dos modelos definidos para representação das políticas de alto nível e das políticas de configuração de QoS independentes de dispositivo. É importante comentar que as classes PCIM / PCIMe usadas neste trabalho seguem o modelo atualizado pela versão CIM *Policy* 2.8.

5.2. Visão Geral da Arquitetura Proposta

A figura 5.1 apresenta uma visão geral da arquitetura proposta. A arquitetura é baseada nos padrões do IETF e definida em três camadas: modelo de política de alto nível (que estende o IETF PCIM), modelo de política independente de dispositivo (que estende o IETF QPIM) e um modelo de política dependente de dispositivo (baseado na estrutura IETF DiffServ PIB). O núcleo da estrutura é o modelo de política de alto nível (HLPM – *high level policy model*), onde o administrador expressa os objetivos de negócio através de regras que associam usuários, aplicações e servidores a níveis de serviço, durante um determinado período de tempo (e.g. O Usuário A, acessando o Servidor X, durante o Horário Comercial recebem nível de serviço Ouro). O HLPM é definido como uma extensão do PCIMe e está descrito em detalhes na seção 5.3. As políticas de alto nível são traduzidas em políticas de configuração que são

independentes de dispositivo, ou seja, políticas que definem o efeito de QoS desejado na rede sem mencionar detalhes específicos dos mecanismos, tais como tipo de escalonador e algoritmo de descarte (e.g. O tráfego com IP de origem = 10.0.0.5/24 e IP de destino = 10.0.1.3/24, no horário entre 13h e 17h, é marcado com DSCP = AF11).

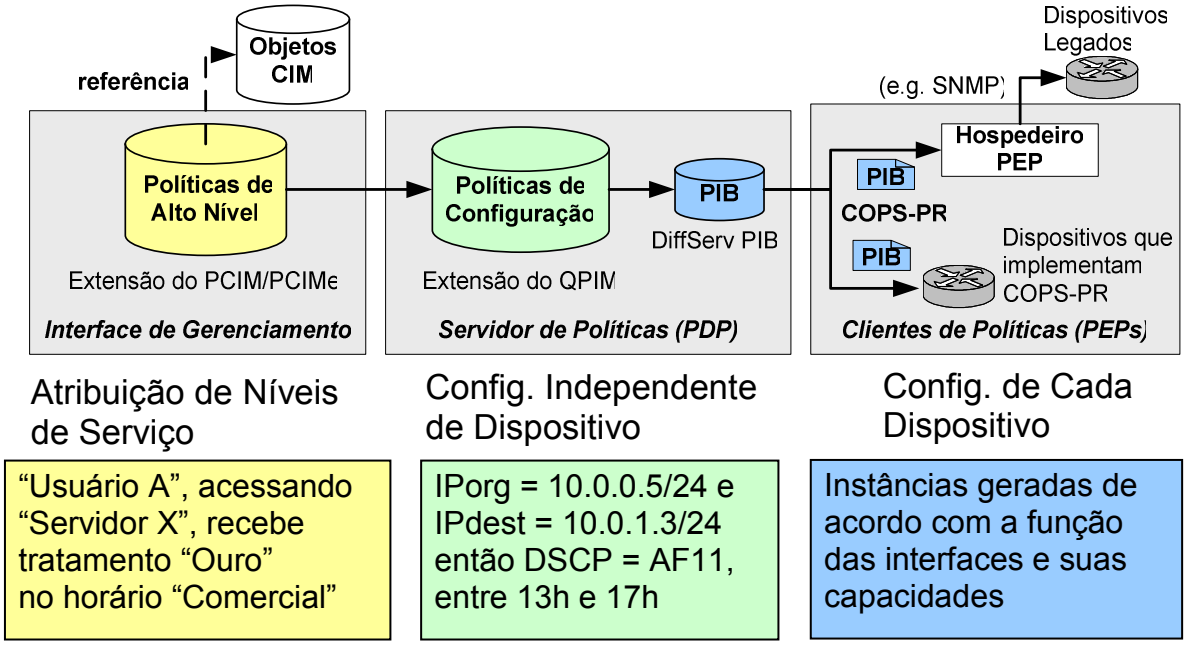

Figura 5.1: Visão geral da arquitetura proposta

O modelo de política de nível de configuração (CLPM – *configuration level policy model*) é definido como uma combinação de classes PCIM/PCIMe e QPIM para possibilitar a representação tanto da identificação de tráfego (condições) como do nível de serviço correspondente (ações). As condições definem a filtragem de pacotes IP e as ações são especificadas através de parâmetros de QoS, tais como controle de congestionamento e alocação de largura de banda. No modelo de configuração, um nível de serviço é representado por um conjunto ordenado de ações QPIM. O CLPM e o processo de tradução de políticas de alto nível em políticas de configuração estão discutidos na seção 5.5. Durante o processo de decisão**,** as informações QPIM são convertidas para instâncias da DiffServ PIB. Este processo é executado quando o PDP recebe a mensagem de requisição COPS-PR, enviada por um PEP para solicitar o aprovisionamento de configuração. A requisição COPS-PR enviada pelo PEP contém dois conjuntos de informações usados como parâmetros de entrada para a decisão do PDP:

- Papel das interfaces gerenciadas para que somente o conjunto de políticas pertinentes àquele PEP seja selecionado;

- Capacidades das interfaces, uma vez que os dispositivos de rede podem suportar diferentes mecanismos para implementar ações de QoS.

Finalmente, a distribuição da configuração dos dispositivos de rede consiste em enviar a *DiffServ* PIB através do protocolo COPS-PR. Duas situações podem ser consideradas:

- Dispositivos que implementas o protocolo COPS-PR e desta forma são capazes de aceitar diretamente a PIB com informação de configuração (i.e. toda a tradução necessária da PIB para comandos específicos do fabricante é implementada internamente no dispositivo);
- Dispositivos legados que necessitam a implementação de um hospedeiro programável, agindo como PEP para converter as informações da PIB em comandos específicos do fabricante usando um protocolo de configuração, por exemplo, SNMP ou CLI;

5.3. Modelo para Representação das Políticas de Alto Nível (HLPM)

O conceito central do modelo proposto para representação de políticas de QoS em alto nível é acomodar as semânticas propostas em [VER01] [VER02] e [SUN03] discutidas no capítulo 4, mas possibilitando ao administrador utilizar as informações dos objetos CIM normalmente existentes no sistema para definir usuários, aplicações e servidores. A figura 5.2 mostra o diagrama UML do modelo HLPM. O conjunto de classes com preenchimento cinza especializa o modelo genérico PCIM/PCIMe, descrito no capítulo 3, para suportar a semântica: "Usuários (ou grupo de usuários), acessando Aplicações (ou grupo de Aplicações) em um Servidor (ou grupo de Servidores), em um determinado Horário, recebe um determinado Nível de Serviço".

Dentro do modelo proposto, uma política é representada por uma instância de *SLSPolicyGroup.* A classe *SLSPolicyGroup* agrupa (através da associação *PolicySetComponent*) as regras de alto nível *SLSPolicyRule*s em conjuntos coerentes definidos pelo atributo *SLSType.* Assim, no tipo Olímpico podem estar contidas as regras Ouro, Prata e Bronze e outros conjuntos podem ser adicionados conforme os objetivos de negócio. No modelo, as condições associadas à *SLSPolicyRule* permitem definir "quem" receberá o nível de serviço e "quando" o serviço estará disponível. A informação "quem" é representada pela classe *CompoundTargetPolicyCondition* e a semântica usuários/aplicações/servidores é possível a partir das associações com as três

classes: *CompoundUserPolicyCondition, CompoundApplicationPolicyCondition e CompoundServerPolicyCondition*. O atributo "*Direction*" é adicionado em *CompoundTargetPolicyCondition* porque nem sempre é desejado que o tráfego no sentido cliente/servidor receba o mesmo nível de serviço que o tráfego no sentido servidor/cliente.

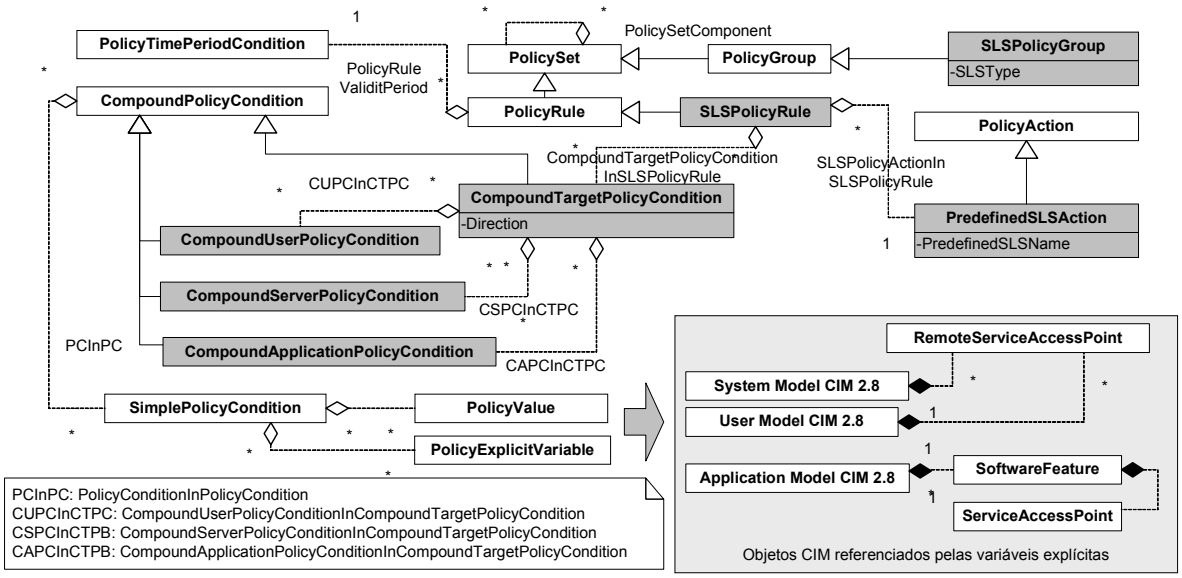

Figura 5.2: Modelo de classes para representação das políticas de negócio
(as extensões são representadas pelas classes com preenchimento cinza escuro)

Uma *CompoundCondition* define os objetos em termos de expressões lógicas. Estas expressões são formadas por *SimplePolicyConditions*, que seguem a semântica variável/valor definida pelo PCIMe. Em alto nível, as variáveis explícitas (*PolicyExplicitVariable*) são usadas para selecionar os objetos CIM cadastrados no sistema e que atendem ao valor especificado. Portanto, fazer com que todas as classes envolvidas na informação "quem" derivem de *CompoundCondition* torna o modelo bastante flexível e permite que as representações de grupos de usuários, grupos de aplicações e grupos de servidores sejam reutilizadas por várias políticas de alto nível. O diagrama de classes da figura 5.3 descreve um grupo de servidores multimídia formado pelos servidores X, Y e Z existentes no ambiente gerenciado. Servidores Multimídia = (ServidorX OU ServidorY OU ServidorZ).

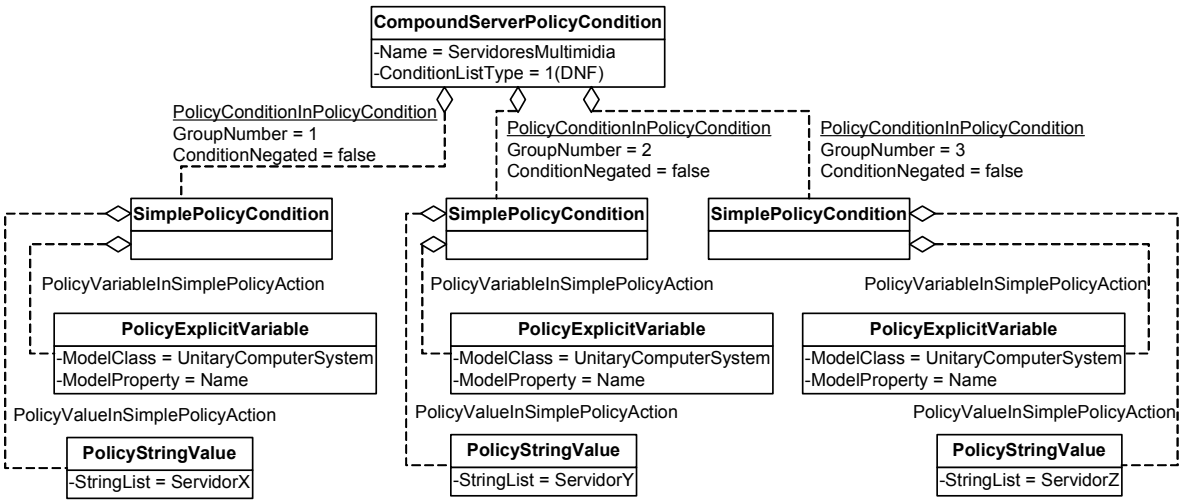

Figura 5.3: Condição Composta de Servidores

A classe *CompoundUserPolicyCondition* seleciona objetos CIM *User,* possibilitando que se obtenha, a partir da associação com *RemoteServiceAccessPoint*, o endereço IP ou a identificação do computador correspondente aos usuários selecionados pela condição. Na classe *CompoundServerPolicyCondition* são selecionados objetos CIM *System* e o endereço IP correspondente aos servidores que satisfazem a condição também é obtido através da associação com *RemoteServiceAccessPoint.* Com a classe *CompoundApplicationPolicyCondition* são selecionados objetos CIM *Application*, possibilitando a obtenção das informações de porta e protocolo através das associações de *SoftwareFeatures* e *ServiceAccessPoint* com as aplicações selecionadas. Os diagramas UML da seção 5.4 apresentam o subconjunto das classes CIM referenciadas pelas políticas de alto nível.

A classe *PolicyTimePeriodCondition* determina os intervalos de validade de cada regra.

Quando as condições de uma *SLSPolicyRule* são satisfeitas, então a correspondente *PredefinedSLSAction* deve ser executada. Na proposta deste trabalho, o atributo *"PredifinedSLSName"* aponta para um conjunto ordenado de ações QPIM definido no nível de configuração. Seguindo este conceito, os níveis de serviço podem estar pré-definidos no sistema, possibilitando o reuso de configurações complexas que foram previamente especificadas, provavelmente por um especialista em *DiffServ*. Esta estratégia torna a tarefa de gerenciamento mais intuitiva para o administrador, que precisa apenas referenciar uma configuração pré-definida para associar um cliente (usuários / aplicações / servidores) a um nível de serviço. Novas configurações de QoS

podem ser acrescentadas no sistema de acordo com as necessidades administrativas de cada ambiente.

5.4. Representação dos Objetos CIM

Como discutido na seção anterior, a informação "quem" (*CompoundTargetPolicyCondition*) de uma regra de alto nível é formada por composições de usuários, aplicações e servidores. Estas composições representam expressões lógicas que selecionam objetos CIM, possibilitando a recuperação dos endereços IP associados aos servidores e estações de usuários e das portas e protocolos das aplicações. Nos diagramas das figuras 5.4, 5.5, 5.6 e 5.7 estão representadas as classes e associações relevantes para arquitetura de gerenciamento de QoS proposta. Apenas alguns atributos estão representados e o detalhamento completo das classes pode ser encontrado na especificação CIM[5].

Na figura 5.4 *UserEntity* é uma classe abstrata que representa usuários, seus nomes, dados de contato e outras informações. A classe *Person* é usada para representar pessoas e possui atributos que indicam os dados de "páginas brancas e amarelas", ou seja, informações particulares e comerciais. Como elemento gerenciado um objeto *Person* pode estar associado a um SAP (*Service Access Point*), no caso especializado pela classe *RemoteSeviceAccessPoint* que define o(s) endereço(s) IP ou nome(s) da(s) estação(ões) de trabalho utilizada(s) pelo usuário.

Um servidor é representado pela classe *UnitaryComputerSystem* no modelo CIM *System* e deriva da classe *ComputerSystem,* conforme mostrado na figura 5.5. A associação *HostedAccessPoint* relaciona um servidor com um endereço IP definido por *RemoteServiceAccesPoint*.

[5] http://www.dmtf.org/standards/cim/cim_schema_v28

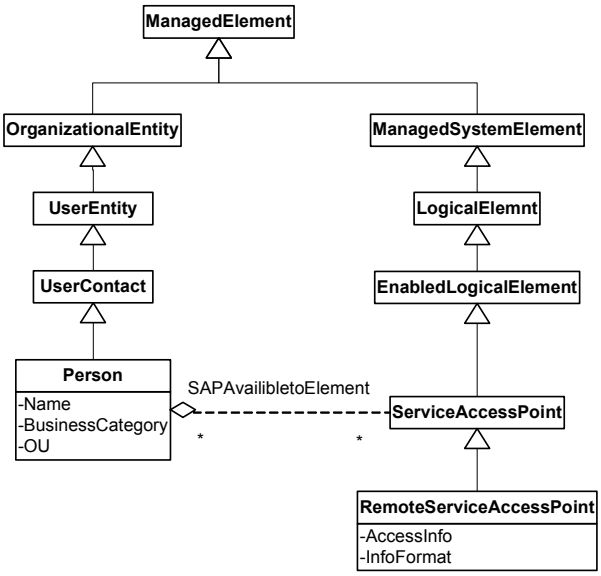

Figura 5.4: Modelo para associação de usuários e endereços IP

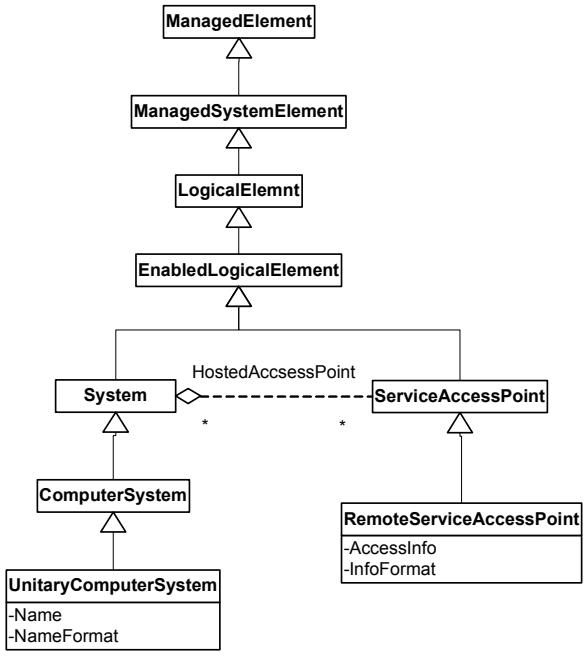

Figura 5.5: Modelo para associação de servidores e endereços IP

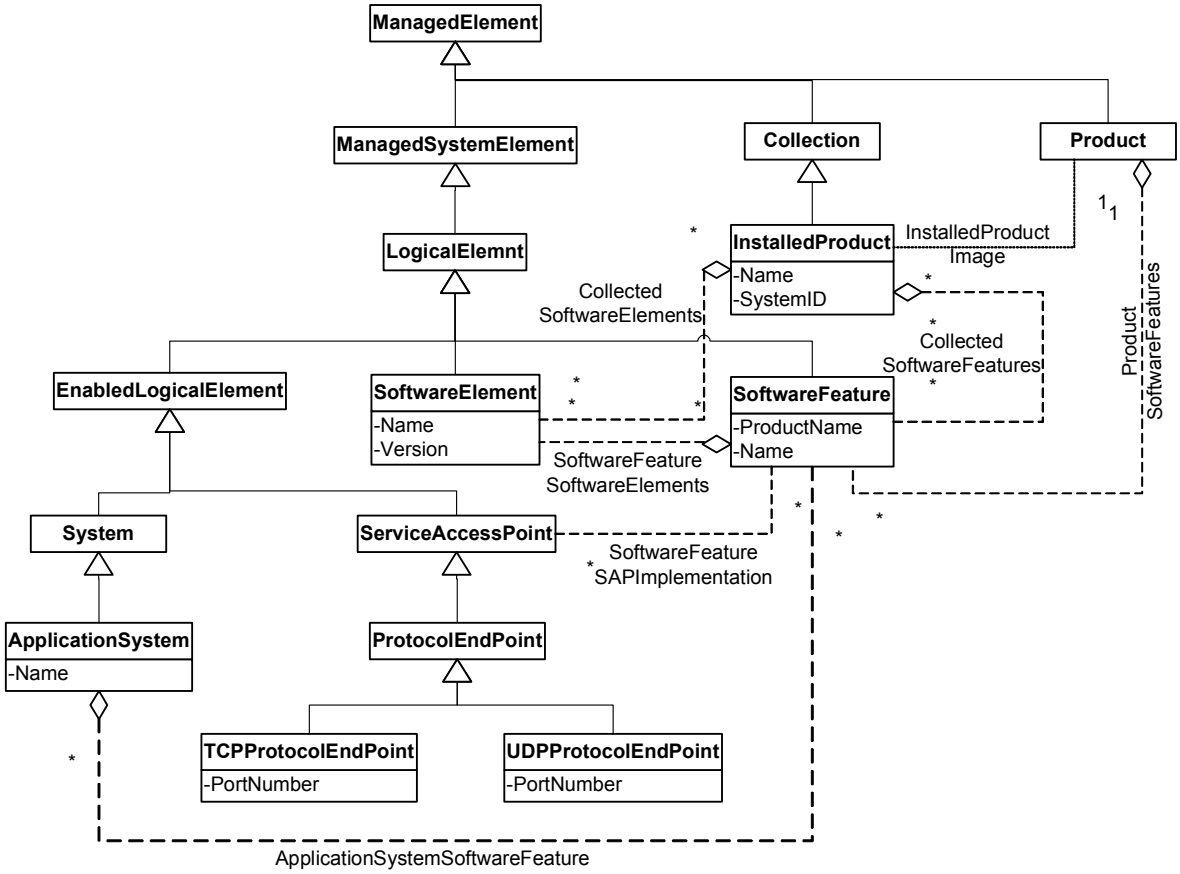

Figura 5.6: Modelo para associação de portas a aplicações

Uma aplicação (figura 5.6) pode ser representada como um sistema de *software* (*ApplicationSystem*) ou um produto instalado (*Installed Product*). A classe *ApplicationSystem* representa uma aplicação ou sistema de *software* que implementa uma função de negócio particular e pode ser gerenciada como uma unidade independente. O sistema pode ser decomposto em seus componentes funcionais através da classe *SoftwareFeature* e usando a associação *ApplicationSystemSoftwareFeature.* Um objeto *SoftwareFeature* pode ser associada com um SAP (*ServiceAccessPoint*) que define a porta ou range de portas e protocolo. A classe *InstalledProduct* representa a instalação de um produto adquirido e de forma análoga à *ApplicationSystem* também pode ser decomposto através da classe *SoftwareFeature,* usando a associação *CollectedSoftwareFeatures.* A classe *SoftwareElement* é usada para refinar ainda mais um objeto *SoftwareFeature,* decompondo as características em conjuntos de elementos gerenciados ou implementados em uma plataforma específica, definida por sua arquitetura de *hardware* e sistema operacional como, por exemplo, *Windows* 2000 com processador Intel.

A figura 5.7 mostra o modelo de classes usado para definir a associação de um endereço IP e uma coleção de papéis a uma sub-rede. A coleção de papéis, por sua vez, está associada a um conjunto de políticas de configuração.

5.5. Modelo para Representação das Políticas de Configuração (CLPM)

Dentro da arquitetura proposta, um SLS pré-definido no sistema é descrito através de ações QPIM. O QPIM foi apresentado no capítulo 3, e suas classes modelam as informações necessárias para gerenciar os mecanismos de QoS (*IntServ* e *DiffServ*). Como neste trabalho é adotada a metodologia *DiffServ*, somente as classes QPIM relevantes para esta abordagem são representadas no modelo CLPM. No que diz respeito a *DiffServ*, o modelo QPIM oferece os elementos necessários para definir os parâmetros de um perfil de tráfego e as respectivas ações para controlar os mecanismos de QoS com intuito de adequar o encaminhamento do tráfego aos níveis especificados. Para completar a representação de políticas no nível de configuração, foi adotada a estratégia sugerida na [SNI03], ou seja, combinar as ações QPIM (ressaltadas pelo quadro cinza) com as condições do PCIM/PCIMe, conforme mostrado na figura 5.8. No modelo proposto, duas extensões de classes foram introduzidas: *ConfigPolicyGroup* e *ConfigPolicyRule*. A configuração de um dispositivo é definida através de instâncias da classe *ConfigPolicyGroup*. Um objeto *ConfigPolicyGroup* agrupa todas as regras de configuração (*ConfigPolicyRules*) que são relevantes para uma determinada combinação de papéis. Por isso a figura 5.8 mostra uma associação entre *PolicyRoleCollection* e *ConfigPolicyGroup*, esta associação permite que sejam definidos "papéis" para um conjunto de configuração. Vale ressaltar que a associação *PolicySetInRoleCollection* não era definida na versão original do PCIMe, mas foi introduzida no modelo a partir da versão CIM *Policy* 2.7.1 e por isto é incluída no modelo de políticas de configuração deste trabalho. De acordo com [WES01] papel é uma característica determinada administrativamente para um elemento gerenciado (e.g. interface de dispositivo de rede) e dentro do conceito de PBNM funciona como um seletor de políticas. Quando uma requisição inicial de aprovisionamento de configuração é enviada, o PEP informa os papéis desempenhados pelas interfaces do dispositivo gerenciado e o PDP seleciona as instâncias *ConfigPolicyGroup* que atendem a estes papéis. Dentro da nossa arquitetura, as instância de *ConfigPolicyGroup* são automaticamente criadas como resultado do

processo de tradução das políticas de alto nível. Portanto, o processo de tradução é também responsável por determinar quais papéis são relacionados para cada instância de configuração, isto é possível a partir da associação da classe CIM *Network* com *PolicyRoleCollection* (quadro pontilhado na figura). Durante a execução do processo de tradução, as informações dos usuários e servidores são traduzidas para endereços IP, estes endereços estão alocados em sub-redes que, por sua vez, estão associadas a uma coleção de papéis, assim são geradas as políticas de configuração associadas com os papéis desta coleção. O processo de tradução é detalhado no capítulo 6.

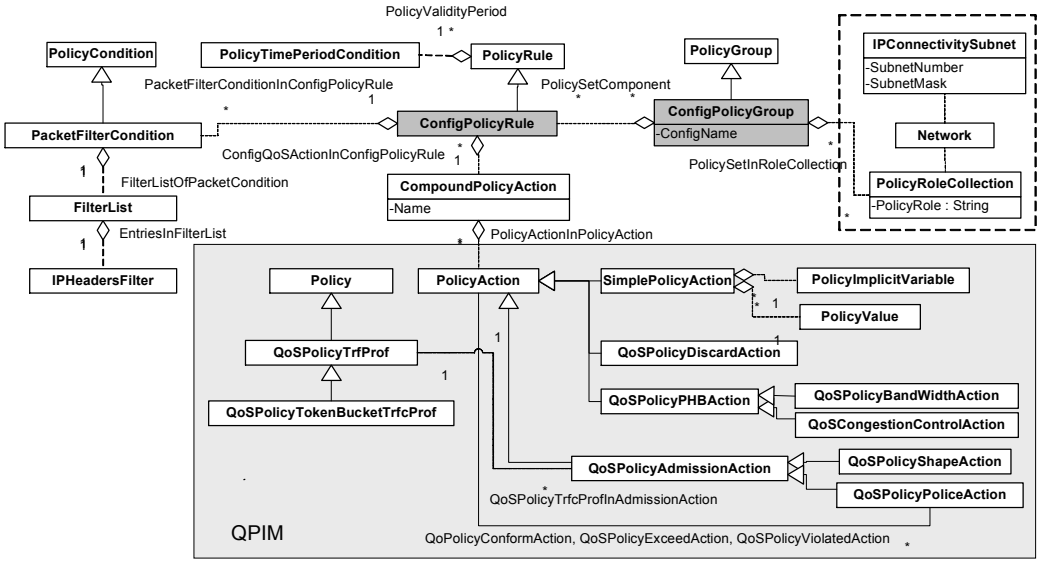

Figura 5.7: Modelo para representação das políticas de configuração de QoS, incluindo as classes PCIM/PCIMe e QPIM

As condições de cada regra *ConfigPolicyRule* são formadas por um conjunto de instâncias *PacketFilterCondition,* esta classe foi introduzida no nível de configuração para completar o modelo QPIM no que diz respeito a identificação de tráfego. A classe *PacketFilterCondition* também não é definida originalmente no PCIMe e foi criada a partir da versão CIM *Policy* 2.7.1. As instâncias *PacketFilterCondition* são compostas por filtros de cabeçalho IP (*IPHeadersFilter*- classe definida no modelo CIM *Network* a partir da versão 2.6) e determinam a seleção dos pacotes que receberão o nível de serviço especificado pela *CompoundPolicyAction* correspondente. Os parâmetros para filtragem dos pacotes IP são obtidos a partir das informações "quem" (informações de usuários, aplicações e servidores) definidas em alto nível, conforme apresentado na seção 5.3. A tabela 5.1 sintetiza as conversões das classes do modelo de alto nível para

o modelo de configuração. É importante deixar claro que a seleção do tráfego também poderia ser representada através do agrupamento de várias condições simples (*SimplePolicyCondition*) com variáveis e valores adequados definidos no PCIMe. Mas o uso das classes *PacketFilterCondition* e *IPHeadersFilter* torna o modelo mais claro e, principalmente, facilita sua implementação.

Tabela 5.1: Conversão das Classes de Alto Nível para Configuração

CompoundTarget PolicyCondition	*PacketFilter Condition*	Converão
Objeto *User* + Objeto *System*	*HdrSrcAdress, HdrSrcMask*, *HdrDestAdress, HdrDestMask*	A partir da informação *RemoteServiceAccessPoint* dos usuários e servidores, obtém-se as informações de IP de origem e destino
Objeto *Application*	*HdrProtocolId HdrSrcPortStart*, *HdrSrcPortEnd*, *HdrDestPortStart*, *HdrDestPortEnd*	A partir da informação *ServiceAccessPoint* das aplicações, obtém-se as informações de protocolo e portas
PredefinedSLSAction	*CompoundPolicyAction*	Converão
PredefinedSLSName	*ConfigName*	O nome de um nível de serviço pré-definido no sistema faz referência a um conjunto de performances de QoS que foi previamente especificado em nível de configuração
PolicyTimePeriodCondition	*PolicyTimePeriodCondition*	Não há conversão

Uma instância *CompoundPolicyAction* é composta por um conjunto ordenado de ações de QoS e representa a performance da especificação de um nível de serviço (SLS) pré-definida no sistema. Para referenciar a performance de QoS desejada, o administrador utiliza o atributo "*PredefinedSLSName*" do modelo de alto nível

apontando para o nome da *CompoundPolicyAction* correspondente. Uma *CompoundPolicyAction* é formada por ações QPIM (mostradas no retângulo cinza da figura).

A classe *PolicyTimePeriodCondition* determina a configuração de um dispositivo de acordo com intervalos de horários.

5.6. Conclusão

Neste capítulo foi apresentada a estrutura geral da arquitetura proposta para automatizar o processo de geração e distribuição de configuração *DiffServ* para dispositivos de rede a partir da definição dos objetivos de negócio descritos em alto nível pelo administrador. Para tal, foi proposto um novo modelo para representação de políticas de negócio, com intuito de simplificar o processo de definição dos objetivos de QoS em alto nível. O principal diferencial em relação aos trabalhos similares encontrados na literatura é que a arquitetura proposta está completamente de acordo com os padrões do IETF no que diz respeito ao gerenciamento de redes baseado em políticas. Os modelos para representação de políticas estendem as classes PCIM/PCIMe e a configuração de QoS é descrita conforme as ações especializadas por QPIM. A configuração de um dispositivo é descrita através de instâncias (PRIs) da *DiffServ* PIB, que contém os elementos funcionais e os parâmetros que compõem o tratamento de QoS recebido por um determinado tráfego. O protocolo COPS-PR é usado para enviar as configurações aos dispositivos.

Um sistema de gerenciamento que segue as diretrizes e os padrões do IETF garante sua compatibilidade com a maior parte dos fabricantes de equipamentos para redes computacionais e possibilita o aproveitamento das informações dos objetos CIM normalmente existentes no ambiente a ser gerenciado.

Os detalhes para efetiva implementação dos modelos propostos são apresentados no próximo capítulo.

Capítulo 6

Implementação dos Modelos de Políticas Propostos e Processo de Tradução

6.1 Introdução

Os modelos propostos no capítulo 5 independem de um esquema particular de repositório ou uma linguagem específica de programação. Entretanto, para a efetiva implementação da arquitetura no ambiente a ser gerenciado, é necessário que as informações modeladas sejam armazenadas em um determinado repositório. Basicamente, havia duas opções para implementação dos repositórios de políticas: 1) utilizar serviços de diretórios baseados no protocolo LDAP; 2) representar as informações em arquivos XML. Neste trabalho, optou-se pelo uso do XML, devido as suas características de linguagem humana, flexibilidade para representação das informações e pela variedade de ferramentas disponíveis para manipulação de documentos XML. O conceito de reuso das informações de políticas é uma premissa básica na implementação do sistema proposto. Neste capítulo é discutido o mapeamento XML, bem como o reuso de informações a partir de referências *XPointer*[W3C01].

Neste capítulo também é descrito o processo de tradução, responsável por converter as políticas de alto nível em políticas de configuração. , o processo de decisão, executado pelo PDP levando em consideração o papel das interfaces, as capacidades dos dispositivos e a grade horária das configurações de QoS, e a implementação do protocolo COPS-PR, o qual é responsável pela comunicação entre PEP e PDP.

6.2. Mapeamento e Reuso das Informações de Políticas em XML

Para implementação da arquitetura, as informações dos modelos propostos foram mapeadas em XML de acordo com o esquema XSD descrito no Anexo A. Os modelos são constituídos por classes estruturais, que representam as informações de políticas, e classes associativas, que além de acrescentar atributos complementares para descrição das políticas, também indicam o relacionamento entre as instâncias das classes estruturais. Neste trabalho, o mapeamento das informações em XML é inspirado nas diretrizes do IETF e DMTF para o mapeamento LDAP [STR04] [DMTF00] e segue as seguintes estratégias:

- Mapeamento das classes estruturais – o mapeamento é um para um, ou seja, cada classe estrutural do modelo de informação é definida como um elemento XML. Os atributos das classes são representados como atributos dos elementos. Apenas os elementos diretamente instanciados são representados nos arquivos XML, estes elementos recebem todos os atributos das classes genéricas superiores.
- Mapeamento das classes associativas (associações ou agregações) – para o mapeamento das classes associativas são adotadas duas abordagens:
    1) Se a associação ou agregação não envolver informações reutilizáveis, é usado o formato superior-subordinado, representado pelo relacionamento entre nó pai e nó filho em XML. Neste caso, a classe associativa não é explicitamente representada e seus atributos são adicionados ao elemento subordinado (ou nó filho).
    2) Se a associação ou agregação envolver informações reutilizáveis, a classe associativa é explicitamente representada como um nó filho em XML e entre seus atributos está a referência *XPointer* que aponta para um específico objeto reutilizável. No caso de associação o nó pai corresponde ao elemento que representa a classe *Antecedent* e a referência *XPointer* aponta para a classe *Dependent.* No caso de agregação o nó pai corresponde ao elemento que representa a classe *GroupComponent* e a referência *XPointer* aponta para a classe *PartComponent.*.

A implementação das estratégias acima ficará mais clara com os exemplos dos arquivos XML que serão apresentados adiante.

No mapeamento, o reuso das informações é implementado através de referências escritas em XML *Pointer Language* (*XPointer). XPointer* define o endereçamento para partes individuais de um documento XML, utilizando a sintaxe *XPath* [W3C01]. O documento referenciado não é especificado na expressão *XPointer*, para tal é usada a URI que precede *#xpointer*.

6.2.1 Estrutura de reuso das informações das políticas de alto nível

A figura 6.1 descreve a organização dos contêineres utilizados para armazenar as informações das políticas de alto nível e o modo como são reutilizadas.

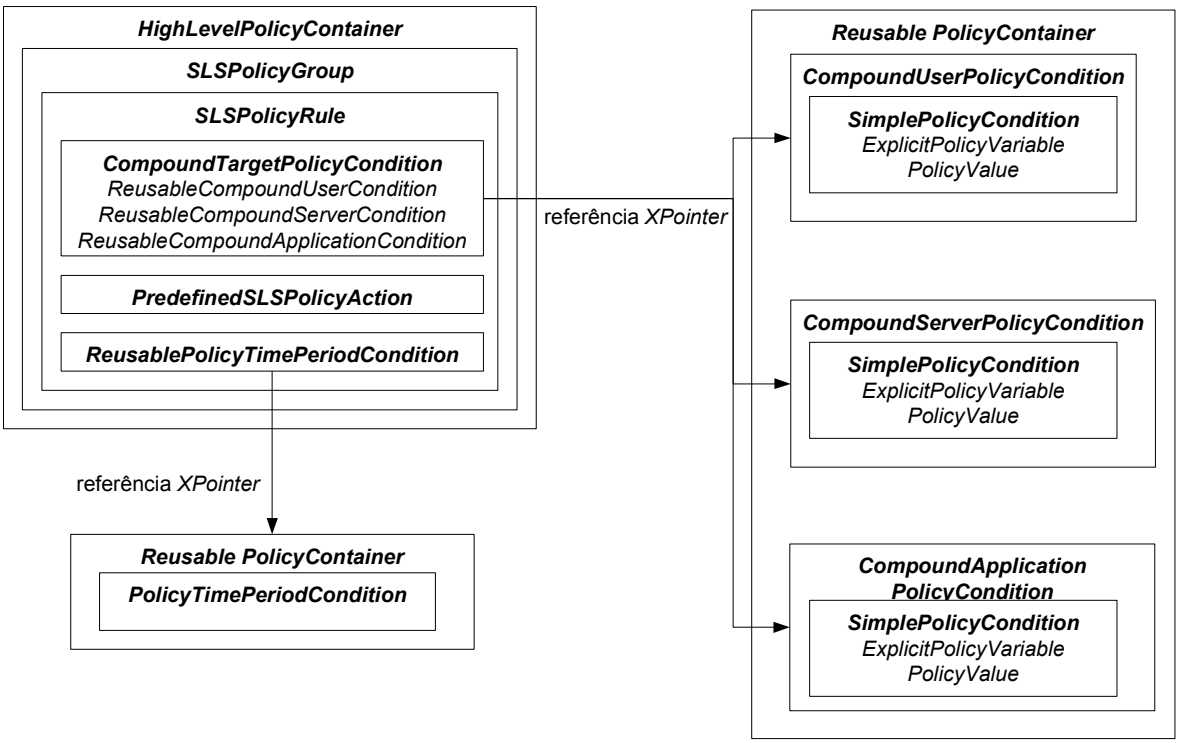

Figura 6.1: Estrutura de reuso para políticas de alto nível

O primeiro contêiner da estrutura é nomeado "*HighLevelPolicyContainer*". Neste contêiner são representados os grupos SLS <*SLSPolicyGroup*> que, por sua vez, contêm as regras SLS <*SLSPolicyRule*>. Em uma regra <*SLSPolicyRule*>, as condições <*CompoundTargetPolicyCondition*> são definidas pela semântica: "Usuários acessando Aplicações em Servidores". Para que as composições de usuários <*CompoundUserPolicyCondition*>, aplicações <*CompoundApplicationPolicyCondition*> e servidores

<*CompoundServerPolicyCondition*> possam ser reaproveitadas por várias <*CompoundTargetPolicyConditions*> estas informações são armazenadas em contêineres reutilizáveis (arquivos XML) <*ReusablePolicyContainer*>. A referência para elementos contidos em arquivos XML é possível a partir de expressões *XPointer* [W3C01]. Os períodos de tempo associados às políticas <*PolicyTimePeriodCondition*> também são formados por informações reutilizáveis. As ações <*PredefinedSLSPolicyAction*> não são indicadas por referências *XPointer*, pois o atributo "*PredefinedSLSName*", contido no próprio elemento, define o SLS selecionado para a regra. A figura abaixo apresenta um exemplo de políticas de negócio mapeadas em XML. O atributo "*SLSType*" do elemento <*SLSPolicyGroup*> indica o conjunto de tipos de serviço adotado no modelo. No caso, "Olímpico" corresponde a um grupo que possui três especificações de níveis de serviço: Bronze, Prata e Bronze.

```
<PolicyContainer Name="PoliticasDeNegocio">
  <SLSPolicyGroup SLSType="Olimpico" PolicyDecisionStrategy="2">
    <SLSPolicyRule Name="RegraBronze" Enabled="1" ConditionListType="1" ExecutionStrategy="2" Priority="1">
        … </SLSPolicyRule>
    <SLSPolicyRule Name="RegraPrata" Enabled="1" ConditionListType="1" ExecutionStrategy="2" Priority="2">
      <CompoundTargetPolicyCondition ConditionListType="1"  GroupNumber="1" ConditionNegated="false">
        <CompoundUserPolicyConditionInCompoundTargetPolicyCondition GroupNumber="1"
          ConditionNegated="false"
          PartComponent="./CUC.xml#xpointer(//CompoundUserPolicyCondition[@Name='CompUsr1']) " />
        < CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition GroupNumber="1"
          ConditionNegated="false"
          PartComponent="./CAC.xml#xpointer(//CompoundApplicationPolicyCondition[@Name='CompApp1']) " />
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition GroupNumber="1"
          ConditionNegated="false"
          PartComponent="./CSC.xml#xpointer(//CompoundServerPolicyCondition[@Name='CompSrv1']) " />
      </CompoundTargetCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="Prata" />
      <PolicyRuleValidityPeriod PartComponent="./Validade.xml#
        xpointer(//PolicyTimePeriodCondition[@Name='Periodo1'])" />
    </SLSPolicyRule>
    <SLSPolicyRule Name="RegraOuro" Enabled="1" ConditionListType="1" ExecutionStrategy="2" Priority="3">
        …</SLSPolicyRule>
  </SLSPolicyGroup>
</PolicyContainer>
```

Figura 6.2: Repositório XML para políticas de alto nível

O arquivo XML da figura 6.2 retrata bem as duas abordagens usadas no mapeamento das classes associativas. O relacionamento entre <*SLSPolicyGroup*> e <*SLSPolicyRule*> ocorre diretamente entre superior e subordinado, assim a associação *PolicySetComponent* fica implícita e seu atributo "*Priority*" é incorporado no elemento <*SLSPolicyRule*>. De modo análogo ocorrem os relacionamentos de <*SLSPolicyRule*>

com <*CompoundTargetPolicyCondition*> e com <*PredifinedSLSPolicyAction*>. No caso do relacionamento entre <*SLSPolicyRule*> e sua condição de período *PolicyTimePeriodCondition* existe reuso de informação, por isto a associação <*PolicyRuleValidityPeriod*> é representada explicitamente como nó filho de <*SLSPolicyRule*> e seu atributo *PartComponent* contem a referência *XPointer* que aponta para um específico elemento <*PolicyTimePeriodCondition*> armazenado em um contêiner reutilizável. De modo análogo ocorrem os relacionamentos entre <*CompoundTagetPolicyCondition*> e suas componentes <*CompoundUserPolicyCondition*>, <*CompoundApplicationPolicyCondition*> e <*CompoundServerPolicyCondition*>.

O elemento que representa o período de tempo no qual a regra é válida é mostrado na figura 6.3 e tem o seguinte significado:
"A regra deve ser aplicada de Segunda a Sexta-Feira, de 09h00 a 17h00, a partir da data de 01 de janeiro de 2004 e sem definição de data de término de validade".

```
<ReusablePolicyContainer Name="Validade">
   <PolicyTimePeriodCondition Name="Periodo1" TimePeriod="20040101T060000/THISANDFUTURE"
    DayOfWekkMask="01111100" TimeOfDayMask="T090000/T170000 />
    <!-- outros períodos de validade -->
</ReusablePolicyContainer>
```

Figura 6.3: Repositório XML para períodos de validade

6.2.2. Estrutura de reuso das informações das políticas de configuração

A figura 6.4 descreve a organização dos contêineres utilizados para armazenar as informações das políticas de configuração e o modo como são reutilizadas.

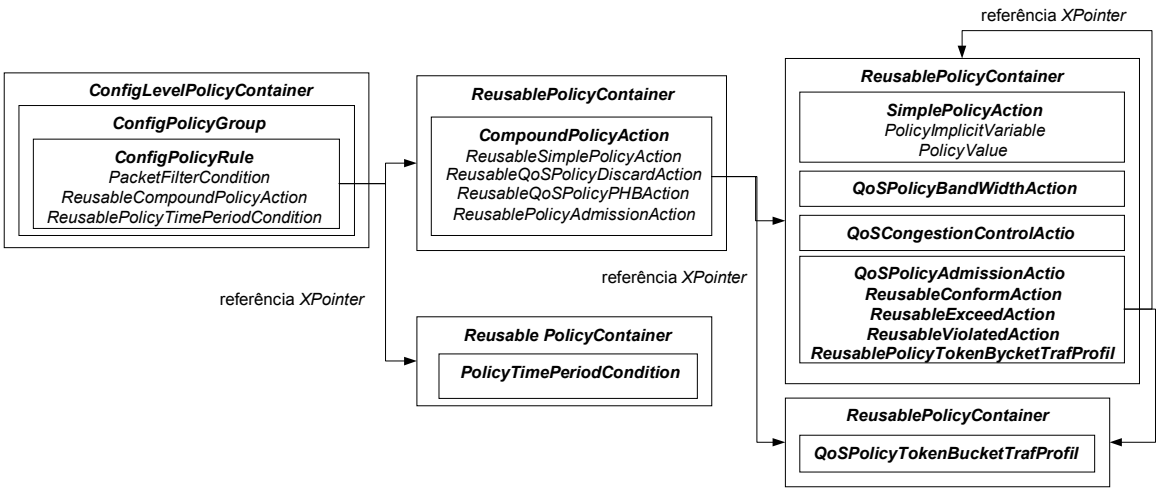

Figura 6.4: Estrutura do mapeamento XML para políticas de configuração

O primeiro contêiner da estrutura é nomeado "*ConfigLevelPolicyContainer*". Neste contêiner são representados os grupos <*ConfigPolicyGroup*> que contêm as regras <*ConfigPolicyRule*> correspondentes a configuração dos dispositivos que desempenham o mesmo papel na rede gerenciada. No exemplo da figura 6.5, cada <*ConfigPolicyGroup*> representa a configuração dos dispositivos em uma sub-rede específica, dentro de uma rede empresarial *DiffServ*. No caso, somente a política de configuração que define o nível de serviço Prata para o Departamento de Engenharia está detalhada.

A condição <*PacketFilterCondition*> é gerada durante a execução do processo de tradução e não é reutilizável. No mapeamento das políticas de configuração as composições de ações <*CompoundPolicyAction*> e os períodos de tempo <*PolicyTimePeriodCondition*> são reutilizáveis e por isto são definidos por referências *XPointer* dentro de <*ConfigPolicyRule*>

```
<PolicyContainer Name="PoliticasDeConfiguracao">
   <ConfigPolicyGroup ConfigName="ConfigQoSDepComercial" PolicyDecisionStrategy="1"> …
   </ConfigPolicyGroup>
   <ConfigPolicyGroup ConfigName="ConfigQoSDepEngenharia" PolicyDecisionStrategy="1">
        <ConfigPolicyRule Enabled="1" ConditionList Type="1" Priority="2">
          <PacketFilterCondition FilterEvaluation="4" GroupNumber="2" ConditionNegated="false">
               <IPHeadersFilter IsNegated="False" HdrIPVersion="4" HdrSrcAddress="0.0.0.0" HdrSrcMask="0"
                 HdrDestAddress="10.0.4.1" HdrDestMask="24" Direction="3"/>
          </PacketFilterCondition>
          <!-- outras condições de filtros -->
          <PolicyRuleValidityPeriod
            PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='Periodo1'])" />
          <ConfigQoSActionInConfigPolicyRule PartComponent="./QoSOlimpic.xml#
            xpointer(//CompoundPolicyAction[@name='AcaoPrata'])" />
        </ConfigPolicyRule>
     </ConfigPolicyGroup>
     <!-- outros grupos de configuração-->
</PolicyContainer>
```

Figura 6.5: Políticas de configuração geradas pelo processo de tradução

A figura 6.6 mostra as composições de ações <*CompoundPolicyAction*> que representam os níveis de serviços (SLSs) pré-definidos no sistema. Nas composições, a ordem de execução das ações é definida como obrigatória no atributo "*SequencedActions*=1" e seguem a indicação do atributo "*ActionOrder*" na associação <*PolicyActionInPolicyAction*>. Desta forma, no exemplo da figura 6.7, o tráfego selecionado por <*PacketFilterCondition*> é primeiramente medido e os pacotes são marcados com DSCP que indicam conformidade, excesso ou violação em relação ao perfil de tráfego definido. Posteriormente os pacotes são enfileirados e, finalmente, encaminhados pelo escalonador correspondente.

```
<ReusablePolicyContainer Name="SLSsOlimpicos">
   <CompoundPolicyAction Name="AcaoBronze" SequencedActions="1"
     ExecutionStrategy="2"> … </CompoundPolicyAction>
   <CompoundPolicyAction Name="AcaoPrata" SequencedActions="1" ExecutionStrategy="2">
     <PolicyActionInPolicyAction ActionOrder="1" PartComponent=
       "./QPIMAction.xml#xpointer(//QoSPolicyPoliceAction[@Name='Police256k'])"/>
       <PolicyActionInPolicyAction ActionOrder="2" PartComponent=
       "./QPIMAction.xml#xpointer(//QoSPolicyCongestionControlAction[@Name='Fila2'])" />
       <PolicyActionInPolicyAction ActionOrder="3" PartComponent=
       "./QPIMAction.xml#xpointer(//QoSPolicyBandwidthAction[@Name='BW2'])" />
   </CompoundPolicyAction>
   <CompoundPolicyAction name="AcaoOuro" SequencedActions="1" ExecutionStrategy="2">…
   </CompoundPolicyAction>
</ReusablePolicyContainer>
```

Figura 6.6: Repositório XML para os níveis de serviço pré-definidos no sistema

Todas as ações QPIM: <*QoSPolicyPoliceAction*>, <*QoSPolicyCongestionControlAction*>, <*QoSPolicyBandwidthAction*>, <*SimplePolicyAction*> são armazenadas no mesmo contêiner reutilizável (fig. 6.7) e descrevem os parâmetros dos SLSs pré-definidos. A estrutura da figura 6.5 indica que as expressões *XPointer* podem referenciar o próprio contêiner porque <*QoSPolicyPoliceAction*> reutiliza as ações determinadas por <*PolicyConformAction*>, <*PolicyExceedAction*> e <*PolicyViolateAction*> que são executadas de acordo com o resultado obtido na comparação com <*QoSPolicyTokenBucketTrfcProf*>.

```
<ReusablePolicyContainer Name="AcoesQPIM">
  <QoSPolicyPoliceAction Name="Police256k" qpAdmissionScope="0">
    <QoSPolicyTrfcProfInAdmissionAction Dependent="./QPIMAction.xml#
      xpointer(//QoSPolicyTokenBucketTrfcProf[@Name='TB256k'])" />
    <PolicyConformAction Dependent="./ QPIMAction.xml #
      xpointer(//SimplePolicyAction[@Name='DSCPAF21'])" />
    <PolicyExceedAction Dependent="./ QPIMAction.xml #
      xpointer(//SimplePolicyAction[@Name='DSCPAF22'])" />
    <PolicyViolateAction Dependent="./ QPIMAction.xml #
      xpointer(//SimplePolicyAction[@Name='DSCPAF23'])" />
  </QoSPolicyPoliceAction>
  <QoSPolicyCongestionControlAction Name="Fila2" qpQueueSizeUnits="1" qpQueueSize="15"
    qpDropMethod="3" qpDropThresholdUnits="0" qpMinThresholdValue="30" qpMaxThresholdValue="45" />
  <QoSPolicyBandwidthAction Name="BW2" qpBandwidthUnits="1" qpMinBandwidth="25" />
  <SimplePolicyAction Name="DSCPAF21">
    <PolicyDSCPVariable />
    <PolicyIntegerValue IntegerList="AF21" />
  </SimplePolicyAction>
  …<!-- outras ações QPIM -->
</ReusablePolicyContainer>
```

Figura 6.7: Repositório XML para ações QPIM pré-definidas no sistema

O repositório da figura 6.8 representa os parâmetros de perfis de tráfego que são reutilizados pelas ações de policiamento <*QoSPolicyPoliceAction*>.

```
<ReusablePolicyContainer Name="TokenBucket">
  <QoSPolicyTokenBucketTrfcProf  Name="TB256k"
    qpTBRate="256" qpTBNormalBurst="64" qpTBExcessBurst="32" />
  <QoSPolicyTokenBucketTrfcProf  Name="TB512k"
    qpTBRate="512" qpTBNormalBurst="128" qpTBExcessBurst="64" />
  <QoSPolicyTokenBucketTrfcProf  Name="TB1M"
    qpTBRate="1024" qpTBNormalBurst="256" qpTBExcessBurst="128"/>
</ReusablePolicyContainer>
```

Figura 6.8: Repositório XML para parâmetros *Token Bucket*

6.3. Processo para Tradução das Políticas de Alto Nível em Políticas de Configuração

Neste algoritmo de tradução, é assumido que:

- Cada regra *SLSPolicyRule* possui 1 ou várias condições *CompoundTargetPolicyCondition*, exatamente 1 ação *PredifinedSLSPolicyAction,* 0 ou vários períodos de validade *TimePeriodCondition*.
- Cada condição *CompoundTargetPolicyCondition* contem 0 ou 1 condição composta de usuários, 0 ou 1 condição composta de aplicações e 0 ou 1 condição composta de usuários.
- Todas as instâncias de *SLSPolicyRule, CompoundTargetPolicyCondition, CompoundUserPolicyCondition, CompoundApplicationPolicyCondition* e *CompoundServerPolicyCondition* sempre possuem o atributo *ConditionListType*="1", ou seja seguem o formato DNF.

Tabela 6.1: Algoritmo de tradução das políticas de negócio para políticas de configuração

1	Fazer com que *SLSPolicyGroups* seja a lista dos grupos de políticas de alto nível contidos no contêiner *HighLevelPolicyContainer*. • *SLSPolicyGroups*(cont: *PolicyContainer*)={sg: *SLSPolicyGroup* | sg ⊂ *BusinessLevelPolicyContainer*}
2	Para cada *SLSPolicyGroup* obtido no passo anterior, determinar a lista de regras associadas ao grupo. • *SLSPolicyRules*(sg: *SLSPolicyGroup*)={sr: *SLSPolicyRule* | sr ∈ sg}
3	Para cada regra *SLSPolicyRule* obtida no passo anterior, determinar as composições *(Targets)* de condições, o período de validade e a ação associados a ela. • *CT*(sr: *SLSPolicyRule*)={ct: *CompoundTargetPolicyCondition* | ct ∈ sr} • *Validity*(sr: *SLSPolicyRule*)={v: *PolicyTimePeriodCondition* | v ∈ sr} • *PredefinedSLSAction*(sr: *SLSPolicyRule*)={sa: *String* | sa ∈ sr}
4	Para cada *CompoundTargetPolicyCondition* obtida no passo anterior, obter a condição composta de usuários, servidores e aplicações que estão associadas a ela. • *CompoundUserCondition*(ct: *CompoundTargetPolicyCondition*)={cuc: *CompoundUserPolicyCondition* | suc ∈ ct}

	• *CompoundServerCondition*(ct: *CompoundTargetPolicyCondition*)={csc: *CompoudServerPolicyCondition* | ssc ∈ ct} • *CompoundApplicationCondition*(ct: *CompoundTargetPolicyCondition*)={cac: *CompounApplicationPolicyCondition* | sac ∈ ct}
5	Para cada condição composta de usuários, servidores e aplicações obtidas no passo anterior, determinar os subconjuntos das condições simples que possuem o mesmo *GroupNumber* e obter o par variável, valor das condição simples que formam o subconjunto. • *SubC*(gn: *GroupNumber*)={subcc: *Vector* | subc ⊃ *SimplePolicyConditions* ∈ a um mesmo *GroupNumber*} • *VarVal*(subc: *Vector*)={vv: Vector | vv ⊃ (*PolicyExplicitVariable*, *PolicyStringValue*) ∈ subc}
6	A partir do vetor VarVal do passo anterior é formada a expressão de busca conforme a sintaxe *XPAth.* Se VarVal está associado com uma condição composta de usuários, o resultado da busca retorna a lista de endereços IP dos usuários que atendem os parâmetros de filtragem. Se, no passo 3, *CompoundUserCondition* = ∅, então o endereço "coringa" 0.0.0.0/0 é retornado. E se toda uma sub-rede é selecionada pela expressão *XPath*, o endereço da sub-rede é retornado. • *IPusr*(busca: *XPAth*)={ipusr: *InetAdress* | ipusr ∈ a um objeto CIM *Person* que atende aos requisitos da busca} • Se *IPusr*(∅), então ipusr=0.0.0.0 • Se *IPusr*(sub-rede), então ipusr=IP da sub-rede Se VarVal está associado com uma condição composta de servidores, o resultado da busca retorna a lista de endereços IP dos servidores que atendem os parâmetros de filtragem. Se, no passo 3, *CompoundServerCondition* = ∅ e *CompoundApplicationCondition* ≠ ∅, então é retornada a lista com o endereço IP de todos os servidores que hospedam as aplicações selecionadas. Se, no passo 3, *CompoundServerCondition* = ∅ e *CompoundApplicationCondition* = ∅, então a lista dos endereços IP de todos os servidores do sistema é retornada. • *IPsrv* (busca: *XPAth*)={ipsrv: *InetAdress* | ipsrv ∈ a um objeto CIM

	UnitaryComputerSystem que atende aos requisitos da busca} • Se *IPsrv* = ∅ ∧ *Portapp* ≠ ∅, então *IPsrv* = {ipsrv: *InetAdress* | ipsrv ∈ a um objeto CIM *UnitaryComputerSystem* que hospeda, pelo menos, uma aplicação que atende ao requisito da busca *XPath* para aplicação} • Se *IPsrv* = ∅ ∧ *Portapp* = ∅, então *IPsrv* = {ipsrv: *InetAdress* | ipsrv ∀ objeto CIM *UnitaryComputerSystem* definido no sistema} Se VarVal está associado com uma condição composta de aplicações, o resultado da busca retorna a lista de vetores (porta,protocolo) das aplicações que atendem aos parâmetros da filtragem. Se, no passo 3, *CompoundApplicationCondition* = ∅, então a informação de porta não é descrita no cabeçalho IP. • Portapp(busc: *XPath*)={portapp: *Vector* | portapp ∈ a um objeto CIM *ApplicationSystem* ou CIM *InstalledProduct* que atende aos requisitos da busca} • Se Portapp = ∅, então portapp=∅
7	A partir das listas {ipusr}, {ipsrv} e {portapp}, obtidas no passo 6, formar a lista das possíveis tuplas {ipsrc, ipdst, portdst ou portsrc} de acordo com o atributo *"Direction"* da *CompoundTargetPolicyCondition* (ct) obtida no passo 3. Se *Direction* = 1 então ipsrc = ipusr, ipdst = ipsrv e portdst = portapp; Se *Direction* = 2 então ipsrc = ipsrv, ipdst = ipusr e portsrc = portapp; Se *Direction* = 3 então as duas tuplas anteriores são retornadas; • Tupla (ipsrc: *InetAdress*, ipdst: *InetAdress*, portsrc ou portdst: *Vector*)={pfc: *PacketFilterCondition* | ipsrc ∈ IPusr ou IPsrv, ipdst ∈ IPusr ou IPsrv, portsrc ou portdst ∈ Portapp}
8	Determinar a(s) sub-rede(s) que são afetadas por cada filtro de pacote IP, obtidos através da lista de tuplas no passo anterior. • *SubNets*(ipsrc: *InetAdress,* ipdst: *InetAdress*)={sn: *NetworkAdress* | sn ⊃ IPsrc ∨ sn ⊃ IPdst}
9	Determinar a coleção de papéis associada às sub-redes obtidas no passo anterior. • *SubNetRoles*(sn: *NetworkAdress*)={rc: *RoleCollection* | rc está associada com sn}
10	Criar o grupo de configuração *ConfigPolicyGroup* correspondente à coleção de papéis obtida no passo anterior. Se o grupo já existir, pular para o próximo passo.

	• *ConfigPolicyGroup*(rc: *RoleCollection*)={cg: *ConfigPolicyGroup* | cg ⊂ *ConfigLevelPolicyContainer*}
11	Criar um elemento <*PacketFilterCondition*> dentro da regra *ConfigPolicyRule* associada ao grupo de configuração do passo anterior. Mapear as informações da tupla usada no passo 8 para um elemento <*IPHeadersFilter*>. • *PacketFilterCondition*(ipsrc: *InetAdress*, ipdst: *InetAdress*, portdst: *Vector*)={pfc: *PacketFilterCondition* | pfc ⊂ *ConfigPolicyRule*}
12	Criar um elemento <*PolicyRuleValidityPeriod*> dentro da mesma regra usada no passo 11. Copiar a mesma referência *XPointer* usada por *Validity* no passo 3. • *PolicyRuleValidityPeriod* (ref: *XPointer*)= {prv: *PolicyTimePeriodCondition* | prv ⊂ *ConfigPolicyRule*}
13	Criar um elemento <*ConfigQoSActionInConfigPolicyRule*> dentro da mesma regra usada no passo 11. Construir uma expressão *XPointer* para referenciar a *CompoundPolicyAction* correspondente, a partir da *String* (sa) obtida no passo 3. • *ConfigQoSActionInConfigPolicyRule*(ref: *XPointer*)= {ca: *CompoundPolicyAction* | ca ⊂ *ConfigPolicyRule*}
14	Verificar se a regra *SLSPolicyRule* do passo 3 possui regra aninhada. • *NestedRule*(sr: *SLSPolicyRule*)={nr: *SLSPolicyRule* | nr ⊂ sr} Se nr ≠ ∅, retornar ao passo 3 fazendo com que sr = nr.

6.4 Mapeamento *DiffServ* PIB em XML

Dentro da proposta deste trabalho, as classes (PRCs) *Framework* PIB e *DiffServ* PIB, apresentadas nas figuras 3.14 e 3.15 respectivamente, são mapeadas em uma estrutura de elementos XML. Esta estrutura é usada tanto na seção <*ActivePEP*> do arquivo XML do PDP como na seção <*Pib*> do arquivo XML do PEP.

Devido às características das PRCs, a estratégia usada no mapeamento dos módulos PIB em XML difere daquela descrita para o mapeamento dos modelos de representação de políticas e segue o esquema mostrado abaixo:

<Prc oid=“ ”> → Início do elemento que representa uma classe ou tabela com seu respectivo OID.

<Prid id=“ ”> → Nó filho que unicamente identifica uma instância da Prc através do atributo id.

<Atributo1 type=“ ”>valor**</Atributo1>**

→ Seqüência de nós filhos de Prid que representam os atributos da classe Prc correspondente, conforme definido pelas RFCs. O atributo “type” é utilizado para representar o tipo do valor de cada atributo (*Integer*, *String*,...).

⋮

<AtributoN type=“ ”>valor**</AtributoN>**

Figura 6.9: Estrutura do mapeamento XML para as PRCs da PIB

A estrutura da PIB implementada neste trabalho possui cinco grupos de classes: <*BasePIB*>, <*DeviceCapabilities*>, <*ClassifierGroup*>, <*DsCapabilities*> e <*DsPolicy*>. Na figura 6.10 é apresentada a seção <*ActivePEP*> do arquivo XML do PDP, onde estão representados os elementos da PIB correspondente a configuração do cliente “*PepDiffServ1*” . As classes (PRCs) comum a todos os tipos de clientes, correspondentes aos grupos <*BasePIB*>, <*DeviceCapabilities*> e <*ClassifierGroup*>, são definidas no módulo *Framework PIB* [SAH03] e as classes (PRCs) específicas para os clientes que implementam os mecanismos DiffServ, correspondente aos grupos <*DsCapabilities*> e <*DsPolicy*>, são definidas no módulo *DiffServ PIB* [RFC3317].

```
- <ActivePep>
 - <pep id="PepDiffServ1">
   - <Pib>
     - <BasePib>
       + <PrcSupport oid="1.3.6.1.2.2.2.1.1">
       + <PibIncarnation oid="1.3.6.1.2.2.2.1.2">
       + <DeviceId oid="1.3.6.1.2.2.2.1.3">
      </BasePib>
     - <DeviceCapabilities>
       + <CapabilitiesSet oid="1.3.6.1.2.2.2.2.1">
       + <InterfaceRoleCombo oid="1.3.6.1.2.2.2.2.3">
      </DeviceCapabilities>
     - <ClassifierGroup>
       + <IPFilter oid="1.3.6.1.2.2.2.3.2">
      </ ClassifierGroup>
     - <DsCapabilities>
       + <dsIfClassificationCaps oid="1.3.6.2.2.4.1.2">
       + <dsIfMeteringCaps oid="1.3.6.2.2.4.1.3">
       + <dsIfAlgDropCaps oid="1.3.6.2.2.4.1.4">
       + <dsIfQueueCaps oid="1.3.6.2.2.4.1.5">
       + <dsIfSchedulerCaps oid="1.3.6.2.2.4.1.6">
       + <dsIfMaxRateCaps oid="1.3.6.2.2.4.1.7">
       + <dsIfElmDepthCaps oid="1.3.6.2.2.4.1.8">
       + <dsIfElmLinkCaps oid="1.3.6.2.2.4.1.9">
      </ DsCapabilities>
     - <DsPolicy>
       + <dsDataPath oid="1.3.6.2.2.4.2.1">
       + <dsClfr oid="1.3.6.2.2.4.2.2">
       + <dsClfrElement oid="1.3.6.2.2.4.2.3">
       + <dsMeter oid="1.3.6.2.2.4.2.4">
       + <dsTBParam oid="1.3.6.2.2.4.2.5">
       + <dsAction oid="1.3.6.2.2.4.2.6">
       + <dsDscpMarkAct oid="1.3.6.2.2.4.2.7">
       + <dsAlgDrop oid="1.3.6.2.2.4.2.8">
       + <dsMQAlgDrop oid="1.3.6.2.2.4.2.9">
       + <dsRandomDrop oid="1.3.6.2.2.4.2.10">
       + <dsQTable oid="1.3.6.2.2.4.2.11">
       + <dsScheduler oid="1.3.6.2.2.4.2.12">
       + <dsMinRate oid="1.3.6.2.2.4.2.13">
       + <dsMaxRate oid="1.3.6.2.2.4.2.14">
     - </DsPolicy>
   </Pib>
 </pep>
</ActivePep>
```

Figura 6.10 Grupos e classes da PIB implementada

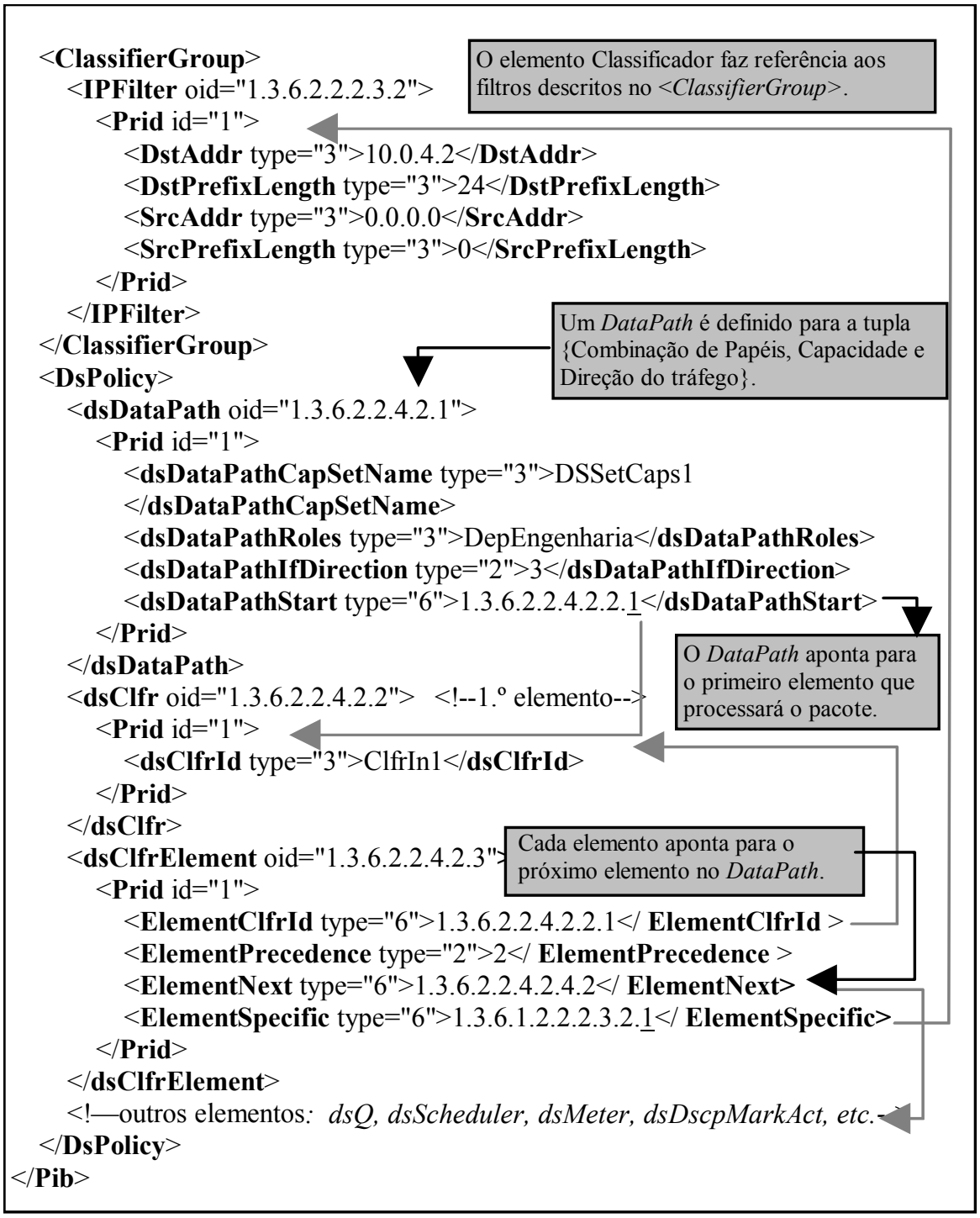

Figura 6.11: Elementos da *DiffServ* PIB

No grupo <*ClassifierGroup*> (fig. 6.11) está a PRC *IPFilter*, que contém instâncias de filtros para descrever o tráfego que irá receber um específico (pré-definido) nível de serviço. Ou seja, estas instâncias correspondem aos elementos

<*PacketFilterCondition*> do modelo de políticas de configuração (CLPM). Um pacote receberá o tratamento correspondente ao primeiro classificador que atender suas características. Por isto, o elemento <*ElementePrecedence*> da classe *dsClfrElement* define a prioridade do filtro IP referenciado, esta informação é obtida a partir do atributo *PolicySetComponent.Priority* que representa a prioridade das regras (*SLSPolicyRule/ConfigPolicyRule)* dentro da agregação dos grupos (*SLSPolicyGroup/ConfigPolicyGroup*), e define a ordem para "filtragem" dos pacotes. No caso do grupo de SLSs Olímpicos, pode-se definir que os filtros correspondentes ao tratamento Ouro possuem prioridade = 1, os filtros correspondentes ao tratamento Prata possuem prioridade = 2 e os filtros correspondentes ao tratamento Bronze possuem prioridade = 3.

O nível de serviço correspondente, ou "ações", é descrito através de instâncias das classes definidas no grupo <*DsPolicy*>. O <*DsPolicy*> é formado por uma seqüência ordenada de elementos funcionais *DiffServ* que processam os pacotes que ingressam ou e egressam da interface gerenciada. Os valores das instâncias dos elementos *DiffServ* são obtidos a partir das informações das instâncias QPIM que definem os SLSs pré-definidos no sistema.

6.5. Conclusão

No presente capítulo foi detalhada a proposta de implementação para a arquitetura de gerenciamento *DiffServ* baseado em políticas de negócio apresentado no capítulo 5. Nesta implementação, optou-se pelo uso de arquivos XML para repositório das informações, principalmente, devido a sua estrutura de representação que possibilita a assimilação intuitiva das informações durante a leitura humana. Esta estratégia, inclusive, facilita o entendimento dos complexos modelos (PCIMe, QPIM, *Framework* PIB, *DiffServ* PIB) usados para representação das políticas nos diferentes níveis de abstração existentes dentro da arquitetura. Para o mapeamento das informações em XML, foram consideradas as diretrizes do IETF e DMTF para armazenamento dos objetos CIM em LDAP. O conceito de reaproveitamento de informações é um fundamento básico dentro deste trabalho, para isto foram utilizadas referências formadas por expressões *Xpointer*.

Também foi descrito o algoritmo executado para transformação das políticas de negócio em políticas de configuração de forma independente de dispositivo.

Capítulo 7

Distribuição das Políticas para os Dispositivos de Rede

7.1. Introdução

O modelo de políticas de alto nível proposto no capítulo 5, permite a especificação de níveis de serviço dependente de horário, ou seja, um SLS pode não ser permanentemente válida e portanto deve ser praticado apenas durante determinados intervalos de tempo. Mas a *DiffServ* PIB não inclui classes para representação de tempo porque, normalmente, os dispositivos de rede (roteadores *e switches*) não possuem capacidade para processar esta informação. Então, mecanismos adicionais precisam ser implementados para que a estrutura definida pelo IETF suporte o gerenciamento de QoS dependente de horário.

Este capítulo descreve o processo de decisão executado pelo servidor de políticas (PDP), que inclui as seguintes etapas: seleção das políticas relevantes para um determinado dispositivo; agenda de configuração dos dispositivos; conversão das políticas em instâncias da *DiffServ* PIB, considerando um conjunto determinado de capacidades; e, finalmente, a distribuição das instâncias da *DiffServ* PIB para os dispositivos utilizando o protocolo COPS/COPS-PR.

Combinando os recursos do protocolo COPS/COPS-PR e da *DiffServ* PIB, três estratégias distintas são discutidas para a atualização dinâmica que corresponde à agenda de configuração dos dispositivos de rede.

7.2. Processo de Decisão Executado pelo Servidor de Políticas (PDP)

O processo de decisão é executado quando o PDP recebe uma requisição de aprovisionamento de configuração enviada por um PEP. Os parâmetros de entrada para este processo são os papéis das interfaces gerenciadas e as suas capacidades, a partir destas informações o servidor de políticas executa três tarefas: 1) seleciona as políticas pertinentes ao papel desempenhado pelas interfaces do cliente (PEP); 2) verifica o período de validade de cada regra e monta a grade horária das configurações; 3) converte as informações de configuração independentes de dispositivo em instâncias da *DiffServ* PIB.

7.2.1. Seleção das políticas de configuração

Para selecionar as políticas que são relevantes para configuração de um determinado dispositivo, o PDP realiza uma sucessão de buscas (formadas por expressões *XPath*) nos repositórios do sistema, utilizando como parâmetro inicial o papel das interfaces do PEP. O diagrama de seqüência da figura 7.1 ilustra as etapas que compõem o processo de seleção das políticas.

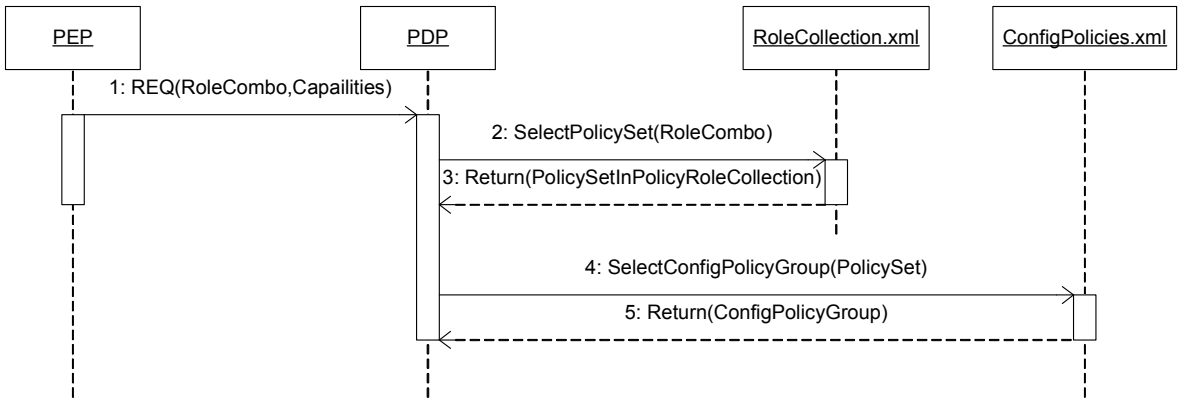

Figura 7.1: Diagrama de seqüência para o PDP selecionar as políticas de configuração.

7.2.2. Grade horária e atualização dinâmica das configurações

O modelo de política de alto nível (HLPM), discutido no capítulo 5, suporta a representação de SLSs dependentes de horário, isto é, uma especificação de nível de serviço pode não ser permanentemente válida e portanto deve ser praticada apenas durante determinados intervalos de tempo. A *DiffServ* PIB não inclui classes para representação de tempo porque, normalmente, os dispositivos de rede (roteadores *e*

switches) não possuem capacidade para processar esta informação. Assim, fica sob responsabilidade do PDP o gerenciamento da validade das configurações. Dentro deste trabalho, o PDP, após selecionar as políticas relevantes para o dispositivo, executa um método para determinar as encarnações da PIB (PIB *Incarnations*) correspondentes à grade horária das regras que compõem uma configuração. Cada encarnação representa uma instância da PIB disjunta e independente. Na atual implementação, a grade horária é representada por vinte e quatro intervalos de uma hora e as configurações são geradas diariamente do lado do PDP. A tabela 7.1 ilustra a implementação do método, utilizando o intervalo entre 08h00 e 12h00. As regras R1 e R2 são válidas entre 08h00 e 12h00. As Regras R3 e R4 são válidas entre 09h00 e 11h00.

Tabela 7.1: Grade horária das encarnações da PIB.

8h	9h	10h	11h	12h
R1	R1	R1	R1	
R2	R2	R2	R2	
	R3	R3		
	R4	R4		
Encarnação0	Encarnação1		Encarnação0	

Para que as diferentes configurações (encarnações da PIB) da grade horária sejam transmitidas ao PEP existem, fundamentalmente, três abordagens:

1) Uma encarnação completa da PIB é enviada ao PEP toda vez que a configuração do dispositivo precisa ser alterada: inicialmente, apenas a encarnação da PIB válida no horário da requisição de aprovisionamento é transmitida ao PEP. Depois disto, quando a validade da encarnação expira, outra encarnação é enviada pelo PDP através de uma mensagem de decisão não solicitada. Ao receber a nova encarnação o PEP substitui completamente a anterior. O diagrama de seqüência da figura 7.2 demonstra esta abordagem. O mesmo identificador de estado do cliente (*Client-Handle*) é usado em todas as mensagens de decisão (DEC 7, 11 e 15) porque apenas uma encarnação da PIB é mantida no PEP.
2) Múltiplas encarnações da PIB são transmitidas durante o processo inicial de aprovisionamento: quando o PEP solicita seu aprovisionamento inicial, o PDP transmite todas as encarnações representando um determinado período da grade

horária (na implementação atual “24 horas”). Somente a encarnação válida no horário da requisição de aprovisionamento é ativada e todas as outras ficam desativadas. Quando a validade da configuração expira, o PDP envia uma mensagem curta de decisão para troca do contexto no PEP (i.e. a antiga encarnação é desativada e a encarnação válida para o novo período é ativada). Como cada encarnação da PIB é uma instância disjunta e independente, diferentes identificadores de estado (*Client-Handle*) devem ser utilizados para endereçar cada contexto. O identificador (*Handle*) sempre é gerado pelo PEP [DUR00] por isso a especificação COPS-PR [CHA01] define um novo *flag* de decisão (0x02) que permite o PDP comandar o PEP para abrir uma nova requisição de estado ou apagar uma já existente. No diagrama 7.3, as mensagens *DEC_Null* (10 e 14) são utilizadas pelo PDP para comandar o PEP a solicitar novas requisições.

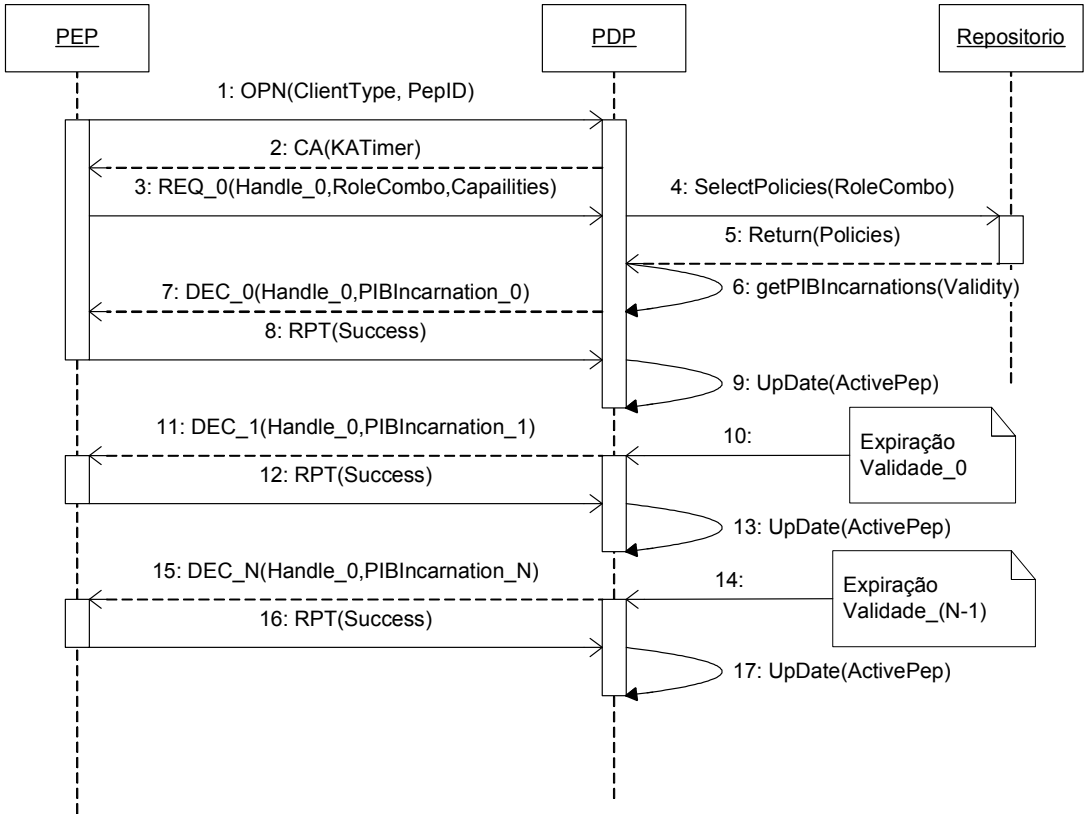

Figura 7.2: Diagrama de seqüência para completa substituição da encarnação da PIB em toda atualização de configuração.

3) Apenas as informações das classes e instâncias (PRCs e PRIs) modificadas são enviadas para alteração da configuração no PEP. Conforme mostrado no diagrama da figura 7.4, uma encarnação completa da PIB, válida no horário da requisição de aprovisionamento, é transmitida ao PEP na decisão inicial (DEC 7) e quando a

validade da configuração expira, uma mensagem de decisão (DEC 11 e 15) transporta as informações para remoção das PRCs e PRIs que não tem mais validade e instalação das novas PRCs e PRIs necessárias.

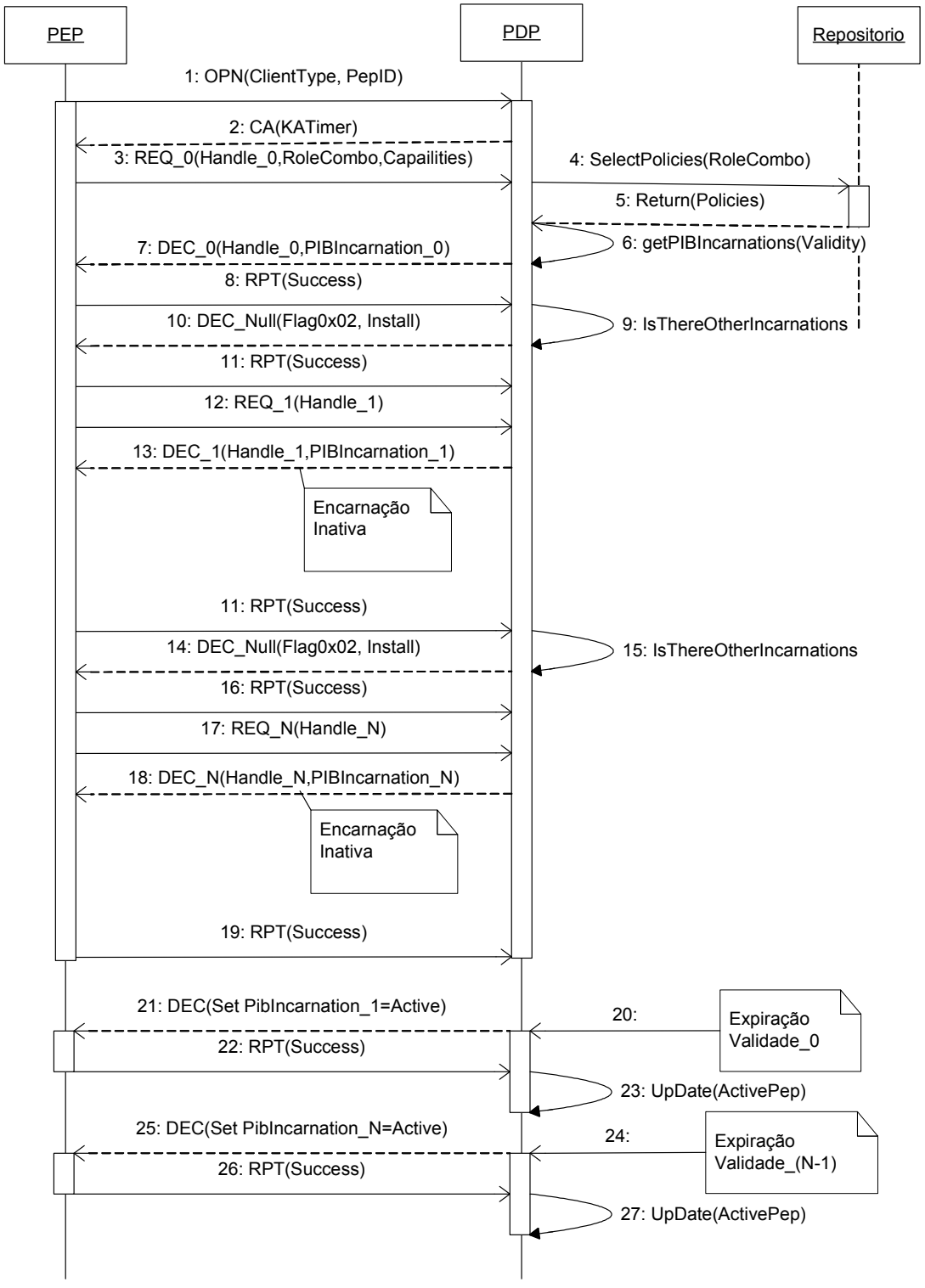

Figura 7.3: Diagrama de seqüência para utilização de múltiplas encarnações no PEP.

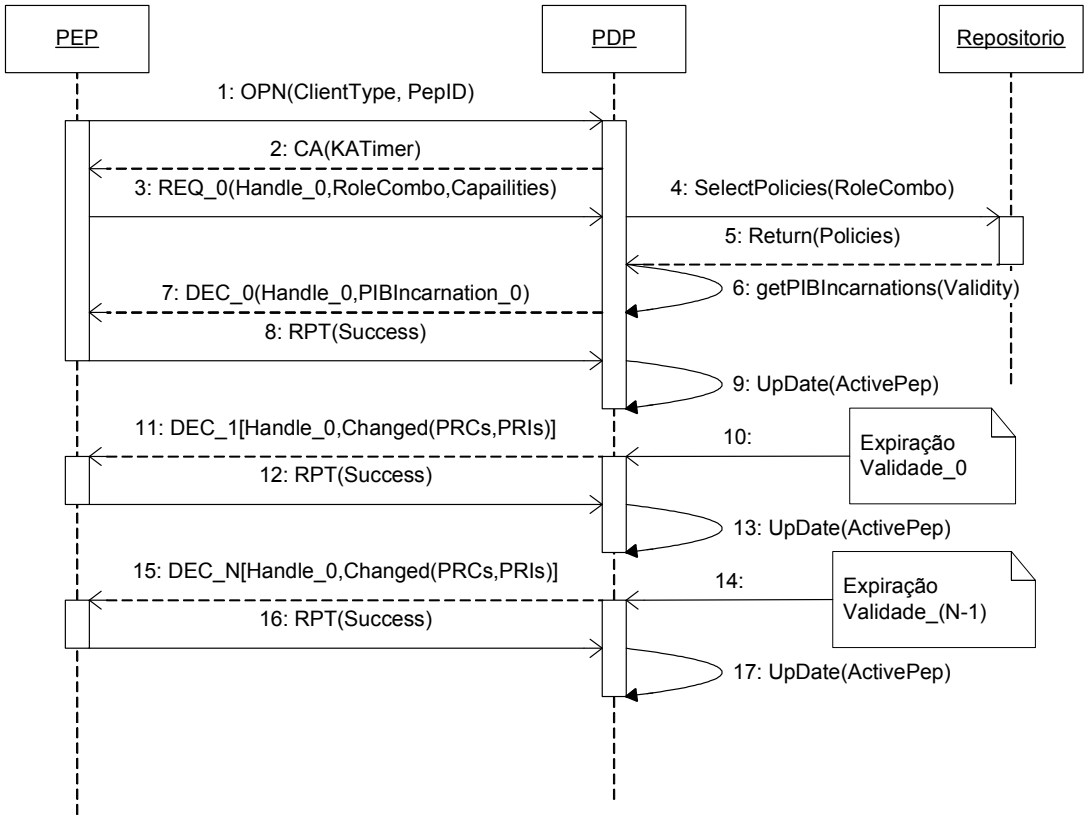

Figura 7.4: Diagrama de seqüência para atualização de PRCs e PRIs no PEP.

Cada uma das abordagens descritas possui vantagens e desvantagens, conforme sintetizado nas tabelas 7.2, 7.3 e 7.4. A escolha da implementação mais eficaz depende das características específicas do ambiente gerenciado.

Tabela 7.2: Prós e contras da atualização completa de uma encarnação da PIB no PEP.

Atualização completa de uma encarnação da PIB no PEP	
Prós	Contras
- Processo simplificado para configuração inicial dos dispositivos. Somente a encarnação válida para o período é transmitida. - Espaço para armazenamento de configuração no PEP é igual a uma encarnação da PIB.	- O processamento de atualização do PEP é mais demorado. A mensagem de decisão montada pelo PDP inclui as informações de uma encarnação completa. - As mensagens de decisão que alteram completamente uma encarnação da PIB geram maior tráfego na rede.

Tabela 7.3: Prós e contras da utilização de múltiplas encarnações da PIB no PEP.

Utilização de múltiplas encarnações da PIB no PEP	
Prós	Contras
- O processamento para atualização do PEP é mais rápido. A mensagem de decisão montada pelo PDP inclui apenas a encarnação da PIB que deve ser ativada. - As mensagens de decisão que alteram o contexto de configuração geram pequeno tráfego na rede.	- Processo para configuração inicial dos dispositivos é mais complexo. O PEP é forçado a solicitar um número de requisições de configuração igual à quantidade disponível de encarnações da PIB. - Espaço para armazenamento de configuração no PEP é proporcional a quantidade de encarnações da PIB transmitidas.

Tabela 7.4: Prós e contra para somente atualizar as PRCs e PRIs necessárias.

Somente atualização das PRCs e PRIs modificadas	
Prós	Contras
- Processo simplificado para configuração inicial dos dispositivos. Somente a encarnação válida para o período é transmitida. - Reuso de informações previamente instaladas. - Espaço para armazenamento de configuração no PEP é igual a uma encarnação da PIB. - As mensagens de decisão que alteram as PRCs e PRIs necessárias geram um tráfego na rede menor do no caso da atualização completa de uma encarnação da PIB.	- Durante o processamento de atualização do PEP é necessário que o PDP verifique as diferenças entre as PRCs e PRIs da nova configuração e da anterior.

O Capítulo 8 apresenta uma avaliação numérica para as diferentes abordagens de atualização de configuração utilizando um roteador Linux.

7.2.3. Conversão das políticas de configuração para instâncias na *DiffServ* PIB

Para converter as políticas de configuração independentes de dispositivos em instâncias (PRIs) da *DiffServ* PIB é necessário que sejam consideradas as capacidades específicas de cada dispositivo, uma vez que diferentes mecanismos (algoritmos de descarte, escalonadores, etc.) podem ser implementados na execução das ações *DiffServ*. Considerando esta premissa, verificou-se que a implementação de um algoritmo de conversão genérico, capaz de suportar qualquer combinação de capacidades, não é uma tarefa simples. Por isso foi adotado o conceito de "bibliotecas de conversão". O parâmetro que determina a invocação das bibliotecas é o conjunto de capacidades do dispositivo. Esta estratégia possibilita o aprimoramento da ferramenta de gerenciamento através da inserção de novas bibliotecas no sistema.

Dentro deste trabalho, foi desenvolvida uma biblioteca de conversão que atende o seguinte conjunto de capacidades:

- Classificação: classificador baseado em IP de origem, IP de destino, Protocolo, Porta de origem, Porta de destino e DSCP.
- Medição: medidor *srTCM* (*single rate Three Color Meter*) [HEI99a]. Este tipo de medidor é definido por três atributos: CIR (*Committed Information Rate*), CBS (*Committed Burst Size*) e EBS (*Excess Burst Size*). Como resultado desta medição os pacotes são marcados como normal (*"green"*) quando o parâmetro CBS não é ultrapassado, excesso (*"yellow"*) quando é ultrapassado o parâmetro CBS mas não o EBS e violação (*"red"*).
- Algoritmo de Descarte: descarte randômico. O algoritmo randômico é capaz de implementar as disciplinas RED [FLO93] e suas derivações [WRED] e GRED [ALM99].
- Escalonamento: escalonadores que suportam o método WRR (*Weighted Round Robin*), ou seja, encaminham o tráfego de diferentes filas, servindo cada uma delas de acordo com seu "peso" ou largura de banda definida. As disciplinas CBQ (*Class Based Queue*) e HTB (*Hierarchical Token Bucket*) são exemplos de implementação do conceito WRR.

Considerando estas capacidades, o mapeamento das classes e atributos do modelo de políticas de configuração (CLPM) para as PRIs da *DiffServ* PIB é executado conforme mostrado nas tabelas a seguir:

Tabela 7.5: Mapeamento dos atributos da classe *IPHeadersFilter* para a PRC *IpFilterTable*

Classe *IPHeadersFilter*	**PRC *IpFilterTable***
IsNegated	*BaseFilterNegation*
HdrIPVersion	*IpFilterAddrType*
HdrSrcAdress	*IpFilterSrcAddr*
HdrSrcMask	*IpFilterSrcPrefixLength*
HdrDestAdress	*IpFilterDstAddr*
HdrDestMask	*IpFilterDstPrefixLength*
HdrProtocolId	*IpFilterProtocol*
HdrSrcPortStart	*IpFilterSrcL4PortMin*
HdrSrcPortEnd	*IpFilterSrcL4PortMax*
HdrDestPortStart	*IpFilterDstL4PortMin*
HdrDestPortEnd	*IpFilterDstL4PortMax*
HdrDSCP	*IpFilterDscp*

Tabela 7.6: Mapeamento dos atributos da classe *QoSPolicyTokenBucketTrfcProf* para a PRC *dsTBParamTable*

Classe *QoSPolicyTokenBucketTrfcProf*	**PRC *dsTBParamTable***
qpTBRate	*dsTBParamRate* (medidor 1)
qpTBNormalBurst	*dsTBParamBurstSize* (medidor 1)
qpTBExcessBurst	*dsTBParamBurstSize* (medidor 2)

A representação do medidor *srTCM* na PIB requer o uso de duas instâncias *dsTBParamTable.* A primeira instância define a taxa CIR (*Committed Information Rate*) e o número de *bytes* para a rajada normal CBS (*Committed Burst Size*). E a segunda instância define o número de *bytes* para a rajada excedente EBS (*Excess Burst Size*). Os valores CIR, CBS e EBS são obtidos, respectivamente, através dos atributos

qpTBRate, qpTBNormalBurst e *qpTBExcessBurst*, definidos no modelo para representação das políticas de configuração. Esta abordagem está ilustrada na figura 7.5.

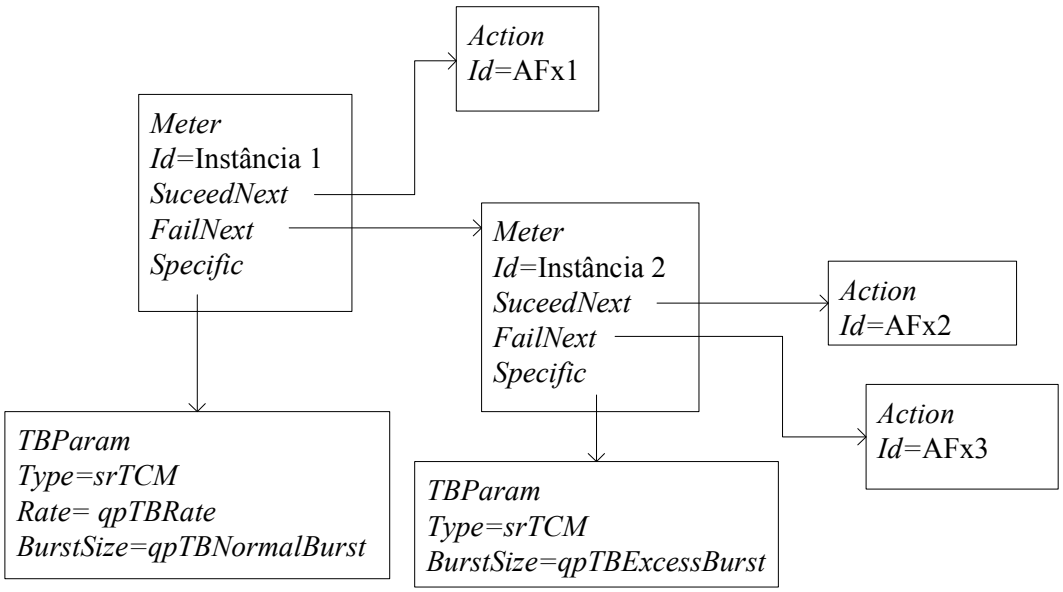

Figura 7.5: Implementação do medidor *srTCM* na *DiffServ* PIB

Tabela 7.7: Mapeamento dos atributos da classe *PolicyDSCPVariable* para a PRC *dsDscpMarkActTable*

Classe *PolicyDSCPVariable*	**PRC *dsDscpMarkActTable***
PolicyValue	*dsDscpMarkActDscp*

Tabela 7.8 Mapeamento dos atributos da classe *QoSPolicyCongestionControlAction* para a PRC *dsAlgDropTable*

Classe *QoSPolicyCongestionControlAction*	**PRC *dsAlgDropTable***
qpQueueSize	*dsAlgDropQThreshold*
qpDropMethod	*dsAlgDropType*
qpThresholdUnits (pacotes ou bytes)	**PRC *dsRandomDropTable***
qpMinThresholdValue	*dsRandomDropMinThreshBytes ou dsRandomDropMinThresPkts*
qpMaxThresholdValue	*dsRandomDropMaxThreshBytes ou dsRandomDropMaxThresPkts*

Para controle de congestionamento foi considerado o algoritmo de descarte randômico. Este algoritmo é capaz de atender o comportamento *Assured Forwarding* [HEI99] definido na metodologia *DiffServ*, no qual existem diferentes prioridades de descarte dentro de uma mesma classe de tráfego. Na estratégia adotada, cada classe de tráfegos é dividida em três diferentes níveis de descarte: tráfego em conformidade (AFx1), tráfego excedente (AFx2) e tráfego de violação (AFx3). O tamanho total do *buffer* alocado para cada classe AFx no dispositivo é definido através do atributo *qpQueueSize* na política de configuração e cada nível AFxy é associado a um algoritmo que descarta pacotes randomicamente, cuja execução depende das quantidades mínima (*qpMinThresholdValue*) e máxima (*qpMaxThresholdValue*) de pacotes ou *bytes* permitidas dentro do espaço reservado para toda a classe. As informações *qpMinThresholdValue* e *qpMaxThresholdValue* são diretamente mapeadas para os parâmetros *dsRandomDropMinThreshBytes(Pkts)* e *dsRandomDropMaxThreshBytes(Pkts)* do descartador específico do tráfego em conformidade. Na implementação deste trabalho, os descartadores específicos dos tráfegos em excesso e violação são mapeados com os parâmetros do tráfego em conformidade dividido por 2 e 4 respectivamente. O uso de pacotes ou *bytes* é definido pelo atributo *qpDropThresholUnits*. A figura 7.6 demonstra esta abordagem.

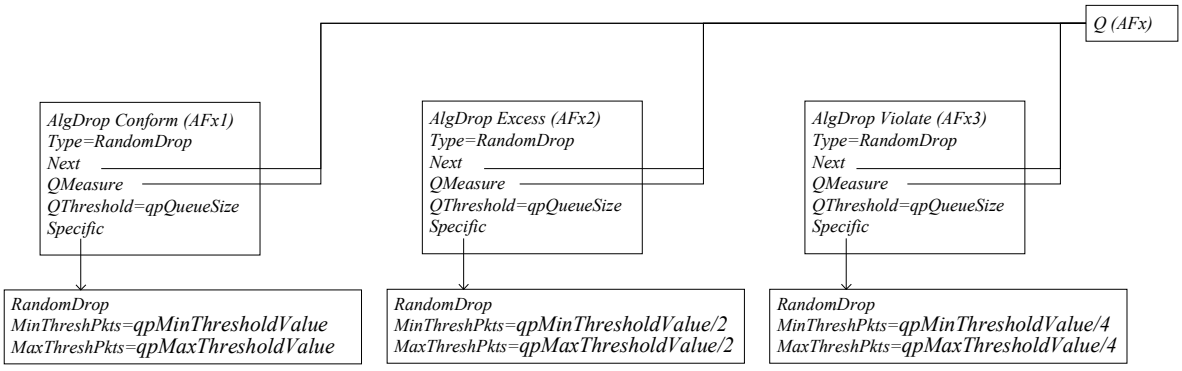

Figura 7.6: Implementação do algoritmo de descarte randômico

Tabela 7.9: Mapeamento da classe *QoSPolicyBandwidthAction* para a PRC *dsQTable*

Classe *QoSPolicyBandwidthAction*	**PRC *dsQTable***
qpMinBandwidth	*dsQMinRate*
qpMaxBandwidth	*dsQMaxRate*

A seqüência para encadeamento dos elementos funcionais *DiffServ* na PIB é obtida a partir da informação *PolicyActionInPolicyAction.ActionOrder*, usada nas composições de ações *CompoundPolicyAction* do modelo CLPM. Vale ressaltar que uma instância PRI que está sendo apontada para indicar um parâmetro, um elemento *DiffServ* ou qualquer outra informação, deve [RFC3317] estar previamente instalada na PIB antes de ser referenciada. Ao receber o conjunto de políticas de configuração selecionado pelo PDP de acordo com o papel desempenhado pelas interfaces, o processo de tradução, responsável por transformar as políticas de configuração independentes de dispositivo em instâncias da *DiffServ* PIB, utiliza o mapeamento definido nas tabelas acima e executa o seguinte algoritmo:

Tabela 7.10: Algoritmo de tradução das políticas de configuração para *DiffServ* PIB

1	Fazer com que *ConfigPolicyRules* seja a lista das regras de políticas de configuração contidas no conjunto de políticas selecionado pelo PDP. • *ConfigPolicyRules*(cg: *ConfigPolicyGroup*)={cr: *ConfigPolicyRule* | cr ⊂ cg }
2	Para cada regra *ConfigPolicyRule* obtida no passo anterior, verificar em qual(is) encarnação(ões) da PIB (is) suas informações devem ser instaladas, de acordo com a grade horária descrita na seção 6.6.2
3	Para cada regra *ConfigPolicyRule* obtida no passo 1, determinar as condições de filtragem de pacotes (*PacketFilterCondition*) e a condição composta (*CompoundPolicyAction*) associados a ela. *PFC*(cr: *ConfigPolicyRule*)={pfc: *PacketFilterCondition* | pfc ∈ cr} • *CA*(cr: *ConfigPolicyRule*)={ca: *CompoundPolicyAction* | ca ∈ cr}
4	Para a ação composta (ca) obtida no passo anterior, determinar a última ação a ser executada dentro da composição, ou seja, a componente que possuir o maior valor no atributo *ActionOrder*. • QA_n(ca: *CompoundPolicyAction*)={qa_n: *PolicyAction* | qa ∈ ca ^ n = maior valor *ActionOrder* dentro da composição}
5	Verificar o tipo da ação QA_n e realizar os mapeamentos correspondentes para a *DiffServ* PIB. Se QA_n = *QoSPolicyBandwidthAction*: 1) Instanciar a PRC *dsScheduler* com o escalonador WRR que realiza o

encaminhamento dos pacotes seguindo a taxa especificada para cada fila na sua entrada. Se a instância do escalonador já tiver sido previamente instalada, apenas é necessário obter sua referência (OID).

2) Instanciar a PRC *dsQ* mapeando os atributos da classe *QoSPolicyBandwidthAction*, conforme descrito na tabela 6-10. Primeiro são instalados os valores correspondentes nas PRCs *dsMinRate* e *dsMaxRate* e depois é feita a referência para estes valores em *dsQMinRate* e *dsQMaxRate*. Como próximo elemento no caminho é apontado o OID da instância *dsScheduler* obtido no item 1).
3) Salvar a referência (OID) da instância *dsQ* que será usada pelo elemento funcional precedente no caminho.
4) Decrementar n. Se n ≥ 1 retornar ao início do passo 5 do algoritmo, caso contrário seguir para o próximo passo.

Se QA_n = *QoSPolicyCongestionControl*:

1) Criar as instâncias da PRC *dsRandomDrop,* mapeando os atributos da classe *QoSPolicyCongestionControl* conforme a tabela 6-9 nos valores específicos de descarte para violação, excesso e conformidade. Salvar as referências destas instâncias.
2) Instanciar a PRC *dsAlgDrop* com *Type = Random*. Os atributos *dsAlgDrop.Next* e *dsAlgDrop.QMeasure* apontam para a mesma fila *dsQ* que foi previamente instalada seguindo os parâmetros da classe *QoSPolicyCongestionControl* definida na regra de configuração (cr) correspondente.
3) Salvar a referência (OID) da instância *dsAlgDrop* que será usada pelo elemento funcional precedente no caminho.
4) Decrementar n. Se n ≥ 1 retornar ao início do passo 5 do algoritmo, caso contrário seguir para o próximo passo.

Se QA_n = *SimplePolicyAction*

1) Instanciar a PRC *dsDscpMarkAct*, mapeando os atributos da classe *PolicyDSCPVariable,* conforme descrito na tabela 6-8. Caso já exista uma instância com o valor DSCP correspondente, obter o OID desta instância.

	2) Instanciar a PRC *dsAction* que aponta para o valor DSCP instalado no passo 1). O atributo *dsActionNext* aponta para o OID correspondente ao elemento funcional instanciado na ação QA_{n+1}. 3) Salvar a referência (OID) da instância *dsAlgDrop* que será usada pelo elemento funcional precedente no caminho. 4) Decrementar n. Se n ≥ 1 retornar ao início do passo 5 do algoritmo, caso contrário seguir para o próximo passo. Se QA_n = *QoSPolicyPoliceAction* 1) Obter as respectivas ações de violação, excesso e conformidade. Verificar o tipo destas ações e mapeá-las para a *DiffServ* PIB conforme as hipóteses anteriores. 2) Obter o perfil de tráfego associado. Criar as instancias da PRC *dsTBParam* mapeando os atributos da classe *QoSPolicyTokenBucketTrfcProf,* conforme descrito na tabela 6-7. Caso já existam instâncias com o mesmo perfil de tráfego, obter os OIDs destas instâncias. 3) Criar as duas instâncias da PRC *dsMeter* que apontam para os parâmetros *Token Bucket* instalados no item 2). O atributo *dsMeterFailNext* do segundo medidor aponta para a ação de violação do item 1). O atributo *dsMeterSuceedNext* do segundo medidor aponta para a ação de excesso do item 1). O atributo *dsMeterFailNext* do primeiro medidor aponta para o OID do segundo medidor. O atributo *dsMeterSuceedNext* aponta para a ação de conformidade do item 1). 4) Salvar a referência (OID) da instância *dsMeter* do primeiro medidor que será usada pelo elemento funcional precedente no caminho. 5) Decrementar n. Se n ≥ 1 retornar ao início do passo 5 do algoritmo, caso contrário seguir para o próximo passo.
6	Para cada condição de filtragem de pacotes (pfc) obtida no passo 2, determinar o filtro de cabeçalho IP associado a ela e mapear os atributos da classe *IPHeadersFilter* para uma instância PRC *IpFilter,* conforme descrito na tabela 6-6. Para cada instância da PRC *IpFilter,* criar uma instância correspondente da PRC *dsClfrElement* cujo atributo *dsClfrElementSpecific*

	aponta para o OID da instância *IPFilter*. O atributo *dsClfrElementNext* aponta para o OID da instância do elemento funcional correspondente a ação do passo 3. O atributo *dsClfrElementPrecedence* recebe o valor do atributo do atributo *Priority* da regra de configuração (cr) a qual a condição de filtragem de pacotes está associada. Um classificador é composto por um ou mais elementos classificadores, o atributo *dsClfrElementClfrId* é usado para indicar a qual classificador o elemento classificador pertence. Na implementação deste algoritmo é assumido que existe apenas um classificador (*ClfrIn*) responsável por selecionar os pacotes que entram em uma determinada interface. Assim todos as instâncias da PRC *dsClfrElement* que pertencem ao classificador de entrada possuem o mesmo valor (*ClfrIn*) para o atributo *dsClfrElementClfrId* e cada pacote seguirá o caminho definido pelo primeiro elemento classificador que atenda as características do pacote. Por isto todo classificador deve ter um elemento classificador "coringa" com o mais baixo nível de precedência que defina o tratamento padrão para os pacotes que não forem selecionados pelas regras de configuração resultantes das políticas definidas pelo administrador. No caso deste trabalho foi adotado que o tratamento padrão é o *best-effort*.
7	Instanciar a PRC *dsClfr* com a informação *dsClfrId* que é utilizada por todos elementos classificadores no sentido de entrada da interface. Salvar a referência (OID) desta instância *dsClfr* que será usada pelo elemento *dsDataPath*.
8	Criar uma instância da PRC *dsDataPath* associada com o conjunto de capacidades e papéis da interface e direção ingresso. O atributo *dsDataPathStart* aponta para a instância *dsClfr* intalada no passo 6. Para direção de egresso nenhum caminho *dsDataPath* é instalado porque neste sentido os pacotes já foram previamente marcados e devidamente tratados pelas interfaces de "*core*".

7.3 Implementação PDP e PEP

Para implementação das entidades *Policy Decision Point* (PDP) e *Policy Enforcement Point* (PEP) foi reaproveitada a base dos códigos Java usados em [NAB03]. [NAB03] adotou a estratégia *outsourcing*, portanto várias alterações foram

feitas para o código passar a suportar a estratégia *provisioning*, incluindo a leitura e escrita de objetos COPS-PR.

7.3.1 PDP

Durante sua ativação, o PDP carrega em memória suas configurações iniciais definidas no arquivo XML mostrado na figura 7.7:

```
<Pdp>
  <Config>
      <PdpId>PdpTeste</PdpId>
      <PdpPort>3288</PdpPort>
      <NextPdpAddr>127.0.0.1</NextPdpAddr>
      <NextPdpPort>3288</NextPdpPort>
      <Ka>10</Ka>
      <SupportedPep>2 4001 4002</SupportedPep>
      <AuthorizedPep>PepDiffServ1 Pep1 Pep2 Pep3</AuthorizedPep>
  </Config>
<ActivePep>…  </ActivePep>
```

Figura 7.7: Arquivo de configuração do PDP

- *<PdpId>* – Corresponde à identificação do servidor de políticas.
- *<PdpPort>* – Define a porta TCP em que o servidor de políticas estará respondendo a requisições. De acordo com a norma COPS [DUR00] e a numeração padronizada pelo IANA, esta porta deve ser definida como 3288.
- *<NextPdpAddr>* – Define o endereço do próximo servidor de políticas. Este campo é utilizado quando o servidor recebe uma solicitação e não tem capacidade para prover as políticas necessárias. Dessa forma, encaminha para um próximo servidor que pode responder às políticas solicitadas.
- *<NextPdpPort>* – Define a porta TCP em que o próximo servidor de políticas estará respondendo.

- *<Ka>* – Define o intervalo de tempo sob o qual será gerado randomicamente uma mensagem de controle que tem como finalidade informar o status de funcionamento do servidor de políticas.
- *<SupportedPep>* – Apresenta uma lista com todos os tipos de clientes PEP suportados por este servidor. Clientes que não estejam nesta lista não são suportados pelo PDP e são encaminhados ao próximo servidor de políticas.
- *<AutorizedPep>* – Apresenta uma lista com todas as identificações de clientes PEP que têm permissão para se conectar a este servidor de políticas. Da mesma forma que a entrada anterior, clientes que não pertençam a esta lista não têm sua conexão permitida.
- *<ActivePep>* – Armazena os estados dos diversos PEP's que venham a se conectar ao PDP.

Após a carga da configuração inicial, o PDP torna-se operacional passando a aceitar as conexões dos PEPs através de um *socket* ativado na porta TCP definida na configuração.

A estrutura do código implementado define um PDP genérico, capaz de controlar uma diversidade de tipos de clientes, bastando para isto que o PDP "conheça" a estrutura do módulo PIB especificamente usado pelo cliente.

Além de efetuar a decisão das políticas a serem executadas, o PDP também mantém o estado de configuração dos PEPs a ele conectados. Estas informações são armazenadas na seção *<ActivePep>* do arquivo XML e correspondem a uma cópia fiel das instâncias (PRIs) enviadas para o PEP.

Os principais métodos implementados pelo PDP são:

Tabela 7.11: Principais métodos implementados no PDP

Método	Descrição
readOPN( )	Método usado para processar a solicitação de abertura de conexão enviada por um PEP. Neste método é verificado se o PEP é autorizado pelo PDP e se o tipo do cliente para o qual o PEP está solicitando a conexão é suportado pelo PDP. Se a requisição for aceita, uma nova instância (identificada com *PEPId*) da estrutura XML correspondente a PIB do cliente é carregada em memória no PDP e a mensagem CAT (*client accept*) é retornada ao PEP. Caso

	contrário, é retornada a mensagem CC (*client close*) e a conexão é cancelada.
CreateCAT( )	Este método constrói a mensagem CAT conforme definido no protocolo COPS e adiciona objeto *KA* (C-Num=10 e C-Type) com o valor do período usado para verificação da conexão entre PEP e PDP.
createCC( )	Método usado para encerrar a conexão entre PEP e PDP. Se for o caso, o objeto *PDP Redirect Address* (C-Num=13 e C-Type=1) ou objetos de erro podem ser incluídos na mensagem. O PEP que estava ativo é removido da base de estados.
readREQ( )	Método usado para processar a solicitação de configuração enviada pelo PEP. As informações das capacidades do PEP, enviadas nos objetos COPS-PR encapsulados em *Named Client SI* (C-Num=9 e C-Type=2), são mapeadas para a instância da PIB correspondente ao *PEPId*, carregada em memória no PDP durante o processamento *readOPN( ).* Após armazenar todas as informações, o PDP invoca o método *createDEC( ).*
createDEC( )	Este método constrói a mensagem COPS de decisão, adicionando o objeto (C-Num=6 e C-Type=1) que indica a instalação ou remoção de configuração e o objeto *Named Decision Data* (C-Num=6 e C-Type=5) usado para encapsular os objetos COPS-PR que contêm as informações de configuração. O conteúdo da configuração é formado pelo método específico de decisão, invocado de acordo com o tipo do cliente.
createDiffServDEC( )	Neste método é implementado o elemento *Device Level Policy Compiler* (DLPC) da arquitetura proposta. Ou seja, as políticas de configuração descritas de forma independente de dispositivos são convertidas, levando em consideração as características do dispositivo, para instâncias (PRIs) que serão instaladas na PIB do cliente. As configurações enviadas para o PEP são mantidas em uma cópia temporária até que a mensagem RPT seja retornada pelo PEP. Quando isto acontece a cópia temporária é definitivamente armazenada no arquivo XML do PDP. O arquivo é atualizado

	sempre que a configuração é alterada.
ReadRPT( )	Se a mensagem enviada pelo PEP for de sucesso, este método transfere a cópia temporária de configuração para a base de estados mantida no arquivo XML.

7.3.2. PEP

Cada tipo de cliente de política suportado pelo PEP é devidamente identificado pelo atributo *Client-Type*. O valor de *Clietn-Type* define as características relativas à aplicação de políticas para o tipo de cliente especificado. Tipos de cliente definidos pelos valores compreendidos entre 0x0001 e 0x3FFF são reservados e registrados no IANA e têm seu comportamento e aplicabilidade descrita em documentos específicos de extensão COPS. Os tipos definidos entre 0x4000 e 0x7FFF são reservados para clientes não padronizados sendo permitido sua utilização para uso privado. Os tipos compreendidos entre 0x8000 e 0xFFFF são para tipos padronizados sem, no entanto, a necessidade de documentos específicos que definam seu funcionamento. O IANA[6] define o valor 0x0002 para clientes de políticas de configuração DiffServ, portanto os clientes implementados neste trabalho possuem o atributo *Client-Type* = 0x0002.

De forma análoga ao PDP, o PEP ao ser iniciado, carrega em memória sua configuração inicial a partir de um arquivo XML. O arquivo de configuração do PEP é composto por duas seções: <Configuração> e <Pib>, conforme mostrado abaixo.

```
<Pep>
<Configuracao>
        <PepId>PepDiffServ1</PepId>
        <PdpAddr>localhost</PdpAddr>
        <PdpPorta>3288</PdpPorta>
        <ClientType>2</ClientType>
        <MaxHops>3</MaxHops>
</Configuracao>
<Pib> ... </Pib>
```

Figura 7.8: Arquivo de configuração do PEP

[6] Internet Assigned Numbers Authority – www.iana.org/numbers.html

- *PepId* – Corresponde à identificação do PEP.
- *PdpAddr* – Corresponde ao endereço IP do servidor de políticas ao qual o cliente deve conectar-se para obter os dados necessários ao seu funcionamento.
- *PdpPorta* – Corresponde ao endereço da porta TCP em que o PEP estará respondendo.
- *ClientType* – Corresponde ao tipo definido para o cliente. Neste caso 0x0002.
- *MaxHops* – Este campo tem como função definir o número máximo de buscas por um servidor de políticas, capaz de atender a esse tipo de cliente. Assim, um PDP recebe uma solicitação de conexão de um PEP e caso não possa atendê-la, encaminha esse PEP a um outro servidor de políticas. Um contador interno do PEP registra esses saltos, limitando-os ao máximo previsto no valor definido no campo. Este mecanismo garante que não ocorra um *loop* na busca por um servidor de políticas. Uma mensagem de erro é gerada quando o limite definido é alcançado.

A seção <Pib> é discutida em 6.3 e contem o mapeamento XML das classes *FrameworkPib* e *DiffServPib* implementadas no PEP.

Os principais métodos implementados pelo PEP são:

Tabela 7.12: Principais métodos implementados no PEP

Método	Descrição
createOPN( )	Este método constrói a mensagem COPS de solicitação de abertura de conexão com o PDP. O valor do *Client-Type* e o objeto *PEPId* (C-Num=11 e C-Type=1) são incluídos na mensagem.
readCAT( )	Método usado processar a informação do período KA enviada pelo PDP na mensagem CAT.
createREQ( )	Este método constrói a mensagem COPS de solicitação de configuração (C-Num=2, C-Type=1 e R-Type=8, M-Type=1) enviada ao PDP. O objeto *Named Client SI* (C-Num=9 e C-Type=2) é acrescentado para encapsular os objetos COPS-PR que, por sua vez, contêm os dados de todas as classes do tipo *notify* e *notify/install* da PIB do cliente. Nestes objetos estão as informações das capacidades, limitações e combinação de papéis das interfaces do dispositivo.
readDEC( )	Método usado para processar as decisões de políticas enviadas pelo

	PDP. As instâncias (PRIs) enviadas nos objetos COPS-PR encapsulados em *Named Decision Data* (C-Num=6 e C-Type=5) são instaladas/removidas na seção <Pib> no arquivo XML do PEP. Após instalar/remover as instâncias recebidas, o PEP envia a mensagem RPT para comunicar ao PDP o sucesso ou falha da configuração.
createRPT( )	Este método constrói a mensagem RPT que inclui o objeto *Report-Type* (C-Num=12 e C-Type=1) com conteúdo=1 para indicar sucesso na configuração ou conteúdo=2 para falha.
createCC( )	Método usado para encerrar a conexão entre PEP e PDP. Objetos de erro podem ser incluídos na mensagem.

Após a instalação das informações de configuração, o PEP torna-se operacional e o dispositivo controlado passa a encaminhar os pacotes IP de acordo com as políticas recebidas e sem a necessidade de nova consulta ao PDP. No caso de dispositivos legados, o PEP utiliza um arquivo do tipo *batch* para transformar as informações contidas na PIB em comandos específicos do protocolo de configuração (SNMP, CLI,...) implementado pelo dispositivo.

7.4. Conclusão

Verificou-se que a implementação de um algoritmo genérico, que seja capaz de tratar qualquer combinação de capacidades, para conversão das políticas de configuração independentes de dispositivo em instâncias da *DiffServ* PIB é uma tarefa complexa. Por isso, neste trabalho, foi considerado um conjunto específico de capacidades para os dispositivos de rede que consomem as políticas (PEPs).

Na arquitetura proposta, o servidor de políticas (PDP) também é responsável por construir e gerenciar a grade horária de configuração dos dispositivos que reflete os períodos de tempo em que as especificações de níveis de serviço devem ser praticadas. Para comunicação entre o servidor (PDP) e os clientes de políticas (PEPs) foi implementado o protocolo COPS/COPS-PR que transporta os dados necessários para provisão de configuração. O uso da estratégia *provisioning* é inerente ao mecanismo *DiffServ*, o qual não usa protocolo de sinalização e necessita que o dispositivo seja previamente configurado para que os pacotes recebam o tratamento de QoS

especificado pelo administrador. Para suportar a atualização dinâmica resultante da grade horária de configuração dos dispositivos, foram discutidas três estratégias distintas, a partir da combinação COPS/COPS-PR e DiffServ PIB. O desempenho destas estratégias é avaliado no capítulo 8.

Capítulo 8

Estudo de Caso e Avaliação da Proposta

8.1. Introdução

Com o objetivo de demonstrar a utilização da arquitetura de gerenciamento de QoS deste trabalho e avaliar a flexibilidade dos modelos propostos para representação de políticas, a seção 8.2 apresenta um estudo de caso para gerenciamento de QoS no ambiente de uma Rede Coorporativa (Intranet). Na seção 8.3., um cenário com roteadores Linux é utilizado para verificação de desempenho do servidor de políticas (PDP), dos clientes configurados (PEPs) e análise do impacto das mensagens COPS/COPS-PR na rede levando em consideração as três estratégias diferentes discutidas no capítulo 7 para atualização da configuração dos dispositivos. Finalmente, a seção 8.4 discute os resultados obtidos.

8.2. Estudos de Caso

O cenário usado para demonstrar a implementação do modelo de política de alto nível proposto no capítulo 5 deste trabalho é o ambiente de uma pequena rede coorporativa. A figura 8.1 mostra que o ambiente é composto por quatro sub-redes interconectadas através de uma rede central. A tabela 8.1 descreve as unidades organizacionais (setores) da empresa e os três cargos de mais alto nível hierárquico utilizados em cada setor. Os funcionários (usuários da rede) são cadastrados no sistema como objetos *CIMPerson*, o atributo "OU" define o setor em que o funcionário trabalha e o atributo "*BusinessCategory*" define o cargo dele. Na tabela 8.2 são mostrados os grupos de aplicações e servidores definidos nos sistema.

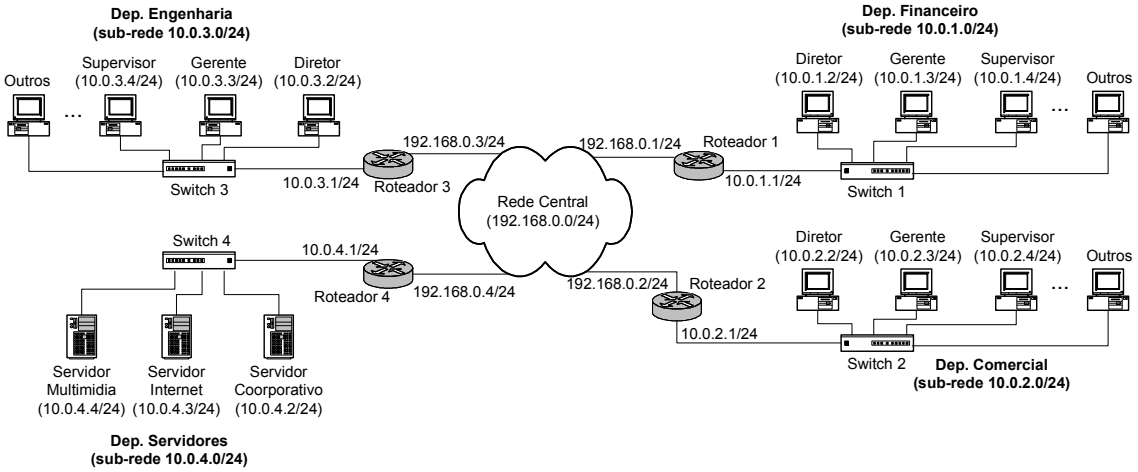

Figura 8.1: Cenário de uso Rede Coorporativa

Tabela 8.1: Usuários da rede coorporativa

	Unidade Organizacional	Cargo
Usuários	Engenharia Comercial Financeiro	Supervisor Gerente Diretor

Tabela 8.2: Grupos de aplicações e servidores definidos no sistema

Aplicações	Internet Multimídia Sistemas Coorporativos	Servidores	Servidor Internet Servidor Multimídia Servidor Coorporativo

Neste exemplo, os objetivos de negócio (políticas de alto nível) são descritos pelo administrador para estabelecer uma alocação hierárquica de largura de banda de acordo com a tripla: usuário / servidor / aplicação e levando também em consideração os intervalos de horário mostrados na tabela 8.3. As regras do primeiro nível hierárquico na tabela 8.4 determinam que durante os intervalos do "Horário1":

1) O tráfego das aplicações de Internet com origem no Servidor de Internet é marcado com DSCP = AF31 e recebe 15% da largura de banda total na interface de borda do Domínio DiffServ, no caso a interface = 10.0.4.1/24.

2) O tráfego das aplicações do Sistema Coorporativo com origem no Servidor Coorporativo é marcado com DSCP = AF21 e recebe 25% da largura de banda total na interface de borda do Domínio DiffServ, no caso a interface = 10.0.4.1/24.
3) O tráfego das aplicações Multimídia com origem no Servidor de Internet é marcado com DSCP = AF11 e recebe 40% da largura de banda total na interface de borda do Domínio DiffServ, no caso a interface = 10.0.4.1/24.

E as sub-regras 1a,b,c; 2a,b,c; 3a,b,c (segundo nível hierárquico) subdividem as larguras de banda do nível superior de acordo com os cargos dos usuários de destino.

Para os intervalos do "Horário2", as regras do primeiro nível hierárquico na tabela 8.5 determinam que:

1) O tráfego das aplicações de Internet com origem no Servidor de Internet é marcado com DSCP = AF21 e recebe 20% da largura de banda total na interface de borda do Domínio *DiffServ*, no caso a interface = 10.0.4.1/24.
2) O tráfego das aplicações do Sistema Coorporativo com origem no Servidor Coorporativo é marcado com DSCP = AF31 e recebe 20% da largura de banda total na interface de borda do Domínio DiffServ, no caso a interface = 10.0.4.1/24.
3) O tráfego das aplicações Multimídia com origem no Servidor de Internet é marcado com DSCP = AF11 e recebe 40% da largura de banda total na interface de borda do Domínio *DiffServ*, no caso a interface = 10.0.4.1/24.

De forma análoga às regras do "Horário1", no "Horário2" as sub-regras 4a,b,c; 5a,b,c; 6a,b,c (segundo nível hierárquico) subdividem as larguras de banda do nível superior de acordo com os cargos dos usuários de destino.

Dentro da arquitetura proposta, o tratamento padrão para os pacotes é o "melhor esforço" (BE), assim todo o tráfego que não atende os requisitos das regras é tratado desta forma.

Tabela 8.3: Grade horária das políticas

Horário	Horário 1 (Seg a Sex)	Horário 2 (Seg. a Sex)
	09h00 – 12h00 14h00 – 17h00	00h00 – 09h00 12h00 – 14h00 17h00 – 24h00

Tabela 8.4: Políticas de negócio para o “Horário 1”

Políticas de Negócio **Grupo “Hierárquico”** **Regras para “Horário 1”**	
SE Aplicação = “Internet” E Servidor = “Servidor de Internet” ENTÃO Nível de Serviço = “AF31+15%BW” para o tráfego na Direção = “2” (Servidor → Cliente)	Regra 1
SE Usuário = “Diretor” ENTÃO Nível de Serviço = “40%BW”	Regra 1a
SE Usuário = “Gerente” ENTÃO Nível de Serviço = “25%BW”	Regra 1b
SE Usuário = “Supervisor” ENTÃO Nível de Serviço = “15%BW”	Regra 1c
SE Aplicação = Sistemas Coorporativos E Servidor = “Servidor Coorporativo” ENTÃO Nível de Serviço = “AF21+25%BW” para o tráfego na Direção = “2” (Servidor → Cliente)	Regra 2
SE Usuário = “Diretor” ENTÃO Nível de Serviço = “40%BW”	Regra 2a
SE Usuário = “Gerente” ENTÃO Nível de Serviço = “25%BW”	Regra 2b
SE Usuário = “Supervisor” ENTÃO Nível de Serviço = “15%BW”	Regra 2c
SE Aplicação = “Multimídia” E Servidor = “Servidor Multimídia” ENTÃO Nível de Serviço = “AF11+40%BW” para o tráfego na Direção = “2” (Servidor → Cliente)	Regra 3
SE Usuário = “Diretor” ENTÃO Nível de Serviço = “40%BW”	Regra 3a
SE Usuário = “Gerente” ENTÃO Nível de Serviço = “25%BW”	Regra 3b
SE Usuário = “Supervisor” ENTÃO Nível de Serviço = “15%BW”	Regra 3c

Tabela 8.5: Políticas de negócio para o "Horário 2"

Políticas de Negócio **Grupo “Hierárquico”** **Regras para “Horário 2”**	
SE Aplicação = “Internet” E Servidor = “Servidor de Internet” ENTÃO Nível de Serviço = “AF21+20%BW” para o tráfego na Direção = “2” (Servidor → Cliente)	Regra 4
SE Usuário = “Diretor” ENTÃO Nível de Serviço = “35%BW”	Regra 4a
SE Usuário = “Gerente” ENTÃO Nível de Serviço = “25%BW”	Regra 4b
SE Usuário = “Supervisor” ENTÃO Nível de Serviço = “20%BW”	Regra 4c
SE Aplicação = Sistemas Coorporativos E Servidor = “Servidor Coorporativo” ENTÃO Nível de Serviço = “AF31+20%BW” para o tráfego na Direção = “2” (Servidor → Cliente)	Regra 5
SE Usuário = “Diretor” ENTÃO Nível de Serviço = “35%BW”	Regra 5a
SE Usuário = “Gerente” ENTÃO Nível de Serviço = “25%BW”	Regra 5b
SE Usuário = “Supervisor” ENTÃO Nível de Serviço = “20%BW”	Regra 5c
SE Aplicação = “Multimídia” E Servidor = “Servidor Multimídia” ENTÃO Nível de Serviço = “AF11+40%BW” para o tráfego na Direção = “2” (Servidor → Cliente)	Regra 6
SE Usuário = “Diretor” ENTÃO Nível de Serviço = “35%BW”	Regra 6a
SE Usuário = “Gerente” ENTÃO Nível de Serviço = “25%BW”	Regra 6b
SE Usuário = “Supervisor” ENTÃO Nível de Serviço = “20%BW”	Regra 6c

A figura 8.2 mostra o contêiner das políticas de negócio mapeadas em arquivo XML, apenas as regras 1 e 1a estão detalhadas na figura. Os arquivos completos utilizados para este estudo de caso estão descritos no Anexo B. Os elementos que determinam as condições de horário (*PolicyTimePeriodCondition*) são armazenados no contêiner da figura 8.3. Na figura 8.4 é representado o arquivo XML que contem as condições compostas reutilizáveis. A ação de cada regra (*PredefinedSLSPolicyAction*) corresponde a um nível de tratamento de QoS descrito no nível de configuração.

```
<PolicyContainer Name="BusinessLevelPolicy">
 <SLSPolicyGroup SLSType="Hierarchical" PolicyDecisionStrategy="2">
  <SLSPolicyRule Name="Rule1" RuleUsage="..." Enabled="1" ConditionListType="1" ExecutionStrategy="2"
   Priority="3">
   <CompoundTargetPolicyCondition ConditionListType="1" Direction="2"  GroupNumber="1"
   ConditionNegated="false">
    <CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
     GroupNumber="1" ConditionNegated="false"
     PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplicationPolicyCondition[@Name='Inter
     net'])"/>
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
     GroupNumber="1" ConditionNegated="false"
     PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolicyCondition[@Name='InternetSe
     rver'])"/>
   </CompoundTargetPolicyCondition>
   <PredefinedSLSPolicyAction PredefinedSLSName="AF31+15%BW"/>
   <PolicyRuleValidityPeriod
    PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='Horario1'])"/>
   <SLSPolicyRule Name="Rule1a" RuleUsage="..." Enabled="1" ConditionListType="1"
   ExecutionStrategy="2" Priority="1"> <! -- regra aminhada -->
    <CompoundTargetPolicyCondition ConditionListType="1" Direction="2" GroupNumber="1"
     ConditionNegated="false">
        <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
      GroupNumber="1" ConditionNegated="false"
      PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyCondition[@Name='Directors
      '])"/>
    </CompoundTargetPolicyCondition>
    <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
   </SLSPolicyRule> <!-- fim da regra 1a -->
        ... <!-- regras 1b e 1c -->
  </SLSPolicyRule> <!-- fim da regra 1 -->
        ... <!-- regras 4 a 6-->
 </SLSPolicyGroup>
</PolicyContainer>
```

Figura 8.2: Repositório XML para Políticas de Alto Nível

```
<PolicyContainer Name="TimePeriodConditions">
 <PolicyTimePeriodCondition Name="Horario1" TimePeriod="20040901T000000/THISANDFUTURE"
  DayOfWeekMask="01111100" TimeOfDayMask="T080000/115959;T140000/175959" LocalOrUtcTime="1"/>
 <PolicyTimePeriodCondition Name="Horario2" TimePeriod="20040901T000000/THISANDFUTURE"
  DayOfWeekMask="01111100" TimeOfDayMask="T000000/075959;T120000/135959;T180000/235959"
  LocalOrUtcTime="1"/>
</PolicyContainer>
```

Figura 8.3: Repositório XML para as Condições de Horário

A *CompoundApplicationPolicyCondition* detalhada na figura, define que as aplicações de "Internet" dentro do sistema correspondem ao produto instalado com Nome = "*ApacheHTTP*" E Versão = "*1.3*"; OU produto instalado com Nome = "*SurgeMail*"; OU produto instalado com Nome = "*SurgeFTP*".

```
<PolicyContainer Name="CompoundConditions">
 <ReusablePolicyContainer Name="ReusableCompoundApplicationConditions">
   <CompoundApplicationPolicyCondition Name="Internet" ConditionListType="1">
     <SimplePolicyCondition GroupNumber="1" ConditionNegated="false">
       <PolicyExplicitVariable ModelClass="InstalledProduct" ModelProperty="Name"/>
       <PolicyStringValue StringList="ApacheHTTP"/>
     </SimplePolicyCondition>
     <SimplePolicyCondition GroupNumber="1" ConditionNegated="false">
       <PolicyExplicitVariable ModelClass="SoftwareFeature" ModelProperty="Version"/>
       <PolicyStringValue StringList="1.3"/>
     </SimplePolicyCondition>
     <SimplePolicyCondition GroupNumber="2" ConditionNegated="false">
       <PolicyExplicitVariable ModelClass="InstalledProduct" ModelProperty="Name"/>
       <PolicyStringValue StringList="SurgeMail"/>
     </SimplePolicyCondition>
     <SimplePolicyCondition GroupNumber="3" ConditionNegated="false">
       <PolicyExplicitVariable ModelClass="InstalledProduct" ModelProperty="Name"/>
       <PolicyStringValue StringList="SurgeFTP"/>
     </SimplePolicyCondition>
   </CompoundApplicationPolicyCondition>
     … <!-- outras condições compostas para Aplicações -->
 </ReusablePolicyContainer>

 <ReusablePolicyContainer Name="ReusableCompoundUserConditions">
 … </ReusablePolicyContainer> <! -- condições compostas para Usuários -->

 <ReusablePolicyContainer Name="ReusableCompoundServerConditions">
 … </ReusablePolicyContainer> <! -- condições compostas para Servidores -->

</PolicyContainer>
```

Figura 8.4: Repositório XML para as Condições Compostas

Conforme discutido no capítulo 6, as políticas de negócio são convertidas para políticas de configuração através da execução de um processo de tradução. A figura 8.5 detalha a política de configuração obtida na tradução da "Regra1" definida pelo administrador em alto nível (tabela 8.4) e considerando o cenário da figura 8.1. No nível de configuração, a regra é mapeada para o grupo de políticas "*ConfigQosDepServidor*" porque o fluxo de tráfego selecionado por esta regra (Servidor de Internet) tem origem na sub-rede "Departamento Servidor", esta estratégia de agrupamento das políticas determina a configuração relevante para as interfaces de borda dos roteadores no domínio DiffServ. No cenário utilizado, o grupo "*ConfigQosDepServidor*" representa a configuração da interface 10.0.4.1/24 (roteador 4).

```
<PolicyContainer Name="ConfigLevelPolicy">
 <ConfigPolicyGroup ConfigName="ConfigQoSDepServidor" PolicyDecisionStrategy="2">
  <ConfigPolicyRule Name="Rule1" Enabled="1" ConditionListType="1" Priority="3" ExecutionStrategy="2">
   <ConfigQoSActionInConfigPolicyRule
    PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name='AF31+15%BW'])">
   </ConfigQoSActionInConfigPolicyRule>
   <PolicyRuleValidityPeriod
    PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='Horario1'])">
   </PolicyRuleValidityPeriod>
   <PacketFilterCondition FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
     <IPHeadersFilter IsNegated="false" HdrIPVersion="4" HdrSrcAddress="10.0.4.3" HdrSrcMask="24"
      HdrDestAddress="0.0.0.0" HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="80" HdrSrcPortEnd=""
      HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
   </PacketFilterCondition>
   … <!-- outros filtros resultantes da tradução da regra 1 -->
   <ConfigPolicyRule Name="Rule1a" Enabled="1" ConditionListType="1" Priority="1"
    ExecutionStrategy="2">
    <ConfigQoSActionInConfigPolicyRule
     PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name='40%BW'])">
    </ConfigQoSActionInConfigPolicyRule>
    <PacketFilterCondition FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
     <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4" HdrSrcAddress="0.0.0.0" HdrSrcMask="0"
       HdrDestAddress="10.0.1.2" HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
       HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
     </FilterList>
    </PacketFilterCondition>
    … <!-- outros filtros resultantes da tradução da regra 1a -->
   </ConfigPolicyRule> <!-- fim da regra 1a -->
   … <!-- regras de configuração 1b e 1c -->
  </ConfigPolicyRule> <!-- fim da regra 1 -->
 </ConfigPolicyGroup> <!-- fim do grupo de políticas referentes ao Departamento Servidor -->
 … <!-- outros grupos de políticas de configuração resultantes do processo de tradução -->
</PolicyContainer>
```

Figura 8.5: Políticas de configuração obtidas no processo de tradução

Na regra "*Rule1*", o filtro IP seleciona os pacotes originados na porta tcp 80 do Servidor de Internet (10.0.4.3/24) e a ação correspondente determina que este tráfego recebe o nível de serviço "*AF31+15%BW*", durante os intervalos de tempo do "*Horario1*". A especificação dos níveis de serviço estão contidas no arquivo "*SLSActions.xml*". A regra ainda possui outros filtros IP que resultaram da tradução da política de alto nível, o arquivo completo das políticas de configuração encontra-se no Anexo B. A sub-regra "*Rule1a*" subdivide a largura de banda alocada para as "*Aplicações de Internet*" hospedadas no "*Servidor de Internet*" e define que os "Diretores" recebem "*40%*" do total alocado pela regra "Rule1". Na figura 8.5 é

mostrado apenas o filtro IP referente aos pacotes destinados ao diretor do departamento financeiro (10.0.1.2/24).

A figura 8.6 mostra a especificação dos níveis de serviço pré-definidos no sistema. Cada nível de serviço é descrito por uma composição de ações (*CompoundPolicyAction*) que, por sua vez, reutilizam as ações de QoS definidas no modelo QPIM. Na figura 8.7, são apresentadas as ações QPIM utilizadas pelo nível de serviço "*AF31+15%BW*".

```
<ReusablePolicyContainer Name="SLSActions">
  <CompoundPolicyAction Name="AF31+15%BW" SequencedActions="1" ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
     PartComponent="./QpimActions.xml#xpointer(//SimplePolicyAction[@Name='AF31'])"/>
    <PolicyActionInPolicyAction ActionOrder="2"
     PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='15%'])"/>
  </CompoundPolicyAction>
  ...<!-- outros níveis de serviço (condições compostas) -->
</ReusablePolicyContainer>
```

Figura 8.6: Descrição dos níveis de serviço pré-definidos no sistema

```
<ReusablePolicyContainer Name="QpimActions">
  <SimplePolicyAction Name="AF31"/>
    <PolicyDSCPVariable/>
    <PolicyIntegerValue IntegerList="AF31"/>
  </SimplePolicyAction>
  <QoSPolicyBandwidthAction Name="15%" qpBandwidthUnits="1" qpMinBandwidth="15"/>
  …<!-- outras ações QPIM -->
</ ReusablePolicyContainer>
```

Figura 8.7: Ações QPIM

8.3 Avaliação dos processos para atualização das configurações

Conforme discutido na seção 7.2, existem, fundamentalmente três estratégias para atualização da configuração de um cliente de políticas (PEP): 1) Uma encarnação completa da PIB é transmitida ao PEP toda vez que a configuração do dispositivo precisa ser alterada; 2) Múltiplas encarnações da PIB são transmitidas ao PEP durante o processo inicial de aprovisionamento e a alteração da configuração é realizada através de uma mensagem enviada pelo PDP para comandar a troca de contexto; 3) Apenas as classes (PRCs) e as instâncias (PRIs) da PIB que precisam ser modificadas são transmitidas ao PEP para alteração da configuração.

O cenário da figura 8.8 foi utilizado para avaliar cada uma das estratégias de atualização de configuração dos dispositivos.

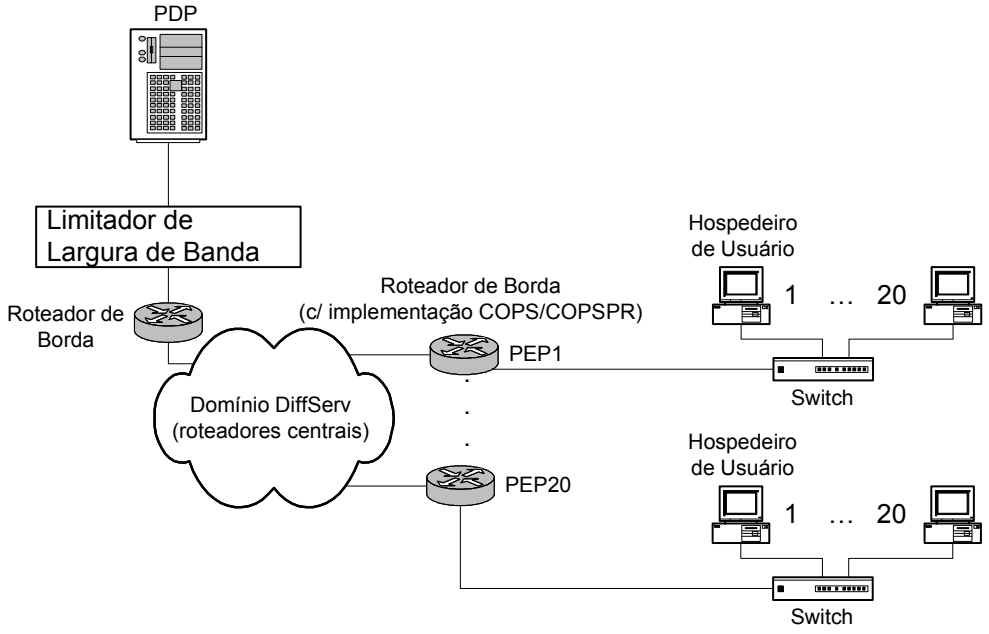

Figura 8.8: Cenário para avaliação de dempenho

O servidor de políticas (PDP) foi implementado em Java e executado em um *PC Intel Pentium IV*, 1,5 GHz com 256 MB RAM e sistema operacional Linux. Os roteadores de borda também foram implementados em Linux e executaram a aplicação desenvolvida em Java para função do PEP, ou seja, receber as mensagens COPS-PR e instanciar as classes da *DiffServ* PIB. Estes roteadores rodaram em *PCs* idênticos ao do PDP.

No processo de avaliação foram realizados experimentos, nos quais o PDP era responsável pela configuração simultânea de até 20 roteadores de borda (PEPs). Cada PEP recebeu uma configuração correspondente a 20 especificações distintas de níveis de serviço (SLSs). Conforme mostrado na figura 8.9, estas especificações resultaram em uma *DiffServ* PIB com: 20 instâncias de classificadores de tráfego que fazem referência a 20 filtros IP; 40 instâncias de medidores (conforme discutido na seção 7.2.3, a implementação de um medidor com três níveis requer o uso de duas instâncias da classe "*dsMeter*") que representam o policiamento individual de cada fluxo de tráfego e determinam se os pacotes estão em conformidade, excesso ou violação em relação ao seu respectivo perfil de tráfego definido através dos parâmetros *token bucket*; 8

instâncias reutilizáveis de parâmetros *token bucket* que definem a taxa de velocidade, o tamanho da rajada normal e o tamanho da rajada excedente para cada classe de serviço pré-definida no sistema (neste caso, Platina, Ouro, Prata e Bronze); e 14 instâncias reutilizáveis de marcação DSCP (12 possíveis combinações AF + EF + BE). O nível de serviço definido como padrão no ambiente gerenciado é o "melhor esforço" (BE), ou seja, todos pacotes que não forem selecionados por nenhum dos filtros da PIB recebem a marcação "BE". A marcação "EF" foi usada para o tráfego de gerenciamento COPS/COPSPR. Note na figura 8.9 que as classes reutilizáveis (desenhadas sem preenchimento) representam as ações de QoS pré-definidas no sistema, por isso a quantidade de instâncias destas classes não é proporcional ao número de SLSs atribuídos a uma mesma interface de dispositivo.

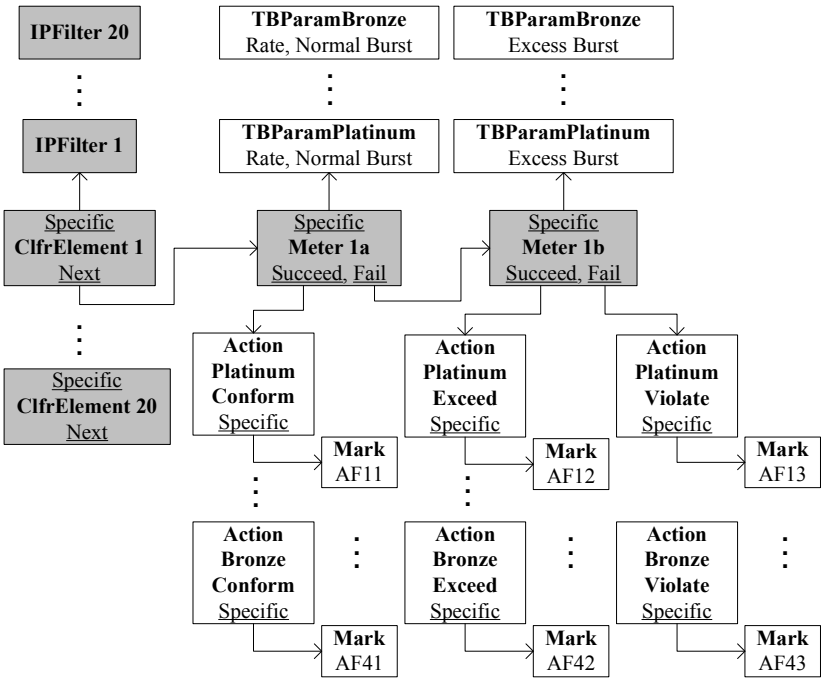

Figura 8.9: Classes (PRCs) e intâncias (PRIs) da DiffServ PIB utilizadas na avaliação

Os dois principais fatores analisados nos experimentos foram: a influência do número de PEPs sendo atualizados simultaneamente e o impacto da restrição da largura de banda disponível para transmissão das mensagens COPS/COPS-PR. Desta forma, pôde-se avaliar a quantidade de largura de banda necessária para a implementação de cada uma das três abordagens propostas e, também, verificar o atraso introduzido na agenda de atualização do serviço quando a largura de banda é restringida. No cenário

usado para a avaliação de desempenho, o domínio *DiffServ* (roteadores centrais) inclui apenas dois saltos entre o PDP e qualquer um dos roteadores de borda, por isso o atraso introduzido pelos roteadores centrais foi considerado desprezível em relação aos outros fatores do experimento. Para restrição da largura de banda foi utilizado o recurso de controle de tráfego do Linux (*linux traffic contro*l), disponível no sistema operacional do hospedeiro do PDP. O controle de tráfego foi configurado através do *script* mostrado na figura 8.10 e o parâmetro *tbf rate* alterado entre os valores 256kbps, 512kbps e 1Mbps. Também foram realizadas medidas sem restrição de largura de banda, ou seja, 100Mbps.

```
# tc qdisc add dev eth0 root handle 1: prio
# tc filter add dev eth0 parent 1:0 prio 1 protocol ip u32 match ip sport 3288 0xfff flow id 1:1
# tc qdisc add dev eth0 parent 1:1 handle 10: tbf rate 256kbit burst 50kb latency 100ms
```

Figura 8.10: *Script* utilizado para restringir a largura de banda do tráfego COPS/COPS-PR gerado pelo PDP

A tabela 8.6 apresenta o tamanho aproximado das mensagens de decisão DEC COPS/COPS-PR para que se tenha uma noção do tráfego gerado em cada uma das estratégias de atualização de configuração. Os tamanhos foram obtidos considerando as instâncias da *DiffServ* PIB da figura 8.9.

Tabela 8.6: Tamanho das mensagens DEC COPS/COPS-PR

Estratégia	Tamanho da mensagem DEC COPS/COPS-PR
Troca da encarnação completa correspondente a 20 SLSs	12kB
Troca de contexto	450B
Alteração das PRCs e PRIs correspondentes a 10 SLSs	5kB

Para análise do efeito do número de PEPs atualizados simultaneamente, foram realizados experimentos com dois diferentes cenários: 10 e 20 roteadores de borda

(PEPs), utilizando a implementação *multithread* do PDP. A seguir são mostradas as medidas coletadas para cada uma das três abordagens.

8.3.1 Avaliação da abordagem que transmite uma completa encarnação da PIB para atualização da configuração dos PEPs

A tabela 8.7 apresenta o tempo médio para atualização dos PEPs. Para ambos os cenários (10 e 20 PEPs), a média foi obtida considerando 3 réplicas de atualização em cada PEP. Os valores expressam a diferença de tempo entre o momento em que o PDP detecta que um SLS deve ser atualizado (definido pela grade horária de configuração) e o momento em que o PEP confirma que a nova configuração foi instalada corretamente (i.e. o PDP recebe a mensagem RPT = sucesso). Dentro da proposta deste trabalho, o PDP constrói previamente todas as encarnações da PIB referentes a um dia inteiro (24 horas) durante o processo inicial de aprovisionamento. Então o atraso introduzido nas atualizações corresponde ao tempo consumido pelo PDP para transformar toda a encarnação da PIB em objetos COPSPR e montar a mensagem de decisão, adicionado com o retardo introduzido pela rede para entregar as mensagens ao PEP.

Tabela 8.7: Tempo médio de atualização dos PEPs com a troca completa da encarnação da PIB

	10 PEPs		20 PEPs	
BW (bps)	Tempo Médio (ms)	Desvio Padrão (ms)	Tempo Médio (ms)	Desvio Padrão (ms)
100M	1178,7	300,16	1465,1	621,48
1M	1537,4	419,14	1761,3	630,57
512k	1668,3	572,91	1887,4	739,70
256k	1760,8	478,47	2443,6	1456,91

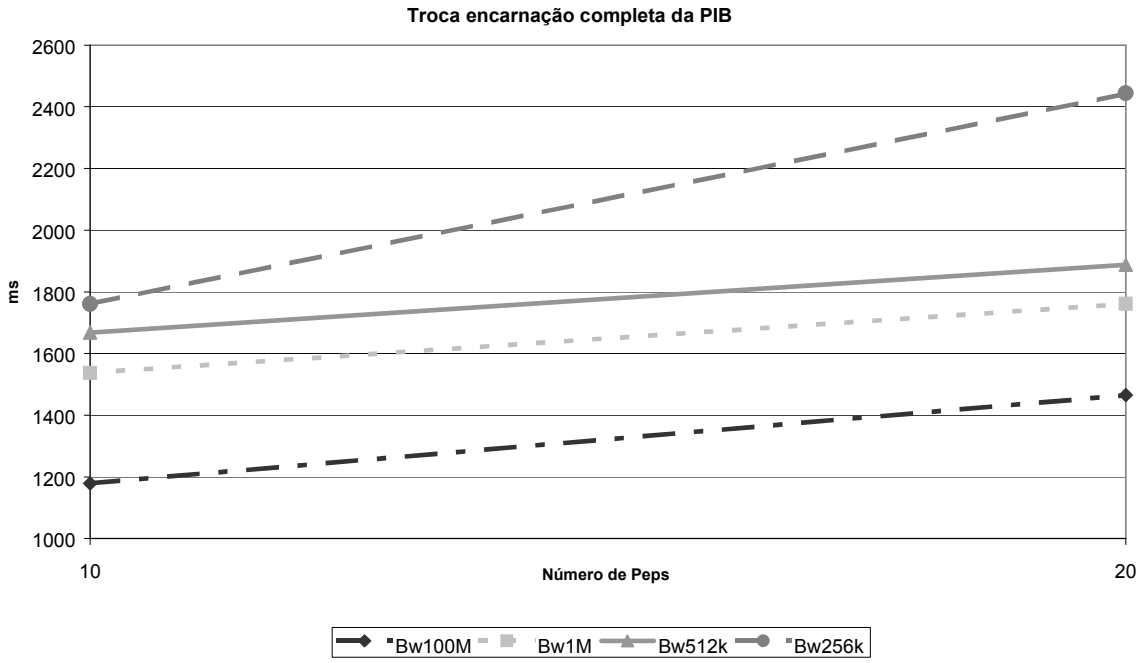

Figura 8.11: Gráfico dos tempos médios para atualização completa da encarnação da PIB

No gráfico 8.11, observa-se que o efeito do aumento de número de PEPs é aproximadamente constante para as larguras de banda entre 512kbps e 1Mbps, ou seja, o atraso introduzido pela rede não é significante (curvas paralelas). Entretanto, o efeito aumenta no caso da restrição de banda igual a 256kbps, indicando que a partir deste ponto o atraso ocasionado pela rede torna-se mais significante. Os valores também mostram que, no cenário de pior caso (20 PEPs e banda restrita em 256kbps), o atraso médio introduzido na atualização da agenda dos serviços é cerca de 2,5 s.

8.3.2 Avaliação da abordagem que transmite apenas uma mensagem de troca de contexto para atualização da configuração dos PEPs

O procedimento do experimento é análogo ao utilizado na seção 8.3.1 e os resultados obtidos são mostrados na tabela 8.8. De acordo com o esperado, esta é a abordagem que introduz o menor atraso médio para atualização da agenda de serviço. Isto ocorre porque, para montar a mensagem de decisão, o PDP precisa apenas transformar uma única classe da PIB (*PibIncarnation)* em objeto COPSPR, resultando

também em pequenas mensagens a serem transportadas pela rede. O cenário de pior caso resultou em um atraso médio de 0,7 s.

Tabela 8.8: Tempo médio de atualização dos PEPs utilizando apenas a troca de contexto

	10 PEPs		20 PEPs	
BW (bps)	Tempo Médio (ms)	Desvio Padrão (ms)	Tempo Médio (ms)	Desvio Padrão (ms)
100M	471,8	185,17	641,2	248,40
1M	611,8	171,50	686,2	239,77
512k	633,3	162,17	702,4	251,10
256k	672,1	222,19	722,64	239,27

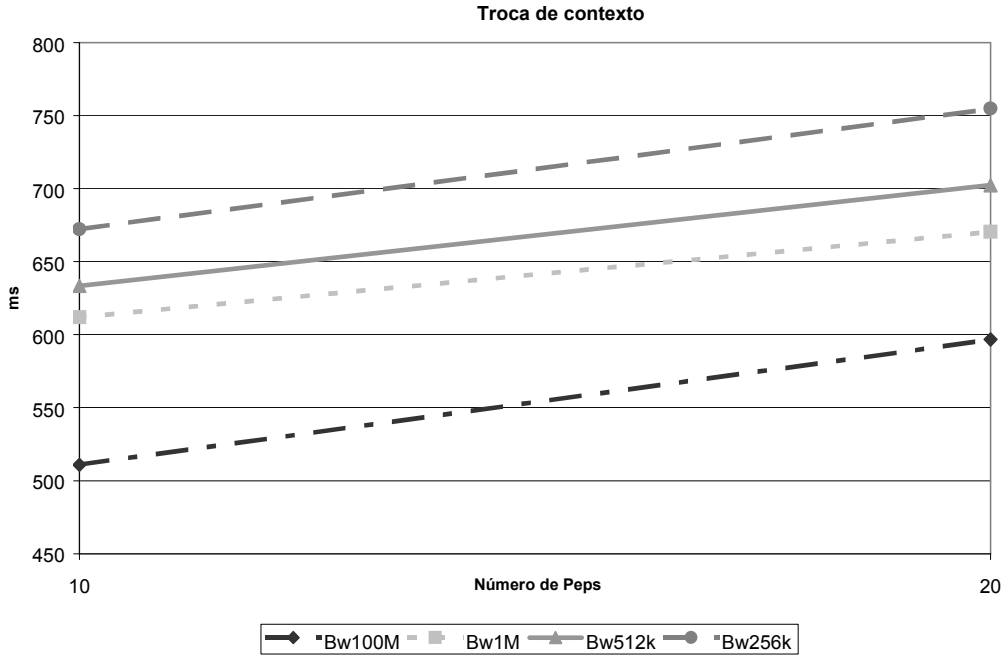

Figura 8.12: Gráfico dos tempos médios para troca de contexto

As curvas paralelas no gráfico da figura 8.12 demonstram que a restrição de banda tem pouco efeito nesta abordagem. Também se percebe que esta é a abordagem com maior potencial de ecalabilidade para o PDP, uma vez que o aumento do número de PEPs representa um efeito pouco significante no atraso médio de atualização.

8.3.3 Avaliação da abordagem que transmite somente as PRCs e PRIs que precisam ser modificadas para atualização da configuração dos PEPs

Dentro da arquitetura proposta, a quantidade de classes (PRCs) que precisam ser alteradas é proporcional ao número de SLSs modificados em relação à configuração anterior da grade horária. Por isso, nesta abordagem, o número de SLSs alterados também foi avaliado no experimento. A tabela 8.9 mostra os resultados para o caso em que 50% dos SLSs são modificados e a tabela 8.10 mostra os resultados para o caso em que 100% dos SLSs são modificados. Como os parâmetros reutilizáveis na PIB (fig. 7.9) são pré-configurados no dispositivo durante o processo inicial de aprovisionamento, a atualização de 1 SLS implica na alteração de 1 elemento classificador, 1 filtro IP e 2 medidores.

Os resultados das tabelas 8.9 e 8.10 refletem que esta abordagem é uma solução intermediária das duas anteriores.

Tabela 8.9: Tempo médio de atualização dos PEPs com a alteração das PRCs e PRIs correspondentes a modificação de 10 SLSs

	10 PEPs		20 PEPs	
BW (bps)	Tempo Médio (ms)	Desvio Padrão (ms)	Tempo Médio (ms)	Desvio Padrão (ms)
100M	888,7	353,03	989,1	364,25
1M	906,1	284,87	1021,0	444,18
512k	921,5	310,83	1058,8	435,78
256k	955,6	381,59	1139,0	413,00

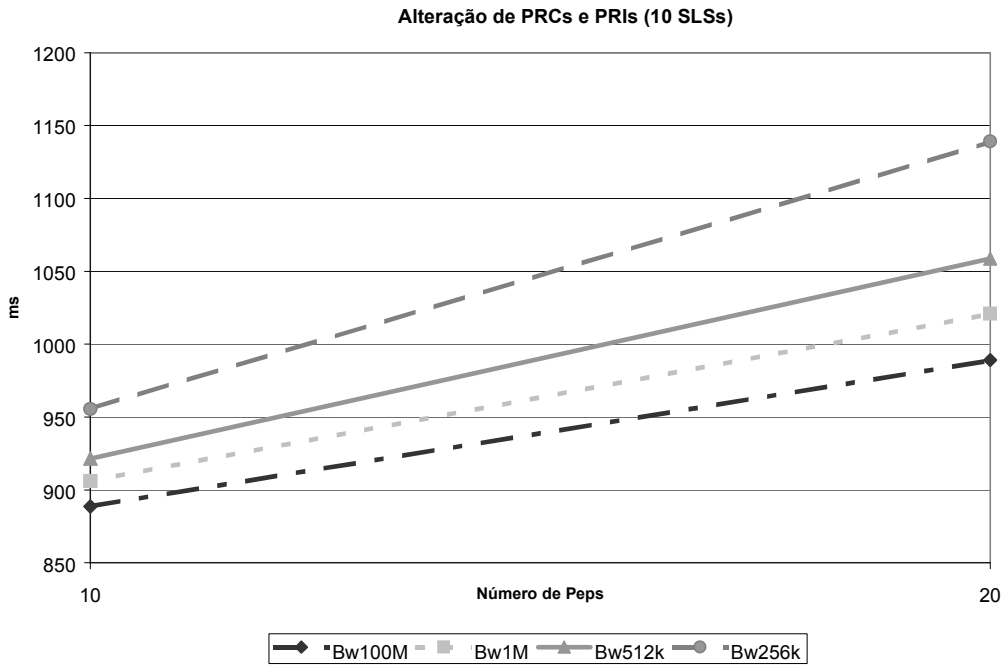

Figura 8.13: Gráfico dos tempos médios para alteração das PRCs e PRIs correspondentes a modificação de 10 SLSs

Tabela 8.10: Tempo médio de atualização dos PEPs com a alteração das PRCs e PRIs correspondentes a modificação de 20 SLSs

	10 PEPs		20 PEPs	
BW (bps)	Tempo Médio (ms)	Desvio Padrão (ms)	Tempo Médio (ms)	Desvio Padrão (ms)
100M	1124,1	428,17	1193,2	465,51
1M	1178,0	477,94	1278,9	464,83
512k	1253,2	294,15	1553,6	659,78
256k	1397,0	425,15	2301,4	1191,51

No caso em que 100% dos SLSs são modificados (tabela 8.9), o reaproveitamento das informações pré-configuradas na PIB é pouco significante, tornando o desempenho similar ao da substituição total da encarnação da PIB. Por isso o gráfico da figura 8.14 demonstra um problema de escalabilidade quando a largura de banda para o tráfego das mensagens COPS/COPSPR é restringida com valores abaixo de 256kbps.

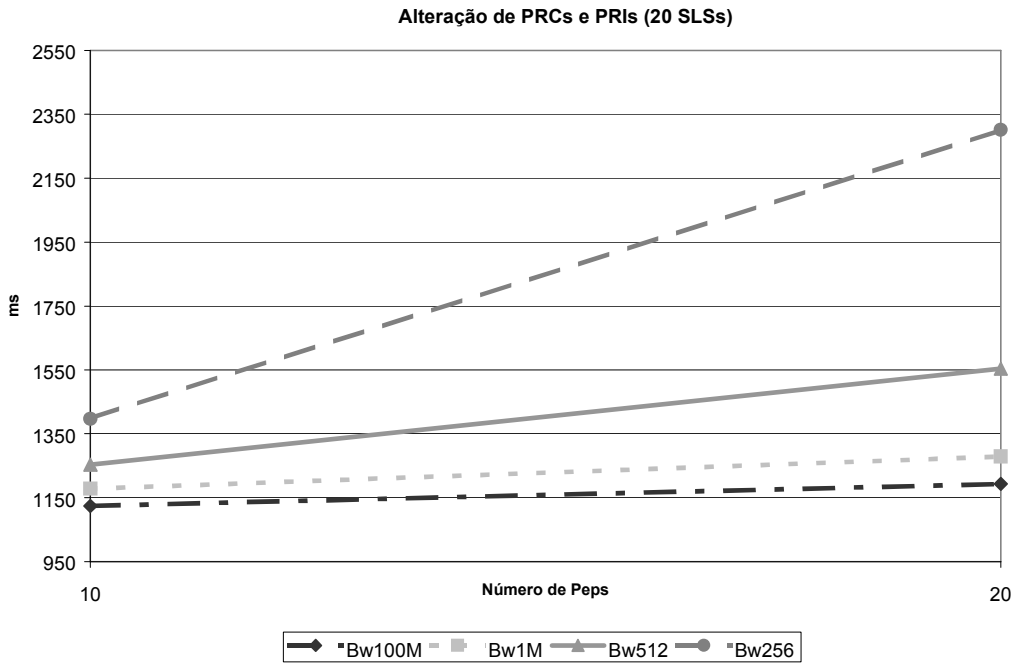

Figura 8.14: Gráfico dos tempos médios para alteração das PRCs e PRIs correspondentes a modificação de 20 SLSs

8.4 Conclusão

Neste capítulo foi considerado um ambiente de rede coorporativa (Intranet) para demonstrar como os modelos propostos são instanciados na descrição das regras de negócio que determinam tratamentos diferenciados de QoS para os fluxos de tráfego na rede. Através do processo de tradução foram gerados os grupos que definem as políticas de configuração para os dispositivos da rede. E, também, foi exemplificado o modo como os níveis de serviço são pré-definidos no sistema através das ações de QoS do modelo QPIM.

O desempenho das três diferentes estratégias discutidas para a atualização da configuração dos dispositivos foi analisado em um cenário com até 20 PEPs, considerando encarnações da *DiffServ* PIB com a configuração de 20 SLSs que determinam o policiamento e a marcação do código DSCP para os pacotes que ingressam nas interfaces de borda do domínio *DiffServ*. O atraso médio resultante de cada uma das abordagens corresponde ao tempo consumido pelo PDP para montar as mensagens de decisão, adicionado com o retardo introduzido pela rede para entrega das mensagens ao PEP. Por isso, conforme esperado, o menor atraso médio é introduzido

pela abordagem em que o PDP, inicialmente, transmite múltiplas encanações da PIB durante ao PEP e, posteriormente, utiliza apenas mensagens de troca de contexto para atualização da agenda de serviço. Entretanto, esta abordagem possui a desvantagem de consumir maior quantidade de recursos de armazenamento no PEP. Esta desvantagem pode se tornar significante de acordo com número de encarnações da PIB gerado pela grade horária de configuração.

Capítulo 9

Conclusões e Trabalhos Futuros

Este trabalho contribui na definição de uma completa arquitetura para gerenciamento de QoS em ambientes *DiffServ*, seguindo os recentes padrões do IETF. Dentro da arquitetura, é proposto um novo modelo para definição de políticas de alto nível e, no nível de configuração, o modelo QPIM é complementado com as classes necessárias para representação da filtragem dos fluxos de tráfego que receberão um determinado conjunto de ações de QoS. O modelo de alto nível da nossa arquitetura permite que uma especificação de nível de serviço (SLS) não seja permanente, ou seja, sua validade é determinada por horários de início e fim ou intervalos periódicos. Mas a *DiffServ* PIB não inclui classes para representação de tempo porque, normalmente, os dispositivos de rede (roteadores *e switches*) não possuem capacidade para processar esta informação. Então novas contribuições são necessárias para que a estrutura do IETF suporte o gerenciamento de QoS dependente de horário. Para controlar a validade das regras determinadas pelo administrador, o servidor de políticas (PDP) deste trabalho monta uma grade horária de configuração que resulta em diferentes encarnações da *DiffServ* PIB. Três diferentes estratégias para alteração dinâmica da configuração dos dispositivos foram discutidas e o desempenho de cada uma delas foi analisado a partir de medições do atraso introduzido na agenda do serviço. Futuramente, estas estratégias serão avaliadas em ambientes onde a atualização dinâmica da configuração depende de eventos ocorridos na rede, ou seja, não é possível que o PDP construa previamente as encarnações da PIB. Por exemplo, definição de níveis de serviço para usuários móveis, neste caso, o processo de atualização poderia ser disparado pelo evento de autenticação do usuário em um ponto de acesso ou *home agent*.

Para que o administrador possa representar intuitivamente o efeito desejado na rede é necessário que sejam implementados algoritmos para verificação de conflitos entre políticas e, principalmente, algoritmos que calculem os recursos disponíveis para evitar a definição de políticas que extrapolem a capacidade de cada nó e de toda a rede. Em trabalhos futuros, pretende-se desenvolver algoritmos de controle de acesso que aliados a mecanismos eficazes para monitoração do estado da rede, serão capazes de otimizar a utilização dos recursos disponíveis no sistema, atendendo aos parâmetros de QoS estipulados. Além disto o modelo de alto nível deve ser capaz de representar: eventos, valores estatísticos para confiabilidade do cumprimento dos contratos e preço dos serviços (contabilidade). Uma evolução do modelo proposto neste trabalho deverá suportar estas informações.

Um importante aspecto no que diz respeito à utilização dos objetos CIM é que as diretivas definidas pelo DMTF e IETF permitem que diferentes estratégias sejam adotadas para o mapeamento das classes associativas dos modelos de informação em repositórios como XML, LDAP ou outro qualquer. Esta flexibilidade representa um obstáculo para reutilização das informações, pois a recuperação dos objetos CIM cadastrados em um sistema depende do conhecimento prévio sobre o esquema específico adotado no repositório. Certamente este é um ponto a ser discutido dentro dos grupos de trabalho do DMTF e IETF.

Publicações

[BEL04] BELLER, A. G.; JAMHOUR, E.; PELLENZ, M. *Defining Reusable Business-Level QoS Policies for DiffServ*. Proceedings of Distributed Systems Operations and Management WorkShop. DSOM 2004, pp. 40-51.

[BEL05] BELLER, A. G.; JAMHOUR, E.; PENNA, M. C. *Time-Dependent QoS Management in DiffServ Networks*. International Conference on Telecommunications. ICT 2005.

[BEL05a] BELLER, A. G.; JAMHOUR, E.; PENNA, M. C. *Uma Arquitetura para Gerenciamento DiffServ com Suporte a Condições de Horário*. Simpósio Brasileiro de Redes de Computadores. SBRC 2005. (*Short paper*)

Referências Bibliográficas

[ALM99] ALMESBERGER, W.; SALIM, J.H.; KUZNETSOV, A. *Differentiated Services on Linux*. Proceedings of Globecom '99, vol. 1, pp. 831-836. 1999.

[AQU00] SALSANO, S.; et al. *Definition and usage of SLSs in the AQUILA consortium*. <draft-salsano-aquila-sls-00.txt>. IETF, November 2000.

[BER02] BERNET, Y.; BLAKE, S.; GROSSMAN, D:; SMITH, A. *An Informal Management Model for DiffServ Routers*. RFC 3290, IETF, 2002.

[BLA01] BLACK, D.; FLOYD, S.; RAMAKRISHNAN, K. *The Addition of Explicit Congestion Notification (ECN) to IP.* RFC 3168, IETF, 2001.

[BLA98] BLAKE, S.; et al. *An Architecture for Differentiated Service.* RFC 2475, IETF, 1998.

[BRA94] BRADEN, R.; CLARK, D.; SHENKER, S. *Integrated Services in the Internet Architecture: an Overview*. RFC 1633, IETF, 1994.

[BRA97] BRADEN, R.; et al. *Resourse Reservation Protocol (*RSVP). RFC 2205, IETF, 1997.

[CAS02] CASE, J.; MUNDY, R.; PARTAIN, D.; STEWART, B. *Introduction and Applicability Statements for Internet Standard Management Framework*. RFC 3410, IETF, 2002.

[CAS88] CASE, J.; et. al. *A Simple Network Management Protocol*. RFC 1067, IETF, 1988.

[CHA01] CHAN, K.; et.al. *COPS Usage for Policy Provisioning (COPS-PR)*. RFC 3084, IETF, 2001.

[CHA03] CHAN, K.; SAHITA, R.; HAHN, S.; MCCLOGHRIE, K. *Differentiated Services Quality of Service Policy Information Base*. RFC 3317, IETF, 2003.

[DAM01] DAMIANOU, N.; et al. The *Ponder Policy Specification Language.* Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks. POLICY 2001, pp 18-39.

[DMTF00] Distributed Management Task Force, Inc. *Guidelines for CIM-to-LDAP Directory Mappings*. DMTF, 2000.

[DMTF03] Distributed Management Task Force, Inc. *CIM Schema: Version 2.8*. DMTF, 2003.

[DUR00] DURHAM, D.; et.al. *The COPS (Common Open Policy Service) Protocol*. RFC 2748, IETF, 2000.

[FLE01] FLEGKAS, P.; TRIMINTZIOS, P.; PAVLOU, G.; ANDRIKOPOULOS, I.; CAVALCANTI, C. F. *On Policy-based Extensible Hierarchical Network Management in QoS-enabled IP Networks.* Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks. POLICY 2001, pp 230-246.

[FLO93] FLOYD, S.; JACOBSON, V. *Random Early Detection Gateways for Congestion Avoidance*. IEEE/ACM Transactions on Networking, vol. 1 no. 4, August 1993.

[GUE00] GUERIN, R.; PENDARAKIS, D.; YAVATKAR, R. *A Framework for Policy-based Admission Control*. RFC 2753, IETF, 2000.

[GUE97] GUERIN, R.; PARTRIDGE, C.; SHENKER, S. *Specification of Guaranteed Quality of Service.* RFC 2212, IETF, 1997.

[HEI99] HEINANEN, J.; et al. *Assured Forwarding PHB Group*. RFC 2597, IETF, 1999.

[HEI99a] HEINANEN, J.; GUERIN, R. *A Single Rate Three Color Marker*. RFC 2697, IETF, 1999.

[HEZ00] HEZORG, S.; et.al. *COPS usage for RSVP*. RFC 2749, IETF, 2000.

[JAC99] JACOBSON, V.; NICHOLS, K.; PODURI, K. *An Expedited Forwarding PHB*. RFC 2598, IETF, 1999.

[LEI00] LEINER, B. M.; et al. *A Brief History of the Internet*. Versão 3.31, http://www.isoc.org/internet/history/, 2000.

[MOO01] MOORE, B.; et al. *Policy Core Information Model*. RFC 3060, IETF, 2001.

[MOO03] MOORE, B.; et al. *Policy Core Information Model Extensions*. RFC 3460, IETF, 2003.

[MOO04] MOORE, B.; et al. *Information Model for Describing Network Device QoS Datapath Mechanisms*. RFC 3670, IETF, 2004.

[NAB03] NABHEN, R.; JAMHOUR, E.; MAZIERO C. *Policy-Based Framework for RBAC*. Proceedings of Distributed Systems Operations and Management WorkShop. DSOM 2003, pp. 181-193.

[PON02] PONNAPPAN, A.; YANG, L.; PILLAI, R.; BRAUN, P. *A Policy Based QoS Management System for the IntServ/DiffServ Based Internet*. Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks. POLICY 2002 .

[RFC791] Darpa Internet Program Protocol Specification. *Internet Protocol*. RFC791, IETF, 1981.

[SAH03] SAHITA, R.; HAHN, S.; CHAN, K.; MACCLOGHRIE, K. *Framework Policy Information Base*. RFC 3318, IETF, 2003.

[SHE97] SHENKER, S.; WROCLAWSKI, J. *General Characterization Parameters for Integrated Service Network Elements.* RFC 2215, IETF, 1997.

[SNI03] SNIR, Y.; et al. *Policy Quality of Service Information Model*. RFC 3644, IETF, 2003.

[STR04] STRASSNER, J.; MOORE, B.; MOATS, R.; ELLESSON, E. *Policy Core Lightweight Directory Access Protocol (LDAP) Schema*. RFC 3703, IETF, 2004.

[SUN03] KAKADIA, D. *Enterprise QoS Based Systems & Network Management.* Sun Microsystems White Paper, Article #8934, Volume 60, Issue 1, SysAdmin Section, 2003.

[TEQ01] GODERIS, D.; et al. *Service Level Specification Semantics, Parameters and negociation requirements.* <draft-tequila-sls-01.txt>. IETF, June 2001.

[VER01] VERMA, D.; BEIGI, M.; JENNINGS, R. *Policy based SLA Management in Enterprise Networks.* Proceedings of the IEEE Workshop on Policies for Distributed Systems and Network. POLICY 2001.

[VER02] VERMA, D. *Simplifying Network Administration using Policy based Management.* IEEE Network Magazine. March 2002.

[W3C03] W3C. *XPointer Framework*. W3C Recommendation, 2003.

[WES01] WESTERINEN, A.; et al. *Terminology for Policy-Based Management.* RFC 3198, IETF, 2001.

[WRED] Technical Specification from Cisco. *Distributed Weighted Random Early Detection*. http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf

[WRO97] WROCLAWSKI, J. *Specification of the Controlled-Load Network Element Service*. RFC 2211, IETF, 1997.

Anexo A

Esquemas XML para os Modelos Propostos

Esquema do contêiner para as políticas de alto nível

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">

  <xsd:element name="PolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SLSPolicyGroup" type="SLSPolicyGroupType"
          maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="SLSPolicyGroupType">
    <xsd:sequence>
      <xsd:element name="SLSPolicyRule" type="SLSPolicyRuleType"
        maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="SLSType" type="xsd:string"/>
    <xsd:attribute name="PolicyDecisionStrategy"
      type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="PolicyRoles" type="xsd:string"/>
    <xsd:attribute name="Enabled" type="xsd:nonNegativeInteger"/>
  </xsd:complexType>

  <xsd:complexType name="SLSPolicyRuleType">
    <xsd:sequence>
      <xsd:element name="CompoundTargetPolicyCondition"
        type="CompoundTargetPolicyConditionType" maxOccurs="unbounded"/>
      <xsd:element name="PredefinedSLSAction">
        <xsd:complexType>
          <xsd:attribute name="PredefinedSLSName" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="PolicyRuleValidityPeriod" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="PartComponent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
```

```
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string"/>
    <xsd:attribute name="ConditionListType"
      type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="RuleUsage" type="xsd:string"/>
    <xsd:attribute name="Enabled" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="Priority" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="ExecutionStrategy"
      type="xsd:nonNegativeInteger"/>
  </xsd:complexType>

  <xsd:complexType name="CompoundTargetPolicyConditionType">
    <xsd:sequence>
      <xsd:element
       name="CompoundUserPolicyConditionInCompoundTargetPolicyCondition
       " minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="GroupNumber"
            type="xsd:nonNegativeInteger"/>
          <xsd:attribute name="ConditionNegated" type="xsd:boolean"/>
          <xsd:attribute name="PartComponent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element
       name="CompoundServerPolicyConditionInCompoundTargetPolicyConditi
       on" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="GroupNumber"
            type="xsd:nonNegativeInteger"/>
          <xsd:attribute name="ConditionNegated" type="xsd:boolean"/>
          <xsd:attribute name="PartComponent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element
      name="CompoundApplicationPolicyConditionInCompoundTargetPolicyCond
      ition" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="GroupNumber"
            type="xsd:nonNegativeInteger"/>
          <xsd:attribute name="ConditionNegated" type="xsd:boolean"/>
          <xsd:attribute name="PartComponent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="ConditionListType"
      type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="Direction" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="GroupNumber" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="ConditionNegated" type="xsd:boolean"/>
  </xsd:complexType>

</xsd:schema>
```

Esquema do contêiner para as composições reutilizáveis de usuários, servidores e aplicações

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">
  <xsd:element name="PolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ReusablePolicyContainer" maxOccurs="3">
          <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
              <xsd:element name="CompoundUserPolicyCondition"
                type="CompoundConditionType"/>
              <xsd:element name="CompoundServerPolicyCondition"
                type="CompoundConditionType"/>
              <xsd:element name="CompoundApplicationPolicyCondition"
                type="CompoundConditionType"/>
            </xsd:choice>
            <xsd:attribute name="Name" type="xsd:string"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="CompoundConditionType">
    <xsd:sequence>
      <xsd:element name="SimplePolicyCondition" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="PolicyExplicitVariable">
              <xsd:complexType>
                <xsd:attribute name="ModelClass" type="xsd:string"/>
                <xsd:attribute name="ModelProperty"
                  type="xsd:string"/>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="PolicyStringValue">
              <xsd:complexType>
                <xsd:attribute name="StringList" type="xsd:string"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="GroupNumber"
            type="xsd:nonNegativeInteger"/>
          <xsd:attribute name="ConditionNegated" type="xsd:boolean"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string"/>
    <xsd:attribute name="ConditionListType"
      type="xsd:nonNegativeInteger"/>
  </xsd:complexType>

</xsd:schema>
```

Esquema do contêiner para as condições de tempo

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">

  <xsd:element name="ReusablePolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PolicyTimePeriodCondition"
           maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="Name" type="xsd:string"/>
            <xsd:attribute name="TimePeriod" type="xsd:string"/>
            <xsd:attribute name="MonthOfYearMask"
              type="xsd:Octetstring"/>
            <xsd:attribute name="DayOfMonthMask"
              type="xsd:Octetstring"/>
            <xsd:attribute name="DayOfWeekMask"
              type="xsd:Octetstring"/>
            <xsd:attribute name="TimeOfDayMask"
              type="xsd:Octetstring"/>
            <xsd:attribute name="LocalOrUtcTime"
              type="xsd:nonNegativeInteger"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Esquema do conteiner para as políticas de configuração

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">

  <xsd:element name="PolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ConfigPolicyGroup"
           type="SLSPolicyGroupType"
          maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="ConfigPolicyGroupType">
    <xsd:sequence>
      <xsd:element name="ConfigPolicyRule" type="ConfigPolicyRuleType"
        maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ConfigName" type="xsd:string"/>
    <xsd:attribute name="PolicyDecisionStrategy"
      type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="Enabled" type="xsd:nonNegativeInteger"/>
  </xsd:complexType>
```

```
<xsd:complexType name="ConfigPolicyRuleType">
  <xsd:sequence>
    <xsd:element name="ConfigQoSActionInConfigPolicyRule">
      <xsd:complexType>
        <xsd:attribute name="ParComponent" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="PolicyRuleValidityPeriod" minOccurs="0">
      <xsd:complexType>
        <xsd:attribute name="PartComponent" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="PacketFilterCondition"
     type="PacketFilterConditionType" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
  <xsd:attribute name="ConditionListType"
    type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="Enabled" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="Priority" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="ExecutionStrategy"
    type="xsd:nonNegativeInteger"/>
</xsd:complexType>

<xsd:complexType name="PacketFilterConditionTypeType">
  <xsd:sequence>
    <xsd:element name="FilterList">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="IPHeadersFilter">
            <complexType>
              <xsd:attribute name="IsNegated" type="xsd:boolean"/>
              <xsd:attribute name="HdrIPVersion"
                type="xsd:nonNegativeInteger"/>
              <xsd:attribute name="HdrSrcAddress"
                type="xsd:InetAdress"/>
              <xsd:attribute name="HdrSrcMask"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrDestAddress"
                type="xsd:InetAdress"/>
              <xsd:attribute name="HdrDestMask"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrProtocolID"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrSrcPortStart"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrSrcPortEnd"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrDestPortStart"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrDestPortEnd"
                type="nonNegativeInteger"/>
              <xsd:attribute name="HdrDSCP"
                type="nonNegativeInteger"/>
            </complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="Direction"
```

```
            type="xsd:nonNegativeInteger"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="FilterEvaluation"
      type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="GroupNumber" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="ConditionNegated" type="xsd:boolean"/>
  </xsd:complexType>

</xsd:schema>
```

Esquema do conteiner para os níveis de serviço pré-definidos no sistema (composições de ações reutilizáveis)

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">

  <xsd:element name="ReusablePolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CompoundPolicyAction"
           maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="PolicyActionInPolicyAction"
                 maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:attribute name="ActionOrder"
                    type="xsd:nonNegativeInteger"/>
                  <xsd:attribute name="PartComponent"
                    type="xsd:string"/>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
            <xsd:attribute name="SequencedActions"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="ExecutionStrategy"
              type="xsd:nonNegativeInteger"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="Name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Esquema do contêiner para as ações QPIM reutilizáveis

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">

  <xsd:element name="ReusablePolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="QoSPolicyBandwidthAction" minOccurs="0"
           maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="Name" type="xsd:string"/>
            <xsd:attribute name="qpMaxPacketSize"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpForwardingPriority"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpBandwidthUnits"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpMinBandwidth"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpMaxBandwidth"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpMaxDelay"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpMaxJitter"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpFairness"
              type="xsd:nonNegativeInteger"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="QoSPolicyCongestionControlAction"
           minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="Name" type="xsd:string"/>
            <xsd:attribute name="qpMaxPacketSize"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpQueueSizeUnits"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpQueueSize"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpDropMethod"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpDropThresholdUnits"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpDropMinThresholdValue"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpDropMaxThresholdValue"
              type="xsd:nonNegativeInteger"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="QoSPolicyPoliceAction"
           type="QoSPolicyPoliceActionType" minOccurs="0"
           maxOccurs="unbounded"/>
        <xsd:element name="SimplePolicyAction "
           type="SimplePolicyActionType" minOccurs="0"
           maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:complexType name="QoSPolicyPoliceActionType">
    <xsd:sequence>
      <xsd:element name="QoSPolicyTrfcProfInAdmissionAction">
        <xsd:complexType>
          <xsd:attribute name="Dependent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="PolicyConformAction">
        <xsd:complexType>
          <xsd:attribute name="Dependent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="PolicyExceedAction">
        <xsd:complexType>
          <xsd:attribute name="Dependent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="PolicyViolateAction">
        <xsd:complexType>
          <xsd:attribute name="Dependent" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string"/>
    <xsd:attribute name="qpAdmissionScope"
      type="xsd:nonNegativeInteger"/>
  </xsd:complexType>

  <xsd:complexType name="SimplePoliceActionType">
    <xsd:sequence>
      <xsd:element name="PolicyDSCPVariable"/>
      <xsd:element name="PolicyIntegerValue">
        <xsd:complexType>
          <xsd:attribute name="IntegerList"
            type="xsd:nonNegativeInteger"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string"/>
  </xsd:complexType>

</xsd:schema>
```

Esquema do contêiner para os parâmetros *token bucket* reutilizáveis (perfis de tráfego)

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">

  <xsd:element name="ReusablePolicyContainer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="QoSPolicyTokenBucketTrfcProf"
          maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="Name" type="xsd:string"/>
            <xsd:attribute name="qpTBRate"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpTBNormalBurst"
              type="xsd:nonNegativeInteger"/>
            <xsd:attribute name="qpTBExcessBurst"
              type="xsd:nonNegativeInteger"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Anexo B

Arquivos XML do Estudo de Caso - Rede Coorporativa

Regras de Alto Nível

```
<PolicyContainer Name="BusinessLevelPolicy">
  <SLSPolicyGroup SLSType="Hierarchical" PolicyDecisionStrategy="2">
    <SLSPolicyRule Name="Rule1" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="2">
      <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">

<CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false" ServerOrClientApp="1"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplication
PolicyCondition[@Name='Internet'])"/>
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolic
yCondition[@Name='InternetServer'])"/>
      </CompoundTargetPolicyCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="AF31+15%BW"/>
      <PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time1'])"/>
      <SLSPolicyRule Name="Rule1a" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Directors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule1b" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
```

```
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Managers'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="25%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule1c" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Supervisors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="15%BW"/>
      </SLSPolicyRule>
    </SLSPolicyRule>
    <SLSPolicyRule Name="Rule2" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="2">
      <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">

<CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false" ServerOrClientApp="1"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplication
PolicyCondition[@Name='EnterpriseSystems'])"/>
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolic
yCondition[@Name='EnterpriseServer'])"/>
      </CompoundTargetPolicyCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="AF21+25%BW"/>
      <PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time1'])"/>
      <SLSPolicyRule Name="Rule2a" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Directors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule2b" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Managers'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="25%BW"/>
      </SLSPolicyRule>
```

```
      <SLSPolicyRule Name="Rule2c" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Supervisors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="15%BW"/>
      </SLSPolicyRule>
    </SLSPolicyRule>
    <SLSPolicyRule Name="Rule3" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="2">
      <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">

<CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false" ServerOrClientApp="1"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplication
PolicyCondition[@Name='Multimedia'])"/>
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolic
yCondition[@Name='MultimediaServer'])"/>
      </CompoundTargetPolicyCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="AF11+40%BW"/>
      <PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time1'])"/>
      <SLSPolicyRule Name="Rule3a" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Directors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule3b" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Managers'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="25%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule3c" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
```

```
ondition[@Name='Supervisors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="15%BW"/>
      </SLSPolicyRule>
    </SLSPolicyRule>
    <SLSPolicyRule Name="Rule4" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="2">
      <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">

<CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false" ServerOrClientApp="1"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplication
PolicyCondition[@Name='Internet'])"/>
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolic
yCondition[@Name='InternetServer'])"/>
      </CompoundTargetPolicyCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="AF21+20%BW"/>
      <PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time2'])"/>
      <SLSPolicyRule Name="Rule4a" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Directors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule4b" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Managers'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="25%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule4c" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Supervisors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="15%BW"/>
      </SLSPolicyRule>
    </SLSPolicyRule>
    <SLSPolicyRule Name="Rule5" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="2">
```

```
      <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">

<CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false" ServerOrClientApp="1"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplication
PolicyCondition[@Name='EnterpriseSystems'])"/>
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolic
yCondition[@Name='EnterpriseServer'])"/>
      </CompoundTargetPolicyCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="AF31+20%BW"/>
      <PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time2'])"/>
      <SLSPolicyRule Name="Rule5a" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Directors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule5b" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Managers'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="25%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule5c" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Supervisors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="15%BW"/>
      </SLSPolicyRule>
    </SLSPolicyRule>
    <SLSPolicyRule Name="Rule6" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="2">
      <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">

<CompoundApplicationPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false" ServerOrClientApp="1"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundApplication
PolicyCondition[@Name='Multimedia'])"/>
```

```
        <CompoundServerPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundServerPolic
yCondition[@Name='MultimediaServer'])"/>
      </CompoundTargetPolicyCondition>
      <PredefinedSLSPolicyAction PredefinedSLSName="AF11+40%BW"/>
      <PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time2'])"/>
      <SLSPolicyRule Name="Rule6a" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Directors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="40%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule6b" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Managers'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="25%BW"/>
      </SLSPolicyRule>
      <SLSPolicyRule Name="Rule6c" RuleUsage="..." Enabled="1"
ConditionListType="1" ExecutionStrategy="2" Priority="1">
        <CompoundTargetPolicyCondition ConditionListType="1"
Direction="2" GroupNumber="1" ConditionNegated="false">
          <CompoundUserPolicyConditionInCompoundTargetPolicyCondition
GroupNumber="1" ConditionNegated="false"
PartComponent="./CompoundConditions.xml#xpointer(//CompoundUserPolicyC
ondition[@Name='Supervisors'])"/>
        </CompoundTargetPolicyCondition>
        <PredefinedSLSPolicyAction PredefinedSLSName="15%BW"/>
      </SLSPolicyRule>
    </SLSPolicyRule>
  </SLSPolicyGroup>
</PolicyContainer>
```

Condições Compostas Reutilizáveis

```
<PolicyContainer>
    <ReusablePolicyContainer Name="ReusableCompoundUserConditions">
      <CompoundUserPolicyCondition Name="Supervisors"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="Person"
```

```
ModelProperty="BusinessCategory"/>
          <PolicyStringValue StringList="Supervisor"/>
        </SimplePolicyCondition>
      </CompoundUserPolicyCondition>
      <CompoundUserPolicyCondition Name="Managers"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="Person"
ModelProperty="BusinessCategory"/>
          <PolicyStringValue StringList="Manager"/>
        </SimplePolicyCondition>
      </CompoundUserPolicyCondition>
      <CompoundUserPolicyCondition Name="Directors"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="Person"
ModelProperty="BusinessCategory"/>
          <PolicyStringValue StringList="Director"/>
        </SimplePolicyCondition>
      </CompoundUserPolicyCondition>
    </ReusablePolicyContainer>

    <ReusablePolicyContainer Name="ReusableCompoundServerConditions">
      <CompoundServerPolicyCondition Name="InternetServer"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="UnitaryComputerSystem"
ModelProperty="Name"/>
          <PolicyStringValue StringList="InternetServer"/>
        </SimplePolicyCondition>
      </CompoundServerPolicyCondition>
      <CompoundServerPolicyCondition Name="MultimediaServer"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="UnitaryComputerSystem"
ModelProperty="Name"/>
          <PolicyStringValue StringList="MultimediaServer"/>
        </SimplePolicyCondition>
      </CompoundServerPolicyCondition>
      <CompoundServerPolicyCondition Name="EnterpriseServer"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="UnitaryComputerSystem"
ModelProperty="Name"/>
          <PolicyStringValue StringList="EnterpriseServer"/>
        </SimplePolicyCondition>
      </CompoundServerPolicyCondition>
    </ReusablePolicyContainer>

    <ReusablePolicyContainer
Name="ReusableCompoundApplicationConditions">
      <CompoundApplicationPolicyCondition Name="Internet"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
```

```
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="InstalledProduct"
ModelProperty="Name"/>
          <PolicyStringValue StringList="ApacheHTTP"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="SoftwareFeature"
ModelProperty="Version"/>
          <PolicyStringValue StringList="1.3"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="2"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="InstalledProduct"
ModelProperty="Name"/>
          <PolicyStringValue StringList="SurgeMail"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="3"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="InstalledProduct"
ModelProperty="Name"/>
          <PolicyStringValue StringList="SurgeFTP"/>
        </SimplePolicyCondition>
      </CompoundApplicationPolicyCondition>
      <CompoundApplicationPolicyCondition Name="Multimedia"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="InstalledProduct"
ModelProperty="Name"/>
          <PolicyStringValue StringList="HelixMediaServer"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="SoftwareFeature"
ModelProperty="Name"/>
          <PolicyStringValue StringList="MMS"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="2"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="InstalledProduct"
ModelProperty="Name"/>
          <PolicyStringValue StringList="HelixMediaServer"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="2"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="SoftwareFeature"
ModelProperty="Name"/>
          <PolicyStringValue StringList="RDP/RDT"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="3"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="InstalledProduct"
ModelProperty="Name"/>
          <PolicyStringValue StringList="NetMeeting"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="3"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="SoftwareFeature"
```

```
ModelProperty="Name"/>
          <PolicyStringValue StringList="H.323Stream"/>
        </SimplePolicyCondition>
      </CompoundApplicationPolicyCondition>
      <CompoundApplicationPolicyCondition Name="EnterpriseSystems"
ConditionListType="1">
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="SoftwareFeature"
ModelProperty="Name"/>
          <PolicyStringValue StringList="CustomerDataBase"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="1"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="SoftwareFeature"
ModelProperty="Version"/>
          <PolicyStringValue StringList="2.3"/>
        </SimplePolicyCondition>
        <SimplePolicyCondition GroupNumber="2"
ConditionNegated="false">
          <PolicyExplicitVariable ModelClass="ApplicationSystem"
ModelProperty="Name"/>
          <PolicyStringValue StringList="FinancialSystem"/>
        </SimplePolicyCondition>
      </CompoundApplicationPolicyCondition>
    </ReusablePolicyContainer>
</PolicyContainer>
```

Usuários

```
<UserEntry>
  <Person Name="AndreBeller" BusinessCategory="Manager"
OU="Commercial">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.2.3/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="EdgardJamhour" BusinessCategory="Director"
OU="Commercial">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.2.2/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="TimothySquair" BusinessCategory="Supervisor"
OU="Commercial">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.2.4/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="EzequielMazza" BusinessCategory="Manager"
OU="Engineering">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.3.3/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
```

```
  <Person Name="MarceloPellenz" BusinessCategory="Director"
OU="Engineering">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.3.2/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="JoseSilva" BusinessCategory="Supervisor"
OU="Engineering">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.3.4/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="MauroFonseca" BusinessCategory="Director"
OU="Financial">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.1.2/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="LuizAugusto" BusinessCategory="Manager"
OU="Financial">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.1.3/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
  <Person Name="MarceloZanetti" BusinessCategory="Supervisor"
OU="Financial">
    <SAPAvailibletoElement>
      <RemoteServiceAccessPoint AccessInfo="10.0.1.4/24"
InfoFormat="3"/>
    </SAPAvailibletoElement>
  </Person>
</UserEntry>
```

Servidores

```
<Server>
  <UnitaryComputerSystem Name="InternetServer">
    <HostedAccessPoint>
      <RemoteServiceAccessPoint AccessInfo="10.0.4.3/24"
InfoFormat="3"/>
    </HostedAccessPoint>
  </UnitaryComputerSystem>
  <UnitaryComputerSystem Name="MultimediaServer">
    <HostedAccessPoint>
      <RemoteServiceAccessPoint AccessInfo="10.0.4.4/24"
InfoFormat="3"/>
    </HostedAccessPoint>
  </UnitaryComputerSystem>
  <UnitaryComputerSystem Name="EnterpriseServer">
    <HostedAccessPoint>
      <RemoteServiceAccessPoint AccessInfo="10.0.4.2/24"
InfoFormat="3"/>
    </HostedAccessPoint>
  </UnitaryComputerSystem>
</Server>
```

Aplicações

```
<Application>
  <InstalledProduct Name="ApacheHTTP">
      <SoftwareFeature Name="WebServer" Version="1.3">
         <TCPProtocolEndPoint PortNumber="80"/>
      </SoftwareFeature>
  </InstalledProduct>
  <InstalledProduct Name="SurgeMail">
      <SoftwareFeature Name="SMTP">
         <TCPProtocolEndPoint PortNumber="25"/>
      </SoftwareFeature>
      <SoftwareFeature Name="POP3">
         <TCPProtocolEndPoint PortNumber="110"/>
      </SoftwareFeature>
      <SoftwareFeature Name="IMAP">
         <TCPProtocolEndPoint PortNumber="143"/>
      </SoftwareFeature>
  </InstalledProduct>
  <InstalledProduct Name="SurgeFTP">
      <SoftwareFeature Name="FTP">
         <TCPProtocolEndPoint PortNumber="21"/>
      </SoftwareFeature>
  </InstalledProduct>
  <InstalledProduct Name="HelixMediaServer">
      <SoftwareFeature Name="HTTP">
         <TCPProtocolEndPoint PortNumber="8080"/>
      </SoftwareFeature>
      <SoftwareFeature Name="RTSP">
         <TCPProtocolEndPoint PortNumber="554"/>
      </SoftwareFeature>
      <SoftwareFeature Name="PNA">
         <TCPProtocolEndPoint PortNumber="7070"/>
      </SoftwareFeature>
      <SoftwareFeature Name="MMS">
         <UDPProtocolEndPoint PortNumber="4090..5000"/>
      </SoftwareFeature>
      <SoftwareFeature Name="RDP/RDT">
         <UDPProtocolEndPoint PortNumber="34445..34459"/>
      </SoftwareFeature>
  </InstalledProduct>
  <ApplicationSystem Name="FinancialSystem">
      <SoftwareFeature Name="Payments" Version="2.0">
         <TCPProtocolEndPoint PortNumber="49152"/>
      </SoftwareFeature>
  </ApplicationSystem>
  <ApplicationSystem Name="CommercialSystem">
      <SoftwareFeature Name="CustomerDataBase" Version="2.3">
         <TCPProtocolEndPoint PortNumber="49153"/>
      </SoftwareFeature>
  </ApplicationSystem>
  <InstalledProduct Name="NetMeeting">
      <SoftwareFeature Name="ILS">
         <TCPProtocolEndPoint PortNumber="389"/>
      </SoftwareFeature>
      <SoftwareFeature Name="ULS">
         <TCPProtocolEndPoint PortNumber="522"/>
      </SoftwareFeature>
```

```
      <SoftwareFeature Name="T.120">
         <TCPProtocolEndPoint PortNumber="1503"/>
      </SoftwareFeature>
      <SoftwareFeature Name="H.323SetUp">
         <TCPProtocolEndPoint PortNumber="1720"/>
      </SoftwareFeature>
      <SoftwareFeature Name="AudioControl">
         <TCPProtocolEndPoint PortNumber="1731"/>
      </SoftwareFeature>
      <SoftwareFeature Name="H.323Stream">
         <UDPProtocolEndPoint PortNumber="1024..65535"/>
      </SoftwareFeature>
  </InstalledProduct>
</Application>
```

Sub-redes

```
<ManagedSystemElement>
  <Network name="financial.company.com">
    <IPConnectivityCollection SubnetNumber="10.0.1.0"
SubnetMask="24"/>
    <PolicyRoleCollectionInSystem
Dependent="./RoleCollection.xml#xpointer(//PolicyRoleCollection[@Polic
yRole='EdgeFinancial'])"/>
  </Network>
  <Network name="commercial.company.com">
    <IPConnectivityCollection SubnetNumber="10.0.2.0"
SubnetMask="24"/>
    <PolicyRoleCollectionInSystem
Dependent="./RoleCollection.xml#xpointer(//PolicyRoleCollection[@Polic
yRole='EdgeCommercial'])"/>
  </Network>
  <Network name="engineering.company.com">
    <IPConnectivityCollection SubnetNumber="10.0.3.0"
SubnetMask="24"/>
    <PolicyRoleCollectionInSystem
Dependent="./RoleCollection.xml#xpointer(//PolicyRoleCollection[@Polic
yRole='EdgeEngineering'])"/>
  </Network>
  <Network name="servers.company.com">
    <IPConnectivityCollection SubnetNumber="10.0.4.0"
SubnetMask="24"/>
    <PolicyRoleCollectionInSystem
Dependent="./RoleCollection.xml#xpointer(//PolicyRoleCollection[@Polic
yRole='EdgeServers'])"/>
  </Network>
  <Network name="backbone.company.com">
    <IPConnectivityCollection SubnetNumber="192.168.0.0"
SubnetMask="24"/>
    <PolicyRoleCollection
Dependent="./RoleCollection.xml#xpointer(//PolicyRoleCollection[@Polic
yRole='Core'])"/>
  </Network>
</ManagedSystemElement>
```

Associação de Papéis a Conjuntos de Políticas

```
<Collection>
```

```
  <SystemSpecificCollection InstanceId="1">
    <PolicyRoleCollection PolicyRole="EdgeCommercial">
      <PolicySetInPolicyRoleCollection
Member="./ConfigResult.xml#xpointer(//ConfigPolicyGroup[@ConfigName='C
onfigQoSCommercial'])"/>
    </PolicyRoleCollection>
  </SystemSpecificCollection>
  <SystemSpecificCollection InstanceId="2">
    <PolicyRoleCollection PolicyRole="EdgeFinancial">
      <PolicySetInPolicyRoleCollection
Member="./ConfigResult.xml#xpointer(//ConfigPolicyGroup[@ConfgiName='C
onfigQoSFinancial'])"/>
    </PolicyRoleCollection>
  </SystemSpecificCollection>
  <SystemSpecificCollection InstanceId="3">
    <PolicyRoleCollection PolicyRole="EdgeEngineering">
      <PolicySetInPolicyRoleCollection
Member="./ConfigResult.xml#xpointer(//ConfigPolicyGroup[@ConfigName='C
onfigQoSEngineering'])"/>
    </PolicyRoleCollection>
  </SystemSpecificCollection>
  <SystemSpecificCollection InstanceId="4">
    <PolicyRoleCollection PolicyRole="EdgeServers">
      <PolicySetInPolicyRoleCollection
Member="./ConfigResult.xml#xpointer(//ConfigPolicyGroup[@ConfigName='C
onfigQoSServers'])"/>
    </PolicyRoleCollection>
  </SystemSpecificCollection>
  <SystemSpecificCollection InstanceId="5">
    <PolicyRoleCollection PolicyRole="Core">
      <PolicySetInPolicyRoleCollection
Member="./ConfigResult.xml#xpointer(//ConfigPolicyGroup[@ConfigName='C
onfigQoSCore'])"/>
    </PolicyRoleCollection>
  </SystemSpecificCollection>
</Collection>
```

Condições reutilizáveis de Horário

```
<Time>
  <PolicyTimePeriodCondition Name="Time1"
TimePeriod="20040901T000000/THISANDFUTURE" DayOfWeekMask="01111100"
TimeOfDayMask="T080000/115959;T140000/175959" LocalOrUtcTime="1"/>
  <PolicyTimePeriodCondition Name="Time2"
TimePeriod="20040901T000000/THISANDFUTURE" DayOfWeekMask="01111100"
TimeOfDayMask="T000000/075959;T120000/135959;T180000/235959"
LocalOrUtcTime="1"/>
</Time>
```

Níveis de Serviço Pré-definidos no Sistema

```
<ReusablePolicyContainer Name="SLSActions">
  <CompoundPolicyAction Name="AF11+40%BW" SequencedActions="1"
ExecutionStrategy="2">
```

```
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//SimplePolicyAction[@Name='
AF11'])"/>
    <PolicyActionInPolicyAction ActionOrder="2"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='4
0%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="AF21+25%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//SimplePolicyAction[@Name='
AF21'])"/>
    <PolicyActionInPolicyAction ActionOrder="2"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='2
5%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="AF31+15%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//SimplePolicyAction[@Name='
AF31'])"/>
    <PolicyActionInPolicyAction ActionOrder="2"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='1
5%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="AF21+20%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//SimplePolicyAction[@Name='
AF21'])"/>
    <PolicyActionInPolicyAction ActionOrder="2"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='2
0%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="AF31+20%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//SimplePolicyAction[@Name='
AF31'])"/>
    <PolicyActionInPolicyAction ActionOrder="2"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='2
0%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="40%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='4
0%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="25%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='2
5%'])"/>
  </CompoundPolicyAction>
  <CompoundPolicyAction Name="15%BW" SequencedActions="1"
ExecutionStrategy="2">
    <PolicyActionInPolicyAction ActionOrder="1"
PartComponent="./QpimActions.xml#xpointer(//QosPolicyBWAction[@Name='1
```

```
5%'])"/>
  </CompoundPolicyAction>
</ReusablePolicyContainer>
```

Ações QPIM reutilizáveis

```
<ReusablePolicyContainer Name="ReusableQpimActions">
  <QoSPolicyBandwidthAction Name="40%" qpBandwidthUnits="1"
qpMinBandwidth="40"/>
  <QoSPolicyBandwidthAction Name="25%" qpBandwidthUnits="1"
qpMinBandwidth="25"/>
  <QoSPolicyBandwidthAction Name="20%" qpBandwidthUnits="1"
qpMinBandwidth="20"/>
  <QoSPolicyBandwidthAction Name="15%" qpBandwidthUnits="1"
qpMinBandwidth="15"/>
  <SimplePolicyAction Name="AF11">
    <PolicyDSCPVariable/>
    <PolicyIntegerValue IntegerList="AF11"/>
  </SimplePolicyAction>
  <SimplePolicyAction Name="AF21">
    <PolicyDSCPVariable/>
    <PolicyIntegerValue IntegerList="AF21"/>
  </SimplePolicyAction>
  <SimplePolicyAction Name="AF31">
    <PolicyDSCPVariable/>
    <PolicyIntegerValue IntegerList="AF31"/>
  </SimplePolicyAction>
</ReusablePolicyContainer>
```

Resultado das Políticas de Configuração

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicyContainer Name="ConfigPolicy">

<ConfigPolicyGroup ConfigName="ConfigQoSServers"
PolicyDecisionStrategy="2">
  <ConfigPolicyRule Name="Rule1" Enabled="1" ConditionListType="1"
Priority="2" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'AF31+15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time1'])">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="80"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
```

```
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="25"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="110"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="143"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="21"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><ConfigPolicyRule Name="Rule1a" Enabled="1"
ConditionListType="1" Priority="1" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
```

```
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule1b" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
```

```
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule1c" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
```

```
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>

</PacketFilterCondition></ConfigPolicyRule></ConfigPolicyRule><ConfigP
olicyRule Name="Rule2" Enabled="1" ConditionListType="1" Priority="2"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'AF21+25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time1'])">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.2" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="49153"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.2" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="49152"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><ConfigPolicyRule Name="Rule2a" Enabled="1"
ConditionListType="1" Priority="1" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
```

```
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule2b" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
```

```
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule2c" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
```

```
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>

</PacketFilterCondition></ConfigPolicyRule></ConfigPolicyRule><ConfigP
olicyRule Name="Rule3" Enabled="1" ConditionListType="1" Priority="2"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'AF11+40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time1'])">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.4" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="udp" HdrSrcPortStart="4090"
HdrSrcPortEnd="5000" HdrDestPortStart="" HdrDestPortEnd=""
HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.4" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
```

```
HdrDestMask="0" HdrProtocolID="udp" HdrSrcPortStart="34445"
HdrSrcPortEnd="34459" HdrDestPortStart="" HdrDestPortEnd=""
HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.4" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="udp" HdrSrcPortStart="1024"
HdrSrcPortEnd="65535" HdrDestPortStart="" HdrDestPortEnd=""
HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><ConfigPolicyRule Name="Rule3a" Enabled="1"
ConditionListType="1" Priority="1" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
```

```
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule3b" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
```

```
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule3c" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
```

```
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>

</PacketFilterCondition></ConfigPolicyRule></ConfigPolicyRule><ConfigP
olicyRule Name="Rule4" Enabled="1" ConditionListType="1" Priority="2"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'AF21+20%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time2'])">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="80"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="25"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="110"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="143"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.3" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="21"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><ConfigPolicyRule Name="Rule4a" Enabled="1"
ConditionListType="1" Priority="1" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
```

```
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule4b" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
```

```
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule4c" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.4"
```

```
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>

</PacketFilterCondition></ConfigPolicyRule></ConfigPolicyRule><ConfigP
olicyRule Name="Rule5" Enabled="1" ConditionListType="1" Priority="2"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'AF31+20%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time2'])">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.2" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="49153"
```

```
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.2" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="tcp" HdrSrcPortStart="49152"
HdrSrcPortEnd="" HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><ConfigPolicyRule Name="Rule5a" Enabled="1"
ConditionListType="1" Priority="1" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
```

```
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule5b" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
```

```
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule5c" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
```

```
</PacketFilterCondition></ConfigPolicyRule></ConfigPolicyRule><ConfigP
olicyRule Name="Rule6" Enabled="1" ConditionListType="1" Priority="2"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'AF11+40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="./Time.xml#xpointer(//PolicyTimePeriodCondition[@Name='
Time2'])">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.4" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="udp" HdrSrcPortStart="4090"
HdrSrcPortEnd="5000" HdrDestPortStart="" HdrDestPortEnd=""
HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.4" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="udp" HdrSrcPortStart="34445"
HdrSrcPortEnd="34459" HdrDestPortStart="" HdrDestPortEnd=""
HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="10.0.4.4" HdrSrcMask="24" HdrDestAddress="0.0.0.0"
HdrDestMask="0" HdrProtocolID="udp" HdrSrcPortStart="1024"
HdrSrcPortEnd="65535" HdrDestPortStart="" HdrDestPortEnd=""
HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><ConfigPolicyRule Name="Rule6a" Enabled="1"
ConditionListType="1" Priority="1" ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'40%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
```

```
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.2"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule6b" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'25%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
```

```
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.3"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition></ConfigPolicyRule><ConfigPolicyRule
Name="Rule6c" Enabled="1" ConditionListType="1" Priority="1"
ExecutionStrategy="2">
  <ConfigQoSActionInConfigPolicyRule
PartComponent="./SLSActions.xml#xpointer(//CompoundPolicyAction[@Name=
'15%BW'])">
  </ConfigQoSActionInConfigPolicyRule><PolicyRuleValidityPeriod
PartComponent="">
  </PolicyRuleValidityPeriod><PacketFilterCondition
FilterEvaluation="2" GroupNumber="1" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.2.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="2" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="3" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.3.4"
```

```
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="4" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="5" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>
  </PacketFilterCondition><PacketFilterCondition FilterEvaluation="2"
GroupNumber="6" ConditionNegated="false">
    <FilterList Direction="1">
      <IPHeadersFilter IsNegated="false" HdrIPVersion="4"
HdrSrcAddress="0.0.0.0" HdrSrcMask="0" HdrDestAddress="10.0.1.4"
HdrDestMask="24" HdrProtocolID="" HdrSrcPortStart="" HdrSrcPortEnd=""
HdrDestPortStart="" HdrDestPortEnd="" HdrDSCP=""/>
    </FilterList>

</PacketFilterCondition></ConfigPolicyRule></ConfigPolicyRule></Config
PolicyGroup></PolicyContainer>
```