MARCELO ZANETTI

PROPOSTA DE ARQUITETURA DE MEDIÇÃO E ESTIMATIVA DE TRÁFEGO EM REDES DIFFSERV

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

2006

MARCELO ZANETTI

PROPOSTA DE ARQUITETURA DE MEDIÇÃO E ESTIMATIVA DE TRÁFEGO EM REDES DIFFSERV

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
Orientador: Prof. Dr. Edgard Jamhour

CURITIBA

2006

Zanetti, Marcelo
Z28p Proposta de arquitetura de medição e estimativa de tráfego em redes
2006 Diffserv / Marcelo Zanetti ; orientador, Edgard Jamhour. – 2006.
xvi, 120 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2006
Inclui bibliografia

1. Arquitetura de redes de computadores. 2. Redes de computação - Protocolos. I. Jamhour, Edgard. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada. III. Título.

CDD 20. ed. – 004.65
004.62

Dedico a meus pais.

Agradecimentos

Agradeço especialmente ao Prof. Edgard Jamhour que durante todo o período de pesquisa e desenvolvimento sempre se colocou a disposição. Manifesto meus sinceros agradecimentos ao mesmo, profissional que faz jus às palavras **Professor** e **Orientador**.

Às contribuições do professor Mauro Fonseca.

A André Beller que forneceu seu código e sempre se pos a disposição para explicar o mesmo.

A minha namorada que sempre esteve presente nos momentos de alegria e principalmente nas horas difíceis incentivando.

Aos companheiros de apartamento de souberam entender meus momentos de irritação quando alguma coisa dava errada.

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

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ACCT	*Timer Value*
AF	*Assured Forwarding*
BA	*Behavior Aggregate*
BE	*Best Effort*
BER	*Binary Encoding Rules*
BMP	*Bandwidth Management Point*
CBQ	*Class Based Queue*
COPS	*Common Open Policy Service*
COPS-PR	*Common Open Policy Service for Policy Provisioning*
CU	*currently Unused*
DEC	*Decision*
DiffServ	*Differentiated Services*
DS	*Differentiated Services*
DSCP	*Differentiated Services CodePoint*
ECN	*Explicit Congestion Notification*
EF	*Expedited Forwarding*
EWMA	*Exponential Weighted Moving Average*
FIAC	*Fair Intelligent Admission Control*
GRED	*Generalized Random Early Detection*
HTB	*Hierarchical Token Bucket*
IETF	*Internet Engineering Task Force*
IntServ	*Integrated Services*
IP	*Internet Protocol*
IPSec	*Internet Protocol Security*
LDAP	*Lightweight Directory Access Protocol*
KA	*Keep Alive*
MF	*Multifield*
MBAC	*Measurement-based AC*
MTU	*Maximum Transmission Unit*
NTP	*Network Time Protocol*
OID	*Object Identifier*

OPN	*Client Open*
PBAC	*Parameter-based AC*
PBN	*Policy Based Network*
PBNM	*Policy Based Network Management*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PHB	*Per-Hop Behavior*
PIB	*Policy Information Base*
PRC	*Provisioning Class*
PRI	*Provisioning Instance*
PRID	*Provisioning Instance Identifier*
QoS	*Quality of Service*
RAP	*Resource Allocation Protocol*
RD	*Resource Discovery*
RED	*Random Early Detection*
REQ	*Requisition*
RFC	*Request for Comments*
RPT	*Report State*
RSVP	*Resource Reservation Protocol*
SLA	*Service Level Agreement*
SRRP	*Sender-initiated Resource Reservation Protocol*
TCA	*Traffic Conditioning Agreement*
TCB	*Traffic Conditioning Block*
TCP	*Transmission Control Protocol*
TM	*Traffic Matrix*
TMAC	*Traffic Matrix based Admission Control*
TLS	*Transport Layer Security*
TOS	*Type of Service*
UDP	*User Datagram Protocol*
U	*Upper-Bound*
WAN	*Wide Area Network*
WFQ	*Weighted Fair Queueing*
WRR	*Weighted Round Robin*
XML	*Extensible Markup Language*

Resumo

Este trabalho apresenta um sistema de medição e estimativa de tráfego para redes que implementam a metodologia de QoS de Serviços Diferenciados. O trabalho propõe a utilização do modelo de Gerência de Redes Baseada em Políticas (PBNM) para realizar a coleta de dados dos roteadores de borda e de núcleo do domínio DiffServ. A arquitetura é baseada nos padrões do IETF (*Internet Engineering Task Force*) e define uma nova PIB (*Policy Information Base*) que permite padronizar a forma como os dados coletados de um domínio de DiffServ são representados. A PIB proposta é uma especialização para DiffServ da PIB genérica definida pelo IETF. Para realizar o transporte das informações estatísticas coletadas dos dispositivos de rede é utilizado o protocolo COPS-PR.

Com base nos dados coletados dos dispositivos de rede e posteriormente agrupados no PDP (*Policy Decision Point*), são feitas estimativas de atraso fim-a-fim para os fluxos de dados transportados pela rede. A estratégia baseada em estimativa do atraso pelos nós é comparada com a estratégia fim-a-fim convencional, utilizando como estudo de caso um fluxo de vídeo transmitido em tempo-real.

Palavras-chave: DiffServ, Medição, PIB, PBNM.

Abstract

This work presents a framework for estimating the delay in networks that implements the Differentiated Services QoS methodology. The work considers the use of the Policy Based Network Management (PBNM) model to perform the collection of data from the core and border routers in a DiffServ domain. The architecture is based on the IETF (Internet Engineering Task Force) standards and defines new PIB (Policy Information Base) which permits standardize the form how the collected data of DiffServ domain are represented. The PIB proposed is a specialization for DiffServ of the generic PIB defined by the IETF. To carry the collected statistical information from the network devices is used the COPS-PR protocol.

Based on the collected data from the network devices and grouped on PDP (Policy Decision Point), are performed valuations of end-to-end delay to the data flows carried by the network. The strategy based on the network devices information is confronted with the conventional end-to-end measurement strategy by using a case study considering the transmission of a real-time video flow.

Keywords*: DiffServ, Measurement, PIB, PBNM.*

Capítulo 1

Introdução

1.1 Desafio

O avanço das redes de computadores e, na mesma medida a estabilidade do protocolo IP (*Internet Protocol*) possibilitou uma rede com uma variedade infinita de sistemas e meios de transmissão. Troca de e-mails e navegação na WEB fazem parte do dia a dia das pessoas a título de trabalho, estudo e lazer. Outras redes como a de telefonia, de rádio e de televisão também estão convergindo para redes IP se aproveitado de sua grande abrangência e flexibilidade, neste cenário, tipos diferentes de tráfego devem ser tratados de modo diferenciado. Algumas aplicações, como as de áudio, têm exigências rígidas quanto ao atraso fim a fim, mas podem tolerar perdas mínimas de pacotes, enquanto outras como transferências de arquivos são sensíveis a este último aspecto, mas as exigências quanto ao atraso são pouco severas [TEC, 99].

A necessidade de Qualidade de Serviço (QoS) é evidente quando se busca utilizar eficientemente canais de comunicação em redes de computadores. O IETF (*Internet Engineering Task Force*) propôs então dois modelos para qualidade de serviço (QoS) [KAM, 00] na rede, IntServ [BRA, 94] e DiffServ [BLA, 98]. DiffServ é um modelo de QoS baseado em acordos (SLAs - *Service Level Agreement*) [DOM, 00]. Um SLA é um acordo estabelecido entre um cliente e um prestador de serviços para determinar como o fluxo de dados deve ser tratado dentro de um domínio de Serviços Diferenciados, definindo os indicadores de qualidade e os níveis que estes devem possuir.

Estes contratos devem definir claramente os níveis de serviço esperados, especificando o desempenho a que o prestador se compromete através de parâmetros objetivos como disponibilidade da infra-estrutura e comunicações, que permitem medir a qualidade do serviço e que serão monitorados durante a duração do tempo do contrato.

Dessa forma, o contrato de SLA especifica os níveis de serviços ou padrões de desempenho, passando a ser uma peça fundamental na comunicação e no negócio da empresa.

Quantifica ainda o serviço mínimo aceitável, fazendo com que a qualidade do nível de serviço contratado seja satisfatório e atenda às necessidades desse usuário.

Os clientes não exigem apenas qualidade, mas também a comprovação do nível de excelência do serviço prestado. Por esse motivo não bastam apenas modelos e contratos, torna-se necessário o monitoramento do serviço e o gerenciamento dos indicadores. Qual a banda de tráfego disponível e desempenho dos serviços? Estas são perguntas que os prestadores de serviço precisam responder quando se comprometem a cumprir um SLA.

O suporte de SLA requer medição de desempenho para aferir se os níveis de qualidade especificados foram alcançados. Quando um provedor Internet ou uma corporação fornece serviços baseados em SLA para seus usuários, diferentes níveis de medições podem ser realizadas pelos administradores para verificar o desempenho que o ambiente DiffServ está entregando ao usuário e para avaliar se os níveis de serviços estão dentro dos parâmetros acordados.

O problema de se realizar medições em um domínio DiffServ é a falta de uma arquitetura e métodos padronizados para realizar as medições e a avaliação dos SLAs acordados com os clientes do domínio de QoS. Tornando-se o grande desafio deste trabalho a proposta de uma arquitetura de medição e estimativa de tráfego.

1.2 Motivação

A medição do tráfego nos nós de um domínio de Serviços Diferenciados é um processo que inclui a configuração e a sua monitorização ao longo do tempo, assim como a alteração da sua configuração de forma a permitir que se possa observar a real utilização do domínio de Serviços Diferenciados e garantir a qualidade dos serviços que são fornecidos aos usuários. Os métodos tradicionais estão centrados principalmente na gestão e configuração de equipamentos individuais, não levando em consideração a estrutura de um domínio de Serviços Diferenciados como um todo. Contudo, nos atuais ambientes computacionais, devido às características de distribuição e diversidade, esse controle tem exigido, cada vez mais, grandes esforços para seu gerenciamento. Nos últimos anos diversas tentativas para o desenvolvimento de métodos para a gerenciamento da rede tem sido feito.

O modelo de Gerência de Redes Baseada em Políticas (PBNM) [CAL, 02] 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.

A arquitetura Gerência de Redes Baseada em Políticas teve origem no trabalho desenvolvido pelo grupo *Policy* Framework do IETF e pretende criar uma infra-estrutura que permita ao administrador um elevado grau de abstração acerca das capacidades dos equipamentos que pretende gerenciar. Esta abstração permite elevar o nível de controle dos equipamentos de rede. A arquitetura definida descreve as principais entidades existentes, sem descrever nenhuma norma para a sua implementação. Esta arquitetura é composta por três entidades principais: Repositório de Políticas, PDP e PEP. A comunicação entre o PDP e o PEP é efetuada utilizando o protocolo para difusão de políticas.

O Repositório de Políticas é o local onde as políticas associadas a um determinado domínio estão armazenadas, ou seja, é a base de dados. Exemplos deste são servidores de diretórios e banco de dados. O modelo mais utilizado é o modelo de diretórios e é complementado por uma estrutura de dados chamada PIB (*Policy Information Base*). A estrutura de uma PIB, em forma de árvore onde cada ramo da árvore corresponde a um diferente tipo de classe de políticas (PRCs – *Policy Rules* ou *Policy Rule Classes*) e as folhas representam o conteúdo dessa mesma política – instâncias desse tipo de política (PRIs – *Policy Rules* ou *Policy Rule Instances*). Cada PRC pode conter múltiplas PRIs.

A PIB contém toda a informação estruturada de acordo com as classes das políticas definidas. A definição de uma PIB é efetuada com um elevado nível de abstração, permitindo que os detalhes de implementação do *hardware* sejam escondidos. Desta forma, pode-se monitorar o comportamento dos vários equipamentos existentes na rede, usando a mesma estrutura de dados para todos, deixando de existir a necessidade de criar novas políticas exclusivamente para determinado PEP, de um ou outro fabricante específico.

O PDP (*Policy Decision Point*) é responsável por efetuar o processamento de políticas definidas para a rede, juntamente com outros dados relevantes para a administração da rede. Ele adquire, distribui e opcionalmente traduz regras em mecanismos de políticas do alvo. O PDP toma as decisões posteriormente transformadas em novas configurações para os PEPs consultando o repositório de políticas.

O PEP (*Policy Enforcement Point*) funciona normalmente nos equipamentos ativos de uma rede realizando as ações impostas pelas regras de políticas. O PEP é responsável por gerenciar cada equipamento de acordo com as instruções recebidas pelo PDP. Possui também a responsabilidade de traduzir e instalar as políticas nestes equipamentos.

Para a troca de informação ser efetuada com sucesso entre os vários componentes da arquitetura existe a necessidade da utilização de protocolos normalizados. Assim, estes protocolos permitirão a comunicação sem restrições entre produtos de vários fabricantes

assegurando assim o caráter geral da solução. Os protocolos principais utilizados para difusão de políticas são: COPS (*Common Open Policy Service*) [DUR, 00] e COPS-PR (*COPS for Policy Provisioning*) [CHA, 01]. O COPS foi criado especialmente para o transporte de políticas. Ele é responsável pelo intercâmbio de informações entre PDPs e PEPs existindo uma relação direta entre os pedidos, ou seja, para cada solicitação de um PEP a uma resposta de um PDP. Já o COPS-PR é uma extensão do protocolo COPS para o modelo de provisionamento, ou seja, os PEPs recebem um conjunto inicial de políticas e são atualizados quando o PDP encontrar novas políticas aplicáveis.

1.3 Proposta

Este trabalho propõe uma arquitetura de medição e estimativa de tráfego para redes DiffServ utilizando o modelo PBNM.

Primeiramente propõe-se a definição de uma nova PIB de avaliação para representação das medições realizadas nos dispositivos de rede pertencentes ao domínio de Serviços Diferenciados monitorado. A PIB proposta tem elevada importância para o desenvolvimento do trabalho, sendo concebida a partir da proposição contida em RFC 3571 [RAW, 03] e de estudos realizados com diversos dispositivos de rede.

Uma vez que o modelo PBNM será utilizado as suas principais entidades PDP e PEP devem ser implementados. As informações coletadas dos dispositivos de redes ativos devem ser repassados pelos PEPs ao PDP. Para realizar o transporte das informações estatísticas coletadas dos dispositivos de rede o protocolo COPS e COPS-PR também devem ser implementados.

A arquitetura proposta deve possibilitar a solicitação e coleta de informações nos nós de um domínio de Serviços Diferenciados, baseada nos dados estatísticos que o administrador do domínio deseja monitorar.

O trabalho em uma segunda etapa propõe a avaliação da arquitetura proposta, para observar a sua escalabilidade, no sentido em que o tráfego gerado pelo processo de medição não comprometa o funcionamento de rede e represente a real utilização dos dispositivos de rede.

1.4 Organização

Este projeto de dissertação de mestrado é composto por 8 capítulos, estruturados da seguinte maneira:

Capítulo 1 – Introdução: Compreende uma apresentação geral do contexto referente ao trabalho.

Capítulo 2 – Serviços Diferenciados: Apresenta a arquitetura de Serviços Diferenciados para QoS.

Capítulo 3 – Padrões de Gerência de Redes Baseada em Políticas: Apresentar a arquitetura de Gerência de Redes Baseada em Políticas (PBNM) proposta pelo IETF (*Internet Engineering Task Force*). São evidenciados os principais aspectos desta arquitetura, desde os protocolos de comunicação até os modelos para representação de informação.

Capítulo 4 – Controle de Medições em Redes DiffServ: São apresentados alguns modelos de medição e estimativa de tráfego existentes para redes DiffServ.

Capítulo 5 – Arquitetura Proposta: Apresenta a arquitetura proposta neste trabalho.

Capítulo 6 – Implementação: Apresenta a implementação da arquitetura proposta, explicado o processo de tradução das decisões enviadas pelo PDP ao PEP para a realização das medições no Sistema Operacional Linux e a implementação do protocolo COPS-PR para o envio dos relatórios ao PDP.

Capítulo 7 – Avaliação: Apresenta avaliação da arquitetura proposta a partir de um estudo de caso realizando no Sistema Operacional Linux para realização das medições.

Capítulo 8 – Conclusão e Trabalhos Futuros: Apresenta as conclusões do trabalho bem como sugestões de continuidade do mesmo.

Capítulo 2

Serviços Diferenciados

2.1 Introdução

A arquitetura DiffServ foi projetada pelo grupo de trabalho Serviços Diferenciados (DS) do IETF (*Internet Engineering Task Force*). O conceito de Serviços Diferenciados está definido em [BLA, 98] e parte da caracterização de domínios. Um domínio consiste de nós de borda que são utilizados para conectar domínios de DiffServ diferentes e nós de núcleo que são utilizados somente dentro do domínio.

Para distinguir o tráfego que está entrando nas extremidades do domínio (bordas) os pacotes IP (*Internet Protocol*) são modificados em um campo específico, chamado bytes DS, que usa o espaço do octeto TOS (*Type of Service*) no cabeçalho de IPv4 e o octeto da classe de tráfego (*Traffic Class*) no cabeçalho IPv6. Após os pacotes serem marcados no byte DS os mesmos são remetidos para dentro do domínio de DiffServ. Os pacotes que foram marcados com o mesmo valor no byte DS são encaminhados para classes de fluxos correspondentes. Utilizando o conceito de classe, é possível classificar fluxos A, B, C, D, etc, onde cada classe recebe um tratamento diferente previamente estabelecido através de um acordo entre as partes chamado SLA (*Service Level Agreement*). O SLA define os serviços que cada cliente vai receber.

Para oferecer este serviço em conformidade com SLA os mecanismos a seguir precisam ser combinados em uma rede:

- Configurar os bits do byte DS nas bordas do domínio;
- Usar esses bits para determinar como os pacotes são tratados pelos roteadores dentro do domínio;
- Condicionar os pacotes marcados nas bordas da rede de acordo com os requisitos de QoS de cada serviço.

A seção 2.2 explica o byte DS, que é utilizado para distinguir tráfego que está entrando no domínio de Serviços Diferenciados. A seção 2.3 explica o comportamento por salto que é o tratamento que cada fluxo de dados vai receber dentro do domínio de DiffServ. Na seção 2.4 é definido um domínio de DiffServ com seus nós de borda e de núcleo. As seções 2.5 e 2.6 explicam classes definidas pelo IETF para o encaminhamento de dados. Por fim na seção 2.6 a conclui este capítulo.

2.2 Campo DS

Para distinguir os pacotes de dados de clientes diferentes, o grupo de trabalho do IETF através da RFC 2474 [NIC, 98] propôs redefinir a estrutura do byte TOS de IPv4 e o campo de classe de tráfego de IPv6 como byte DS. As especificações do byte DS substituem as definições do octeto TOS de IPv4. Todo o tráfego da rede dentro de um domínio recebe um serviço que depende da classe de tráfego especificada no byte DS. A figura 2.1 ilustra a estrutura do byte DS.

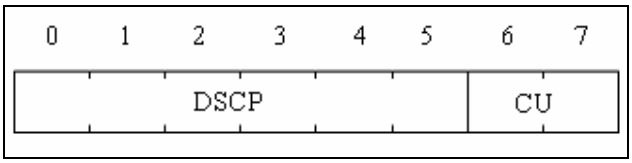

Figura 2.1: Byte DS.

Seis bits do campo DS são usados como DSCP (*Differentiated Services CodePoint*) para selecionar a classe de tráfego que um pacote utiliza em cada nó de um domínio de DiffServ. O campo CU (*currently unused*) com dois bits que não estavam sendo utilizados, atualmente representam o campo ECN (*Explicit Congestion Notification*) [FLO, 01].

Cada dispositivo de rede que suporta Serviços Diferenciados precisa ser informado como os pacotes com bytes DS diferentes devem ser processados. Na especificação de Serviços Diferenciados, essa informação é chamada de PHB (*Per-Hop Behavior*).

2.3 O comportamento Por Salto (PHB)

Basicamente um comportamento por salto (PHB) é um tratamento particular que um grupo de pacotes marcados com um mesmo DSCP (*CodePoint*) vai receber a cada nó da rede

ao longo de seu caminho. Todos os roteadores em um domínio de Serviços Diferenciados precisam ser informados que serviço um pacote com um PHB específico deve receber.

Para ter certeza de que os comportamentos por salto em cada roteador são funcionalmente equivalentes, certos grupos de PHBs comuns estão definidos nas RFCs *Assured Forwarding* PHB [HEI, 99] e *Expedited Forwarding* PHB [JAC, 99], tópicos 2.5 e 2.6 respectivamente. Com estas especificações pode-se evitar que o mesmo valor do byte DS apresente diferentes tratamentos em roteadores distintos de um domínio de DiffServ.

Um padrão de tratamento precisa estar disponível em todos os nós compatíveis de Serviços Diferenciados. Ele representa o comportamento padrão de encaminhamento *best-effort* disponível nos roteadores existentes. Quando não houver nenhum acordo efetivo, será assumido que os pacotes pertencem a esse nível de serviço. O valor DSCP recomendado para definir o PHB padrão do byte DS é 000000.

2.4 Domínio de Serviços Diferenciados

Um domínio de DS é uma porção contígua de nós DiffServ na qual um conjunto consistente de planos de ação de Serviços Diferenciados é administrado de uma forma coordenada e com as mesmas definições de PHB [BLA, 98]. Um domínio de DS tem limites bem definidos, que consiste de nós de borda que são usados para conectar domínios DS diferentes entre si, e componentes que são usados somente dentro dos domínios. A figura 2.2 apresenta um domínio de DiffServ com seus nós de borda e de núcleo.

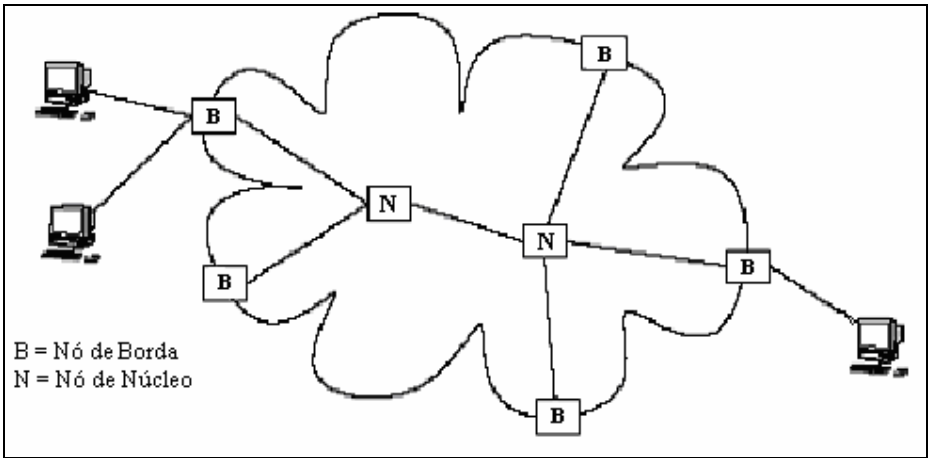

Figura 2.2: Domínio DS.

Um domínio de DS regularmente consiste em uma ou mais redes sobre a mesma administração. A administração do domínio DS é responsável pela garantia de que os recursos adequados serão disponibilizados e reservados para suportar e cumprir com os SLAs oferecidos pelo domínio aos seus clientes.

2.4.1 Nós de Borda de Serviços Diferenciados

Todos os pacotes de dados de um domínio de DiffServ precisam passar por um nó de borda que pode ser um roteador, um *host*, ou um *firewall*. Um nó de borda pode ser de egresso ou/e de ingresso. A interface de ingresso processa o tráfego que está entrando em um em um nó e a interface de egresso realiza a função de condicionamento do tráfego que é encaminhado para o próximo nó de um domínio de DiffServ. O condicionamento de tráfego é realizado em um nó de borda pelo condicionador de tráfego. Este classifica, marca e possivelmente condiciona os pacotes que entram ou deixam o domínio de Serviços Diferenciados. A figura 2.3 ilustra os componentes do condicionador de tráfego.

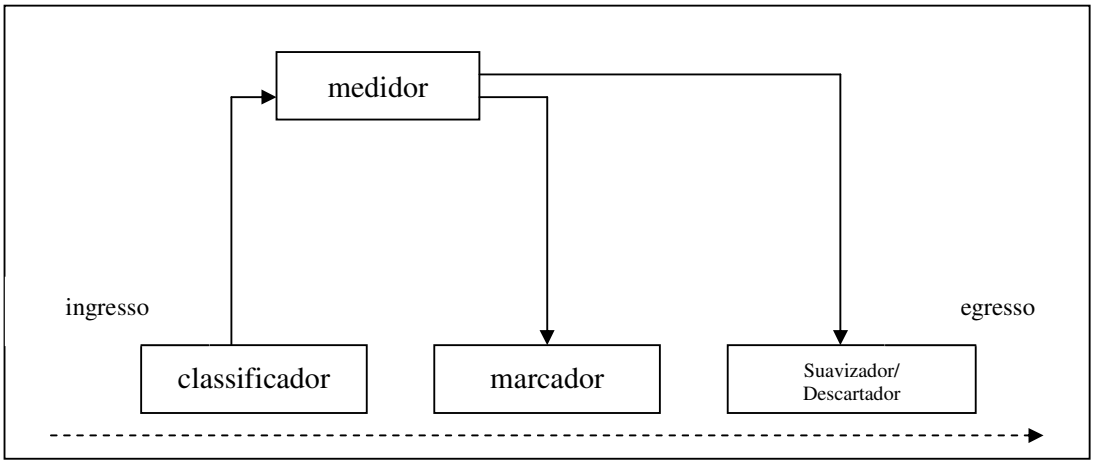

Figura 2.3: Condicionador de Tráfego DS.

Um condicionador de tráfego consiste dos seguintes blocos:

- Classificador: Seleciona os pacotes através dos cabeçalhos, e encaminha aqueles que correspondem às regras de classificação para processamento posterior. O modelo DiffServ específica dois tipos de classificadores:
    - Multicampos (MF): Pode classificar com base no byte DS bem como com base em qualquer outro campo do cabeçalho IP, como endereço IP, porta, etc;

    - Comportamento Agregado (BA): A classificação é baseada somente no byte DS. BA é definido como uma coleção de pacotes com o mesmo DSCP que cruza uma rede em uma direção particular;
- Medidor: Verifica se o fluxo de tráfego selecionado pelo classificador está de acordo com o perfil do tráfego especificado no contrato (SLA) estabelecido entre o cliente e o provedor de serviços. Um medidor passa informações de estado para outras funções de condicionamento, para que se possa tomar uma ação particular para cada pacote que atende ou não os requisitos QoS;
- Marcador: Responsável pela marcação ou remarcação do byte DS. A marcação é realizada 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.
- Suavizador / Descartador: É responsável pela adaptação e descarte dos pacotes de acordo com as propriedades estabelecidas pelo PHB. Um suavizador geralmente tem um *buffer* de tamanho definido e os pacotes podem ser descartados se não houver espaço no *buffer* suficiente para armazenar os pacotes atrasados. Os descartadores realizam o descarte de alguns ou de todos os pacotes baseados em regras específicas. Por exemplo, podem-se aceitar todos os pacotes até a taxa máxima permitida e descartar todos eles que excedem a taxa configurada.

O condicionador de tráfego em um dispositivo de borda garante que os pacotes que vão passar pelo domínio sejam marcados de acordo com um PHB de um dos grupos de PHB suportados pelo domínio. Isso é necessário porque domínios DS diferentes podem ter grupos diferentes de PHB, o que significa que a mesma entrada no byte DS pode ser interpretada de formas distintas em domínios diferentes.

Se um pacote passar através de vários domínios, o byte DS pode ser remarcado em cada dispositivo de borda, para garantir que a QoS contratada no SLA seja cumprida.

2.4.2 Nós de Núcleo de Serviços Diferenciados

Os nós de núcleo selecionam o comportamento de encaminhamento dos pacotes com base no byte DS. Como o valor do byte DS normalmente não muda dentro de um domínio DS, todos os roteadores internos precisam usar o mesmo plano de ação de encaminhamento para atender ao acordo de QoS. O valor do byte DS só muda dentro de um domínio de DiffServ se o tráfego estiver fora de perfil. Como todos os roteadores de núcleo de um

domínio usam as mesmas funções de plano de ação para o tráfego entrante, o condicionador de tráfego dentro de um nó de núcleo é feito somente por um classificador de pacotes. Ele seleciona os pacotes com base em seu valor de PHB e direciona-os para sua correspondente fila no nó.

2.5 PHB *Assured Forwarding* (PHB AF)

No modelo PHB AF [HEI, 99] são determinadas quatro classes de encaminhamento. Para cada classe é alocada certa quantidade de recursos (espaço de *buffer*, banda). Pacotes de IP que desejam usar os serviços providos pelo grupo de PHB AF são nomeados pelo cliente ou pelo provedor de domínio DS em uma ou mais destas classes de AF de acordo com os serviços que o cliente solicitou.

Cada uma destas classes AF é dividida em três níveis de preferência de descarte. No caso de congestionamento, a preferência de descarte de um pacote determina a importância relativa do pacote dentro da classe de AF. Um nó de DiffServ congestionado tenta proteger pacotes com valor um mais baixo de precedência 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 PHB AF, depende (1) quantidade de recurso para encaminhamento alocado para a classe AF a qual o pacote pertence; (2) qual é a carga atual da classe de AF, e no caso de congestionamento dentro da classe; (3) qual é o nível preferência de descarte do pacote.

Ainda na RFC 2597 é dito que:

- Um nó de Serviços Diferenciados deveria implementar as quatro classes de uso geral de AF;
- Pacotes em uma classe de AF devem ser remetidos independentemente de pacotes em outra classe AF;
- Um nó de DiffServ não deve agregar duas ou mais classes AF;
- Um nó de DS não deve reordenar pacotes de AF do mesmo microfluxo quando eles pertencerem à mesma classe de AF embora haja precedência de descarte destes pacotes.

Finalmente na tabela 2.1 os valores de AF indicados para o campo DSCP. Quatro classes e três subdivisões de classe (precedência de descarte). Como se pode perceber são seis

bits como especificado em [NIC, 98]. Classes estão definidas com os três primeiros bits, e subdivisões de classe estão definidas com os últimos três bits.

Tabela 2.1: Valores DSCP para o Grupo PHB AF.

	Classe 1	Classe 2	Classe 3	Classe 4
Baixo nível de descarte	001010	010010	011010	100010
Médio nível de descarte	001100	010100	011100	100100
Alto nível de descarte	001110	010110	011110	100110

2.6 PHB *Expedited Forwarding* (PHB EF)

O PHB EF [JAC, 99] também é conhecido como *Premium Service* e simula um *link* dedicado com banda fixa entre dois *hosts* (fim a fim), estabelecendo rígidos limites de atrasos e praticamente sem perdas de pacotes. O atraso e perda de pacotes acontecem devido à formação de filas nos roteadores ao longo do caminho é quase impossível evitar filas, então dado um fluxo VIP é necessário criar para ele um caminho especial nos roteadores que evitem as filas.

Para ter-se um serviço *Premium* são necessários dois pontos:

1. Configurar o dispositivo de rede de forma que o agregado tenha uma taxa de partida mínima, bem definida e em particular, independente da intensidade de outro tráfego ao nó;
2. Condicionar o agregado de forma que sua taxa de chegada a qualquer nó sempre seja menor que a taxa de partida mínima configurada naquele nó.

Devido a estas características o PHB EF representa o mais alto nível de priorização de tráfego dentro do domínio DiffServ. Para evitar-se que o tráfego PHB EF cause danos a outros fluxos deve-se determinar parâmetros como taxa máxima e tamanho de rajadas, e descartar o tráfego que exceder estes limites.

O valor de DSCP definido para o PHB EF na RFC 2598 é 101110.

2.7 Conclusão

Serviços Diferenciados estão se tornando um mecanismo atraente principalmente para provedores de rede, uma vez que o cliente paga pela QoS desejada. No modelo *best-effort* isto não é possível, pois todos os pacotes são tratados de modo equivalente. Com Serviços Diferenciados é possível distinguir e priorizar os pacotes de dados de acordo com a necessidade de cada aplicação e perfis dos clientes.

A desvantagem é que o respeito das SLAs depende da carga efetiva aplicada na rede. Se esta carga for mal dimensionada, SLAs podem não ser respeitadas. Os trabalhos de pesquisa atuais concentram-se no desenvolvimento de mecanismos de controle de admissão para redes DiffServ, objetivando resolver este problema.

Capítulo 3

Padrões de Gerência de Redes Baseada em Políticas

3.1 Introdução

Este capítulo tem por objetivo apresentar a arquitetura de Gerência de Redes Baseada em Políticas (PBNM) proposta pelo grupo de trabalho RAP (*Resource Allocation Protocol*) do IETF (*Internet Engineering Task Force*). São evidenciados os principais aspectos desta arquitetura, desde os protocolos de comunicação até os modelos para representação de informação. Os conceitos apresentados neste capítulo são a base para o desenvolvimento deste trabalho.

Na seção 3.2 apresenta-se a arquitetura PBNM com as suas principais entidades. A seção 3.3 apresenta os modelos de comunicação para estabelecer a comunicação entre o PDP (*Policy Decision Point*) e PEP (*Policy Enforcement Point*). A seção 3.4 apresenta o modelo para a representação de informações proposto pelo IETF conhecida como PIB (*Policy Information Base*). Apresentando ainda a estrutura da Framework PIB e da PIB Framework *Feedback* que é de fundamental importância para o desenvolvimento deste trabalho. Por fim na seção 3.5 as conclusões deste capítulo.

3.2 Gerencia Baseada em Políticas

A arquitetura de gerencia baseada em políticas descreve de um modo geral, as principais entidades existentes (figura 3.1), sem, contudo descrever nenhuma norma para a sua implementação.

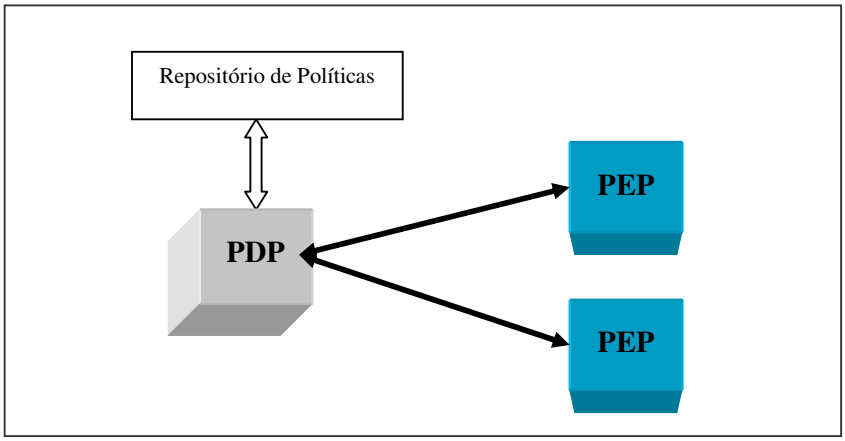

Figura 3.1: Principais componentes da Arquitetura PBNM.

Os três elementos apresentados na figura 3.1 são descritos abaixo:

- O Repositório de Políticas é o local onde as políticas associadas a um determinado domínio estão armazenadas, ou seja, é à base de dados. Exemplos deste são servidores de diretórios e banco de dados. O modelo mais utilizado é o modelo de diretórios e é complementado por uma estrutura de dados chamada PIB (*Policy Information Base*) (seção 3.4);
- Um PDP (*Policy Decision Point*) é entidade responsável por efetuar a tradução de políticas de alto nível para um nível inferior de abstração, mais próximo do formato usado para configurar os equipamentos. O PDP acessa o repositório de políticas e processa as informações juntamente com outros dados que conhece, como por exemplo, as capacidades de cada equipamento a que o PEP está associado. A combinação desta informação dá origem à configuração de cada PEP que o PDP controla;
- O PEP (*Policy Enforcement Point*) é o componente que trabalha diretamente com os equipamentos existentes na rede, sendo responsável por assegurar que os dados enviados pelo PDP que o controla serão postos em prática no equipamento que está sob sua administração, efetuando aqui uma tradução destes dados para comandos reconhecidos pelo equipamento controlado. O PEP deve sempre obedecer aos comandos enviados pelo PDP, podendo também ele próprio enviar comandos e informação para o PDP.

Para a comunicação entre o PDP e o PEP é utilizando o protocolo COPS (*Common Open Policy Service*) apresentado na próxima seção.

3.2.1 *Common Open Policy Service* (COPS)

Para comunicação entre PDP e PEP o IETF propôs a utilização do protocolo COPS (*Common Open Policy Service*) [DUR, 00]. COPS é transportado sobre TCP (*Transmission Control Protocol*), que estabelece uma comunicação segura entre cliente e servidor. A conexão de TCP entre o PDP e o PEP é iniciada pelo PEP e é mantida até o fim da negociação. A porta *default* TCP em qual o PDP aguarda conexões do PEP é 3288. A figura 3.2 ilustra o formato do cabeçalho de uma mensagem COPS.

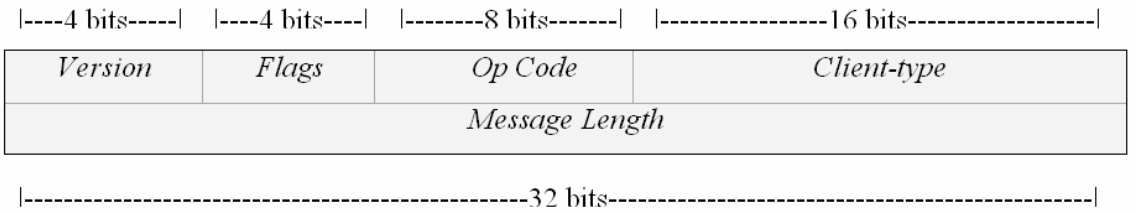

Figura 3.2: Cabeçalho COPS.

Uma mensagem COPS é formada pelos campos apresentados na figura 3.2. A tabela 3.1 descreve sucintamente a função destes campos.

Tabela 3.1: Descrição dos campos do protocolo COPS.

Version	Versão COPS atual é 1;
Flags	Composto por 4 bits que podem ser de valor 0x1 para mensagem solicitada ou 0x0 para as demais mensagens. Este *flag* é setado quando a mensagem é solicitada por outra mensagem COPS;
Op Code	Determina o tipo da mensagem 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; • 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;
Client-type	Identifica o tipo do cliente;
Message Length	Descreve o tamanho da mensagem em bytes, inclui o cabeçalho.

O cabeçalho do COPS apresentado na figura 3.2 define o tipo da mensagem. A figura 3.3 demonstra como os elementos das políticas são estruturadas no corpo da mensagem COPS.

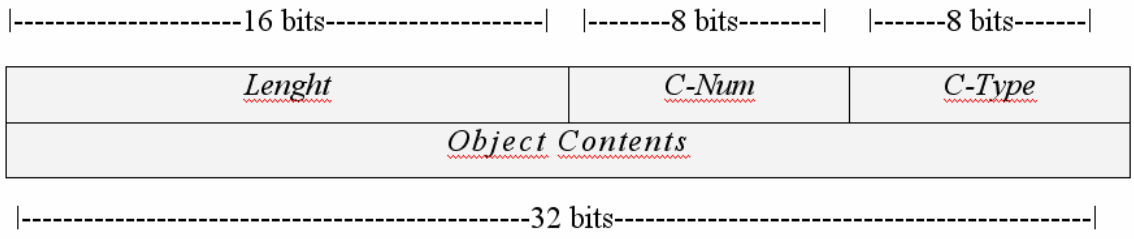

Figura 3.3: Objeto COPS.

As mensagens entre o PDP e o PEP são construídas a partir destes objetos ilustrados na figura 3.3. A tabela 3.2 descreve sucintamente a função destes campos.

Tabela 3.2: Elementos do objeto COPS.

Lenght	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	Identifica a classe da informação contida no objeto;
C-Type	Depende do *C-Num* e identifica o subtipo ou versão da informação da informação contida no objeto.

3.2.2 Principais mensagens COPS trocadas entre o PDP e o PEP

As principais mensagens COPS, utilizadas neste trabalho, são ilustradas na figura 3.4, em seguida apresenta-se uma breve explicação sobre cada mensagem.

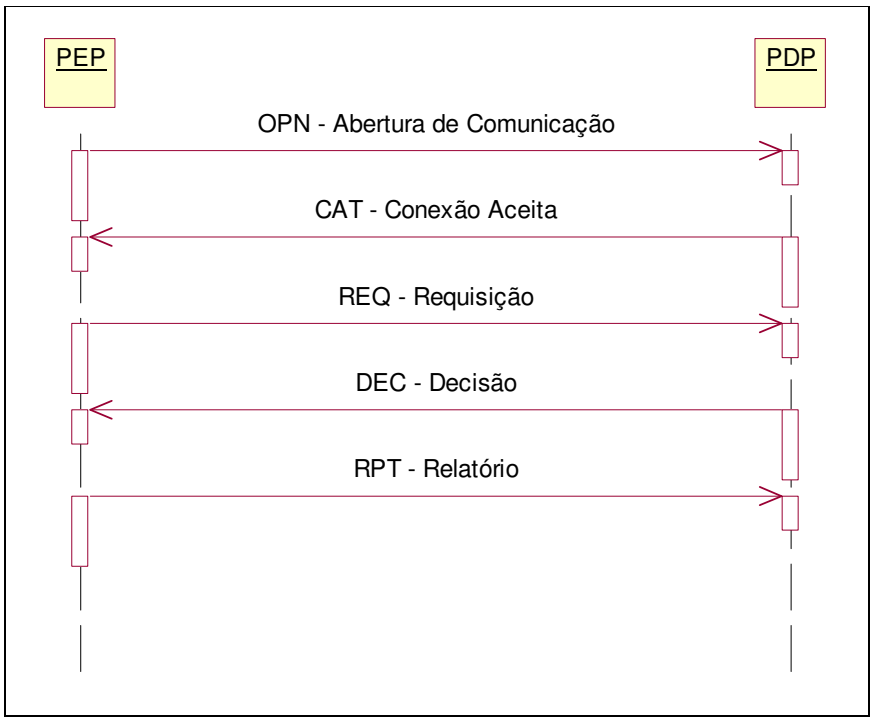

Figura 3.4: Troca de mensagens entre PEP e PDP.

O processo de troca de mensagens segue os seguintes passos:

1. O PEP estabelece a conexão com um PDP remoto e envia mensagens *Client-Open* (OPN). A negociação de mecanismos de segurança, quando necessário, ocorre também nesta etapa;

2. O PDP responde com uma mensagem *Client-Accept* (CAT) em separado para cada cliente requisitado pelo PEP. Na mensagem *Client-Accept* é especificado também o intervalo de tempo entre as mensagens *Keep-Alive* e o valor de *acct* (*Timer Value*) que permite ao PDP controlar o intervalo de tempo a qual mensagens de relatórios podem ser enviados pelo PEP;
3. Se algum tipo de cliente requisitado pelo PEP não for suportado pelo PDP, este último responde com uma mensagem *Client-Close* (CC), 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 (REQ) ao PDP, identificando-se e informando das suas capacidades e limitações;
5. Quando o PDP recebe uma mensagem de requisições o mesmo envia uma mensagem de Decisão (DEC) com as políticas que devem ser implementadas pelo PEP;
6. Em resposta a mensagem de Decisão enviada pelo PDP o PEP responde com uma mensagem de relatório (RPT) para comunicar o sucesso ou a falha na execução da Decisão enviada pelo PDP. As mensagens de RPT também podem ser usadas para prover informações especificas de um determinado cliente. Neste caso o tipo de relatório deve ser especificado pelo PEP.

3.2.3 *Common Open Policy Service for Policy Provisioning* (COPS-PR)

Enquanto COPS estabelece comunicação básica entre o PEP e o PDP, o COPS-PR (*Common Open Policy Service for Policy Provisioning*) [CHA, 01] baseia-se em uma estrutura de dados conhecida como PIB (*Policy Information Base*) (seção 3.4) comum ao PEP e PDP, adicionando funcionalidades requeridas pelo modelo *provisioning* (seção 3.3) da arquitetura PBNM.

Um módulo PIB é constituído por classes PRCs (*Provisioning Classes*) e suas instâncias PRIs (*Provisioning Instances*). O formato genérico dos objetos COPS-PR é ilustrado na figura 3.5 e possibilita a identificação de uma determinada instância conhecida como PRID (*Provisioning Instance Identifier*), e a representação dos dados correspondentes a ela.

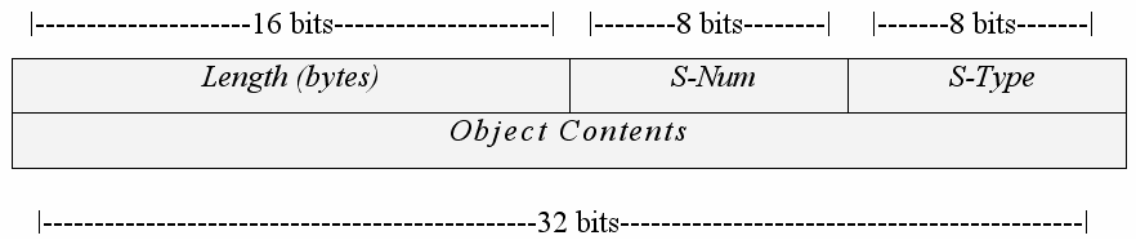

Figura 3.5: Objeto COPS-PR.

Os campos *S-Num* e *S-Type* são similares aos campos *C-Num* e *C-Type* definidos sobre o objeto COPS. A diferença é que estes objetos estão sempre encapsulados nos objetos *Named ClientSI* ou *Named Decision Data*, existentes no COPS. *Named ClientSI* é utilizado em mensagens de requisição, enviadas pelo PEP ao PDP e *Named Decision Data* é utilizada pelo PDP para enviar mensagens de decisão ao PEP. *Named Decision Data* também é utilizada pelo PEP para enviar mensagens de relatório ao PDP.

Na arquitetura de medição e estimativa proposta, o protocolo COPS/COPS-PR é implementado na comunicação entre PEP e PDP, transportando as informações configuração e *accouting*.

3.3 Modelos de Comunicação entre PEP e PDP

A arquitetura PBNM estabelece a interação entre PDPs e PEPs usando o modelo cliente-servidor. Nesta arquitetura podem ser distinguidos dois modelos distintos de comunicação: *outsourcing* e *provisioning*.

No modelo *outsourcing*, no caso de existir um evento que o PEP não saiba tratar de acordo com os critérios que tem instalados, é enviada mensagem de requisição de política (REQ), ao PDP correspondente, notificando-o da ocorrência desse mesmo evento. O PDP então processa o evento de forma a obter as políticas correspondentes ao PEP que realizou a requisição, e responde ao mesmo com uma mensagem de decisão (DEC) que contém as informações que devem ser instaladas de acordo com a notificação realizada. Este modelo é síncrono, visto que o PDP reage a eventos originados nos PEPs por ele controlados.

O modelo *outsourcing* é principalmente implementado em redes *IntServ*, assim quando o protocolo de sinalização RSVP (*Resource Reservation Protocol*) 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 [BEL, 05].

No modelo *provisioning*, quando o PEP realiza a conexão inicial com o PDP este envia todas as informações existentes, aplicável a esse mesmo PEP. Estas políticas são armazenadas no PEP e todos os eventos que venham a ocorrer serão tratados usando estas mesmas informações. Podemos chamar a este modelo *push* pelo fato de que o PDP envia antecipadamente todas as políticas para os PEPs que controla, ou seja a relação 1:1 não é mantida como no modelo *outsourcing*. A partir deste momento a troca de informações entre o PDP e o PEP é feita somente com sinalização, ou caso ocorra uma modificação das políticas ou dos dispositivos envolvidos.

O modelo *provisioning* atende aos requisitos da arquitetura de DiffServ, na qual recursos de rede são previamente configurados com base nos de contratos (SLAs) que são relativamente estáticos. No desenvolvimento da estratégia *provisioning*, o protocolo COPS-PR transporta as informações de configuração [BEL, 05].

3.4 *Policy Information Base* (PIB)

O modelo de dados PIB (*Policy Information Base*) é apresentado no âmbito da sua utilização com os protocolos de difusão de políticas COPS e COPS-PR. No protocolo COPS-PR, cada cliente tem que conter uma base de dados chamada PIB onde armazena a informação recebida.

Uma PIB pode ser representada por uma estrutura em árvore (figura 3.6), onde cada ramo da árvore corresponde a um diferente tipo de política ou classe de políticas definida como PRCs (*Policy Rules* ou *Policy Rule Classes*) e as folhas representam o conteúdo dessa mesma política – instâncias desse tipo de política PRIs (*Policy Rules* ou *Policy Rule Instances*). Cada PRC é identificada por um OID (*Object Identifier*) e pode conter múltiplas PRIs. Cada PRI é identificada por um identificador PRID (*Provisioning Instance Identifier*), sendo que um PRID pode ser considerado como um nome único nos objetos COPS.

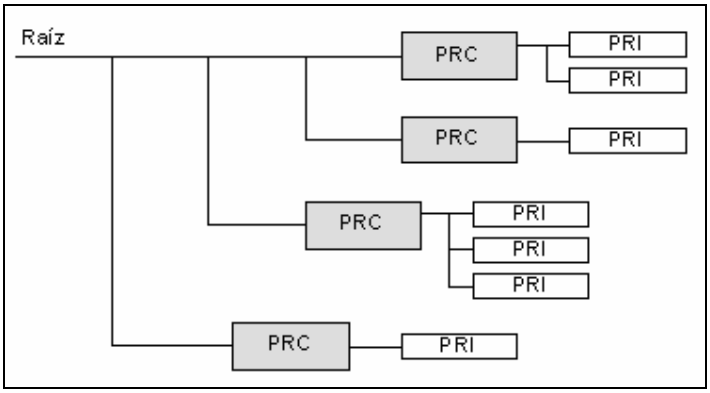

Figura 3.6: Estrutura de árvore da PIB.

Um exemplo de uma PRID pode ser o identificador “1.3.6.2.2.4.2.7.11” ilustrado na figura 3.7, em que os primeiros oito números representam à classe PRC (“1.3.6.2.2.4.2.7”) e o último número representa a PRI (“11”). A figura 3.8 ilustra a PRC (“1.3.6.2.2.4.2.7”) com suas PRIs (“11” e “21”).

```
- <dsAction oid="1.3.6.2.2.4.2.6">
  - <Prid id="2">
      <dsActionNext type="6">1.3.6.2.2.4.2.11.2</dsActionNext>
      <dsActionSpecific type="6">1.3.6.2.2.4.2.7.11</dsActionSpecific>
    </Prid>
```

Figura 3.7: Exemplo de PRID.

```
- <dsDscpMarkAct oid="1.3.6.2.2.4.2.7">
  - <Prid id="11">
      <dsDscpMarkActDscp type="3">0x28</dsDscpMarkActDscp>
    </Prid>
  - <Prid id="21">
      <dsDscpMarkActDscp type="3">0x38</dsDscpMarkActDscp>
    </Prid>
  </dsDscpMarkAct>
```

Figura 3.8: Exemplo de PRC e PRIs.

A definição de uma PIB é realizada com um elevado nível de abstração, permitindo que os detalhes de implementação do *hardware* sejam escondidos. Desta forma, pode-se controlar o comportamento dos vários equipamentos existentes na rede, usando a mesma estrutura de dados para todos. O PDP pode instalar novas PRIs ou alterar as PRIs existentes num PEP, enviando as respectivas mensagens COPS. Pode ainda eliminar determinada PRI através de uma decisão que contenha o PRID do PRI que se pretende eliminar. As políticas

são construídas através de um conjunto de PRIs existentes na PIB. Deste modo pelo fato que o PDP poder acrescentar, alterar ou remover PRIs de cada PEP ele pode implementar as políticas que pretende em cada equipamento.

A PIB permite, ainda, ações que possibilitam sua modificação ou extensão, tornando-a flexível entre PEP e PDP, como: adicionar ou remover novas PRCs; adicionar os retirar atributos das classes existentes; e uma PRC existente pode ser estendida através de novas PRCs definidas em outra PIB.

3.4.1 Framework PIB

O Framework PIB definido pelo IETF através da RFC 3318 [SAH, 03] descreve um modelo para especificar informações de políticas comuns a todos os clientes. A estrutura é definida com classes (PRCs) e instâncias destas classes (PRIs). O Framework PIB específica quatro grupos de classes (PRCs). A figura 3.9 ilustra o Framework PIB.

Grupo *Base PIB* - Descreve as classes e os atributos suportados pelo PEP, suas limitações e a sua configuração atual. Usando estas informações o PDP pode potencialmente escolher a política mais próxima da capacidade do dispositivo.

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.

Grupo *Classifier* - As classes deste grupo representam filtros IP, filtros IEEE 802 e rótulo interno (*Internal Label*). O conjunto de filtros consiste de uma tabela genérica que é herdada por todos os filtros do grupo *Classifier*. A Tabela *Internal Label* é utilizada para possibilitar a classificação baseada em rótulos internos para fluxos que estão percorrendo o caminho entre a interface de ingresso e interface de egresso, sem a necessidade de reclassificação. A tabela *IPFilter* define um filtro para pacotes IP.

Grupo *Marker* - As classes deste grupo representam o marcador 802 e o marcador *Internal Label*. A implementação do marcador *internal label* é específica e pode ser usada para outras funções relacionadas a políticas, tais como contabilidade ou outros tratamentos no caminho de dados.

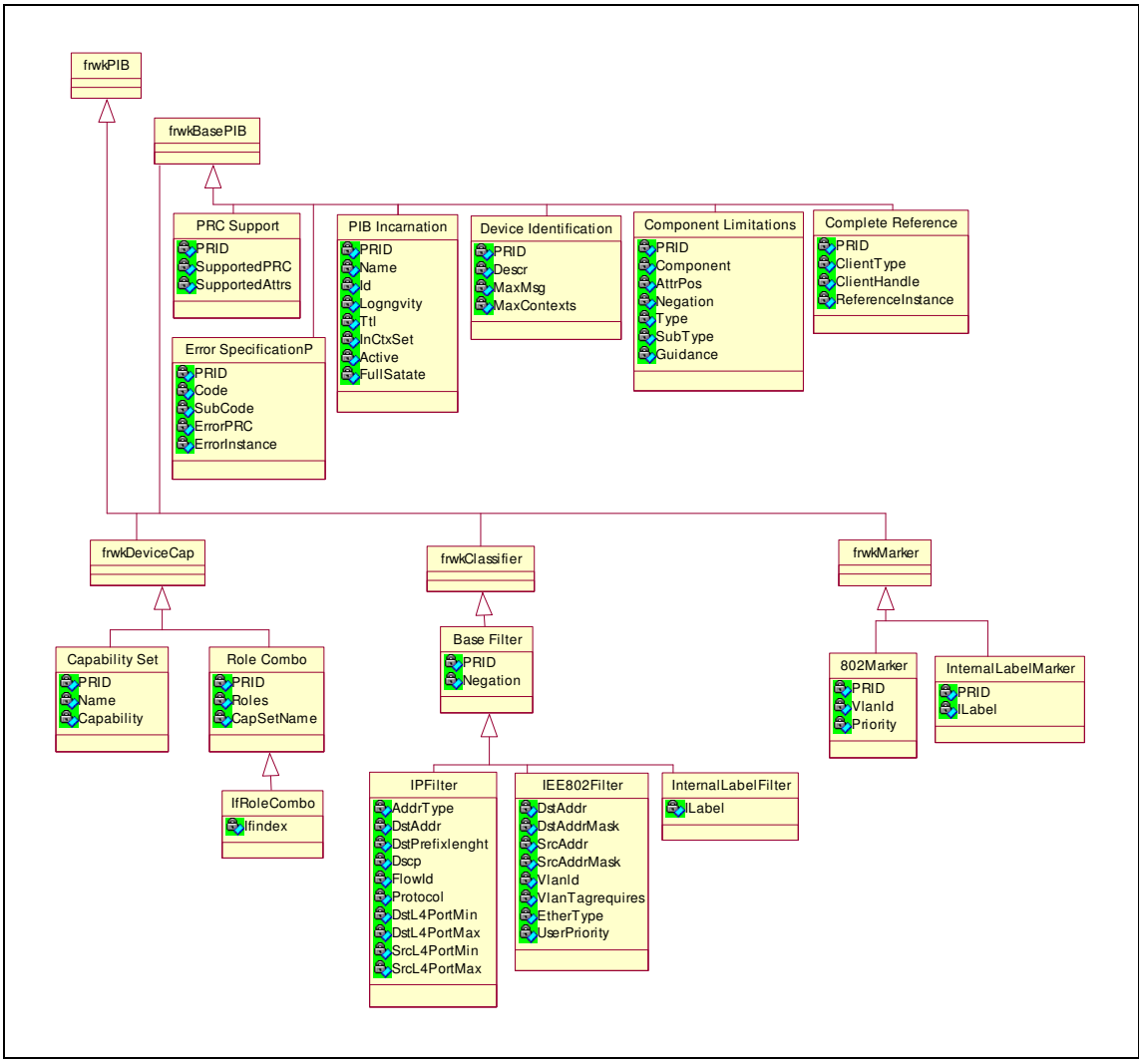

Figura 3.9: Framework PIB.

3.4.2 PIB Framework *Feedback*

A PIB Framework *Feedback* proposta pelo IETF através da RFC 3571 [RAW, 03] define classes genéricas que permitem a representação de informações, previamente configuradas no PEP, ou seja, o monitoramento, avaliação e relatórios de políticas que foram postas em prática nos dispositivos de rede. Para aplicações específicas pode-se estender o modelo, a fim de incluir outras informações necessárias. Como neste trabalho deseja-se realizar a avaliação de um domínio DiffServ, será necessário estender a PIB Framework *Feedback* de forma a atender as necessidades de avaliação de um domínio de Serviços Diferenciados.

Para realizar o *feedback* a PIB Framework *Feedback* define três tipos de políticas básicas.

Política de critérios de seleção - é instalada pelo PDP, define as condições que devem ser utilizadas pelo PEP para realizar o trabalho de monitoramento e coleta de informações baseado na combinação de filtros e papéis. Através desta política é possível informar ao PEP o papel da interface que se deseja monitorar e o filtro que deve ser utilizado. Uma política de critérios de seleção pode ser reutilizada por várias vezes, uma vez que não define nenhuma informação específica. Por exemplo, imagine que se deseja monitorar o papel Departamento de Vendas associado à interface 1 de um roteador. Ao definirem-se os critérios de seleção é selecionado todo um objeto, ou seja, nenhuma informação específica do objeto é selecionada. Para definir que informação se deseja obter do papel selecionado deve-se utilizar uma política de uso.

Uma política de uso - define os atributos que devem ser registrados pelo PEP. Cada política de uso específica um contador relacionado a uma ação específica como, por exemplo, contar o número de pacotes descartados, relacionados à política de critérios de seleção instanciada.

Uma política de uso e uma política de critérios de seleção são firmemente associadas uma com a outra. Uma terceira política é utilizada para associar as políticas de critérios de seleção e de uso. Uma política de associação - também específica o intervalo entre a avaliação das políticas de uso e relatórios.

Estas três políticas são especificadas na PIB Framework *Feedback* por três grupos distintos de classes. A política de critério de seleção é representada pelo grupo *Selection* composto por uma tabela. A política de uso é representada pelo grupo *Usage* que é composto por duas tabelas, e a política de associação é representada pelo grupo *Link* composto por cinco tabelas. A figura 3.10 ilustra os três grupos de classes da PIB Framework *Feedback*.

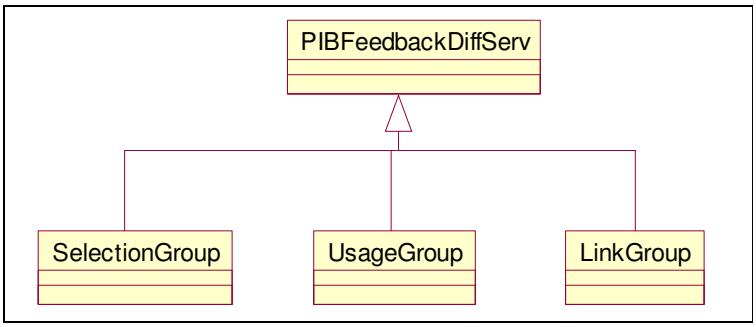

Figura 3.10: Grupos de classes que compõem a PIB Framework *Feedback*.

Grupo *Selection* - este grupo tem como função definir as condições usadas pelo PEP para monitorar e registrar uma política de uso. A figura 3.11 ilustra o grupo de seleção.

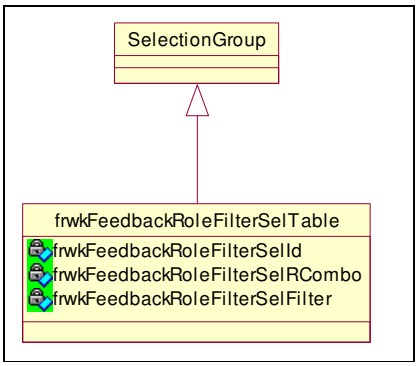

Figura 3.11: Grupo S*election*.

Imagine que se deseja monitorar na interface 1 de um roteador o papel Departamento de Compras, todos os pacotes marcados com DSCP 0x28. Para representação destes dados na PIB Framework *Feedback* utiliza-se a classe *frwkFeedbackRoleComboFilterSelectionTable* que contém a identificação do papel e do filtro que deve ser utilizado para realizar o trabalho de monitoramento. Definidas as condições que devem ser utilizadas para realizar o trabalho de monitoramento é necessário definir que política de uso deve ser registrada (contabilizada) do objeto selecionado.

Grupo *Usage* - este grupo tem como função descrever que parâmetros são registrados pelo PEP. A PIB Framework *Feedback* define duas classes de política de uso, *frwkFeedbackTraffic* e *frwkFeedbackIfTraffic*. A figura 3.12 ilustra o grupo de uso.

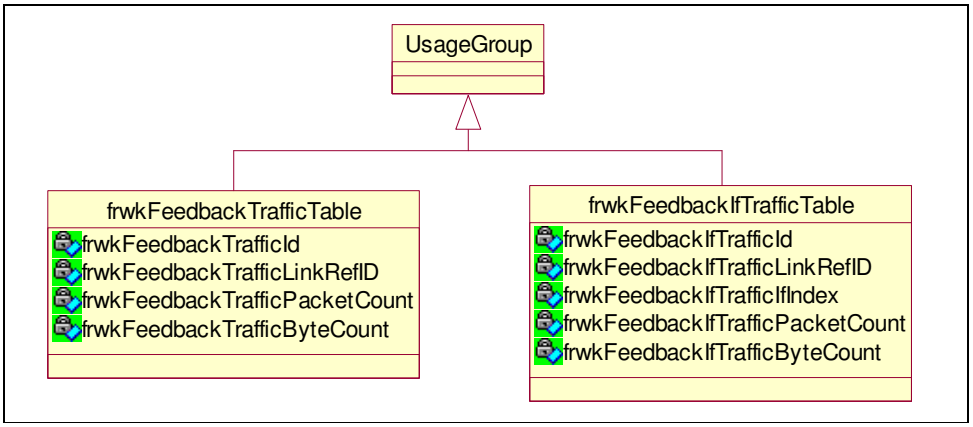

Figura 3.12: Grupo *Usage*.

No grupo *Selection* foi selecionado o objeto que se deseja monitorar, porém não foi especificado que dados se deseja registrar. Para especificar os dados que se deseja registrar é utilizado o grupo *Usage*. Imagine que se deseja monitorar o número de pacotes descartados do papel Departamento de Vendas associado ao DSCP 0x28. Para isto se utiliza a tabela *frwkFeedbackTraffic* uma vez que deseja-se monitorar somente uma das interfaces do roteador. Para casos que se deseja monitorar mais de uma interface do roteador deve-se utilizar a tabela *frwkFeedbackIfTraffic*. È importante observar que em nenhum lugar no grupo *Usage* é informado que se deseja realizar a contabilidade de pacotes descartados. A PIB Framework *Feedback* deixa em aberto, uma vez que é um Framework. Neste trabalho é realizado um estudo para definir os dados que podem ser monitorados em um domínio de DiffServ para que assim se possa especificar na PIB que dados podem ser instanciados.

Grupo *Link* - este grupo tem como função descrever as características do PEP e as combinações de políticas de uso e de seleção. A figura 3.13 ilustra o grupo *Link*.

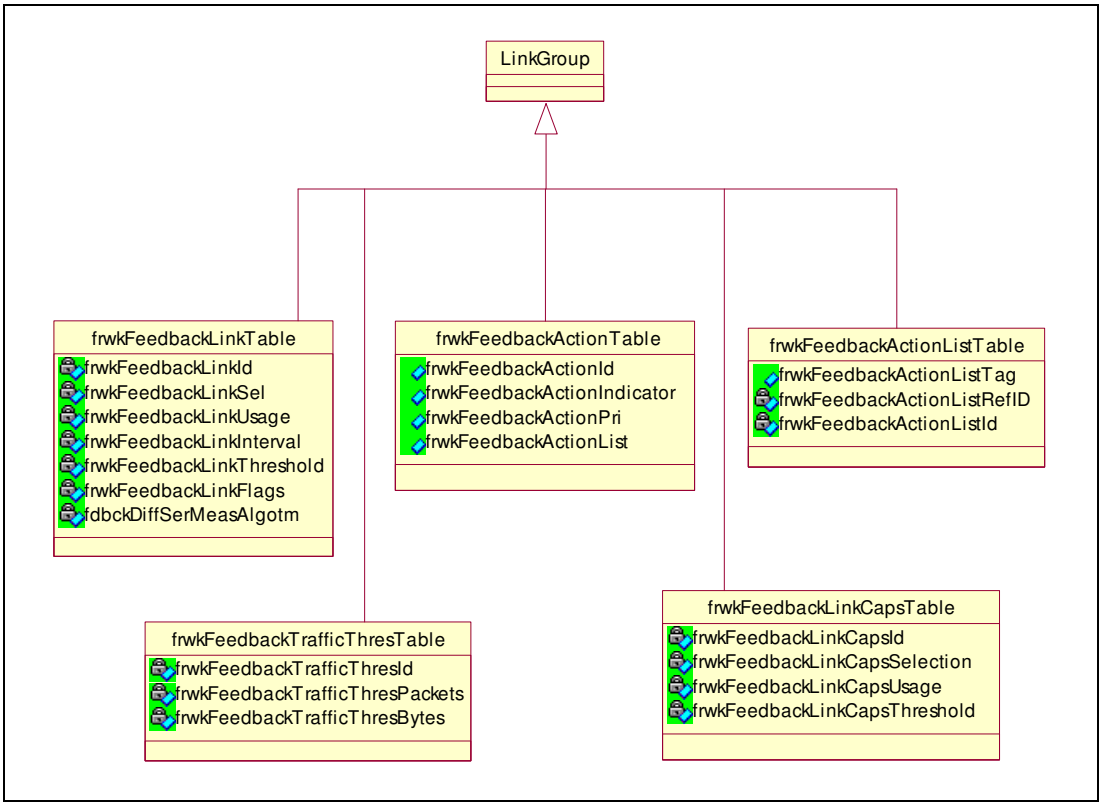

Figura 3.13: Grupo *Link*.

Definido o que se deseja monitorar através do grupo *Selection*, é quais dados devem ser registrados no grupo *Usage* é necessário se realizar a associação entre as políticas de seleção e de uso. A tabela *frwkFeedbackLinkTable* realizar esta associação, o atributo *frwkFeedbackLinkSel* contém o identificador da instancia da política de critérios de seleção, e o atributo *frwkFeedbackLinkUsage* contém o valor da PRC da política de uso. Na tabela *frwkFeedbackLinkTable* ainda é definido o intervalo para a coleta de dados e o tipo de relatório que deve ser enviado ao PDP (ver a explicação dos tipos de relatório na seção 3.4.2.1).

A tabela *frwkFeedbackActionTable* define a ação que deve ser executada para as políticas de uso instanciadas (suspender relatório e medições, suspender relatório, voltar a enviar relatórios, solicitar relatório). A ação pode afetar todas as instancias da tabela *frwkFeedbackLinkTable* ou apenas uma lista. No caso de afetar todas as instâncias da tabela *frwkFeedbackLinkTable* deve-se atribuir o valor zero ao atributo *frwkFeedbackActionPri*. Quando se deseja aplicar a ação a uma lista de instâncias da *frwkFeedbackLinkTable* deve-se atribuir o valor um ao atributo *frwkFeedbackActionPri* que indica que a ação deve ser aplicada a lista de instancias identificadas pelo atributo *frwkFeedbackActionList*.

A tabela frwk*FeedbackActionListTable* define o conjunto de instancias da tabela *frwkFeedbackLinkTable* as quais vão sofrer a ação especificada na tabela *Action*. O atributo *frwkFeedbackActionTag* é o identificador da lista das instancias e o atributo *frwkFeedbackActionListRefId* contém o conjunto das instancias da tabela *frwkFeedbackLinkTable* que vão sofrer a ação.

A tabela *frwkFeedbackSelectionUsageCombinationCapabilityTable* define os critérios de seleção válidos, limiar e combinações de PRC suportadas pelo PEP.

A tabela *frwkFeedbackTrafficStatisticsThresholdTable* é utilizada para prover valores de limiar para os atributos descrito nas classes de uso.

3.4.2.1 Relatórios

Os relatórios providos pelo PEP de acordo com as medições realizadas pelas instâncias do grupo de avaliação de uso (*Usage*) podem ser de três tipos, periódicos, periódicos informando condições ou solicitados.

Relatórios periódicos - Os intervalos a qual os relatórios não solicitados são providos pelo PEP estão definidos na tabela *frwkFeedbackLink* pelo atributo *Interval* vezes o valor definido no campo *acct* da mensagem CAT enviada pelo PDP ao PEP. O atributo *Interval* é na verdade um multiplicador do tempo definido no campo *acct*. Por exemplo, se no

campo *acct* é especificado que o intervalo mínimo entre uma mensagem de relatório e a próxima é de 1,2s, e o atributo *Interval* foi instanciado com o valor 2, significa que o intervalo entre uma medição e a próxima medição é de 2,4s.

Relatórios periódicos informando condições – são gerados quando as condições especificadas dentro da instancia de *frwkFeedbackLink* são atingidas no intervalo de tempo monitorado. O atributo *frwkFeedbackLinkFlags* pode assumir duas condições:

- *ChangeOnly* - Se este *flag* é fixado no atributo *frwkFeedbackLinkFlags*, a instancia de uso associado só é incluída dentro um relatório periódico não solicitado se seu valor mudar desde o último relatório não solicitado;
- Limiar (*Threshold*) - Se este *flag* é fixado no atributo *frwkFeedbackLinkFlags*, a instancia de uso associado só é incluída dentro um relatório periódico não solicitado se a condição de limiar definida no campo de *frwkLinkThreshold* é atingida pela instância de uso associada.

Podem ser combinadas ambas as condições em um objeto de *frwkFeedbackLinkUsage*. Neste caso, ambas as condições precisam ter sucesso para a instancia de uso a ser informada.

Relatório Solicitado - O PDP pode solicitar avaliação de política de uso emitindo uma Decisão não solicitada. O PEP então prove um relatório de avaliação e continuará realizando avaliações periódicas de acordo com os intervalos estabelecidos na aceitação de conexão inicial pelo PDP.

As condições informando (*ChangeOnly* e *Threshold*) não afeta relatórios solicitados.

3.5 Conclusão

Este capítulo apresentou os principais padrões que serão utilizados na implementação sugerida por este trabalho.

O uso do modelo PBNM adiciona a DiffServ uma solução para facilitar a configuração, avaliação e a administração da rede. Um administrador pode realizar a avaliação todo um domínio DiffServ usando o modelo PBNM. Para a troca de mensagens entre as principais entidades da arquitetura PBNM (PEP e PDP) será utilizado o protocolo COPS-PR. A modelagem da PIB para realizar a avaliação de um domínio de Serviços Diferenciados será realizada de acordo com a proposição contida no COPS-PR associada ao Framework *Policy Information Base* [SAH, 03] e Framework *Policy Information Base for Usage Feedback* [RAW, 03].

Capítulo 4

Controle de Medição em Redes DiffServ

4.1 Introdução

Neste capítulo são descritas algumas estratégias existes para realizar o monitoramento, a medição e estimativa da carga de tráfego de um domínio de DiffServ. Realizando medições em um domínio de Serviços Diferenciados é possível avaliar o nível do serviço prestado aos clientes, e realizar a estimativa de tráfego se os contratos estabelecidos com os clientes (SLAs) estão sendo cumpridos, quais SLAs estão ocupando todos os seus recursos e ainda a banda de tráfego disponível no domínio monitorado.

Para realização de medidas os algoritmos existentes podem ser classificados amplamente em duas categorias: Baseado em Parâmetros (*Parameter-based AC* - PBAC) e Baseado em Medidas (*Measurement-based AC* - MBAC). A estratégia PBAC [HWA, 01] permite estimar a carga da rede baseando-se sempre no pior caso. Ou seja, que todos os fluxos que estão passando pelo link estão ocupando toda a reserva de recursos destinada a eles.

Uma alternativa ao PBAC é o algoritmo Baseado em Medidas (MBAC). Este faz seu cálculo baseado nas medidas de tráfego coletado em intervalos de tempo regulares. A taxa comum de cada amostra é calculada dividindo a soma de pacotes colecionados pelo período de tempo avaliado. O controle de admissão está então baseado nestes valores em lugar de assumir o pior caso. Os mais recentes trabalhos têm utilizado a estratégia MBAC para realizar a medição em domínios de Serviços Diferenciados. Dentro da estratégia MBAC existem ainda duas classificações para as medições: medições fim a fim e medições nos nós da rede que são explicadas na seção 4.2. Nas seções seguintes são apresentados trabalhos desenvolvidos para realizar o monitoramento e a medição da carga de tráfego de um domínio de Serviços Diferenciados.

4.2 Estratégias de Medição

Na literatura encontram-se basicamente duas grandes estratégias de medições de SLAs aplicadas ao domínio de redes DiffServ: estratégia de medição por fluxo ou fim a fim e estratégia de medição baseada nos nós (roteadores) da rede.

Medições fim a fim é uma estratégia que está limitada a medir os efeitos da arquitetura de QoS através da medição dos pontos terminais de um domínio de Serviços Diferenciados, não permitindo desta forma observar o comportamento dos dispositivos do núcleo do domínio de Serviços Diferenciados. Isso impossibilita determinar qual nó do caminho de dados pode estar prejudicando o desempenho da rede.

A medição em nós da rede que permite ao administrador da rede observar o estado atual de cada nó. Assim se o mesmo verificar que um determinado nó está prejudicando o desempenho da rede, é possível estabelecer uma nova rota para o fluxo de dados que está sendo monitorado. As próximas seções apresentam algoritmos desenvolvidos que utilizam as estratégias apresentadas nesta seção.

4.3 FIAC - *Fair Intelligent Admission Control*

Fair Intelligent Admission Control (FIAC) [MIN, 03] é um algoritmo de controle de admissão baseado em medidas, que necessita realizar a sondagem de todo o caminho por onde o fluxo de dados deve passar. A figura 4.1 ilustra o esquema utilizado pelo algoritmo FIAC para realizar o controle de admissão de novos fluxos para o domínio de Serviços Diferenciados.

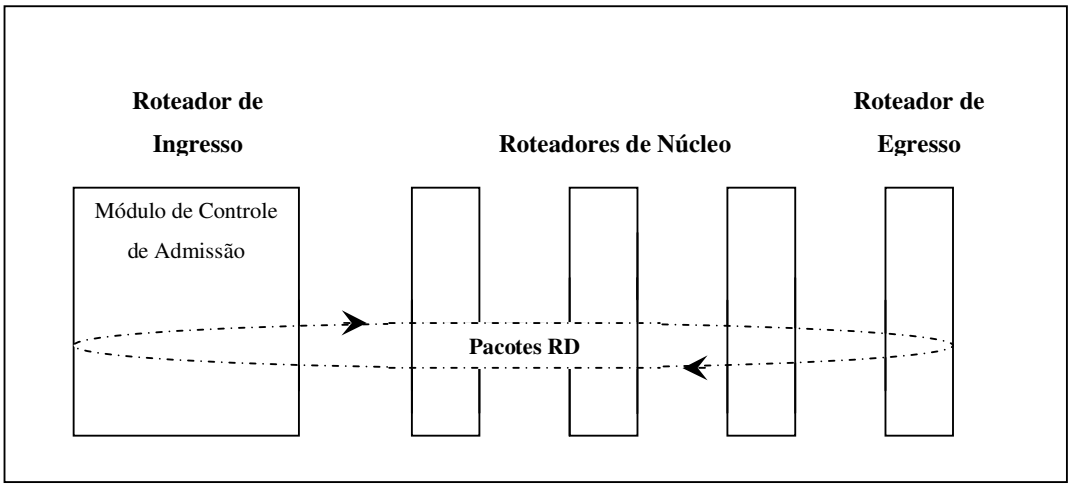

Figura 4.1: Esquema utilizado pelo algoritmo FIAC.

Como ilustrado na figura 4.1 o algoritmo de FIAC é composto por dois módulos adicionais: Protocolo de Descoberta de Recursos (RD - *Resource Discovery*) e pelo Módulo de Controle de Admissão.

- Protocolo RD executa duas funções: primeiramente, é responsável por gerar pacotes RD para avaliação do caminho entre os roteadores de ingresso e egresso do domínio; secundariamente, comunica o Módulo de Controle de Admissão informando dinamicamente capacidade de rede interna. Os pacotes RD gerados são compostos por três campos: DSCP – contém o valor do byte do agregado do fluxo; DIRECTION – específica o destino do pacote RD; e MER – estabelece a taxa explicita do tráfego.
- O Módulo de Controle de Admissão toma decisão de admissão de um novo fluxo baseado no relatório de capacidade da rede do protocolo RD, e nas exigências de QoS do fluxo que está chegando ao nó de ingresso.

Para cada domínio de DiffServ o roteador de ingresso gera pacotes RD que se destinam para o roteador de egresso. Ao receber um pacote de RD, o roteador de núcleo verifica o campo DSCP para descobrir qual classe está sendo avaliada em termos de QoS. Então modifica o campo MER do pacote RD de acordo com a capacidade disponível e envia adiante, para outros roteadores ao longo do caminho para o roteador de egresso. O roteador de egresso é responsável para devolver o pacote RD para o Módulo de Controle de Admissão localizado no roteador de ingresso.

O Módulo de Controle de Admissão ao receber o pacote RD utiliza o algoritmo de FIAC para tomar a decisão de admissão. O algoritmo de FIAC para realizar seu cálculo, verifica qual é a taxa deseja pelo tráfego que está entrando no domínio e o relatório de capacidades do protocolo RD. Se as exigências de QoS do tráfego que está entrado no domínio de Serviços Diferenciados são apoiadas nós de núcleo do domínio, a conexão será admitida; caso contrário, o algoritmo de FIAC controlará o tráfego que está chegando de acordo com a capacidade da rede do núcleo do domínio.

4.4 SRRP - *sender-initiated resource reservation protocol*

O protocolo *sender-initiated resource reservation protocol* (SRRP) [ZHA, 01] é utilizado para estabelecer QoS fim a fim em cima de redes heterogêneas (IntServ e DiffServ).

Para isto implementa um mecanismo de reserva de recursos iniciado pela origem do tráfego de dados.

Nesta estratégia a origem envia uma mensagem de pedido (PT_REQUEST) com a taxa e o pico do fluxo desejado. Esta mensagem é enviada para todos os nós ao longo da rota. Quando a mensagem PT_REQUEST chega a um nó de borda de uma região de Serviços Diferenciados, esta é comparada com os recursos disponíveis na região. A disponibilidade de recursos é determinada pelo contrato estabelecido com o cliente (SLA).

A mensagem PT_REQUEST é ignorada em todos os roteadores internos de um domínio de DiffServ e processada novamente no nó de borda de egresso do domínio. Quando a mensagem PT_REQUEST localiza o receptor, o receptor devolve, a mensagem de pedido como uma mensagem de resposta (PT_APPROVED ou PT_REJECTED) indicando a origem se o pedido foi aceito ou foi rejeitado. A origem recebe a mensagem de resposta, interpreta esta para saber que taxa de tráfego foi admitida para o serviço especificado. A mensagem PT_APPROVED também pode trazer informações de qual valor de marcação do byte DS esta de acordo com a taxa de tráfego admitida no domínio de DiffServ. Desta maneira, é possível obter QoS fim a fim, combinando redes IntServ e redes DiffServ.

4.5 *Sink Trees*

A idéia básica de *sink trees* [CHO, 00] é construir árvores para representar a banda disponível e o *overhead* de cada caminho possível dentro de um domínio de Serviços Diferenciados. Para a construção das árvores foi desenvolvido um algoritmo heurístico que produz um conjunto de árvores em tempo polinomial com mínima demora [CHO, 00].

Uma árvore é constituída por um nó de ingresso, um ou vários nós de núcleo e de egresso. Um nó de ingresso representa o nó pai da árvore, os nós de núcleo representam os ramos e os nós de egresso representam as folhas da árvore.

Cada nó de ingresso do domínio de Serviços Diferenciados possui dois tipos de informação:

1. Uma tabela de roteamento que contém o IP de destino e IP do nó de egresso;
2. Uma segunda tabela que contém dados correspondentes a árvore do roteador de egresso. Esta tabela armazena a banda disponível na árvore, o nó pai, o *overhead* e o número de conexões que esta árvore possui.

A figura 4.2 ilustra as informações contidas nos roteadores de ingresso.

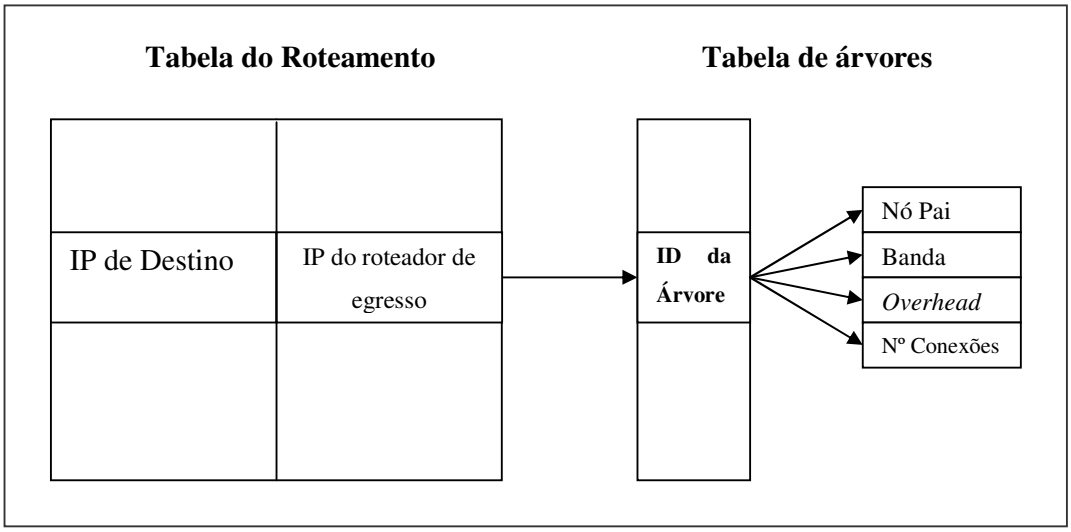

Figura 4.2: Informações contidas nos nós de Ingresso.

Uma vez que o nó de ingresso recebe um novo pedido para aceitar um fluxo de dados, este observa sua tabela de roteamento para verificar o correspondente nó de egresso. Após encontrar na tabela de roteamento o nó de egresso e a correspondente árvore que deve ser utilizada para chegar ao destino é verificado se o fluxo pode ser admitido, baseado na banda disponível e no *overhead* da árvore.

Se o pedido for aceito, cada pacote do fluxo de dados é rotulado de acordo com a árvore que pertence. Assim cada pacote que deixa o nó de ingresso tem um rótulo. Este rótulo é utilizado pelos nós de núcleo do domínio para verificar qual é o próximo salto dentro da árvore. Quando o pacote chega ao roteador de egresso do domínio de Serviços Diferenciados o rótulo é apagado.

4.6 TMAC - *Traffic Matrix based Admission Control*

Traffic Matrix based Admission Control (TMAC) [KER, 04] é um algoritmo de controle de admissão baseado em medidas que não necessita realizar a sondagem de todo o caminho por onde o fluxo de dados deve passar. TMAC realiza a sondagem apenas dos nós de borda de um domínio de Serviços Diferenciados e usa uma matriz de tráfego (TM - *Traffic Matrix*) para predizer o impacto global de admitir um novo fluxo. A matriz de tráfego TM(ingresso, egresso) é uma matriz que representa o volume de tráfico entre todos os pares de roteadores de ingresso e egresso em uma rede Serviços Diferenciados. TMAC também usa uma matriz *Upper-Bound* (U(ingresso, egresso)) que representa o limite máximo de tráfego

admitido entre nós de ingresso e egresso de uma rede de Serviços Diferenciados. A figura 4.3 ilustra as matrizes de tráfego.

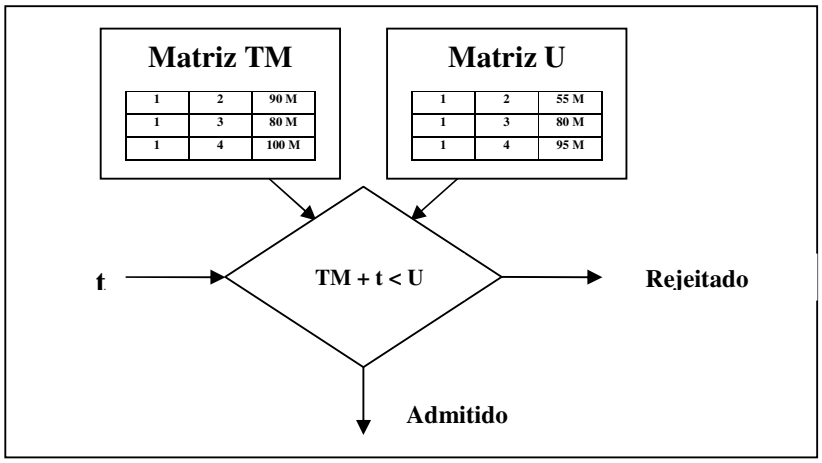

Figura 4.3: Matrizes utilizadas pelo algoritmo TMAC.

Quando um nó de ingresso recebe um novo pedido de fluxo, este pedido é remetido a matriz de tráfego que esta localizada em um nó central. São usados os endereços de origem e destino do pacote IP para identificar os pontos finais do fluxo (roteadores de ingresso e egresso). O novo fluxo será admitido se a matriz TM(ingresso, egresso) mais a taxa do fluxo do pedido (t) forem menores que U(ingresso, egresso). Caso contrário o fluxo será rejeitado.

Uma das vantagens desta abordagem é que TMAC é um algoritmo de controle de admissão que não sonda todos os roteadores do caminho diminuindo assim o consumo de banda.

4.7 Dynamic QoS Adaptation Using COPS and Network Monitoring Feedback

A proposta do artigo [AHM, 03] é baseada em monitorar todos os nós de núcleo de um Domínio de Serviços Diferenciados. Quando um nó de núcleo percebe um uma mudança significativa no estado de rede este envia um relatório ao PDP. O PDP posterior, dependendo do estado de rede, realiza uma nova configuração dos nós de borda do domínio.

Quando o PEP é iniciado, o mesmo faz um pedido de conexão ao PDP local para receber todas as informações das políticas que traficam pelo domínio. O protocolo COPS é utilizado para a comunicação entre o PEP e o PDP. As políticas enviadas pelo PDP ao PEP são processadas e instaladas de acordo com as regras definidas pelo administrador do domínio

de Serviços Diferenciados. A partir deste momento o PEP associado ao nó de núcleo começa a realizar o trabalho de monitoramento do domínio de Serviços Diferenciados. Para realizar o trabalho de monitoramento e calcular o uso da banda o PEP utiliza Modelo de Média Móvel Ponderada Exponencialmente, EWMA - (*Exponential Weighted Moving Average*). O modelo EWMA aplica pesos maiores para dados mais recentes coletados das interfaces do nó, do que para os dados mais antigos. Como resultado, a volatilidade reage mais rapidamente a qualquer mudança no uso da banda. Assim que o PEP percebe uma mudança significativa na banda disponível do nó, um evento externo é ativado para informar ao PDP qual a disponibilidade atual de banda. A mensagem de informação é enviada ao PDP através do protocolo COPS-PR. O PDP então realiza uma nova configuração de QoS de acordo com a disponibilidade de banda informada pelo PEP.

A figura 4.4 ilustra a troca de mensagens entre os PEPs e o PDP.

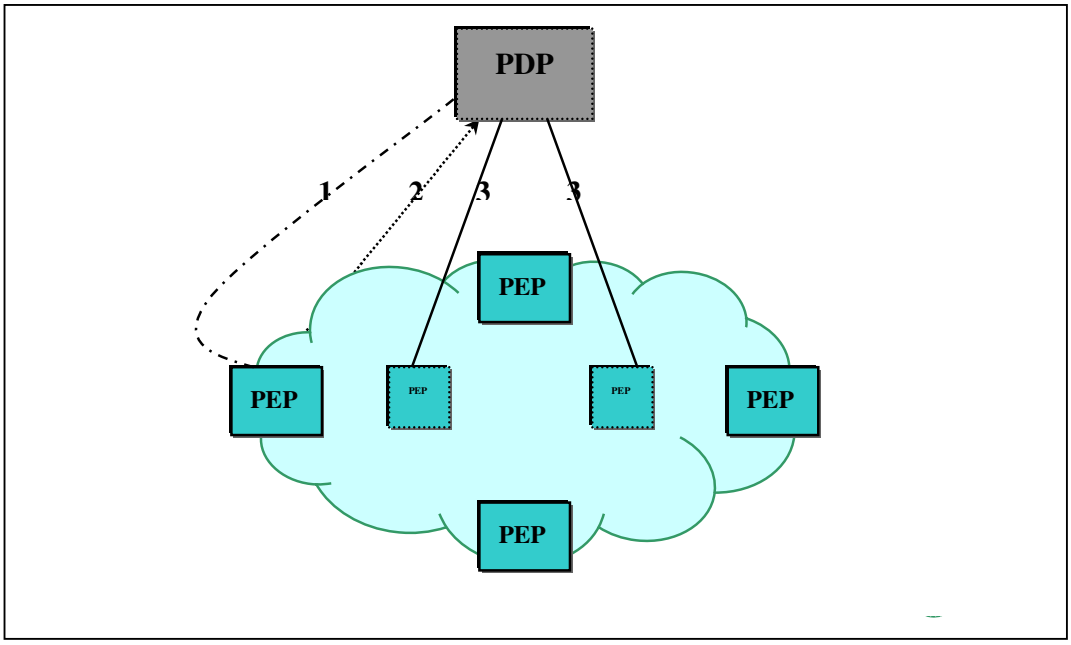

Figura 4.4: Troca de mensagens entre os PEPs e o PDP.

A linha número 1 ilustra a configuração dos nós de borda do domínio na conexão inicial. As linhas identificadas com o número 3 ilustram os relatórios enviados ao PDP pelos PEPs assim que uma mudança na carga de tráfego de um nó de núcleo é percebida e a linha número 2 ilustra a nova configuração que o PDP realiza de acordo com as mensagens de relatório recebidas dos PEPs.

4.8 Conclusão

Os algoritmos fim a fim apresentados têm como desvantagem a exigência que todos os dispositivos da rede suportem os protocolos que geram as mensagens para realizar a sondagem da rede. Outra desvantagem das estratégias apresentadas nas seções 4.2 e 4.3 é o grande número de mensagens injetadas na rede para realizar o monitoramento da rede.

A estratégia 4.4 é baseada no principio de Serviços Integrados e também necessita que os nós da rede suportem o protocolo de sinalização proposto.

O algoritmo TMAC apresentado na seção 4.6 realiza apenas medições dos nós de borda, o que gera um baixo número de mensagens na rede. Porém possui como desvantagem a impossibilidade de observar como está a carga de tráfego dos nós de núcleo do domínio de Serviços Diferenciados.

A estratégia de medição proposta neste trabalho segue a abordagem exposta na seção 4.7. A arquitetura apresentada na seção 4.7 tem como vantagem a utilização de um modelo de gerenciamento proposto pelo IETF o que garante a compatibilidade com qualquer tipo de equipamento de rede.

As principais diferenças em relação ao trabalho apresentado na seção 4.7 é que este propõem uma PIB *Feedback* DiffServ para a representação dos dados monitorados em um domínio de Serviços Diferenciados. A arquitetura proposta também realiza o monitoramento dos nós de ingresso e egresso, o que possibilita também realizar avaliações de tráfegos individuais e não somente do agregado como a abordagem exposta na seção 4.7. Outra vantagem em relação ao modelo anterior é que este está constantemente informando ao PDP o estado atual da rede e não somente quando um sensível aumento no consumo de banda é percebido pelos dispositivos de rede. No próximo capítulo é apresentada a arquitetura proposta.

CAPÍTULO 5

Arquitetura Proposta

5.1 Introdução

Na arquitetura atual proposta pelo IETF, quando um usuário se inscrever para um serviço particular, e estabelece um contrato (SLA) com o administrador do domínio de Serviços Diferenciados. Não lhe é disponibilizada nenhuma garantia que o seu contrato será cumprido.

Os padrões propostos pelo IETF não complementam medições e estimativa de tráfego em redes de Serviços Diferenciados. A arquitetura proposta neste trabalho está baseada em monitorar todos os nós de um domínio de Serviços Diferenciados e reunir as informações coletadas em cada nó do domínio em um dispositivo central denominado PDP. Com todas estas informações reunidas no PDP é possível realizar uma estimativa de tráfego do domínio de Serviços Diferenciados.

Na seção 5.2 são descritos os elementos da arquitetura de medição e estimativa de tráfego que está sendo proposta. Na seção 5.3 é apresentada a PIB proposta. Na seção 5.4 e 5.5 é descrito o funcionamento do PDP e PEP respectivamente. Na seção 5.6 é apresentada ferramenta de estimativa, e por fim na seção 5.7 a conclusão deste capítulo.

5.2 Visão Geral da Arquitetura Proposta

A arquitetura proposta neste trabalho complementa três aspectos:

1. A proposta de uma nova PIB para representar as medições realizadas nos nós da rede DiffServ;
2. Proposta de uma estratégia para agrupar os relatórios recebidos dos PEPs no PDP;

3. Proposta de uma ferramenta para a estimativa de tráfego, que trabalhe com os dados das medições coletados nos dispositivos de rede e posteriormente agrupadas no PDP.

A figura 5.1 ilustra a arquitetura proposta.

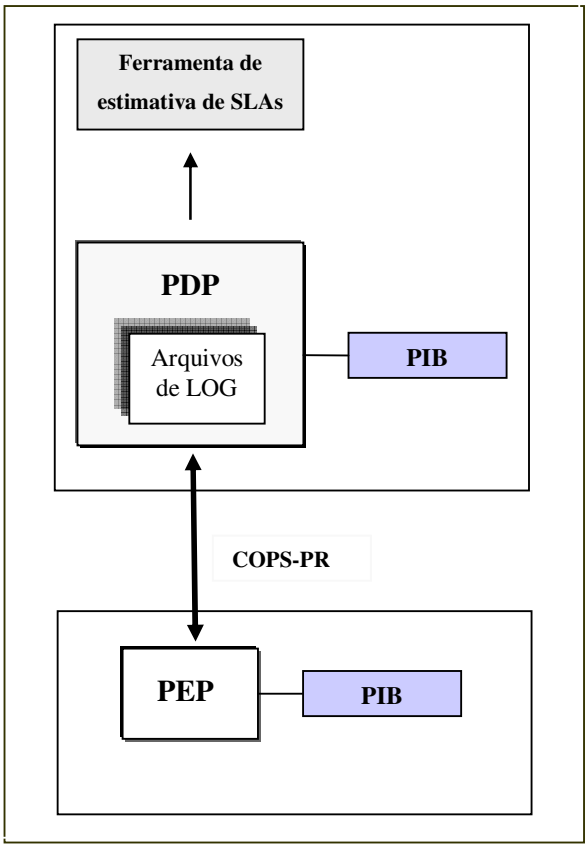

Figura 5.1: Arquitetura proposta.

As próximas seções explicam os elementos apresentados na figura 5.1.

5.3 PIB

Como já apresentado no capítulo 03, a PIB é uma estrutura de dados em forma de árvore, em que os ramos representam as classes e as folhas representam instâncias destas classes.

Antes de se apresentar a estrutura da PIB proposta neste trabalho é necessário realizar alguns explicações para justificar a escolha dos elementos e da estrutura da PIB, nomeada como PIB *Feedback* DiffServ.

5.3.1 Elementos da PIB *Feedback* DiffServ

Para propor a PIB *Feedback* DiffServ, foi necessário realizar estudos com alguns roteadores, para observar quais elementos de medição DiffServ cada equipamento fornece quando Serviços Diferenciados é configurado. Os três roteadores estudados foram: Avaya, Cisco, Linux RedHat. O resultado deste estudo é demonstrado na tabela 5.1. O X são os elementos suportados pelos fabricantes.

Tabela 5.1: Comparativo entre elementos de medição dos roteadores.

Elemento/Descrição	Linux	Avaya	Cisco
Bytes enviados – informa o acumulado de bytes que foram enviados pelo roteador.	X	X	X
Pacotes enviados – informa o acumulado de pacotes que foram enviados pelo roteador.	X	X	X
Pacotes descartados – informa o acumulado de pacotes que foram descartados pelo roteador.	X	X	X
Pacotes *overlimits* – informa quantas vezes a classe solicitou enviar um pacote, mas este não pode ser enviado por restrições de taxa.	X	X	
Número de pacotes na fila – informa o número atual de pacotes na fila do roteador.		X	X
Número de bytes na fila –	X	X	X

informa o número atual de bytes na fila do roteador.			
Informações de configuração – exibe dados da configuração DiffServ realizada no roteador.	X	X	X
Número de pacotes doados pela classe para outra classe – informa o número de pacotes doados para outras classes.	X		
Número de pacotes tomados emprestados da classe pai – informa quantos pacotes à classe tomou emprestado de sua classe superior.	X	X	
Número de pacotes maiores que o mtu (*Maximum Transmission Uni*) configurado no condicionador de tráfego – informa a quantidade de pacotes maiores que o configurado para o condicionador de tráfego.	X		
Número de pacotes que excedem a taxa configurada para cada classe – informa o número de pacotes que excedem a taxa configurada para cada classe instanciada.			X

Os elementos de medição presentes nestes roteadores são propostos para fazerem parte da estrutura da a PIB *Feedback* DiffServ para a representação dos dados monitorados nos

dispositivos de rede, uma vez que a PIB *Feedback* DiffServ deve ser suportada por qualquer roteador.

5.3.2 Estrutura da PIB *Feedback* DiffServ

A PIB Framework *Feedback* RFC 3571 define contadores genéricos que permitem a representação informações previamente configuradas no PEP. Para representar as informações de avaliação de um domínio de DiffServ é necessário estender a PIB Framework *Feedback* a fim de representar as informações coletadas em um domínio de DiffServ. Para realizar a extensão da PIB Framework *Feedback* havia duas alternativas, o uso de atributos nas tabelas já existentes da PIB Framework *Feedback,* ou a criação de um novo grupo de classes. Foi então realizado um estudo para verificar qual era a melhor forma para a PIB proposta representar os dados coletados durante as medições. O estudo baseou-se na comparação de alguns pontos de interesse, como:

- **Complexidade do PDP**: é necessário que a estrutura não venha a prejudicar o desempenho do PDP;
- **Complexidade no PEP**: como no PDP, se a complexidade for aumentada pode haver uma sobrecarga nos dispositivos reduzindo assim seu desempenho;
- **Capacidades**: deve ser possível que o PEP informe para o PDP, na conexão inicial, que parâmetros de medição são suportados pelos dispositivos (PEPs);
- **Extensibilidade para outros tipos de contadores**: a estrutura deve admitir a adição de novos contadores, quando necessário;
- **Estrutura da PIB Framework *Feedback***: é importante que a estrutura proposta não altere a estrutura da PIB **Framework** *Feedback*, uma vez que esta já é uma RFC;
- **Tamanho da PIB**: é importante que o tamanho da PIB em Bytes não aumente exorbitantemente.

A tabela 5.2 ilustra o comparativo realizado entre as classes e os atributos, na escolha por uma das opções para propor a PIB *Feedback* DiffServ.

Tabela 5.2: Comparativo entre Classes e atributos.

Características	Classes	Atributos
1) Complexidade do PDP	O aumento de complexidade no PDP é desconsiderável. A única diferença é que no modelo atual da PIB *Feedback* o PDP tem apenas duas opções de PRCs para indicar ao PEP. Com a adição de novas tabelas o PDP terá várias opções de Classes de Uso para informar ao PEP.	A adição de um novo atributo nas tabelas existentes causa um considerável aumento na complexidade do PDP. Este atributo seria responsável por informar qual parâmetro deve ser medido pelo PEP. O algoritmo para manipular este novo atributo seria maior do que o para indicar uma PRC ao PEP.
2) Complexidade no PEP	A adição de novas classes provoca um aumento desconsiderável de complexidade no PEP, uma vez que ele é informado pelo PDP através da tabela de Ligação que PRC deve instanciar para realizar as medições. Assim se o PEP tiver uma tabela para cada parâmetro de medição o PDP informa qual PRC o PEP deve utilizar para realizar as medições, não havendo assim praticamente nenhuma mudança no algoritmo para instanciar novas classes.	A complexidade do PEP sofre um pequeno aumento. O PEP terá que ficar mascarando o novo atributo das tabelas de Uso. Há um aumento de complexidade no que se refere à carga semântica, uma vez que o mesmo atributo é utilizado por várias vezes para parâmetros estatísticos diferentes. Neste caso o algoritmo para realizar o mascaramento do atributo seria muito maior que o para instanciar novas classes.
3) Capacidades	Com a adição de tabelas torna-se fácil informar ao PDP quais parâmetros de	Com a adição do atributo nas tabelas de Uso, não é possível informar ao PDP que

	medições são suportados pelo PEP. Uma vez que cada tabela (PRC) é responsável por um parâmetro estatístico. Também é possível reaproveitar estas informações, em uma próxima conexão.	parâmetros de medição o PEP suporta.
4) Extensibilidade para outros tipos de contadores	Cada tabela pode ter novos atributos além dos herdados da classe superior. Para alguns parâmetros estatísticos os dois atributos herdados da classe superior são insuficientes para realizar a coleta de informações dos parâmetros estatísticas.	Com atributo não é possível à adição de novos atributos de medição para um determinado parâmetro estatístico. A única maneira seria adicionar um novo atributo às classes de Uso, no entanto esta estratégia provocaria uma nova modificação na estrutura da PIB *Feedback*.
5) Estrutura da PIB Framework *Feedback*	A adição de novas classes não altera a estrutura atual da PIB Framework *Feedback*, sendo que só são criadas sub-classes das classes de Uso. Estas novas classes compõem a PIB *Feedback* DiffServ.	A adição de atributo provoca modificações na estrutura atual da PIB Framework *Feedback*, uma vez que é necessário inserir um novo atributo nas PRCs de Uso para indicar que parâmetro está sendo medido.

6) Tamanho da PIB	O tamanho da PIB aumenta consideravelmente com a adição de novas classes uma vez que mais bytes são necessários para armazenar o OID de cada nova tabela.	Com a adição de atributo nas tabelas de Uso o tamanho da PIB sofre uma pequena variação. Uma vez que um novo atributo é inserido em cada classe. Este atributo exige mais alguns bytes.

Dos seis pontos estudados cinco foram favoráveis ao uso de classes. As classes não alteram a estrutura da PIB Framework *Feedback*, pois, só é necessário, à adição de novas PRCs herdeiras das Classes de Uso. Com a criação de tabelas também é possível informar ao PDP que classes ou que parâmetros de medição são suportados por cada dispositivo. As tabelas também possibilitam adicionar novos atributos de medição para os parâmetros que apenas contagem de bytes e pacotes são insuficientes.

A estratégia de uso de atributos nas classes existentes em comparação com a criação de novas classes apresentaram muitos problemas. Para utilizar atributos seria necessário adicionar novos parâmetros nas Classes de Uso e de Seleção, mudando assim a estrutura da PIB Framework *Feedback*. Além disso, haveria um sensível aumento na complexidade do PDP e do PEP, isto, pois o mesmo atributo seria utilizado por várias vezes para representar diferentes tipos de informação, havendo assim uma sobrecarga semântica em ambos os dispositivos (PDP e PEP). Além destes problemas os atributos não possibilitariam informar ao PDP na conexão inicial quais parâmetros de medição estão disponíveis nos PEPs, o que poderia acabar tornando uma ordem do PDP inválida uma vez que o mesmo não teria o conhecimento de quais parâmetros que são suportados pelos PEPs. Com o uso de uma classe (PRC) para cada parâmetro de medição, quando o PEP e PDP estabelecerem a conexão inicial, o PEP pode informar ao PDP que PRCs suporta, evitando assim que PDP tome uma decisão incorreta no momento em que solicitar relatórios aos PEPs.

5.3.3 A PIB *Feedback* DiffServ Proposta

A figura 5.2 ilustra os grupos da PIB *Feedback* DiffServ proposta. Estes grupos compõem a PIB *Feedback* DiffServ, que congrega, tanto conceitos apresentados no *Framework* DiffServ (*GrupoClassifier* e *GrupoDeviceCapabilities*) e na PIB Framework

Feedback (*GrupoSelection*, *GrupoLink* e *GrupoUsage*) – que correspondem aos elementos comuns, necessários a qualquer tipo de cliente – como a forma proposta para a representação e implementação da arquitetura de medição e estimativa de tráfego para redes DiffServ (*UsageDiffServGroup*).

Nesta visão geral da representação PIB *Feedback* DiffServ deve-se ter em mente como abordado no capítulo 3, que tabelas (classes PRCs) são utilizadas a fim de agrupar as informações, todos os atributos são constituídos a partir de instâncias (PRI) e ponteiros baseados em identificações únicas OID, utilizados para implementar a associação entre classes. Esta abordagem é necessária para que seja possível a utilização do protocolo COPS-PR.

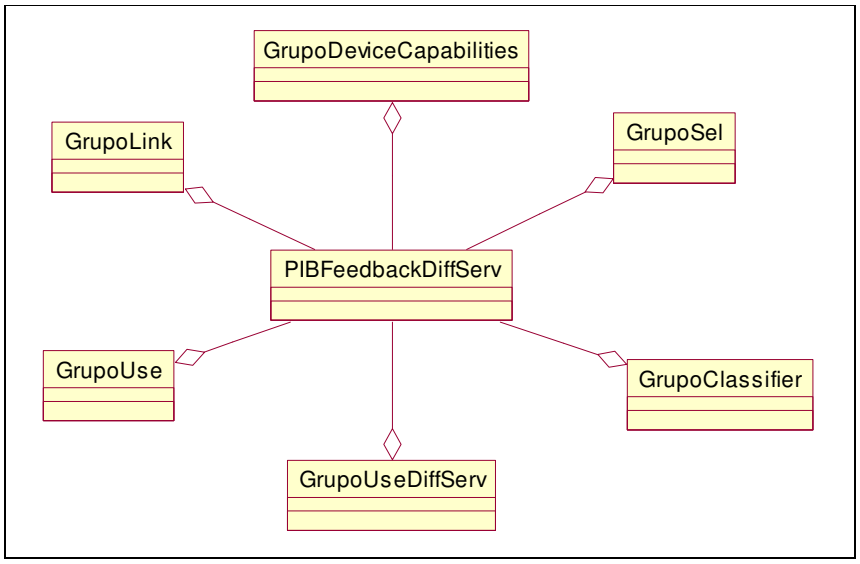

Figura 5.2: PIB *Feedback* DiffServ.

A partir desta caracterização serão descritos, a seguir, os grupos constantes da PIB *Feedback* DiffServ.

Grupo *Device Capabilities*: Este grupo tem como função descrever as características do PEP e as combinações de papéis possíveis para o mesmo. A figura 5.3 ilustra o grupo *Device Capabilities*.

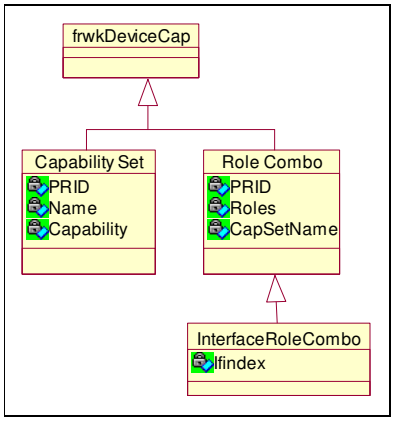

Figura 5.3: Grupo *DeviceCapabilities*.

Este grupo é composto por duas tabelas:

- *CapabilitiesSet* – Esta tabela descreve o conjunto de capacidades disponíveis nas interfaces do dispositivo. É composta por três campos: *Prid* que identifica unicamente a instância, *Name* que descreve o nome do conjunto de capacidades devendo ser único e não nulo e *Capability* que corresponde ao identificador da classe e instância (OID) de uma capacidade associada.
- *InterfaceRoleCombo* – Esta tabela relaciona interfaces a papéis desempenhados e é composta por quatro campos: *Prid,* que identifica unicamente a instância, Roles que relaciona os papéis associados à interface, *CapSetName* que relaciona a uma capacidade e *IfIndex* que define a interface.

Grupo *Classifier*: o grupo *Classifier* tem como função permitir a criação de filtros e classificadores que atuem sobre o protocolo IP. A figura 5.4 ilustra o grupo *Classifier*.

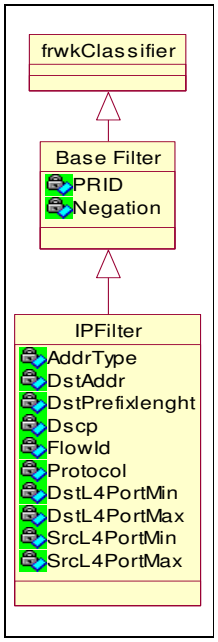

Figura 5.4: Grupo *ClassifierGroup*.

Este grupo é composto por uma única tabela:

- *IPFilter* – Define um filtro para pacotes IP. É composto por quatorze campos: *Prid* que define unicamente uma instância, *Negation* que corresponde a um *BaseFilter* e define se o filtro será aplicado ou não, *AddrType* corresponde ao tipo do endereço contido no pacote *IPv4* ou *IPv6*, *DStAddr* corresponde endereço de destino, *DstPrefixLength* corresponde ao tamanho relativo à mascara do endereço de destino, *SrcAddr* corresponde ao endereço de origem do pacote, *SrcPrefixLength* corresponde ao tamanho relativo à máscara do endereço de origem, *Dscp* corresponde ao valor que o *dscp* deve corresponder no pacote, *FlowId* corresponde ao identificador do fluxo num pacote *IPv6*, *Protocol* corresponde ao nome do protocolo constante do pacote, *DstL4PortMin*, *DstL4PortMax*, *SrcL4PortMin*, *SrcL4PortMax* definem os intervalos para os valores relativos às portas de origem e destino de um pacote camada 4.

Grupo *Selection*: este grupo tem como função definir as condições usadas pelo PEP para monitorar e registrar uma política de uso. A figura 5.5 ilustra o grupo de seleção.

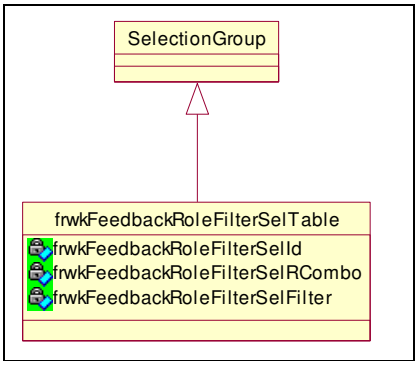

Figura 5.5: Grupo *Selection*.

Este grupo é composto por uma única tabela:

- *frwkFeedbackRoleComboFilterSelectionTable* - é composta por três campos: *RoleFilterSelId* que define unicamente uma instancia, *RoleFilterSelRCombo* que corresponde a identificação da instancia da tabela *InterfaceRoleCombo* e *RoleFilterSelFilter* que corresponde a identificação de instancias de filtros.

Maiores detalhes sobre este grupo encontra-se no capítulo 3 seção 3.4.2.

Grupo *Usage*: este grupo tem como função descrever que atributos são registrados pelo PEP. A PIB Framework *Feedback* define duas classes de política de uso, frwkFeedbackTraffic e *frwkFeedbackIfTraffic*. A figura 5.6 ilustra o grupo de uso.

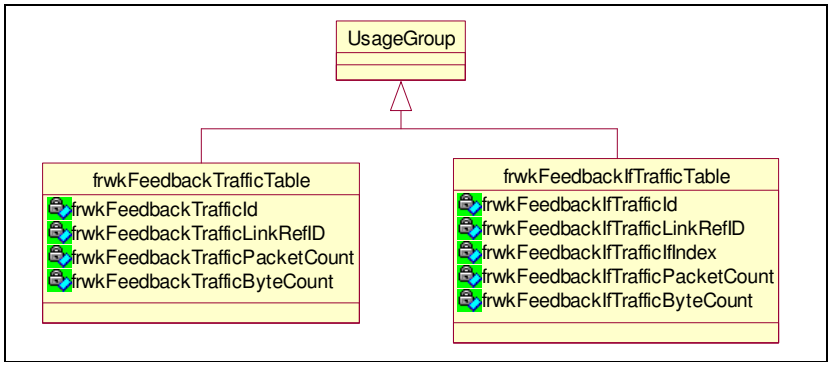

Figura 5.6: Grupo *Usage*.

Maiores detalhes sobre este grupo encontram-se no capítulo 3 seção 3.4.2.

Grupo *Link*: alguns elementos estatísticos como, por exemplo, número de pacotes na fila, não fornece o acumulado de pacotes durante um determinado período de medição. Para

casos como esse, foi necessário criar um novo atributo na tabela *frwkFeedbackLinkTable* nomeado *fdbckDiffServMeasAlgotm*. Este atributo é responsável por informar ao PEP qual valor coletado durante as medições no período de tempo monitorado deve ser envaido nas mensagens de relatório ao PDP. Há três possibilidades: enviar o maior valor, a média dos valores coletados ou ainda o menor valor. O algoritmo responsável por executar a configuração imposta pelo PDP é descrito na seção 5.3. O restante da explicação dos atributos da tabela *frwkFeedbackLinkTable*, e das outras tabelas do grupo *Link* encontram-se no capítulo 3 na seção 3.4.2. A figura 5.7 ilustra o grupo *Link*.

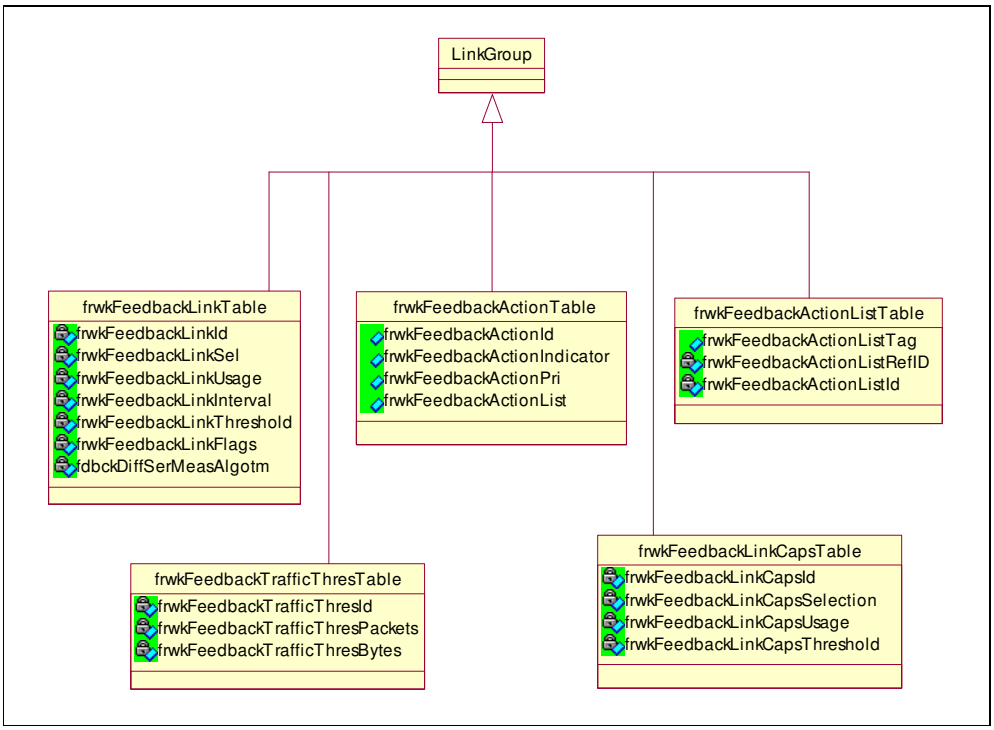

Figura 5.7: Grupo *Link*.

Grupo *UsageDiffServ*: este grupo tem como função descrever que atributos de medição DiffServ que são registrados pelo PEP. Este novo grupo de tabelas herdam os atributos das classes de Uso *frwkFeedbackTrafficTable* e *frwkFeedbackIFTrafficTable* respectivamente. Cada uma das respectivas tabelas é super-classe de quinze tabelas do grupo Classes de Uso DiffServ, assim o grupo *UsageDiffServ*, é composta por trinta tabelas.

As tabelas do grupo *frwkFeedbackIFTrafficTable* são idênticas as do grupo *frwkFeedbackTrafficTable*, a única diferença é que as tabelas do grupo *frwkFeedbackIFTrafficTable* tem um campo a mais (*frwkFeedbackIFtrafficIndex*) em cada

tabela para casos em que se deseja medir mais de uma interface do equipamento de rede. A figura 5.8 ilustra apenas as tabelas do grupo *UsageDiffServ* que são filhas da classe *frwkFeedbackTrafficTable*.

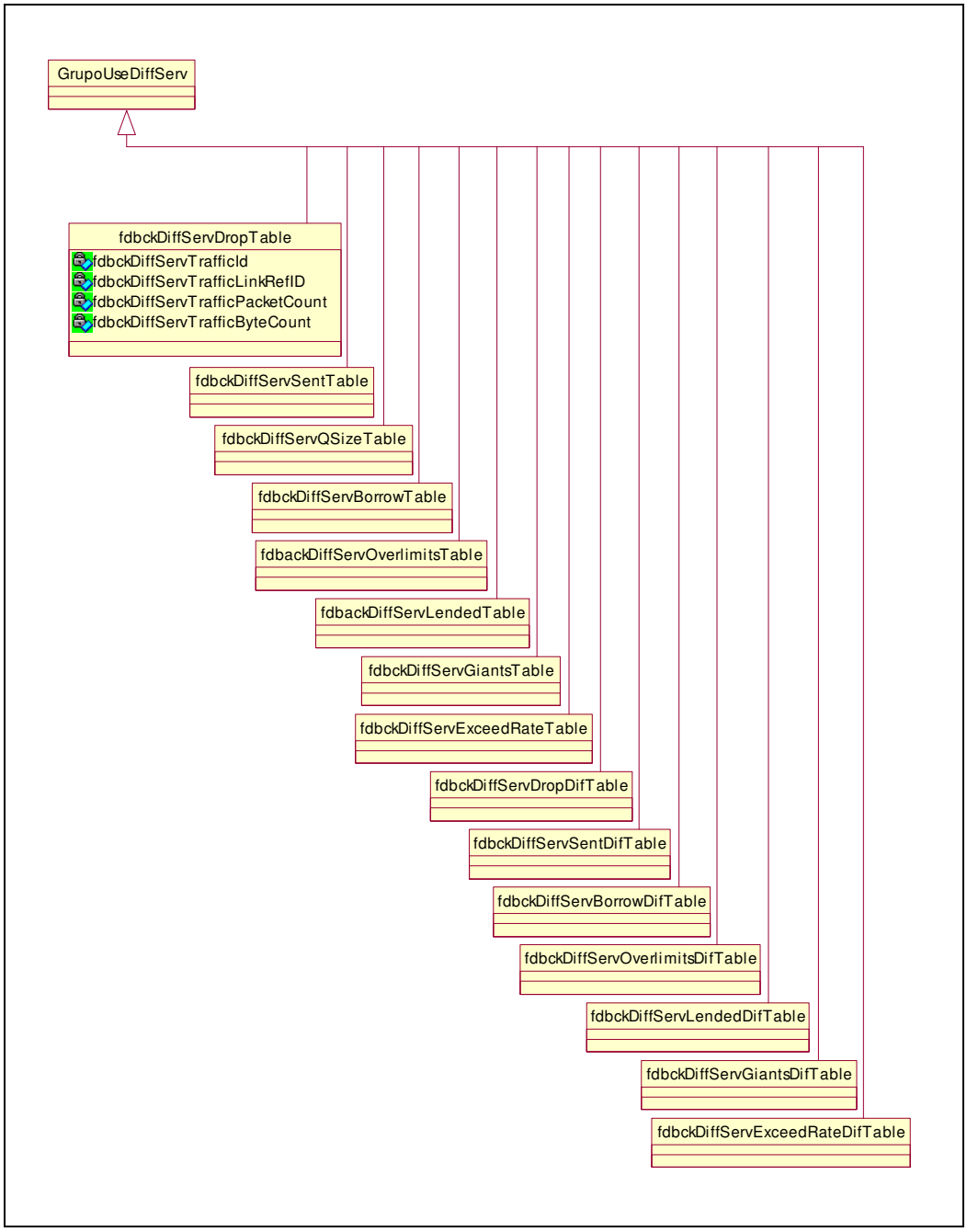

Figura 5.8: Grupo *UsageDiffServGroup*.

Os atributos ilustrados na tabela *fdbckDiffSevDropTable* na figura 5.8 estão presentes em todas as classes do grupo *UsageDiffServ* , uma vez que estes atributos são herdados das tabelas *frwkFeedbackTrafficTable* e *frwkFeedbackIFTrafficTable* respectivamente.

Pode-se dividir as tabelas do grupo *UsageDiffServ* em dois grupo distintos. As tabelas *fdbckDiffSevDropTable*, *fdbckDiffServSentTable*, *fdbckDiffServQSizeTable, fdbckDiffServBorrowTable*, *fdbckDiffServOverlimitsTable*, *fdbackDiffServLendedTable, fdbckDiffServGiantsTable* e *fdbckDiffServExceedRateTable* realizam medidas instantâneas ao final de cada intervalo de tempo (*acct* vezes o valor do atributo *Interval*) e enviam o valor coletado para o PDP. As tabelas *fdbckDiffSevDropDifTable*, *fdbckDiffServSentDifTable*, *fdbckDiffServBorrowDifTable*, *fdbckDiffServOverlimitsDifTable*, *fdbackDiffServLendedDifTable, fdbckDiffServGiantsdifTable* e *fdbckDiffServExceedRateDifTable* enviam somente a diferença medida no período de tempo especificado no campo *acct* vezes o valor do atributo *Interval*. Nesta estratégia o PEP tem que realizar uma coleta no início do intervalo medido e outra ao final do intervalo. A seguir são descritas as particularidades de cada tabela do grupo de *UsageDiffServ*:

- *fdbckDiffSevDropTable*: informa o acumulado de pacotes que foram descartados pelos roteadores;
- *fdbckDiffServSentTable*: informa o acumulado de bytes e pacotes que foram enviados pelo roteadores;
- *fdbckDiffServQSizeTable*: informa o tamanho atual da fila;
- *fdbckDiffServBorrowTable*: informa o acumulado de pacotes que foram tomados emprestados da classe superior;
- *fdbckDiffServOverlimitsTable*: informa o acumulado do roteador de quantas vezes a classe solicitou enviar um pacote mas ele não pode ser enviado por restrições de taxa;
- *fdbackDiffServLendedTable*: informa o acumulado de pacotes doados por esta classe de sua taxa para outras classes. Enquanto a tabela *fdbckDiffServBorrowTable* informa acumulado de pacotes tomados emprestados da classe superior a tabela *fdbackDiffServLendedTable* informa o acumulado de pacotes que esta classe emprestou a outras classes;
- *fdbckDiffServGiantsTable*: informa o acumulado de pacotes maiores que o *mtu* configurado no condicionador de tráfego dos dispositivos de rede;
- *fdbckDiffServExceedRateTable*: informa o acumulado de pacotes que excedem a taxa configurada para cada classe instanciada;

- *fdbckDiffSevDropDifTable*: informa quantos pacotes foram descartados pelos roteadores no período medido;
- *fdbckDiffServSentDifTable*: informa quantos bytes e pacotes foram enviados pelo roteadores no período medido;
- *fdbckDiffServBorrowDifTable*: informa quantos pacotes foram tomados emprestados da classe superior no período medido;
- *fdbckDiffServOverlimitsDifTable*: informa quantas vezes a classe solicitou enviar um pacote mas ele não pode ser enviado por restrições de taxa, no período medido;
- *fdbackDiffServLendedDifTable*: informa o número de pacotes doados por esta classe de sua taxa para outras classes, no período de tempo medido. Enquanto a tabela *fdbckDiffServBorrowDifTable* informa o número de pacotes tomados emprestados da classe superior a tabela *fdbackDiffServLendedTable* informa o número de pacotes que esta classe emprestou a outras classes, no período de tempo medido;
- *fdbckDiffServGiantsDifTable*: informa o número de pacotes maiores que o *mtu* configurado no condicionador de tráfego dos dispositivos de rede, para o período de tempo medido;
- *fdbckDiffServExceedRateDifTable*: informa o número de pacotes que excedem a taxa configurada para cada classe instanciada, no período de tempo medido.

5.4 Descrição do Funcionamento do PDP

Quando o PDP recebe as mensagens de relatório dos PEPs associados a si, é necessário reunir todas estas informações em um único local no PDP de forma a facilitar a compressão e o trabalho com as informações obtidas através das medições realizadas nos dispositivos de rede. Neste trabalho propõe-se que sejam criados arquivos de *log* para reunir as informações das medições realizadas.

Quando o PDP aceita uma nova conexão, um arquivo de *log* é criado para o PEP que está se conectando ao PDP. O arquivo de *log* armazena todas as informações enviadas pelo PEP ao qual faz referência.

As mensagens de relatório que o PDP recebe dos dispositivos de rede contêm os dados da medição realizada e referência da instancia da tabela *frwkFeedbackLinkTable* que instanciou a medição no correspondente dispositivo de rede monitorado. O algoritmo descrito na tabela 5.3 realiza a leitura da mensagem de relatório que o PDP acabou de receber para verificar a qual instancia da tabela *frwkFeedbackLinkTable* o relatório faz referência. Após o

algoritmo obter todas as informações necessárias o mesmo realiza a gravação dos dados no arquivo de *log*. Para que posteriormente a ferramenta de estimativa de tráfego proposta na seção 5.7 possa fazer uso das informações obtidas através das medições realizadas.

Tabela 5.3: Algoritmo para inserir dados no arquivo de *log*.

Passo	Descrição
1	Realizar a leitura do campo *handle* no cabeçalho da mensagem de relatório recebida para verificar qual arquivo de *log* deve receber as novas informações.
2	Após obter a identificação do PEP que enviou a mensagem de relatório é realizada a leitura do conteúdo da mensagem para obter as informações das medições realizadas e a referencia da tabela *frwkFeedbackLinkTable* que instanciou a medição no dispositivo de rede.
3	Procurar na PIB de configuração a PRID da tabela *frwkFeedbackLinkTable* que contém o mesmo valor contido na mensagem de relatório recebida pelo PDP.
4	Ao encontrar a PRID da tabela *frwkFeedbackLinkTable* na PIB de configuração, ler o atributo *frwkFeedbackLinkSel*.
5	Localizar na PIB de configuração o valor instanciado no atributo *frwkFeedbackLinkSel*.
6	Ao localizar PRID referente ao atributo *frwkFeedbackLinkSel* realizar a leitura dos atributos *frwkFeedbackRoleFilterSelRCombo* e *frwkFeedbackRoleFilterSelFilter* para descobrir o papel da interface e o critério de seleção que foi utilizado.
7	Ao encontrar os dados do passo 6, inseri-los no arquivo de *log*, juntamente com os dados da medição recebidos na mensagem de relatório.
8	Fechar o arquivo de *log*.

5.5 Funcionamento do PEP

5.5.1 Algoritmo de tradução da PIB para o dispositivo de rede

Quando o PEP receber a PIB *Feedback* DiffServ contendo o conjunto de regras de configuração selecionado pelo PDP. O PEP deve efetuar aqui uma tradução destas regras para

comandos reconhecidos pelo equipamento de rede controlado por si. A tabela 5.4 ilustra o algoritmo de tradução da PIB para o dispositivo de rede.

Tabela 5.4: Algoritmo de tradução da PIB para o dispositivo de rede.

Passo	Descrição
1	Fazer leitura da PRC *frwkFeedbackActionTable* para saber que ação deve ser aplicada as instancias da PRC *Link*. Neste momento o algoritmo tem duas opções: suspender as medições que estão sendo realizadas, ou instanciar novas medições. O algoritmo verifica se a opção escolhida é para todas as instancias da PIB *Feedback* DiffServ ou apenas para um grupo específico de instancias. Se a decisão for aplicada a todas as instancias o algoritmo segue para o passo 3. Caso contrário vai para o passo 2.
2	Nesta fase o algoritmo verifica quais instancias da PRC *frwkFeedbackLinkTable* vão sofrer a ação escolhida no passo 1.
3	Neste momento o algoritmo instancia que parâmetros devem sofrer a ação escolhida no passo 1. O algoritmo consulta a tabela *frwkFeedbackLinkTable* que contém as PRC que devem ser instanciadas para realizar as medições. O intervalo de tempo é definido de acordo com o valor do campo *acct* vezes o atributo *frwkFeedbackLinkInterval* da tabela *frwkFeedbackLinkTable*.
4	Neste momento uma nova Thread é criada para realizar as medições na política específica.

5.5.2 Algoritmo de leitura no dispositivo de rede

Este algoritmo é utilizado no PEP quando é necessário realizar o monitoramento de um elemento de medição que não fornece o acumulado de bytes ou pacotes no intervalo de tempo monitorado. Como exemplo pode-se citar quantidade de pacotes na fila que estão variando a todo o momento. Para parâmetros de medição como estes que não fornecem o acumulado pode-se através do algoritmo proposto na tabela 5.5 realizar várias medições no período de tempo monitorado e setar o valor que deve ser enviado ao PDP.

Tabela 5.5: Algoritmo para realizar a leitura no dispositivo de rede.

Passo	Descrição
1	Realizar várias leituras no dispositivo de rede no intervalo de tempo monitorado e armazenar os valores coletados em um vetor.
2	Verificar qual a opção selecionada pelo administrador da rede no atributo *fdbckDiffServMeasAlgotm* da tabela *frwkFeedbackLinkTable* ao instanciar a nova política no PEP.
3	Caso se tenha optado pelo maior valor, realizar uma varredura no vetor procurando pelo maior valor coletado.
4	Se a opção selecionada for à média dos valores coletados, realizar a somatória de todos os elementos armazenados no vetor e dividir pelo respectivo número de elementos do vetor.
5	Caso se tenha optado pelo menor valor, realizar uma varredura no vetor procurando pelo menor valor coletado.

5.6 Ferramenta de estimativa de tráfego proposta

A ferramenta de estimativa de tráfego proposta neste trabalho utiliza os arquivos de *log* criados no PDP para realizar a estimativa de tráfego de uma rota de fluxo de dados pertencente ao domínio de Serviços Diferenciados monitorado. A figura 5.9 ilustra os elementos que devem ser informados para que a ferramenta de estimativa possa realizar a avaliação do tráfego.

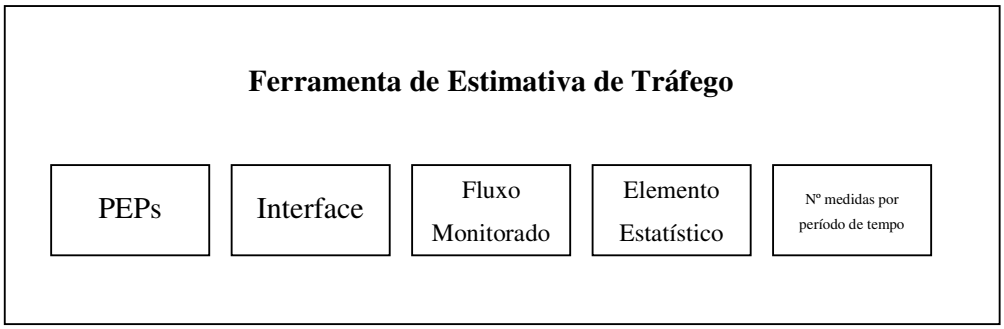

Figura 5.9: Ferramenta de Estimativa.

Na ferramenta de estimativa de tráfego proposta o administrador deve informar os PEPs (cada PEP possui um arquivo de *log*) que pertencem a rota a qual se deseja realizar uma estimativa de tráfego. E a interface de cada equipamento de rede que deseja monitorar.

Após administrador escolher os PEPs que pertencem rota, ou seja, o caminho do tráfego de dados, o mesmo tem que informar a ferramenta de estimativa de tráfego o fluxo de dados que deseja obter estimativas. Neste momento o administrador pode optar por realizar uma estimativa para um fluxo de dados em particular ou para um agregado. Se administrador deseja obter uma estimativa de um fluxo particular este pode ser classificado por porta, protocolos e endereços IP. Caso o tráfego que se deseja obter informações seja o agregado do domínio de DiffServ, o administrador deve escolher o valor DSCP do agregado que deseja obter as estimativas.

Após o administrador informar os nós do domínio de Serviços Diferenciados e o fluxo que deseja obter as estimativas é necessário informar ainda o elemento que deseja obter as estimativas. Neste momento o administrador tem diversas opções. Pode escolher quantidade de pacotes descartados, pacotes enviados, atraso, pacotes emprestados, etc. Ou seja, todos os elementos que foram definidos no grupo de Uso DiffServ da PIB *Feedback* DiffServ.

Quando estes dados forem informados pelo administrador a ferramenta de estimativa realiza uma leitura dos arquivos de *log* (PEPs que deseja obter as informações) para montar o relatório com as estimativas. Por exemplo, se o administrador deseja obter o número de pacotes descartados do agregado de fluxo de dados marcado com valor DSCP AF11, o mesmo deve informar os PEPs por aonde o agregado passa no seu domínio de DiffServ. Se o administrador informar ao programa que os dados do agregado AF11 passam pelos PEPs A, B e C, o sistema vai realizar a leitura dos três arquivos de *log* respectivamente para obter o número de pacotes descartados neste caminho. A tabela 5.6 ilustra os passos do algoritmo utilizado pela ferramenta de estimativa proposta.

Tabela 5.6: Algoritmo utilizado pela ferramenta de estimativa.

Passo	**Descrição**
1	Aguardar até que todas as informações necessárias para a execução do algoritmo sejam informadas a ferramenta de estimativa, caso algum dado não for informado, avisar ao usuário que todos os campos devem ser preenchidos.
2	Ler o conteúdo do arquivo de *log* do PEP informado na interface da ferramenta de estimativa. Caso não encontrar o arquivo de *log* informar ao usuário que o PEP não possui arquivo de *log*.
3	Encontra no conteúdo do arquivo a interface que se deseja obter os dados estatísticos, caso não for encontrada avisar ao usuário que o arquivo de *log*

	não possui informações sobre esta interface.
4	Se a interface for encontrada, verificar a qual fluxo de dados as informações fazem referencia, se a identificação do fluxo de dados for igual ao fluxo de dados informado na interface da ferramenta passar para o passo 5. Caso contrário passar para a próxima linha do arquivo de *log*.
5	Procurar pelo elemento estatístico informado a ferramenta no arquivo de *log*, ao encontrar armazenar no vetor A o valor coletado e voltar ao passo 3. Se o número de medidas coletadas for igual ao informado passar ao passo 7. Se for o fim do arquivo passar para o passo 6.
6	Voltar ao passo 2 até que todos os arquivos de *log* informados na interface de ferramenta de estimativa de tráfego sejam lidos. Se todos os arquivos já foram lidos passar para o passo 8.
7	Encontrar no vetor A o maior valor coletado do arquivo de *log* caso o parâmetro estatístico for a fila do roteador. Para os outros parâmetros estatísticos realizar a soma dos valores armazenados no vetor A. Armazenar o valor obtido no vetor B, e limpar o vetor A. Voltar ao passo 3.
8	Para cada arquivo de *log* lido, foi criado um vetor B. Somar a posição zero de todos os vetores B e armazenar no vetor C, realizar a soma da posição um dos vetores B e armazenar no vetor C. Realizar a soma até que todas as posições do vetor B de cada PEP sejam lidas e armazenadas no vetor C.
9	Exibir o relatório das estimativas obtidas.

5.7 Conclusão

Neste capítulo foi apresentada a estrutura geral da arquitetura proposta para automatizar o processo de leitura das estatísticas fornecidas pelos dispositivos de rede. Para tal, foi proposta uma nova PIB para configurar quais parâmetros estatísticos devem ser monitorados e posteriormente enviados relatórios ao PDP. 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 a requisições de relatórios e das leituras realizadas nos dispositivos de redes. A solicitação de relatórios ao PEP e o envio de relatórios ao PDP é realizada através de instâncias (PRIs) da PIB *Feedback* DiffServ, que contém os elementos funcionais e os parâmetros que se deseja obter relatórios. O protocolo COPS-PR é utilizado para configurar as políticas que devem ser medidas nos dispositivos e para enviar os relatórios ao PDP.

Um sistema de medição que segue as diretrizes e os padrões do IETF garante sua compatibilidade com a maior parte dos fabricantes de equipamentos de redes.

Os detalhes utilizados para o desenvolvimento da arquitetura proposta são apresentados no próximo capítulo.

CAPÍTULO 6

Implementação

6.1 Introdução

A arquitetura proposta no capítulo anterior é independente de uma linguagem de programação específica. Entretanto, para a efetiva implementação do trabalho no ambiente a ser monitorado, é necessário que as informações de decisão enviadas ao PEP sejam armazenas em um determinado repositório. Havia duas possibilidades: LDAP (*Lightweight Directory Access Protocol*) e XML (*Extensible Markup Language*). Neste trabalho optou-se por representar as informações em arquivos XML, devido as suas características de flexibilidade para representação das informações e pela variedade de ferramentas disponíveis para manipulação de documentos XML.

Neste capítulo também é explicado o processo de tradução das decisões enviadas pelo PDP ao PEP para a realização das medições no Sistema Operacional Linux, e o desenvolvimento do protocolo COPS-PR para o envio dos relatórios ao PDP. Na seção 6.5 é apresentada a ferramenta de estimativa e de tráfego desenvolvida. A figura 6.1 da uma visão geral de toda a arquitetura desenvolvida neste trabalho.

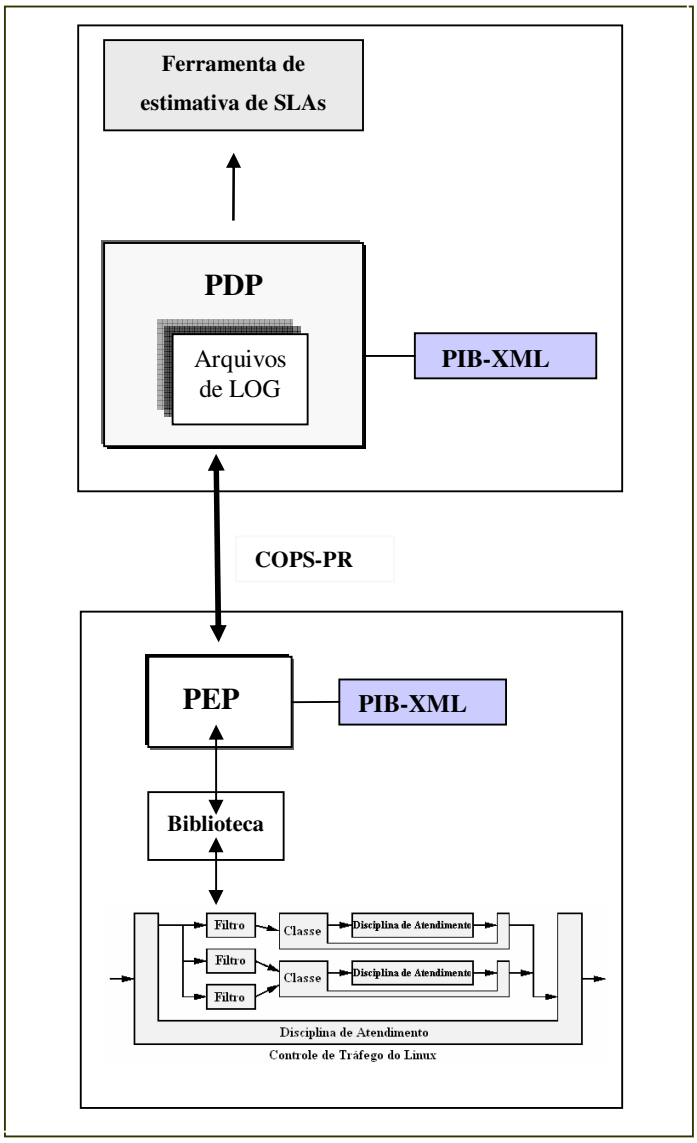

Figura 6.1: Visão geral de toda a arquitetura implementa.

A estrutura deste capítulo segue a figura 6.1.

6.2 Mapeamento PIB *Feedback* DiffServ em XML

Dentro da proposta deste trabalho, as classes (PRCs) da PIB *Feedback* DiffServ são mapeadas em uma estrutura de elementos XML. A estratégia utilizada no mapeamento dos módulos PIB em XML segue o esquema ilustrado na figura 6.2.

```
<Prc oid=“ ”>
        <Prid id=“ ”>
                <Atributo1 type=“ ”>valor</Atributo1>
                                .
                                .
                                .
                <AtributoN type=“ ”>valor</AtributoN>
        </Prid>
</Prc>
```

Figura 6.2: Estratégia de mapeamento para XML.

Os elementos apresentados na figura 6.2 são explicados a seguir:

- **Prc oid**: Indica o início do elemento que representa uma classe ou tabela com seu respectivo “oid”;
- **Prid id**: representa um nó filho que unicamente identifica uma instância da Prc através do atributo “id”;
- **Atributo1 type**: Representa os atributos de cada classe, o “type” é utilizado para representar o tipo de valor de cada atributo (inteiro, string, byte, etc);
- **/Prid**: Indica o fim da instância;
- **/Prc**: Indica o fim da classe.

Cada classe da PIB proposta é definida como um elemento XML e é identificada por um “oid” único que permite implementar a associação entre classes. No caso, o identificador 1.3.6.1.2.2.5 foi utilizado para representar o novo conjunto de contadores. É importante ressaltar que a representação em XML é utilizada apenas internamente na implementação do PEP e PDP. Durante o processo de transmissão de informação do PEP para o PDP, utilizando o protocolo COPS-PR, as variáveis são identificadas apenas pelo “oid”, e seu conteúdo é formatado de acordo como o padrão BER (*binary encoding rules*).

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. A figura 6.3 mostra a estrutura da PIB DiffServ representada em XML.

A implementação da estratégia acima ficará mais clara com os exemplos dos arquivos XML que serão apresentados adiante. A figura 6.3 permite ilustrar os elementos da classe

frwkFeedbackActionTable mapeados para a estrutura XML utilizando a estratégia ilustrada na figura 6.2.

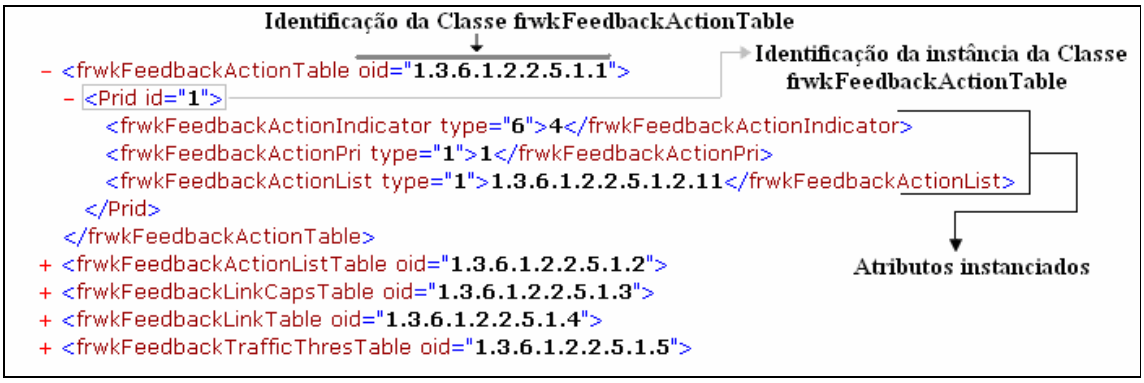

Figura 6.3: Mapeamento da classe *frwkFeedbackActionTable* para estrutura XML.

A estrutura da PIB implementada neste trabalho possui seis grupos de classes mapeadas para a estrutura XML: <*DeviceCapabilities*>, <*ClassifierGroup*>, <*LinkGroup*>, <*UsageGroup*>, <*SelectionGroup*> e <*UsageDiffSevGroup*>. É importante ressaltar que o grupo *Link* no momento de realizar a representação em XML foi nomeado como *frwkFeedabackGroupClasses* de acordo com as normas definidas na RFC 3571. Na figura 6.4 é apresentado o arquivo XML do PDP, onde estão representados os elementos da PIB correspondente a configuração dos PEPs. As classes (PRCs) comum a todos os tipos de clientes, correspondentes aos grupos <*DeviceCapabilities*> e <*ClassifierGroup*>, são definidas no módulo Framework PIB; <*frwkFeedabackGroupClasses* >, <*UsageGroup*>, <*SelectionGroup*> são definidas no módulo Framework PIB *Feedback*. As classes específicas para os clientes que implementam os mecanismos *Feedback* DiffServ, correspondem ao grupo <*UsageDiffServGroup*> e estão definidas na PIB *Feedback* DiffServ proposta.

Figura 6.4 Grupos e classes da PIB implementada.

A figura 6.5 ilustra a associação entre as classes da PIB *Feedback* DiffServ. No grupo <*LinkGroup*> (figura 6.5) está a PRC *frwkFeedbackActionTable*, que contém instâncias de ações que podem ser aplicadas as regras que se deseja monitorar. O atributo *frwkFeedbackActionList* aponta para a lista de instâncias para as quais a ação especificada deve ser aplicada. Na PRC *frwkFeedbackActionListTable* a instancia identificada pela "Prid id

= 11" contém o conjunto de instâncias (atributo *frwkFeedbackActionListRefId*) da PRC *frwkFeedbackLinkTable* que devem sofrer a ação especificada na PRC *frwkFeedbackActionTable*. Na PRC *frwkFeedbackLinkTable* as instancias de critérios de seleção (*frwkFeedbackRoleFilterSelTable*) e de uso (*UsageDiffServGrup*) são unidas para definir os objetos que devem ser monitorados.

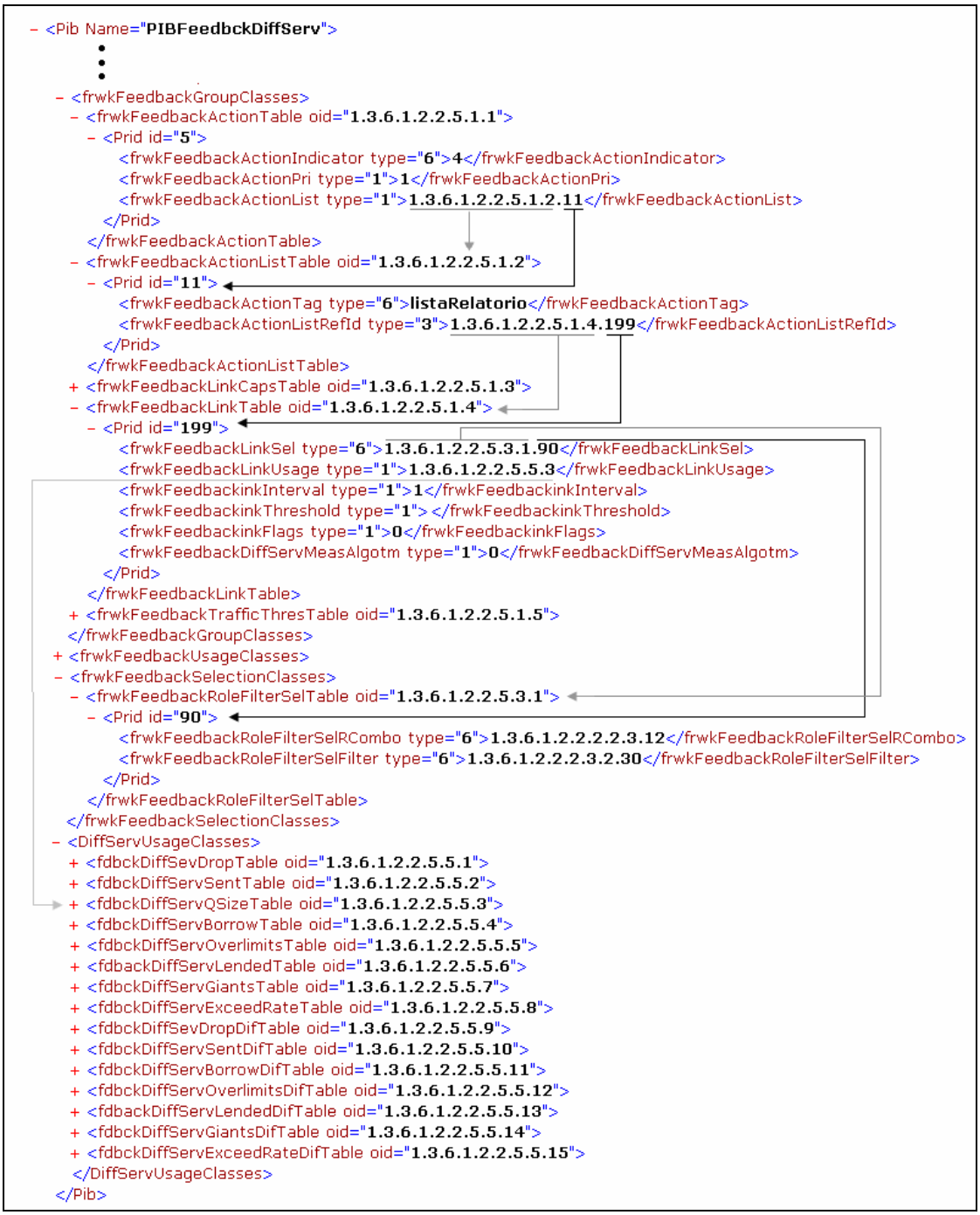

Figura 6.5: Associação entre as classes da PIB *Feedback* DiffServ.

6.3 Biblioteca de Conversão da PIB para comandos do Linux

Para converter as políticas de configuração independentes de dispositivos em instâncias (PRIs) da PIB *Feedback* DiffServ é necessário que sejam consideradas as capacidades específicas de cada dispositivo, uma vez que diferentes mecanismos de medição (descarte, tamanho da fila, pacotes enviados, etc.) podem ser monitorados na execução das ações da PIB *Feedback* DiffServ. Considerando esta premissa, verificou-se que a implementação de um algoritmo de conversão genérico, capaz de suportar qualquer combinação de medições, não é uma tarefa simples. Por isso foi adotado o conceito de "Biblioteca de conversão". Esta estratégia possibilita o aprimoramento da ferramenta de medição de tráfego através da inserção de novas bibliotecas no sistema.

Dentro deste trabalho, foi desenvolvida uma biblioteca de conversão específica da PIB *Feedback* DiffServ para o Controlador de Tráfego do Sistema Operacional Linux de acordo com as estatísticas fornecidas pelo Sistema Operacional (para o entendimento do Controlador de Tráfego do Linux se assume que o leitor tenha conhecimento do Anexo A).

O Sistema Operacional Linux possui quatro tipos de escalonadores, neste trabalho optou-se por trabalhar com o escalonador HTB (*Hierarchical Token Bucket*) [DEV, 05] uma vez que o mesmo é um aprimoramento do escalonador CBQ (*Class Based Queue*). O Linux possui diversas disciplinas de atendimento, para a biblioteca desenvolvida foi utiliza a disciplina GRED (*Generalized Random Early Detection*) [BAL, 04] por ser a que esta de acordo com as definições de Serviços Diferenciados.

Quando as estatísticas do controlador de tráfego do Linux são solicitadas para o escalonador HTB e para a disciplina de atendimento GRED os seguintes dados são fornecidos:

```
#tc -s -d qdisc show dev eth0

qdisc gred 8001:
 DP:1 (prio 2) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 21 Scell_log 9
 DP:2 (prio 3) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 20 Scell_log 9
 DP:3 (prio 4) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 19 Scell_log 9
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
```

```
qdisc htb 2: r2q 10 default 0 direct_packets_stat 0 ver 3.10
Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
```

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 PIB *Feedback* DiffServ, utiliza o mapeamento definido abaixo e executando algoritmo definido na tabela 5.4

- Medição de pacotes e bytes enviados: A biblioteca é capaz de implementar a disciplina GRED. Além do escalonador HTB que encaminha o tráfego a diferentes filas, para verificar a quantidade de pacotes e bytes que foram enviadas por cada classe de serviço;
- Medição de pacotes e bytes descartados: A biblioteca faz a verificação de quantos pacotes e bytes foram descartados nas classes de serviço HTB e nas filas GRED de cada classe.
- Medição do tamanho da fila: A biblioteca desenvolvida suporta a medição na fila do dispositivo do algoritmo GRED;
- Medição dos pacotes emprestados: esta biblioteca realiza a contagem dos pacotes que foram emprestados pelas classes superiores quando havia banda sobrando nas classes. Suporta a medição no escalonador HTB;
- Medição de *overlimits*: este algoritmo realizada a medição de quantas vezes a classe tentou enviar um pacote, mas este não pode ser enviado por restrições. Suporta o HTB.

6.4 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 [BEL, 05]. [BEL, 05] realizava a configuração DiffServ, portanto várias alterações foram feitas para o código passar a suportar o *feedback* das configurações que haviam sido previamente realizadas.

6.4.1 PDP

Durante sua ativação, o PDP carrega em memória suas configurações iniciais definidas no arquivo XML mostrado na figura 6.6.

```
- <Pdp>
  - <Config>
      <PdpId>PdpTeste</PdpId>
      <PdpPort>3288</PdpPort>
      <NextPdpAddr>127.0.0.1</NextPdpAddr>
      <NextPdpPort>3288</NextPdpPort>
      <Ka>10</Ka>
      <acct>0.5</acct>
      <SupportedPep>2 5 4002</SupportedPep>
      <AuthorizedPep>P1 Pep2 Pep3 Pep4</AuthorizedPep>
    </Config>
    <ActivePep />
  </Pdp>
```

Figura 6.6: Arquivo de configuração do PDP.

- *PdpId* – Corresponde à identificação do PDP;
- *PdpPort* – Define a porta TCP em que o PDP estará aguardando a requisições. De acordo com [DUR, 00] esta porta deve ser definida como 3288;
- *NextPdpAddr* – Define o endereço do próximo PDP, para casos em que o PDP receber requisições que não te suporte;
- *NextPdpPort* – Define a porta TCP em que o próximo PDP estará respondendo;
- *Ka* – Define o intervalo de tempo em que os PEPs devem enviar o seu status de funcionamento ao PDP;
- <*acct*> - Define o intervalo de tempo mínimo sob o qual o PEP pode enviar relatórios. Tem como finalidade evitar que o PEP congestione a rede com mensagens de relatório;
- *SupportedPep* – Apresenta uma lista com todos os tipos de clientes PEP suportados por este PDP. Clientes que não estejam nesta lista não têm permissão para se conectarem e são encaminhados ao próximo PDP;
- *AutorizedPep* – Apresenta uma lista com todas as identificações de clientes PEP que têm permissão para se conectar a este PDP. Da mesma forma que a entrada anterior, clientes que não pertençam a esta lista não têm sua conexão permitida.

Após carregar a 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. Os principais métodos implementados pelo PDP são apresentados na tabela 6.1.

Tabela 6.1: 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 e um arquivo de log é criado no PDP. A mensagem CAT (*client accept*) é retornada ao PEP para confirmar que a conexão foi aceita. 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, é adicionado também o objeto *acct* (C-Num=15 e C-Type) com o período de tempo mínimo que o PEP deve aguardar antes de enviar uma nova mensagem de relatório ao PDP [BEL, 05].
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 [BEL, 05].
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, levando em consideração as características do dispositivo, para instâncias (PRIs) que serão instaladas na PIB do cliente.
ReadRPT( )	Este método realiza a leitura do da mensagem RPT 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. Se a mensagem for de relatório quem contém os dados das medições realizadas no PEP o método *ReadRelatorio()* é chamado.
ReadRelatorio()	Ao receber a mensagem RPT o PDP verifica a qual instância da tabela *feedbackLink* a medição recebida faz referência. Após o PDP identificar a política que a medição pertence os dados são armazenados em um arquivo de *log* referente ao PEP que está enviando as informações ao PDP.

6.4.2 PEP

Os principais métodos implementados pelo PEP são descritos na tabela 6.2.

Tabela 6.2: 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 ou removidas na seção <Pib> no arquivo XML do PEP. Após instalar ou 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.
createRPTRelatorio( )	Este método constrói a mensagem RPT que inclui o objeto *Report-Type* (C-Num=12 e C-Type=3) com conteúdo=3 que indica relatório e o objeto *Named Decision Data* (C-Num=9 e C-Type=2) usado para encapsular os objetos COPS-PR que contêm as informações de leitura. O conteúdo do relatório é construído levando em consideração a PRID que o instanciou e as medições realizadas, que vão ser enviadas ao PDP.
createCC( )	Método usado para encerrar a conexão entre PEP e PDP. Objetos de erro podem ser incluídos na mensagem [BEL, 05].

Após a instalação das informações de configuração, o PEP torna-se operacional e o dispositivo controlado passa a monitorar as políticas recebidas e sem a necessidade de nova consulta ao PDP.

6.5 Arquivos de *Log*

Quando o PDP recebe uma solicitação de abertura de conexão enviada por um PEP, 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, um arquivo de *log* é criado para o PEP correspondente.

Ao receber a mensagem RPT (mensagens de relatório) o PDP realizar a leitura do campo *handle* no cabeçalho da mensagem de relatório recebida para verificar qual arquivo de *log* deve receber as novas informações. Então utiliza o algoritmo ilustrado na tabela 5.3 para ler o conteúdo da mensagem e gravar as informações no arquivo de log.

As informações são inseridas no arquivo de *log* na seguinte seqüência:

- Interface: informa a qual interface do roteador monitorado as informações foram obtidas;
- Política: informa qual classe está sendo monitorada no caso dos nós de núcleo; no caso dos nós de borda o monitoramento pode ser realizado baseado no endereço IP do cliente que está chegando ao domínio;
- Tipo de medida: informa quais parâmetros estatísticos foram monitorados. Como exemplo pode-se citar: tamanho da fila, descartes, etc;
- Pacotes e bytes contabilizados: as duas próximas informações informam a quantidade de pacotes e bytes foram contabilizadas referente a medida monitorada;
- Prid: identifica a instancia de medição responsável pela medida recebida do dispositivo de rede;
- RefId: faz referencia a instancia da tabela Link que instanciou as medições no dispositivo de rede.

A figura 6.7 ilustra um arquivo de *log*.

```
( interface eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 19871 prid: 251 RefId: 1.3.6.1.2.2.5.1.4.199 )
( interface eth0 politica AF12 tipodemedida Descartes PacketCount: 49 ByteCount: 10000 prid: 100 RefId: 1.3.6.1.2.2.5.1.4.21 )
( interface eth0 politica AF11 tipodemedida BytesFila PacketCount:  0 ByteCount: 7291 prid: 89 RefId: 1.3.6.1.2.2.5.1.4.199 )
( interface eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 1579 prid: 393 RefId: 1.3.6.1.2.2.5.1.4.199 )
( interface eth0 politica AF13 tipodemedida Descartes PacketCount: 90 ByteCount: 10384 prid: 890 RefId: 1.3.6.1.2.2.5.1.4.11 )
( interface eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 15060 prid: 98 RefId: 1.3.6.1.2.2.5.1.4.199 )
( interface eth0 politica AF12 tipodemedida Descartes PacketCount: 675 ByteCount: 19871 prid: 21 RefId: 1.3.6.1.2.2.5.1.4.21 )
( interface eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 10000 prid: 11 RefId: 1.3.6.1.2.2.5.1.4.199 )
( interface eth0 politica AF13 tipodemedida Descartes PacketCount: 767 ByteCount: 7291 prid: 89 RefId: 1.3.6.1.2.2.5.1.4.11 )
( interface eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 1579 prid: 55 RefId: 1.3.6.1.2.2.5.1.4.199 )
```

Figura 6.7: Arquivo de *log*.

6.6 Ferramenta de Estimativa

A ferramenta de estimativa tem por objetivo fornecer os dados das medições realizadas nos dispositivos de rede. Na interface gráfica o administrador do domínio pode escolher que PEPs quer obter as estimativas de tráfego. Quando é selecionado um PEP para obter os dados das estimativas realizadas pelo sistema de medição, a ferramenta de estimativa consulta o arquivo de *log* armazenado no PDP referente ao PEP que se deseja obter as informações. O administrador pode escolher através da interface gráfica quais parâmetros de cada PEP que deseja obter as informações de estimativa. No caso do administrador escolher vários PEPs para obter os dados, as estatísticas de cada PEP são somadas, e o sistema fornece o valor final.

6.7 Conclusão

Como conclusão da implementação verificou-se que desenvolver um algoritmo genérico para a conversão das políticas de decisão independentes de dispositivo em instancias da PIB *Feedback* DiffServ é uma tarefa complexa devido a grande quantidade de opções fornecidas pelos equipamentos. Portanto neste trabalho foi desenvolvida uma biblioteca específica para o Linux, que complementa o algoritmo de escalonamento HTB e a disciplina GRED. O desenvolvido é capaz de fornecer estatísticas por classe de serviço.

Outro aspecto importante que deve ser lembrado é que a estratégia de provisionamento foi utilizada, pois o PDP pode realizar novas configurações dos PEPs a hora que julgar necessário. A estratégia provisionamento também foi utilizada, pois possibilita que os PEPs possam enviar relatórios periódicos sem a necessidade de novas mensagem do PDP, não é necessário o PDP enviar uma mensagem ao PEP toda vez que receber um novo relatório.

Um aspecto muito importante neste trabalho é que a ferramenta de estimativa de tráfego desenvolvida permite informar, por exemplo, o atraso que um fluxo de dados pode sofrer ao passar por determinados pontos da rede DiffServ. Dados dois pontos da rede o sistema fornece uma estimativa, por exemplo, do atraso que determinado fluxo de dados pode sofre ao passar pelo domínio que esta sendo monitorado. O desempenho destas estratégias é avaliado no capítulo 7.

CAPÍTULO 7

Avaliação

7.1 Introdução

Os capítulos 5 e 6 apresentaram a arquitetura de medição. Este capítulo tem como objetivo apresentar três visões em relação à arquitetura apresentada. Primeiramente, será realizada uma análise do impacto das mensagens COPS/COPS-PR na rede. Em um segundo momento será apresentado como a arquitetura é utilizada para coletar e representar as informações de leituras realizadas nos dispositivos de rede. Está visão será apresentada através da exemplificação de um cenário com roteadores Linux para verificação de desempenho do modelo. Por fim é realizada uma análise dos resultados obtidos com as medições, para avaliar a arquitetura de medição e estimativa de tráfego proposta.

7.2 Descrição do Cenário de testes

Para possibilitar as medições de um domínio de Serviços Diferenciados em uma rede IP, foi montado um domínio DiffServ conforme ilustrado na figura 7.1. Este domínio é composto por 4 roteadores, sendo dois de boda e dois de núcleo, 1 servidores e 2 clientes. A conexão entre os roteadores foi realizada utilizando canais Ethernet 100Mbps. O roteador de borda1 conecta-se ao servidor, enquanto que o roteador de borda2 conecta a rede local de clientes. Todos os roteadores estão conectados ao PDP.

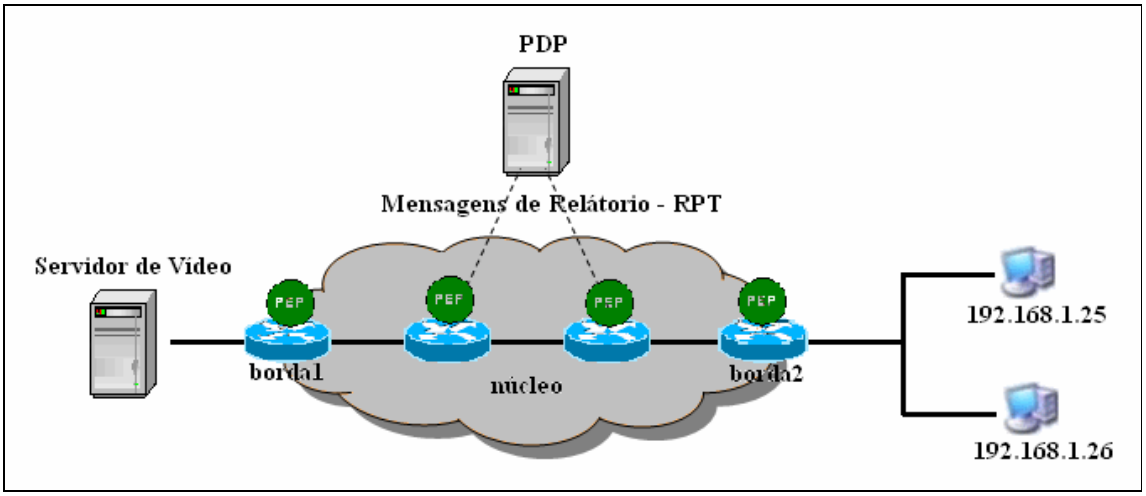

Figura 7.1: Cenário de Testes.

As máquinas utilizadas eram Pentium 1,5 GHz, 256MB, executando Sistema Operacional Linux distribuição Red Hat 9.0.

Nos roteadores de borda e de núcleo foram definidas políticas de DiffServ levando em consideração os seguintes itens:

- A identificação dos perfis de fluxos de tráfego existentes e sua forma de agregação;
- A associação dos tipos de serviços (EF, AF ou BE) aos perfis; e
- A alocação da largura de banda para cada perfil.

Os nós de borda são responsáveis por realizar a identificação e a classificação do tráfego e realizar a sua marcação. No cenário de testes foi utilizado o script ilustrado na figura 7.2 nos roteadores de borda.

```
tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 default_index 5
tc class change dev eth0 parent 1:0 classid 1:1 dsmark mask 0x3 value 0x28
tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip dst 192.168.1.25/24 classid 1:1
```

Figura 7.2: Script utilizado nos roteadores de borda.

Os nós de núcleo são responsáveis por alocar banda para cada perfil de tráfego. A figura 7.3 ilustra o script de configuração utilizado nos roteadores de núcleo.

```
tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 default_index 5 set_tc_index
tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2 pass_on
tc filter add dev eth0 parent 1:0 protocol ip prio 1 handle 0xa tcindex classid 1:1
tc qdisc add dev eth0 parent 1:0 handle 2:0 htb
tc class add dev eth0 parent 2:0 classid 2:1 htb rate 256kbit
tc qdisc add dev eth0 parent 2:1 gred setup DPs 3 default 2 grio
tc qdisc change dev eth0 parent 2:1 gred limit 60KB min 15KB max 45KB \
burst 20 avpkt 1000 bandwidth 10Mbit DP 1 probability 0.02 prio 2
tc qdisc change dev eth0 parent 2:1 gred limit 60KB min 15KB max 45KB \ burst 20 avpkt 1000
bandwidth 10Mbit DP 2 probability 0.04 prio 3
tc qdisc change dev eth0 parent 2:1 gred limit 60KB min 15KB max 45KB \
burst 20 avpkt 1000 bandwidth 10Mbit DP 3 probability 0.06 prio 4
tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 1 tcindex classid 2:1
```

Figura 7.3: Script utilizado nos roteadores de núcleo.

7.3 Descrição dos Experimentos

De modo a avaliar a arquitetura de medição proposta um conjunto de experimentos foi realizado. Estes experimentos foram divididos em duas fases.

7.3.1 Experimentos, Fase I – Intervalo Entre Mensagens de Relatório

Nesta fase, o objetivo principal foi avaliar qual seria um bom intervalo de tempo entre uma mensagem de relatório e outra ao PDP de forma a não congestionar a rede com mensagens de relatório e demonstrar a real utilização do dispositivo de rede. Para realizar esta análise, o cenário descrito na seção 7.2 foi utilizado.

Para descobrir o tamanho de cada pacote de mensagens de relatório a ferramenta Ethereal foi utilizada. O tamanho de cada pacote que contém a mensagem de relatório do PEP é de 158bytes. Descoberto o tamanho do pacote que continha a mensagem de relatório foi necessário encontrar um intervalo de tempo ideal de forma a representar a real utilização do dispositivo de rede. Após a avaliação de vários tempos de teste entre uma mensagem de relatório e outra se avaliou que um intervalo de tempo de 200ms é ideal para demonstrar a real utilização do dispositivo de rede. A figura 7.4 ilustra o envio de mensagens de relatório do PEP ao PDP a cada intervalo de 200ms.

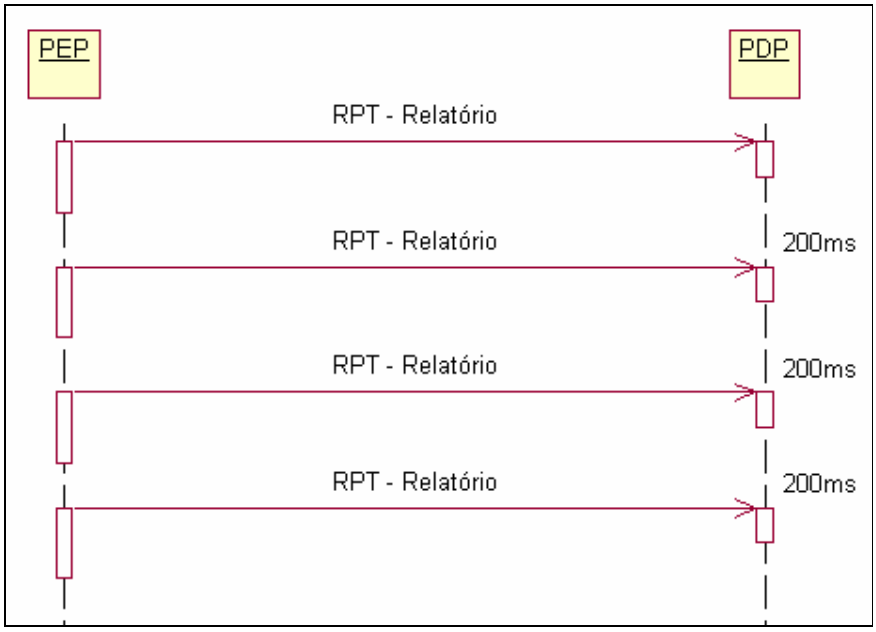

Figura 7.4: Mensagens de relatório a cada 200ms.

O consumo de banda causado pelas mensagens de relatório mostrou-se bastante baixo. Uma vez que um pacote de 158bytes é enviado a cada 200ms.

Descoberto o tempo ideal o campo *acct* que tem seu valor setado com 200ms toda vez que um novo PEP realizar o pedido de conexão ao PDP e este for aceito, uma mensagem CAT é enviada ao PEP informado o intervalo mínimo que deve aguardar entre o envio de uma mensagem de relatório e a próxima.

7.3.2 Experimentos, Fase II – Avaliação da Arquitetura de Medição.

Na fase II o objetivo foi analisar o desempenho da arquitetura de medição proposta. Para isto foi suposto que um contrato de serviço (SLA) foi estabelecido com um cliente do domínio de Serviços Diferenciados construído.

7.3.2.1 Definição de Contrato

Neste exemplo, o administrador do domínio de Serviços Diferenciados estabelece contratos (SLAs) com seus clientes. Onde os detalhes de utilização são acertados. Um contrato de SLA deve definir no seu escopo alguns itens essenciais, tais como [PIN, 04]:

- Serviços cobertos
- O horário ou a freqüência;
- Tempo de Resposta;

- Disponibilidade e confiabilidade;
- Vazão;
- Perda de pacotes.

A figura 7.5 ilustra um contrato estabelecido com o cliente A.

Serviços cobertos: transmissão de vídeo
Horário: 17:00Hs as 23:00Hs
Tempo de Resposta: <= 1,5 segundos
Disponibilidade e confiabilidade: 95%
Vazão: 256kbits

Figura 7.5: Contrato estabelecido com o cliente A.

7.3.2.2 Arquivo de Configuração enviado aos PEPs

Baseado nos contratos estabelecidos com seus clientes o administrador do domínio realiza as medições nos nós de seu domínio para verificar se as condições estabelecidas nos contratos estão sendo cumpridas e para verificar a carga de serviço de cada dispositivo de rede.

Neste exemplo é suposto que o administrador da rede estabeleceu um contrato com um cliente A com o IP 192.168.1.25. Onde o cliente A deve receber tratamento ouro para transmissão de vídeo das 17:00Hs as 23:00Hs, e seus pacotes de dados não devem ter um tempo de resposta maior que 1500ms. A classe que recebe tratamento ouro no domínio deste administrador é a classe de serviço AF11. O cliente A com é classificado pelos roteadores de borda e tem seus pacotes marcados com o valor 0x28 (classe AF11).

Para realizar a verificação da classe AF11 o administrador do domínio instancia as medições nos respectivos nós de seu domínio por onde os dados do cliente A vão trafegar. O arquivo de configuração das medições é apresentado na figura 7.6.

Após o administrador realizar a configuração do arquivo o mesmo é enviado pela mensagem DEC através dos protocolos COPS/COPS-PR para os PEPs.

Quando os dispositivos de rede recebem a configuração enviada pelo PDP, os mesmos realizam a leitura da PIB recebida para verificar quais ações devem executar. O arquivo recebido pelos PEPs é o mesmo apresentado na Figura 7.6.

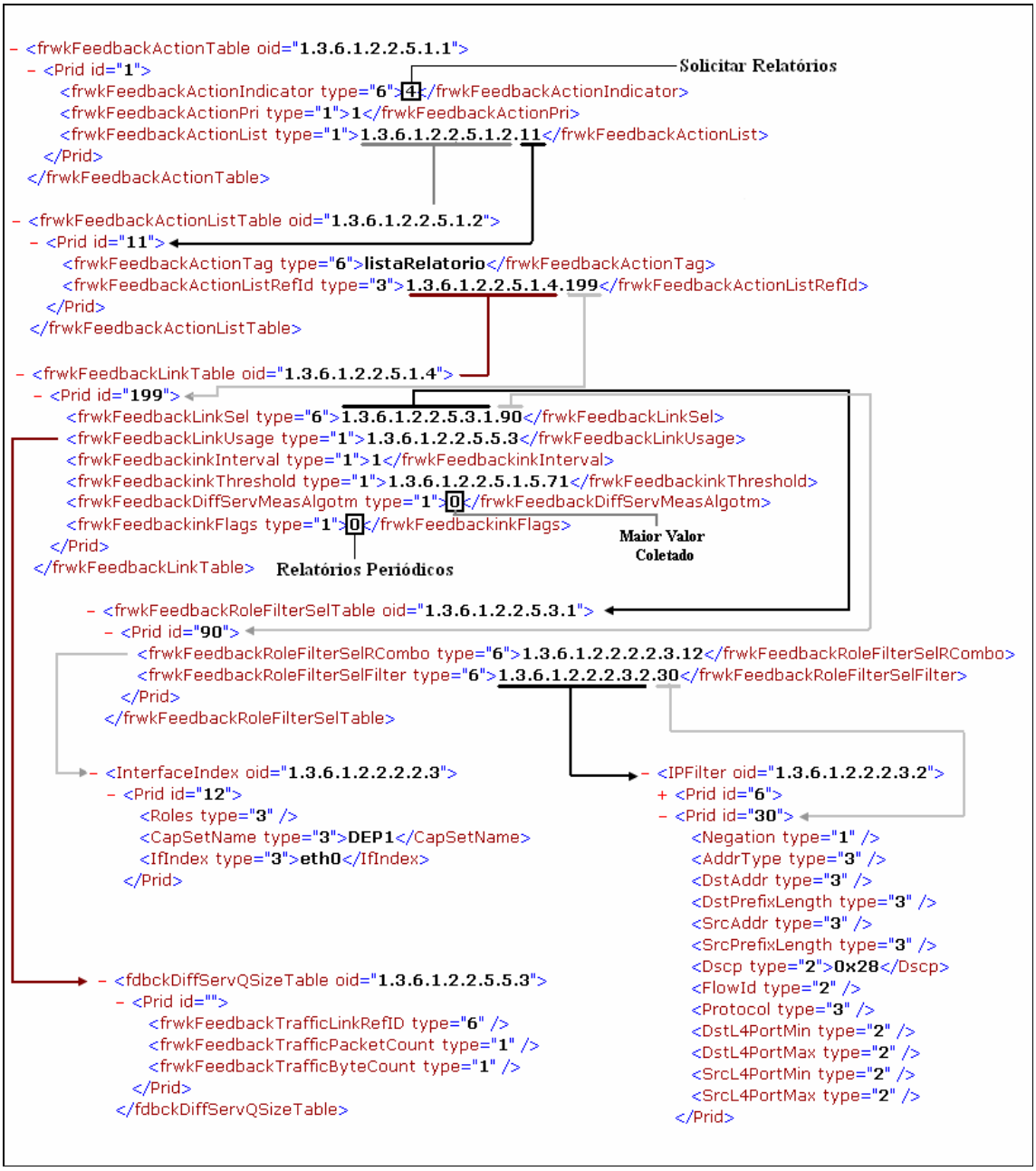

Figura 7.6: Arquivo de configuração.

Neste exemplo a tabela *frwkFeedbackActionTable* informa que relatórios devem ser fornecidos ao PDP para o grupo de instancias da referenciados pela tabela *frwkFeedbackActionLinkTable*. A tabela *frwkFeedbackActionLinkTable* contém a identificação das instâncias da tabela *frwkFeedbackLinkTable* que devem ser monitoradas nos dispositivos de rede. A instancia da tabela *frwkFeedbackLinkTable* contém no atributo *frwkFeedbackLinkSel* a política que deve ser monitorada, neste caso a classe de serviço 0x28 (AF11) do papel DEP1. O atributo *frwkFeedbackLinkUsage* é um ponteiro para a classe de

uso que deve ser utilizada para monitorar a política especificada. Neste arquivo de configuração a classe de uso que deve ser utilizada é a classe *fdbackDiffServQsizeTable*. O intervalo de tempo em que as leituras devem ser realizadas está definido no atributo *frwkFeedbackLinkInterval* da tabela *frwkFeedbackLinkTable*, vezes o valor definido em *acct*. Neste exemplo o intervalo de medição é o mesmo valor do *acct*, pois 1 vezes 200ms = 200ms.

7.3.2.3 Tradução da PIB para o dispositivo de rede

Com os dados fornecidos pelo PDP o algoritmo descrito na tabela 5.4 é utilizado para realizar a conversão da PIB *Feedback* DiffServ para o dispositivo de rede para que as medições possam ser iniciadas no sistema operacional Linux, a seguinte linha de comando é montada de acordo com os dados que foram recebidos do PDP.

```
tc –s –d qdisc show dev eth0
```

Está linha de comando é executada a cada intervalo de tempo. Toda vez que a linha é executada o seguinte conteúdo é fornecido pelo Sistema Operacional Linux:

```
qdisc gred 8001:
 DP:1 (prio 2) Average Queue 17741b Measured Queue 19871b
         Packet drops: 40 (forced 0 early 0)
         Packet totals: 89073 (bytes 9875273)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 21 Scell_log 9
 DP:2 (prio 3) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 20 Scell_log 9
 DP:3 (prio 4) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 19 Scell_log 9
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc htb 2: r2q 10 default 0 direct_packets_stat 0 ver 3.10
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc dsmark 1: indices 0x0040 default_index 0x0005 set_tc_index
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
```

Neste ponto dos testes, verificou-se que quando se realiza a medição das filas há uma particularidade em relação às outras políticas. As outras políticas fornecem o acumulado, enquanto a fila fornece a todo instante o seu tamanho atual. Foi necessário então realizar testes para ver qual seria um bom intervalo de medição na fila de forma a representar a real

utilização da fila do dispositivo de rede. Os testes demonstraram que um intervalo de medição ideal entre uma leitura e a próxima na fila é de 20ms. Após realizar as medições a cada 20ms no intervalo de tempo monitorado havia outro problema. Qual dos valores medidos deveria ser enviado ao PDP de forma a representar a real utilização da fila do dispositivo de rede.

Havia basicamente três opções, enviar o menor valor, a média, ou o maior valor. Quando se optou por trabalhar com o menor valor e/ou a média os resultados obtidos foram insatisfatórios, não era possível representar a real utilização da fila do dispositivo de rede. No momento em que se passou a trabalhar com o maior valor coletado da fila o sistema conseguia realizar uma estimativa da taxa de ocupação da fila corretamente. O administrador do domínio quando desejar pode escolher qual valor deve ser encaminhado para si nos relatórios através do atributo *fdbckDiffServMeasAlgotm* da tabela *frwkFeedbackLinkTable*.

7.3.2.4 Envio das mensagens de Relatório ao PDP

Com as medidas já realizadas o PEP armazena em memória os dados que devem se enviados ao PDP e fica aguardando o próximo intervalo de tempo para enviar as mensagens de relatório. A cada intervalo de acordo com o que foi especificado pelo PDP no campo *acct* o PEP faz uma verificação em memória para observar se há algum relatório para ser encaminhado ao PDP, caso houver o PEP monta a PIB de relatório e envia ao PDP. A figura 7.7 ilustra o relatório enviado ao PDP.

```
- <Pib Name="Relatorio">
  - <DiffServUsageClasses>
    - <fdbckDiffServQSizeTable oid="1.3.6.1.2.2.5.5.3">
      - <Prid id="258">
          <frwkFeedbackTrafficLinkRefID type="6">1.3.6.1.2.2.5.1.4.199</frwkFeedbackTrafficLinkRefID>
          <frwkFeedbackTrafficPacketCount type="1">0</frwkFeedbackTrafficPacketCount>
          <frwkFeedbackTrafficByteCount type="1">19871</frwkFeedbackTrafficByteCount>
        </Prid>
      </fdbckDiffServQSizeTable>
    </DiffServUsageClasses>
  </Pib>
```

Figura 7.7: Conteúdo das mensagens de relatório enviadas ao PDP.

7.3.2.5 União das informações no PDP

Após o PDP receber as mensagens de relatório o mesmo realiza a leitura do conteúdo da mensagem recebida utilizando o algoritmo descrito na tabela 5.3 e monta o arquivo de *log*. A figura 7.8 ilustra o arquivo de *log* criado durante os testes realizados.

```
( face eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 19871 prid: 251 RefId: : 1.3.6.1.2.2.5.1.4.199 )
( face eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 10000 prid: 100 RefId: 1.3.6.1.2.2.5.1.4.199 )
( face eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 7291 prid: 89 RefId: 1.3.6.1.2.2.5.1.4.199 )
( face eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 1579 prid: 393 RefId: 1.3.6.1.2.2.5.1.4.199 )
( face eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 10384 prid: 890 RefId: 1.3.6.1.2.2.5.1.4.199 )
( face eth0 politica AF11 tipodemedida BytesFila PacketCount: 0 ByteCount: 15060 prid: 948 RefId: 1.3.6.1.2.2.5.1.4.199 )
```

Figura 7.8: Arquivo de log formado no PDP.

7.3.2.6 Ferramenta de Estimativa de Tráfego

Com os arquivos de *log* é possível fazer a estimativa de tráfego do domínio de DiffServ. Neste momento a ferramenta de estimativa de tráfego desenvolvida é utilizada para fazer este trabalho.

Para prover os relatórios da ferramenta de estimava verificou-se que um bom intervalo entre uma coleta de dados e outra nos arquivos de *log* é de 10s. O maior valor é obtido do arquivo a cada 10s de cada arquivo de *log*.

7.4 Avaliação dos testes

Para comparar os resultados obtidos pelo sistema de medição foi executada no servidor uma rotina desenvolvida em C, baseada em um gerador de tráfego, para leitura de um arquivo de *log* contendo informações sobre as características de tráfego de vídeo. O arquivo utilizado foi disponibilizado pela Universidade Tecnológica de Berlin (http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html e continha informações sobre os quadros (tamanho e intervalo entre o envio de um quadro e o próximo). O Arquivo representava a reprodução do filme Star Trek em formato H.263 (que simula uma transmissão ao vivo de vídeo) a uma taxa media de 256 kbit/s. A figura 7.9 ilustra o formato do arquivo de *log* do filme Star Trek utilizado nos testes.

```
#Time [ms]      Frametype       Length [byte]
#_____________________________________________________
0               I                       2241
40              P                       1877
120             P                       3199
200             P                       2817
280             P                       3317
400             P                       3252
520             P                       3388
600             P                       3207
```

Figura 7.9: Representação do arquivo do filme Star Trek.

O tempo de saída de cada pacote era colocado dentro do pacote antes do mesmo sair do servidor de vídeo e realizada a conta do atraso quando o pacote de dados chegava ao cliente. Para utilizar esta estratégia as máquinas do cenário apresentado na seção 7.2 tiveram de ser sincronizadas, uma vez que para o cálculo do atraso a fim em um único sentido exige-se que os relógios das estações de medição, geradora e receptora do tráfego, estejam sincronizados dentro de uma determinada precisão.

Neste caso, as duas estações tiveram seus relógios sincronizados através do NTP (*Network Time Protocol*) [NTP, 05] com uma precisão superior a 10ms que é suficiente para esta aplicação. Esta precisão no ajuste dos relógios das estações de medição foi obtida através da sincronização de ambas as estações a cada minuto, com base em um único servidor de NTP posicionado em um segmento de rede Ethernet.

O gráfico 7.10 ilustra a caracterização do tráfego de vídeo. As linhas mais claras representam o atraso fim a fim e a linha escura o atraso estimado pelo sistema.

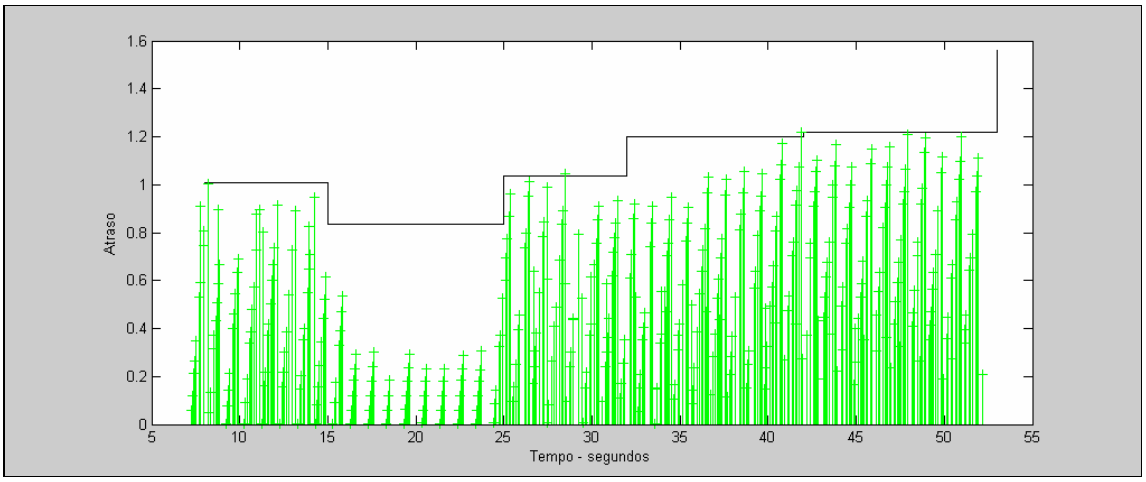

Figura 7.10: Caracterização do tráfego de vídeo.

Neste gráfico e possível visualizar o atraso fim a fim de cada pacote na transmissão do vídeo Star Trek. O tráfego de vídeo se caracteriza por pequenas rajadas, assim o primeiro pacote que chega a fila do roteador encontra a fila vazia e quase não sofre atraso, os próximos pacotes que chegam na fila do roteador já encontram a fila com pacotes e sofrem um atraso maior.

O próximo gráfico (figura 7.11) ilustra o maior atraso fim a fim e o maior atraso estimado pelo sistema a cada 10s durante a transmissão do vídeo Star Trek. A linha

pontilhada (mais clara) representa o atraso real fim a fim enquanto a linha escura representa o atraso estimado pelo sistema desenvolvido.

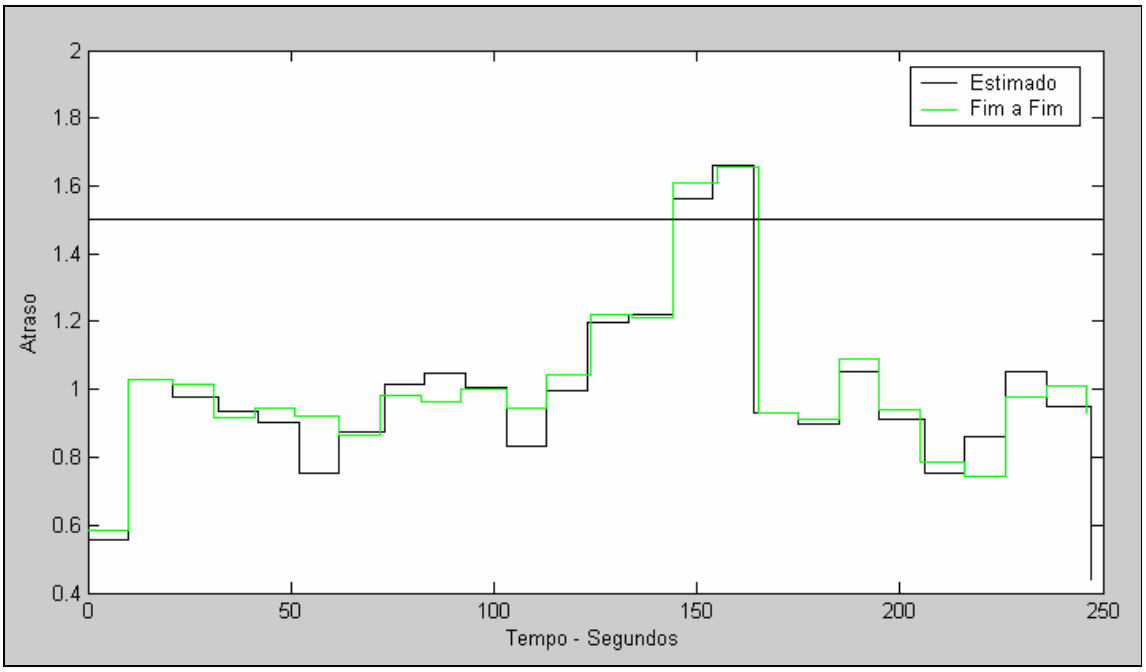

Figura 7.11: Maior atraso a cada 10s.

A tabela 7.1 ilustra os resultados obtidos em relação ao atraso comparando o atraso fim a fim e o estimado pelo sistema de medição.

Tabela 7.1: Dados obtidos utilizando os maiores atrasos.

Atraso	Fim a Fim	Estimado Pelo Sistema
Maior	1.655230ms	1.659188ms
Média	1.009095ms	0.998900ms
Menor	0.583006ms	0.554688ms
Desvio Padrão	0.227911ms	0.235081ms

É possível observar que o maior atraso obtido fim a fim e inferior ao atraso estimado pelo sistema de medição. Porém como ilustrado no gráfico 7.11 em vários momentos o atraso fim a fim foi maior que o estimado pelo sistema de medição desenvolvido.

Como se desejava obter a linha de tráfego fim a fim abaixo do estimado pelo sistema então se procurou achar para que porcentagem do tráfego fim a fim o sistema de estimativa

conseguia trabalhar bem. Neste primeiro gráfico foi tirado os 3% piores pacotes a cada intervalo de 10s do atraso fim a fim, ou seja, da linha mais clara. A figura 7.12 ilustra o gráfico obtido.

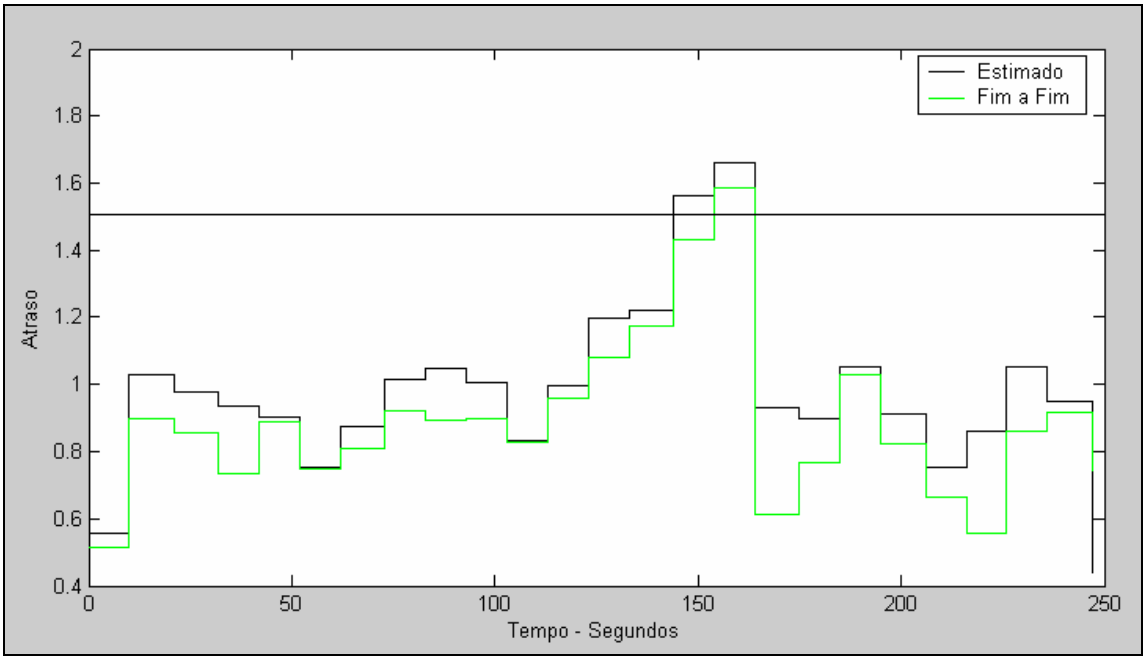

Figura 7.12: Atraso fim a fim sem os 3% piores atrasos.

A tabela 7.2 ilustra os resultados obtidos em relação ao atraso comparando o atraso fim a fim e o estimado pelo sistema de medição, excluindo os 3% piores pacotes a cada intervalo de 10s do atraso fim a fim.

Tabela 7.2: Dados obtidos utilizando os maiores atrasos, excluindo os 3% piores pacotes do atraso fim a fim.

Atraso	Fim a Fim	Estimado Pelo Sistema
Maior	1.588087ms	1.659188ms
Média	0.888590ms	0.998900ms
Menor	0.513280ms	0.554688ms
Desvio Padrão	0.241638ms	0.235081ms

É possível observar que retirado os 3% piores pacotes a cada 10s da medição fim a fim o resultado obtido é satisfatório, em momento algum o atraso estimado pelo sistema de medição este abaixo do atraso fim a fim.

7.5 Conclusão

Neste capítulo a montagem de um ambiente para realizar experimentos envolveu a implementação de Serviços Diferenciados no Sistema Operacional Linux para representar as ações de Serviços Diferenciados. Também foram utilizadas máquinas com Sistema Operacional Linux para representar as entidades utilizadas na arquitetura de medição e estimativa proposta. Uma vez definidas as ações de DiffServ foram realizados experimentos para avaliar a arquitetura proposta em termos de consumo de banda pelas mensagens de relatório e a realização de testes para avaliar a arquitetura de medição e estimativa proposta.

Os testes para avaliar o consumo de banda causado pela mensagem COPS-PR que utiliza a PIB *Feedback* DiffServ para representar as informações de leitura realizadas nos dispositivos de rede mostraram que a banda consumida pelo sistema de medição proposto é extremamente baixo e demonstrando assim que arquitetura PBNM proposta pelo IETF pode ser utilizada para realizar o monitoramento dos nós de um domínio de Serviços Diferenciados.

Em um segundo momento os teste também demonstraram que a arquitetura proposta possibilita a realização da avaliação dos SLAs estabelecidos com os clientes do domínio de Serviços Diferenciados. Os dados de medição recebidos pelo PDP dos PEPs possibilitaram realizar a avaliação dos SLAs. Nos testes realizados o sistema de medição e estimativa demonstrou que consegue ilustrar a desempenho dos serviços disponibilizados para 97% do tráfego real do usuário. Ou seja, para 97% do tráfego do usuário o sistema de medição consegue proferir o atraso sofrido pelo fluxo de dados.

Desta forma a arquitetura de medição proposta demonstra ser uma ótima ferramenta para realizar a estimativa tráfego de um domínio de Serviços Diferenciados.

Capítulo 8

Conclusões e Trabalhos Futuros

Este trabalho apresentou uma arquitetura de medição e estimativa de tráfego para ambientes DiffServ seguindo as recentes recomendações do IETF. A utilização do modelo PBNM permitiu que toda a monitoração da rede pudesse ser executada a partir de um único ponto único, possibilitando assim, a simplificação e racionalização das medições em um domínio de Serviços Diferenciados. Dentro da arquitetura, foi proposta uma PIB de avaliação para representação dos dados obtidos dos dispositivos de rede o que garante a possibilidade da utilização da arquitetura de medição em qualquer segmento da rede uma vez que segue as recomendações do IETF. A proposta da PIB é uma especialização do Framework PIB *Feedback* definida pelo IETF através da RFC 3571 [RAW, 03]. A definição desta PIB mostrou-se bastante difícil devido a diversidade de parâmetros de medição fornecidos pelos dispositivos de rede. Cada fabricante fornece diferentes dados de medição em seus equipamentos.

O desenvolvimento dos algoritmos para realizar as medições também trouxe novas contribuições. Algumas descobertas interessantes foram realizadas no sentido da freqüência de amostragens da fila dos dispositivos de rede. Percebeu-se que a fila dos mesmos varia de tamanho a todo o instante, o que requer uma freqüência de medição bastante elevada. Nesta fase foi necessário avaliar o intervalo ideal entre uma coleta e a próxima na fila do dispositivo de rede de forma a representar a real utilização da fila. Ainda, depois de realizar a coleta na fila dos dispositivos, foi verificado que era necessário realizar um tratamento com os dados coletados, para decidir qual valor deveria ser enviado ao PDP. Neste trabalho foram realizados vários testes para descobrir qual valor coletado deveria ser enviado ao PDP. Os testes demonstram que o maior valor coletado possibilita uma maior visualização da ocupação da fila do dispositivo de rede. Para monitoramentos que se desejar obter outro valor (menor, média) foi proposto um novo atributo na tabela *frwkFeedbackLinkTable* da PIB *Feedback* DiffServ. Assim o administrador do domínio pode escolher o valor medido que deseja receber

nas mensagens de relatório, no momento em que realizar a configuração dos dispositivos de rede.

O sistema de medição e estimativa proposto realiza o monitoramento dos nós de borda e de núcleo. Nas bordas é possível realizar o monitoramento de fluxos individuais, enquanto no núcleo é possível observar o comportamento do tráfego agregado. Nos nós de núcleo é possível realizar o monitoramento de qualquer um dos parâmetros estatísticos cobertos pela PIB *Feedabck* DiffServ. A arquitetura de medição mostrou-se bastante eficiente na estimava do atraso na comunicação entre os dois nós do domínio de Serviços Diferenciados. Para realizar a estimativa de atraso foram realizados testes com fluxos de vídeo que apresentam um comportamento bastante variável durante a transmissão. Demonstrando que a arquitetura proposta pode ser utilizada para a avaliação e monitoramento do tráfego, uma vez que apresentou bons resultados com tráfego de vídeo que é mal comportado. Desta forma é possível que o administrado do domínio possa comprovar a desempenho do serviço prestado ao usuário.

Conclui-se também, através deste trabalho, que o uso dos protocolos COPS e COPS-PR para o envido de informações estatísticas ao PDP apresentam potencialidades interessantes para serem escolhidos em soluções de monitoramento dos dispositivos de rede devido à baixa taxa de consumo de banda apresenta nos testes realizados e pela completitude do protocolo.

A arquitetura proposta apresenta como problema a exigência da sincronização dos PEPs para que o PDP não venha a realizar estimativas incorretas de um determinado período de tempo. Problema que pode ser resolvido com protocolos de sincronização de relógios.

Em trabalhos futuros, propõe-se que seja desenvolvida uma estratégia para a descoberta da rota do fluxo de dados, que o mesmo possa utilizar ao passar pelo domínio, uma vez que no presente trabalho assumiu-se que a rota do fluxo de dados era conhecida. Os resultados apresentados neste trabalho se concentraram em poucas variáveis uma vez que não foi possível elaborar um estudo para estabelecer uma correlação entre as mesmas. Os arquivos de *log* obtidos através dos testes contêm grande volume de informações que podem ser utilizados para a elaboração de uma análise estatística. Para a continuação deste trabalho sugere-se que sejam realizadas estudos com estes dados, para obter um perfil de tráfego e desta forma realizar a previsão de rompimento de SLAs antes que os mesmos venham a ocorrer. Em um segundo momento propõe-se o desenvolvimento de um sistema de medição inteligente. Assim, que o PDP perceber que um SLA está sendo violado, o mesmo pode realizar uma nova alocação de banda para o tráfego de dados que estiver com falta de recursos.

Referências Bibliográficas

[BAL, 04] BALLIACHE, Leonardo. "**Practical QOS**". http://www.opalsoft.net/qos/. 2004.

[BEL, 05] BELLER, Andre. "**Uma Arquitetura para Gerenciamento de QoS Baseado em Políticas**". Pontifícia Universidade Católica do Paraná. 2005.

[BER, 05] BERT Hubert, NETHERLABS, BV; GRAF, Thomas. "**Linux Advanced Routing & Traffic Control HOWTO**". http://lartc.org/howto/. 2005.

[BLA, 01] BLACK, D; FLOYD, S; RAMAKRISHNAN, K. "**The Addition of Explicit Congestion Notification (ECN) to IP**". RFC 3168, IETF. 2001.

[BLA, 98] BLAKE, S; et al. "**An Architecture for Differentiated Service**". RFC 2475, IETF. 1998.

[BRA, 94] BRADEN, R; CLARK, D; SHENKER, S. "**Integrated Services in the Internet Architecture: an Overview**". RFC 1633, IETF, 1994.

[BRO, 04] BROWN, Martin A. "**Traffic Control HOWTO**". http://linux-ip.net/articles/Traffic-Control-HOWTO/. 2004.

[CAL, 02] CALDEIRA, Felipe Manuel. "**Gestão Por Políticas Aplicação a Sistemas de Firewall**". Universidade de Coimbra. Coimbra. 2002.

[CHA, 03] CHAN, K; et.al. "**Differentiated Services Quality of Service Policy Information Base**". RFC 3317, IETF. 2003.

[CHA, 01] CHAN, K; et.al. "**COPS Usage for Policy Provisioning (COPS-PR)**". RFC 3084, IETF. 2001.

[CHO, 00] CHOI, Byung Kyu; BETTATI, Ricardo. "**Efficient Resource Management for Hard Real-Time Communication over Differentiated Services**

Architectures". Dept. of Comput. Sci., Texas A&M Univ., College Station, TX, USA. 2000.

[DEV, 05] DEVIK, Martin Devera Aka. "**HTB Linux queuing discipline manual - user guide**". http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm. 2005.

[DOM, 00] DOMINGUEZ, Kelly S. P; JUNIOR, Nilton A; "**Modelos de Qualidade de Serviço – Aplicações em IP**". 2000.

[DUR, 00] DURHAM, D; et.al. "**The COPS (Common Open Policy Service) Protocol**". RFC 2748, IETF. 2000.

[FLO, 05] FLOYD, S; JACOBSON, V. "**Random Early Detection**". http://www.icir.org/floyd/papers/red/. 2005.

[FLO, 01] FLOYD, S; BLACK, D; RAMAKRISHNAN, K. "**The Addition of Explicit Congestion Notification (ECN) to IP**". RFC 3168, IETF. 2001.

[GUE, 00] GUERIN, R; PENDARAKIS, D; YAVATKAR, R. "**A Framework for Policy-based Admission Control**". RFC 2753, IETF. 2000.

[HEI, 99] HEINANEN, J; et al. "**Assured Forwarding PHB Group**". RFC 2597, IETF. 1999.

[HUB, 04] HUBERT, Bert; NETHERLABS, Bv. "**Linux Advanced Routing & Traffic Control HOWTO**". http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/. 2004.

[HWA, 01] HWANG, Junseok; OKUMUS, Ibrahim T; MANTAR, Haci A. "**Edge-To-Edge Resource Provisioning and Admission Control in DiffServ Networks**". Syracuse University. 2001.

[JAC, 99] JACOBSON, V; NICHOLS, K; PODURI, K. "**An Expedited Forwarding PHB**". RFC 2598, IETF. 1999.

[KAM, 00] KAMIENSKI, Carlos Alberto. **"Qualidade de Serviço na Internet**". Disponível em www.cin.ufpe.br/~cak/publications/jai2000.pdf. 2000.

[KER, 04] KERALAPURA, Ram; CHUAH, Chen-Nee. "**Traffic Matrix Based Admission Control**". 2004.

[KUL, 03] KULKARNI, A; et al. "**Framework for Policy Usage Feedback for Common Open Policy Service with Policy Provisioning (COPS-PR)**". RFC 3483, IETF. 2003.

[LEM, 00] LEMPONEN, Jussi. "**Implementation of Differentiated Services Policy Information Base on Linux**". 2000.

[MIN, 03] MING, Li; Doan B.Hoang; Andrew J. Simmonds "**Fair Intelligent Admission Control over DiffServ Network**". International Conference on Information and Communication Technologies. Thailand. 2003.

[MUR, 01] MURHAMMER, M; ATAKAN, O; BRETZ, S. "**TCP/IP Tutorial and Technical Overview**". Disponível em http://publib-b.boulder.ibm.com/redbooks.nsf/RedbookAbstracts. 2001.

[NIC, 98] NICHOLS, K; et al. "**Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers**". RFC 2474, IETF. 1998.

[NTP, 05] NTP.ORG. **"NTP: The Network Time Protocol"**. http://www.ntp.org/. 2005.

[PIN, 05] PINHEIRO, José Mauricio. "**Gerenciamento de Níveis de Serviços de Serviços**". www.projetoderedes.com.br. 2005.

[SAH, 03] SAHITA, R; et.al. "**Framework Policy Information Base**". RFC 3318, IETF. 2003.

[SEM, 01] SEMERIA, Chuck. "**Supporting Differentiated Service Classes: Queue Scheduling Disciplines**". Juniper Networks. 2001.

[RAD, 04] RADHAKRISHNANR, Saravanan. **"Linux - Advanced Networking Overview**". http://qos.ittc.ku.edu/howto/. 2004.

[RAW, 03] RAWLINS, D; et al. "**Framework Policy Information Base for Usage Feedback**". RFC 3571, IETF. 2003.

[TEC, 99] Technologies, Stardust Technologies. "**Introduction to QoS Policies**". http://www.qosforum.com/white-papers/qospol_v11.pdf. 1999.

[ZHA, 01] ZHANG, G; MOUFTAH, H.T. "**End-to-End QoS Guarantees Over DiffServ Networks**". Sixth IEEE Symposium on Computers and Communications. 2001.

Anexo A

Serviços Diferenciados em Linux

A.1 Introdução

A prototipação deste trabalho foi realizada em Linux. O Kernel do Linux possibilita configurar nós de borda e nós de núcleo, assim como fazer a marcação dos pacotes IP que estão chegando ao domínio de DiffServ. Os componentes do controle de tráfego no Linux são apresentados na seção A.2. Na seção A.3.1 é demonstrado um exemplo da configuração de um nó de borda e explicado cada um dos seus componentes. A seção A.3.2 mostra a configuração de um nó de núcleo para a configuração realizada no tópico A.3.1, e explica as estratégias adotadas para configurar um nó de núcleo de DiffServ. No tópico A.4, é exposto às estatísticas obtidas do controle de tráfego no Linux.

A.2 Controle de tráfego no Linux

O princípio básico na implementação de DiffServ em Linux é mostrado na figura A.1.

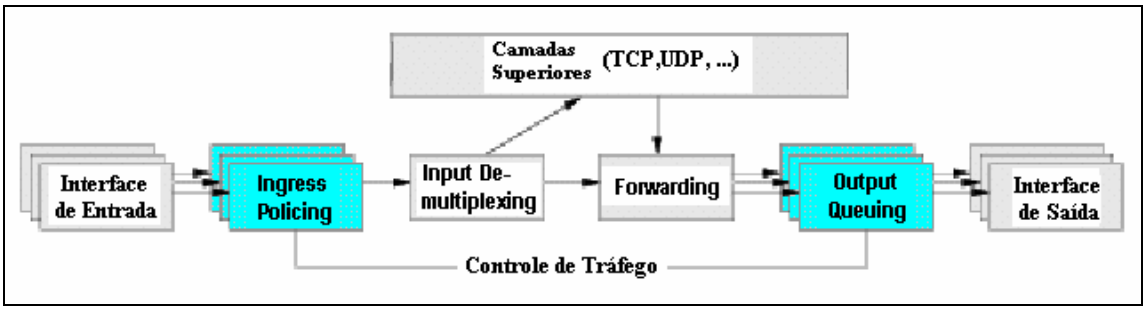

Figura A.1: Processando dados recebidos da rede.

Esta figura mostra como o Kernel processa pacotes de dados recebidos da rede, e como gera novos dados a serem enviados na rede. Pacotes chegam em uma interface de entrada é o primeiro ponto de controle, onde pacotes indesejáveis podem ser descartados. No

próximo passo os pacotes são enviados ao bloco *de-multiplexer* que verifica se estes pacotes são destinados ao nó local. Nesse caso, estes pacotes são enviados à camada superior. Senão, envia os pacotes ao bloco *forwarding*. O bloco *forwarding* que pode também receber pacotes gerados localmente de suas camadas superiores consulta a tabela de roteamento e determina o próximo salto para o pacote [RAD, 04]. Depois disto, os pacotes são transmitidos para a fila de saída, é neste momento que o Controle de Tráfego do Linux atua novamente. Os pacotes são enfileirados, descartados, atrasados ou priorizados de acordo com a política adotada no nó.

O controle de tráfego do Linux possui os seguintes componentes:

- Disciplina de Serviço (qdiscs);
- Classes;
- Filtros;
- Policiador de Tráfego;

A figura A.2 é uma forma prática e fácil de representar estes componentes. Pacotes entram no qdisc principal à esquerda e imediatamente são classificados pelos filtros. Dependendo da estratégia de configuração adotada o fluxo de dados pode ser policiado, ou não. Em seguida os pacotes de dados são encaminhados a uma classe. Cada classe possui uma disciplina de serviço associada.

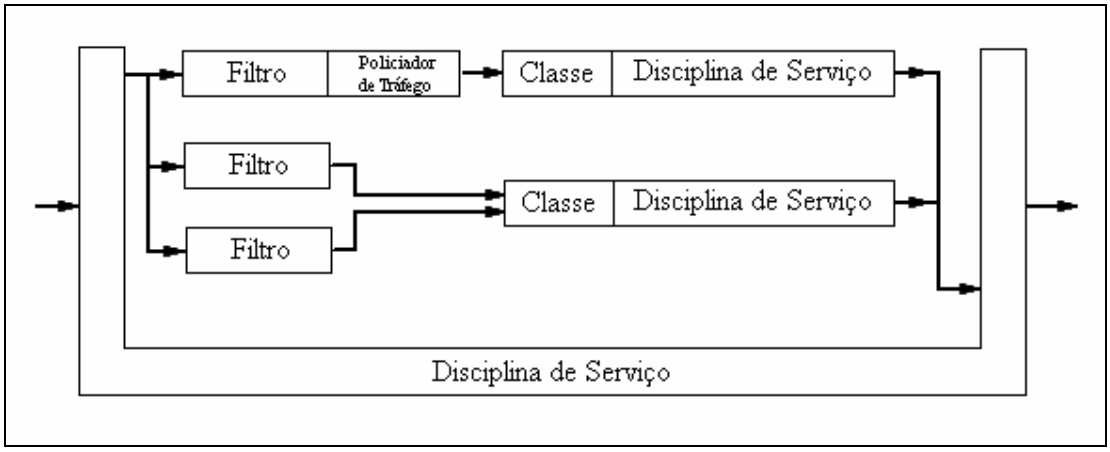

Figura A.2: Componentes de controle de tráfego do Linux.

.

A figura A.3 mostra a relação dos elementos de DiffServ com a arquitetura de Controle de Tráfego no Linux.

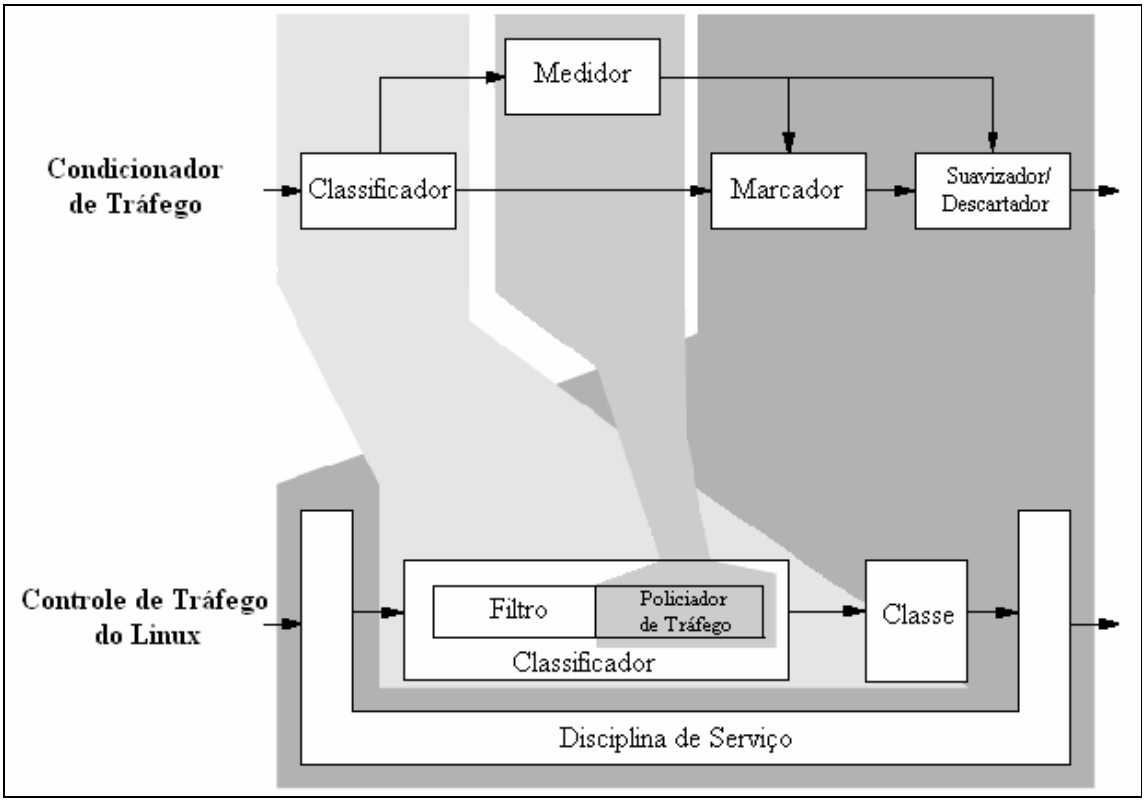

Figura A.3: Relação da arquitetura DiffServ com a arquitetura de controle de tráfego do Linux.

A.2.1 Qdisc

Os qdiscs, abreviação de *Queueing Disciplines* (Disciplina de serviço) [BRO, 04], são as filas de saída dos pacotes. O Linux possui dois tipos de qdisc: *classless* e *classful*. Os qdiscs *classless* não podem conter classes definidas pelo administrador, embora alguns deles possuam mais de uma fila de saída. Já os qdiscs *classful* podem conter classes definidas pelo administrador, permitindo assim a separação e a atribuição de quantidades diferentes de banda para cada tipo de fluxo.

Cada interface de rede possui um qdisc root. O padrão do Linux é atribuir um qdisc do tipo pfifo_fast [BER, 05] para o qdisc root, mas obviamente, isso pode ser mudado.

A.2.2 Classes

As classes só existem dentro de qdiscs *classful* e são uma forma de se dividir o tráfego para um tratamento diferenciado. Uma classe não manipula um pacote diretamente, para isso

ela recorre a uma disciplina de serviço (qdisc), sendo possível um grande número de combinações.

A.2.3 Filtros

É o mecanismo através do qual um pacote é atribuído a uma classe. Um filtro é chamado no momento que um pacote chega a uma disciplina de serviço. Isso é feito com o auxílio de um classificador. Além disso, um filtro pode exercer a função de policiamento, tomando uma ação sempre de acordo com os limites estabelecidos para o fluxo que ele está classificando.

Dentro de um filtro, um classificador é utilizado para identificar certos padrões ou características dos pacotes e fluxos permitindo assim a separação em classes.

A.2.4 Policiador de Tráfego

Policiador de Tráfego é o mecanismo pelo qual o tráfego pode ser limitado. Assim é possível garantir que um determinado tipo de tráfego não consuma mais banda do que foi destinado a ele. O policiamento no Linux e é feito através dos filtros.

A.3 Exemplo de DiffServ no Linux

Este tópico apresenta as configurações de nós de borda e nós de núcleo. A maior diferença entre estas duas configurações é basicamente que, nós de borda dividem o tráfego em diferentes classes de acordo com critérios predefinidos pelo administrador da rede. Enquanto nós de núcleo executam a divisão baseados somente em DSCPs. Os exemplos apresentados aqui são baseados em [BAL, 04] e [LEM, 00].

A.3.1 Nó de Borda

Este exemplo mostra como implementar a funcionalidade de um nó de borda para uma classe do grupo de PHB AF. As outras classes do grupo de PHB AF foram omitidas, uma vez que a configuração é semelhante. A figura A.4 descreve este exemplo.

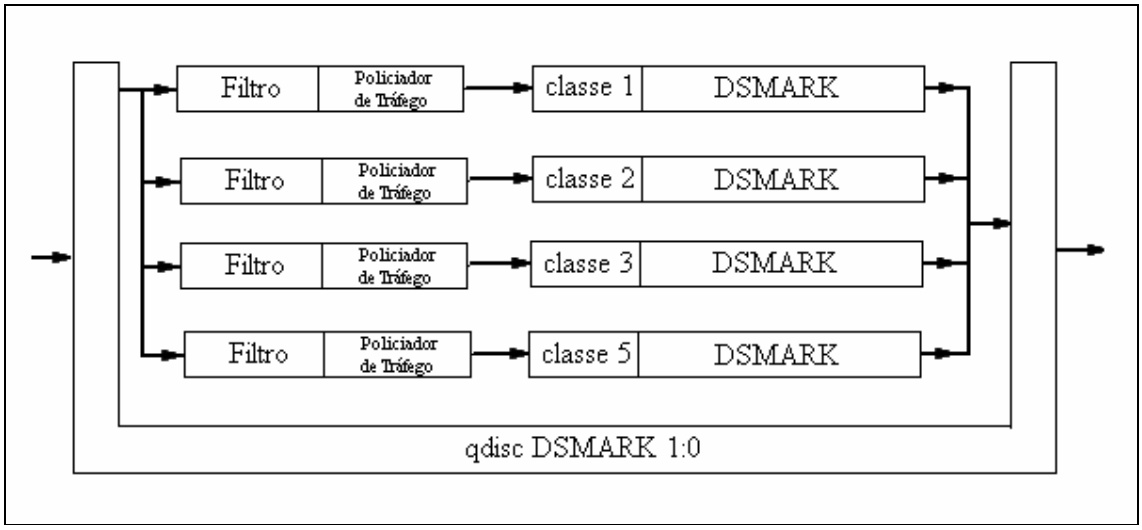

Figura A.4: Configuração de um nó de borda em Linux.

qdisc root

1. tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 default_index 5

Classes dsmark para especificar o campo DSCP

2. tc class change dev eth0 parent 1:0 classid 1:1 dsmark mask 0x3 value 0x28

3. tc class change dev eth0 parent 1:0 classid 1:2 dsmark mask 0x3 value 0x30

4. tc class change dev eth0 parent 1:0 classid 1:3 dsmark mask 0x3 value 0x38

5. tc class change dev eth0 parent 1:0 classid 1:5 dsmark mask 0x3 value 0x0

Filtros

6. tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip src 192.168.1.0/24 \
match ip dst 192.168.3.0/25 police rate 500kbit burst 20k drop classid 1:1

7. tc filter add dev eth0 parent 1:0 protocol ip prio 3 u32 match ip dst 192.168.9.0/24 \
police rate 500kbit burst 20k drop classid 1:2

8. tc filter add dev eth0 parent 1:0 protocol ip prio 3 u32 match ip src 192.168.7.0/24 \
`match ip sport 22 0xffff` police rate 500kbit burst 20k drop classid 1:3

Para configurar DiffServ em Linux é necessário adicionar a interface que está sendo configurada uma regra de enfileiramento (qdisc) e vincular a esta um manipulador, neste caso "1:0". Este manipulador será referenciado por comandos para ajudar a criar a estrutura hierárquica.

O campo DSCP dos pacotes IP são marcados das linhas 2 a 5. O qdisc dsmark [BER, 05] é responsável pela marcação. Os pacotes nomeados para estas classes são marcados com

um novo valor no byte DS, enquanto os dois bits menos significantes do campo de ToS são deixados intactos como especificado pela RFC 2475.

Os filtros mostrados das linhas 6 a 8 são usados para selecionar pacotes para serem marcados em seu campo DS. Os pacotes podem ser selecionados com base em qualquer campo do cabeçalho IP, como por exemplo, endereços IP e/ou portas TCP/UDP. Os filtros também têm o parâmetro *police* responsável pelo policiamento do tráfego que está chegando. Tráfico que estiver até o limite especificando é aceito para esta classe, o excedente será descartado, ou enviado a outro filtro.

A.3.2 Nó de Núcleo

Este exemplo mostra como implementar a funcionalidade de um nó de núcleo para o grupo PHB AF. Novamente, só a classe 1 de AF é demonstrada para maior brevidade. A figura A.5 descreve este exemplo.

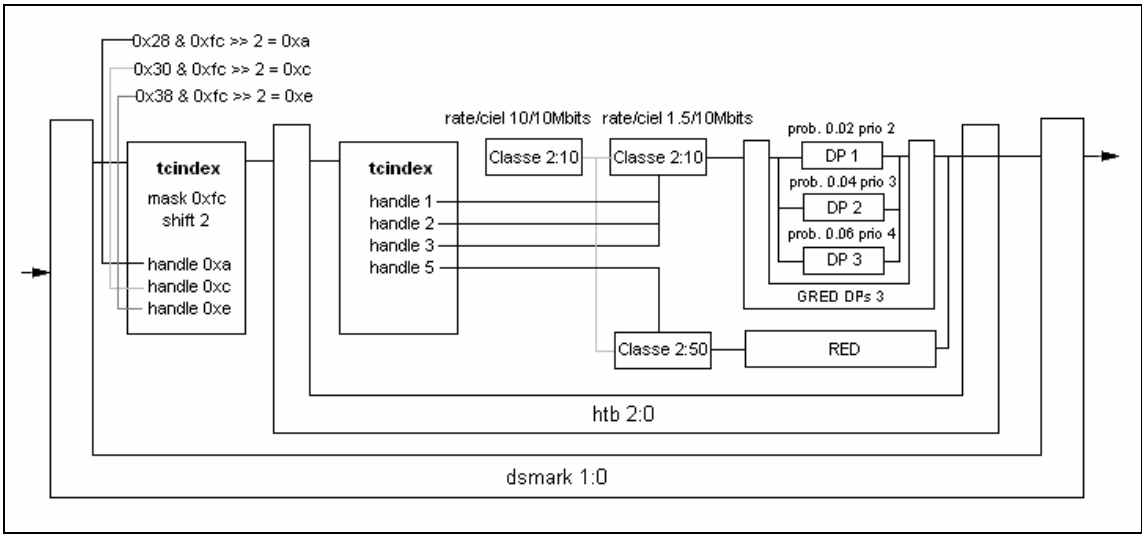

Figura A.5: Configuração de um nó de núcleo em Linux.

qdisc root

1. tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 default_index 5 set_tc_index

Filtros

2. tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2 pass_on
3. tc filter add dev eth0 parent 1:0 protocol ip prio 1 handle 0xa tcindex classid 1:1
4. tc filter add dev eth0 parent 1:0 protocol ip prio 2 handle 0xc tcindex classid 1:2
5. tc filter add dev eth0 parent 1:0 protocol ip prio 3 handle 0xe tcindex classid 1:3

Divisão de Banda

6. tc qdisc add dev eth0 parent 1:0 handle 2:0 htb

7. tc class add dev eth0 parent 2:0 classid 2:1 htb rate 10Mbit ceil 10Mbit

8. tc class add dev eth0 parent 2:1 classid 2:10 htb rate 1500Kbit ceil 10Mbit

9. tc class add dev eth0 parent 2:1 classid 2:50 htb rate 1500Kbit ceil 10Mbit

Disciplinas de Serviço das Classes

10. tc qdisc add dev eth0 parent 2:10 gred setup DPs 3 default 2 grio

11. tc qdisc change dev eth0 parent 2:10 gred limit 60KB min 15KB max 45KB \
burst 20 avpkt 1000 bandwidth 10Mbit DP 1 probability 0.02 prio 2

12. tc qdisc change dev eth0 parent 2:10 gred limit 60KB min 15KB max 45KB \ burst 20 avpkt 1000 bandwidth 10Mbit DP 2 probability 0.04 prio 3

13. tc qdisc change dev eth0 parent 2:10 gred limit 60KB min 15KB max 45KB \
burst 20 avpkt 1000 bandwidth 10Mbit DP 3 probability 0.06 prio 4

14. tc qdisc add dev eth0 parent 2:50 red limit 60KB min 15KB max 45KB \
burst 20 avpkt 1000 bandwidth 10Mbit probability 0.4

Filtros

15. tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 1 tcindex classid 2:10

16. tc filter add dev eth0 parent 2:0 protocol ip prio 2 handle 2 tcindex classid 2:10

17. tc filter add dev eth0 parent 2:0 protocol ip prio 3 handle 3 tcindex classid 2:10

18. tc filter add dev eth0 parent 2:0 protocol ip prio 5 handle 5 tcindex classid 2:50

Aqui classificação dos pacotes está baseada somente valor do campo DSCP, assim nenhuma classe de dsmark precisa ser criada. O qdisc root dsmark é configurado para que o valor do byte DS seja copiado para a variável *skb->tc_index*. O valor da variável *skb->tc_index* então passa por uma operação para que somente o valor DSCP seja utilizado, descartando os dois bits do campo ECN. O valor adquirido é então passado aos filtros, se o valor do campo DSCP não for encontrado no primeiro filtro, este valor é repassado ao próximo filtro até que seja encontrado um filtro responsável pelo valor.

A divisão de banda é feita utilizando o qdisc HTB (*Hierarchical Token Bucket*) [DEV, 05] que está preso ao qdisc dsmark. Ambas classes estão limitadas efetivamente a 1500Kbit podendo pedir banda emprestada a classe pai se esta tiver disponível. As duas classes têm uma disciplina de serviço (qdisc) ligada a si, uma vez que elas não manipulam diretamente um pacote.

O qdisc GRED (*Generalized Random Early Detection*) [BAL, 04] é preso à classe "2:10". GRED é uma disciplina de serviço que realiza um descarte preventivo de pacotes. É configurado com três filas virtuais (linhas 11, 12 e 13). Cada fila virtual tem uma prioridade de envio de pacotes e uma probabilidade de descarte de pacotes.

O qdisc RED (*Random Early Detection*) [FLO, 05] está ligado à classe "2:50", é a disciplina de serviço que originou GRED. O qdisc RED não possui filas virtuais para priorizar tráfego como GRED.

Por último, nas linhas 15 a 19, são criados os filtros que direcionam os pacotes de dados às classes de HTB correspondentes. A classificação é baseada no valor em *skb->tc_index* fixado nas linhas 1, 3, 4 e 5.

A.4 Entendendo estatísticas

A ferramenta de controle de tráfego do Linux permite obter estatísticas sobre as regras de enfileiramento no Linux. Os resultados fornecidos abaixo foram obtidos durante a simulação do item A.3.2.

Primeiro as estatísticas dos qdiscs:

```
#tc -s -d qdisc show dev eth0

qdisc red 8002: limit 60Kb min 15Kb max 45Kb ewma 3 Plog 17 Scell_log 9
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc gred 8001:
 DP:1 (prio 2) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 21 Scell_log 9
 DP:2 (prio 3) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 20 Scell_log 9
 DP:3 (prio 4) Average Queue 0b Measured Queue 0b
         Packet drops: 0 (forced 0 early 0)
         Packet totals: 0 (bytes 0)
 limit 60Kb min 15Kb max 45Kb ewma 3 Plog 19 Scell_log 9
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc htb 2: r2q 10 default 0 direct_packets_stat 0 ver 3.10
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc dsmark 1: indices 0x0040 default_index 0x0005 set_tc_index
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
```

Os dois primeiros qdiscs (RED e GRED) são filhos das classes do qdisc HTB. No primeiro bloco o qdisc RED através do parâmetro *sent* informa quantos pacotes/bytes foram enviados por intermédio de sua fila. O parâmetro *dropped* informa quantos pacotes foram descartados pela fila e *overlimits* diz quantas vezes a regra atrasou um pacote.

No segundo bloco são fornecidas as estatísticas do qdisc GRED. Este foi configurado com três filas virtuais. Cada fila virtual informa quantos pacotes foram enviados por ela (parâmetro *Packet totals*), e quantos pacotes foram descartados pela fila (parâmetro *Packet drops*).

No terceiro e no quarto bloco são fornecidas as estatísticas de quantos pacotes foram enviados, quantos pacotes foram descartados, ou atrasados pelos qdiscs HTB e dsmark respectivamente.

Estatísticas das Classes:

```
#tc –s –d class show dev eth0
class htb 2:1 root rate 10Mbit ceil 10Mbit burst 14704b/8 mpu 0b cburst
14704b/8 mpu 0b level 7
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 9191 ctokens: 9191

class htb 2:10 parent 2:1 leaf 8001: prio 0 quantum 19200 rate 1500Kbit
ceil 10Mbit burst 3519b/8 mpu 0b cburst 14704b/8 mpu 0b level 0
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 15018 ctokens: 9191

class htb 2:50 parent 2:1 leaf 8002: prio 0 quantum 19200 rate 1500Kbit
ceil 10Mbit burst 3519b/8 mpu 0b cburst 14704b/8 mpu 0b level 0
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 15018 ctokens: 9191
```

- *sent*: Número de bytes enviados pela classe, valor também fornecido em pacotes;
- *overlimits*: Mostra quantas vezes a classe solicitou enviar um pacote mas ele não pode ser enviado por restrições de *rate*/*ceil*;
- *rate, pps*: Mostram a taxa atual (média 10 seg) da classe. Esta é a mesma taxa usada pela passagem;
- *lended*: É o número de pacotes doados por esta classe (de sua taxa);
- *borrowed*: São pacotes tomados emprestados da classe pai. *Lends* (empréstimos concedidos) são sempre computados classe-local enquanto *borrows* (empréstimos tomados) são transitivos. Por exemplo, se a classe “2:50” fosse fila de “2:5” e esta

filha de “2:1” (quando “2:50” toma emprestado de “2:5” que por sua vez toma emprestado de “2:1” ambos contadores *borrow* de “2:50” e de “2:5” são incrementados);

- *giants* (gigantes): É o número de pacotes maiores que *mtu* configurado no condicionador de tráfego.

Estatísticas dos Filtros:

```
#tc –s –d filter show dev eth0


filter parent 1: protocol ip pref 3 u32 fh 800:[80000000]  ht divisor 1
filter parent 1: protocol ip pref 3 u32 fh 800::800[80000800]  order 2048
key ht 800 bkt 0 flowid 1:1
police 1 action drop rate 500Kbit burst 20Kb mtu 2Kb [00040000]
  match c0a80100/ffffff00 at 12

 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
filter parent 1: protocol ip pref 3 u32 fh 800::801[80000801]  order 2049
key ht 800 bkt 0 flowid 1:2
police 2 action drop rate 500Kbit burst 20Kb mtu 2Kb [00040000]
  match c0a80100/ffffff00 at 12
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
```

Os filtros configurados no item A.3.1, também fornecem estatísticas quando o parâmetro *police* é configurado. São informados quantos pacotes/bytes foram enviados, e a quantidade de pacotes descartados ou atrasados.