

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA  
ELÉTRICA**

**SERVIÇOS DIFERENCIADOS EM REDES IP:  
MEDIÇÕES E TESTES PARA APLICAÇÕES  
ENVOLVENDO MÍDIAS CONTÍNUAS**

Dissertação submetida à  
Universidade Federal de Santa Catarina  
como parte dos requisitos para a  
obtenção do grau de Mestre em Engenharia Elétrica.

**RENATO DONIZETE VILELA DE OLIVEIRA**

Florianópolis, Abril de 2001.

# SERVIÇOS DIFERENCIADOS EM REDES IP: MEDIÇÕES E TESTES PARA APLICAÇÕES ENVOLVENDO MÍDIAS CONTÍNUAS

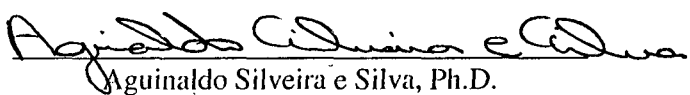
Renato Donizete Vilela de Oliveira

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Informática Industrial*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'



---

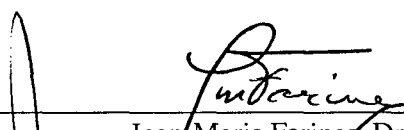
Jean-Marie Farines, Dr.  
Orientador



---

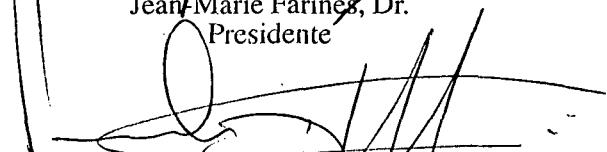
Aginaldo Silveira e Silva, Ph.D.  
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:



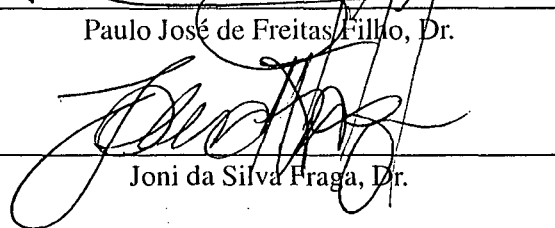
---

Jean-Marie Farines, Dr.  
Presidente




---

Paulo José de Freitas Filho, Dr.



---

Joni da Silva Fraga, Dr.



---

Elvis Melo Vieira, M.Sc.

*Tudo o que vale a pena possuir, vale a pena esperar o tempo de Deus ...*

*(Banda Taus - Franca/SP)*

*Aos meus pais ...*  
*Gilda e Nenén Ouro*

## *Agradecimentos*

*Inicialmente quero agradecer a Deus, criador de todas as coisas visíveis e invisíveis, o qual me concedeu saúde, paz e amor no decorrer de meu trabalho.*

*Quero agradecer a minha mãe Gilda e ao meu pai Nenén Ouro, por todo carinho, compreensão e apoio. Aos meus irmãos Fernando, Júnior, Roberto e Eliana pelo afeto a mim transmitido.*

*Agradeço a meu orientador Jean-Marie Farines, pelo ensinamento e incentivo ao longo do trabalho.*

*Aos membros da banca examinadora, que contribuíram opinando e sugerindo.*

*Sincero agradecimento para meus ex-professores Gualberto Rabay Filho e Marcos Augusto Francisco Borges, por terem me incentivado em seguir pelo caminho do mestrado.*

*Também agradeço a Cesar Torrico, pela colaboração e sugestões.*

*A CAPES pelo financiamento econômico parcial.*

*Finalmente agradeço aos Amigos: Fred Paes, Terezinha Faria, Carlos Ogawa, Rodrigo Yoneda, Marcos Peninha, Rafael Obelheiro, Carlos Brandão, Sobral, Lau, Frank, Michele Wangham, Andréa Rosada, Fabiano, Rottava, Max Magalhães, Alexandre Matiello, Denise Bendo, Kétner Bendo, Jucimara, Emilaura, Rogerio, Camila, Cristiane, Alexandre Camargo ... ; pelo companheirismo e os bons momentos nestes dois anos.*

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

# SERVIÇOS DIFERENCIADOS EM REDES IP: MEDIÇÕES E TESTES PARA APLICAÇÕES ENVOLVENDO MÍDIAS CONTÍNUAS

**Renato Donizete Vilela de Oliveira**

Abril 2001

Orientador: Jean-Marie Farines, Dr.

Área de Concentração: Controle, Automação e Informática Industrial

Palavras-chave: **Qualidade de Serviço, Serviços Diferenciados, Internet.**

Número de Páginas: 79

A atual arquitetura da Internet não possibilita a provisão da Qualidade de Serviço (*Quality of Service* - QoS) exigida por diferentes fluxos de aplicações multimídias heterogêneas. Essa incapacidade nos leva a pensar na necessidade de serviços adicionais a serem oferecidos pela Internet, possivelmente com tarifas diferenciadas. A arquitetura de Serviços Diferenciados propõe-se a oferecer QoS de forma escalável às novas aplicações multimídias surgidas na Internet. Nesse sentido, o presente trabalho consiste na implementação e uso de uma plataforma de testes - *Testbed* - para aplicações que envolvam mídias contínuas (áudio e vídeo) sobre redes IP, procurando estabelecer um determinado nível de QoS fim a fim para fluxos agregados. Nesse propósito, este *testbed* é implementado baseando-se nas abordagens do IETF para IP QoS, na forma de Serviços Diferenciados. Durante os testes procurou-se avaliar alguns parâmetros pertinentes com relação a aplicações que exigem da rede uma garantia fim a fim. Dentre os parâmetros analisados estão a taxa de perda de pacotes, o atraso médio por pacote (*delay*) e o atraso médio entre os pacotes (*jitter*). A seqüência de ensaios realizados sobre a plataforma, avalia o desempenho dos serviços oferecidos ao tráfego gerado por fontes de taxa constante (*Constant Bit Rate* - CBR) pelo esquema de Encaminhamento Expresso (*Expedited Forwarding* - EF). Os resultados para a implementação do Encaminhamento Expresso demonstram que é possível, dentro de um domínio DS, obter uma certa preservação da qualidade de transmissão para um fluxo através da provisão de prioridade de encaminhamento em relação ao fluxo do tipo Melhor Esforço (*Best Effort*).

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree. of Master in Electrical Engineering.

# **DIFFERENTIATED SERVICES IN IP NETWORK: MEASUREMENTS AND TESTS FOR APPLICATIONS INVOLVING CONTINUOUS MEDIAS**

**Renato Donizete Vilela de Oliveira**

April 2001

Advisor: Jean-Marie Farines, Dr.

Area of Concentration: Control, Automation and Industrial Computing

Keywords: **Quality of Service, Differentiated Services, Internet.**

Number of Pages: 79

The current Internet architecture does not provide Quality of Service (QoS) required by different flows from heterogeneous multimedia applications. This incapacity indicates the need of additional services to be offered in the Internet, possibly with differentiated tariff. The Differentiated Services architecture proposes to offer QoS in a scalable manner to the new multimedia applications in the Internet. In this sense, the present work consists of the implementation and use of a platform of tests for continuous media applications (particularly, videoconference). Such platform is a communication support based on the IP networks with multicast support. It allows to establish an end-to-end QoS level for joined flows. The tests bases on the approaches of IETF for IP QoS, in the form of Differentiated Services using the Expedited Forwarding (EF) service class. The network support used was ATM (Constant Bit Rate - CBR - class). The tests permitted to evaluate the behavior of some QoS parameters such as: package loss rate, delay and jitter. They showed that Differentiated Services allow to keep these QoS parameters inside a range by priority establishing.

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Objetivo do Trabalho . . . . .	3
1.2	Organização do texto . . . . .	4
<b>2</b>	<b>Qualidade de Serviço na Internet</b>	<b>5</b>
2.1	Serviços Integrados . . . . .	5
2.1.1	O Modelo de Referência para Implementação . . . . .	6
2.1.2	O Serviço Garantido . . . . .	7
2.1.3	O Serviço de Carga Controlada . . . . .	8
2.1.4	O Protocolo RSVP . . . . .	9
2.1.5	Problemas com IntServ . . . . .	10
2.2	Serviços Diferenciados . . . . .	11
2.3	Multiprotocol Label Switching (MPLS) . . . . .	12
2.4	Roteamento baseado em QoS (QoSR) . . . . .	14
2.5	Engenharia de Tráfego . . . . .	16
2.6	Conclusões do capítulo . . . . .	17
<b>3</b>	<b>Serviços Diferenciados na Internet</b>	<b>19</b>
3.1	Serviços Diferenciados . . . . .	19
3.1.1	Marcação do campo DS . . . . .	20
3.1.2	Condicionamento de Tráfego . . . . .	21
3.1.3	PHB para implementação de Serviços Diferenciados . . . . .	23
3.2	Encaminhamento Expresso . . . . .	23
3.3	Encaminhamento Assegurado . . . . .	25
3.4	Gerenciamento Ativo de Filas . . . . .	27
3.4.1	RED . . . . .	28
3.4.2	Extensões ao RED . . . . .	30
3.4.2.1	RED adaptativo . . . . .	30
3.4.2.2	FRED . . . . .	32
3.4.3	Generalização do RED . . . . .	33



3.4.3.1	RIO . . . . .	34
3.5	Aplicabilidade de Serviços Diferenciados . . . . .	36
3.6	Conclusões do capítulo . . . . .	37
<b>4</b>	<b>Plataforma de Testes</b>	<b>39</b>
4.1	A Ferramenta TMG . . . . .	40
4.1.1	Descrição da Ferramenta . . . . .	40
4.1.2	A Composição da Ferramenta . . . . .	41
4.2	Controlador de Tráfego . . . . .	44
4.2.1	Componentes . . . . .	45
4.2.1.1	Disciplinas de Fila . . . . .	45
4.2.1.2	Classes . . . . .	47
4.2.1.3	Filtros . . . . .	48
4.2.2	A Ferramenta TC . . . . .	50
4.3	Arquitetura da Plataforma de testes . . . . .	50
4.3.1	Configuração do roteador de borda ( <i>Edge</i> ) . . . . .	53
4.3.2	Configuração do roteador interno ( <i>Core</i> ) . . . . .	53
4.4	Conclusões do capítulo . . . . .	55
<b>5</b>	<b>Experimentos e Resultados</b>	<b>56</b>
5.1	Topologia do ambiente de teste . . . . .	57
5.1.1	Arquitetura do <i>testbed</i> . . . . .	57
5.1.2	Ferramentas de medição . . . . .	57
5.1.3	Estratégia para os testes . . . . .	58
5.2	Experimentos e Resultados Obtidos . . . . .	59
5.2.1	Caso tráfego EF - Um Fluxo . . . . .	59
5.2.2	Caso tráfego EF - Dois Fluxos . . . . .	61
5.2.3	Caso Tráfego EF (um fluxo) e Tráfego BE . . . . .	63
5.2.4	Caso Tráfego EF (dois fluxos) e Tráfego BE . . . . .	69
5.3	Comentários sobre os Experimentos . . . . .	75
5.4	Comentários sobre o ambiente de teste . . . . .	77
<b>6</b>	<b>Conclusão</b>	<b>79</b>

# Lista de Figuras

2.1	Estrutura de Serviços Integrados . . . . .	6
2.2	Sinalização RSVP . . . . .	10
2.3	Comparação qualitativa entre modelos de serviço . . . . .	12
2.4	Encaminhamento de pacotes em um domínio MPLS . . . . .	13
2.5	Encaminhamento de pacotes a) sem ET; b) com ET . . . . .	17
3.1	Domínio de Serviços Diferenciados - DS . . . . .	20
3.2	Estrutura do campo DS . . . . .	20
3.3	Funções de Condicionamento de Tráfego . . . . .	22
3.4	Implementação do PHB de Encaminhamento Expresso . . . . .	24
3.5	Implementação do PHB de Encaminhamento Assegurado . . . . .	27
3.6	Parâmetros do algoritmo RED . . . . .	29
3.7	Comportamento do algoritmo RED Adaptativo . . . . .	31
3.8	Exemplo de WRED com 03 classes . . . . .	34
3.9	Parâmetros do Algoritmo RIO . . . . .	35
3.10	Serviços Diferenciados providos por mais de um domínio . . . . .	36
3.11	Implementação conjunta dos PHBs EF e AF . . . . .	37
4.1	Controle de Tráfego no Linux . . . . .	44
4.2	Disciplina de Fila simples com múltiplas classes . . . . .	46
4.3	Combinação de prioridade, TBF e disciplinas de fila FIFO . . . . .	47
4.4	Estrutura do Filtros . . . . .	49
4.5	Espaço do usuário para comunicação com o kernel usando o TC . . . . .	50
4.6	Arquitetura do <i>testbed</i> . . . . .	52
4.7	Arquitetura do roteador de borda . . . . .	53
4.8	Arquitetura do roteador interno . . . . .	54
5.1	Perda de pacotes com relação ao tráfego gerado . . . . .	60
5.2	Atrasos sofrido pelo fluxo EF com relação ao tráfego gerado . . . . .	61
5.3	Perda de pacotes numa transmissão envolvendo dois fluxos EF . . . . .	62
5.4	Atraso médio por pacote numa transmissão envolvendo dois fluxos EF . . . . .	63
5.5	Atraso médio entre pacotes antes e depois da alocação de mais recursos na rede . . . . .	63

5.6	Perda de pacotes - tráfego EF (250Kbps) e BE . . . . .	65
5.7	Perda de pacotes - tráfego EF (500Kbps) e BE . . . . .	65
5.8	Perda de pacotes - tráfego EF (550Kbps) e BE . . . . .	66
5.9	Atraso médio por pacote - tráfego EF (250Kbps) e BE . . . . .	66
5.10	Atraso médio por pacote - tráfego EF (500Kbps) e BE . . . . .	67
5.11	Atraso médio por pacote - tráfego EF (550Kbps) e BE . . . . .	67
5.12	Atraso médio entre pacotes - tráfego EF (250Kbps) e BE . . . . .	68
5.13	Atraso médio entre pacotes - tráfego EF (500Kbps) e BE . . . . .	68
5.14	Atraso médio entre pacotes - tráfego EF (550Kbps) e BE . . . . .	69
5.15	Perda de pacotes - tráfegos EF1 (125Kbps), EF2 (125Kbps) e BE . . . . .	70
5.16	Perda de pacotes - tráfegos EF1 (250Kbps), EF2 (250Kbps) e BE . . . . .	71
5.17	Perda de pacotes - tráfegos EF1 (275Kbps), EF2 (275Kbps) e BE . . . . .	71
5.18	Atraso médio por pacote - tráfego EF1 (125Kbps), EF2 (125Kbps) e BE . . . . .	72
5.19	Atraso médio por pacote - tráfego EF1 (250Kbps), EF2 (250Kbps) e BE . . . . .	73
5.20	Atraso médio por pacote - tráfego EF1 (275Kbps), EF2 (275Kbps) e BE . . . . .	73
5.21	Atraso médio entre pacotes - tráfego EF1 (125Kbps), EF2 (125Kbps) e BE . . . . .	74
5.22	Atraso médio entre pacotes - tráfego EF1 (250Kbps), EF2 (250Kbps) e BE . . . . .	74
5.23	Atraso médio entre pacotes - tráfego EF1 (275Kbps), EF2 (275Kbps) e BE . . . . .	75

# Capítulo 1

## Introdução

A Internet é de fato a grande rede mundial e, conseqüentemente, o ambiente de desenvolvimento de novas aplicações de rede. As decisões de projeto realizadas para a Internet não mais afetam apenas uma pequena comunidade técnica, mas podem causar impacto sobre o público em geral. Entretanto, a compreensão de algumas limitações da Internet atual justifica a necessidade de alterações em sua arquitetura.

Desde a sua concepção inicial, na forma de uma rede que permitisse interligar centros de pesquisas e universidades, a Internet sempre teve como característica o fato de adotar um modelo de serviço de melhor esforço (*best effort*). Este tipo de serviço oferecido pelo protocolo IP não fornece um suporte à qualidade de serviço (*Quality of Service* - QoS) satisfatório. Algumas aplicações de tempo-real (vídeo sob demanda, teleconferência multimídia, realidade virtual, etc.) não funcionam de maneira eficiente devido a atrasos nas filas dos roteadores e a perdas de pacotes em pontos mais congestionados da rede. Além disso, a Internet atual passa por um crescimento notório e vertiginoso com relação ao número de máquinas conectadas à rede e, como conseqüência disso, aumenta-se também a quantidade de tráfego transportado, sem que este seja diferenciado.

Entretanto, vários tipos de serviços foram incluídos recentemente com a finalidade de atender a diferentes tipos de aplicações e usuários. Serviços esses que podem ser agrupados em classes. Uma classe de serviço a considerar, pode ser a que fornece um baixo atraso assim como uma baixa variação de atraso (*jitter*) para aplicações como telefonia IP e videoconferência. Empresas podem interessar-se em pagar por um serviço diferenciado que suporte uma videoconferência com qualidade, economizando assim, tempo de deslocamento e reduzindo custos. Uma outra classe de serviço a qual devemos considerar, é a que pode prover o suporte necessário para empresas que realizam negócios na rede. Empresas desse tipo podem ter interesse em pagar um determinado preço por serviços confiáveis e que dêem a seus usuários uma maior rapidez

no acesso a seus sites na rede. A classe de serviço de melhor esforço continua a ser oferecida para usuários, no caso de aplicações que são menos exigentes em termos de tempo e segurança, desejando apenas conectividade.

Dentro dessa linha, podemos então ressaltar um aspecto muito importante que diz respeito ao gerenciamento de tráfego na Internet. Existe uma grande necessidade por parte das operadoras de comunicação em se controlar (e cobrar) o compartilhamento da banda de transmissão entre as diversas classes de serviço e entre as diversas entidades que utilizam estes recursos. Deve ser possível assegurar uma “porcentagem” mínima do enlace de comunicação para cada classe (entidade) nos instantes de sobrecarga e, nos outros instantes, possibilitar o uso das bandas “ociosas” para aquelas classes que precisam de mais recursos. Tal gerenciamento é conhecido como compartilhamento controlado do enlace (*controlled link-sharing*).

Uma dúvida que pode surgir aqui é: “realmente é necessário mecanismos de gerenciamento de tráfego para a provisão de QoS?” (Xiao e Ni 1999). Por um lado, alguns dizem que não, pois a utilização de fibras óticas em conjunto com a tecnologia de Multiplexação por Divisão de Comprimentos de Onda (*Wavelength Division Multiplexing - WDM*) tornará a oferta de banda passante tão abundante e barata que a QoS para qualquer aplicação estará automaticamente fornecida. Já por outro lado, os defensores da adoção destes mecanismos argumentam que independentemente da quantidade de banda passante que uma rede possa ofertar, novas aplicações serão desenvolvidas para consumir toda esta banda passante. Além do mais, o fato de se ter uma maior quantidade de banda, até mesmo servirá de incentivo para surgimento de tais aplicações. Mais ainda, uma grande quantidade de banda passante, não significa garantia de baixo atraso, pois esse parâmetro está relacionado ao tempo de espera em filas que decorre da carga da rede em determinado momento. Se o problema da provisão de QoS é solucionado a partir da abundância de banda passante, há a obrigação de diminuir o retardo de todos os fluxos para o atendimento dos requisitos de apenas um fluxo. De uma forma ou de outra, a provisão de QoS na Internet demanda e continuará demandando a adoção de mecanismos que tratem de maneira diferenciadas, fluxos provenientes de aplicações que exigem um tratamento distinto durante sua transmissão. Neste trabalho estudaremos e analisaremos alguns dos mecanismos que permitem prover QoS na Internet.

Dentre as abordagens para provisão de QoS em uma rede IP, a arquitetura de Serviços Diferenciados (Blake et al. 1998), proposta pelo grupo de trabalho DiffServ (*Differentiated Services*) do IETF (*The Internet Engineering Task Force*)(Group 2001), pode ser considerada a que mais provavelmente será utilizada para QoS na Internet devido a sua relativa simplicidade e as suas características de escalabilidade. Essa arquitetura mostra-se hoje a mais viável para suportar o conjunto de aplicações cujos requisitos de QoS exigem serviços melhores que de melhor esforço. Dentre os muitos estudos sendo realizados nessa área, podemos destacar os estudos sobre a estratégia de marcação de pacotes (Heinanen e Guerin 1999b, Heinanen e

Guerin 1999a, Heinanen e Guerin 1999c), sobre tarifação dos serviços diferenciados (Semret et al. 1999), sobre justiça na marcação de pacotes (Kim 1999), sobre suavizadores de tráfego adaptativos (Bonaventure e Cnodder 1999), além dos trabalhos citados no decorrer do texto.

Para o ambiente e plataforma de testes (*testbed*) Qbone (Teitelbaum 1999) desenvolvida no contexto do projeto da Internet2, escolheu como implementação inicial a arquitetura de Serviços Diferenciados. Nessa escolha, foram levados em consideração, além de outras, três características principais: a flexibilidade, a escalabilidade e a interoperabilidade. A intenção desta plataforma é submeter os mecanismos propostos a testes reais.

Através dos mecanismos encontrados na arquitetura de Serviços Diferenciados, serviços distintos podem ser oferecidos. Cada classe de serviço pode estar mais apta a atender às exigências de QoS específicas provenientes de uma categoria entre a variedade de aplicações multimídias que vêm surgindo. O transporte de áudio e vídeo desperta grande interesse nesse contexto devido a seus potenciais de aplicação. Inclusive com relação a áudio, podemos encontrar estudos preliminares já em andamento. Entre outros, Naser et al. (Naser et al. 1998) analisam o tráfego de voz transmitido sobre a estrutura original de Serviços Diferenciados (Nichols et al. 1999), onde o conceito de PHB ainda não estava definido.

## 1.1 Objetivo do Trabalho

O objetivo desse trabalho é o estudo e a análise de alguns dos serviços diferenciados. Para tal é necessária a implementação e utilização de um ambiente de teste (*testbed*) para aplicações que envolvam mídias contínuas (áudio e vídeo) (Xue e et al. 1999), especialmente as de vídeo-conferência. A plataforma desejada deve ser construída sobre um suporte de comunicação baseado em redes IP, onde é possível estabelecer um determinado nível de Qualidade de Serviço (QoS) fim a fim para fluxos agregados. Ela deve ser implementada baseando-se nas abordagens do IETF para IP QoS, na forma de Serviços Diferenciados. No presente trabalho é focado o desempenho do PHB de Encaminhamento Expresso (EF) (Jacobson et al. 1999) no transporte de tráfegos de mídias contínuas CBR (*Constant Bit Rate*).

A análise para o tráfego CBR inclui, dentre os algoritmos que implementam o serviço EF, o mecanismo com uma fila prioritária (*Priority Round-Robin* - PRR), a influência do número de fluxos presentes e quão eficientemente cada tipo de tráfego utiliza uma alocação extra de banda passante.

## 1.2 Organização do texto

O capítulo 2 apresenta as abordagens de Qualidade de Serviço na Internet as quais são aplicadas sobre redes IP, e estão atualmente em constante discussão entre os pesquisadores dessa área. O capítulo 3 apresenta a proposta de Serviços Diferenciados (Blake et al. 1998) do IETF, estudando os PHBs de Encaminhamento Expresso e Encaminhamento Assegurado, dando uma atenção especial para o primeiro, sobre o qual está focado esse trabalho. O capítulo 4 apresenta uma descrição do ambiente de teste (*testbed*) desenvolvido para esse trabalho. O capítulo 5 apresenta e discute os resultados obtidos através dos ensaios realizados sobre o ambiente de teste descrito no capítulo anterior. Finalmente o capítulo 6 apresenta as conclusões e perspectivas deste trabalho.

# Capítulo 2

## Qualidade de Serviço na Internet

NESTE capítulo apresentamos as principais abordagens que estão sendo discutidas atualmente para o oferecimento de QoS na Internet e os relacionamentos que podem existir entre elas. Existe um grande interesse acadêmico e comercial (fabricantes, provedores) nesse tema, mas o principal fórum de discussão está no âmbito do IETF (*Internet Engineering Task Force*), o órgão responsável pelas questões de engenharia na Internet.

### 2.1 Serviços Integrados

A Internet tradicionalmente oferece um único modelo de serviço chamado de melhor esforço, que apresenta um desempenho razoável para aplicações a quais não exigem grandes garantias de tempo e segurança, como transferência de arquivos e mensagens de correio eletrônico. Entretanto, à medida que nós avançamos em direção à era das comunicações multimídia, estão sendo desenvolvidas novas aplicações de tempo real com uma grande sensibilidade ao atraso da rede. Para essas aplicações, o modelo de melhor esforço é totalmente inadequado, mesmo em redes com cargas leves. Embora esse problema possa ser aliviado introduzindo o maior grau possível de adaptabilidade em certas aplicações, existe uma necessidade de garantias mais rígidas em termos de largura de banda, atraso e perda de pacotes.

Nesse contexto, o IETF criou o grupo de trabalho IntServ (*Integrated Services*) (Bernet e et al. 1999, Braden et al. 1994) para viabilizar o surgimento de uma rede de serviços integrados. O termo serviços integrados é empregado para designar um modelo de serviços para a Internet, que inclui o serviço de melhor esforço, serviços de tempo real e serviços de compartilhamento controlado de enlace (Braden et al. 1994). Esse último está relacionado com o oferecimento de algumas Classes de Serviços. Os objetivos iniciais do grupo IntServ são a criação de um modelo



de serviços integrados e de um modelo de referência para implementação. Como resultado já foram produzidas pelo IETF várias RFCs (*Requests for Comments*) e alguns dos tópicos mais importantes são abordados a seguir. Em um modelo de Serviços Integrados, um roteador deve suprir uma determinada QoS para cada fluxo, de acordo com seu modelo de serviço. Essa função é realizada pelo controle de tráfego que se divide em três componentes: o controle de admissão, o classificador de pacotes e o escalonador de pacotes. A estrutura proposta é ilustrada na Figura 2.1.

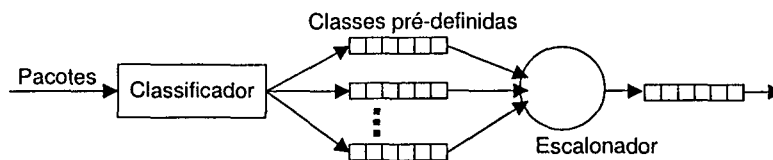


Figura 2.1: Estrutura de Serviços Integrados

### 2.1.1 O Modelo de Referência para Implementação

Este modelo visa estender a funcionalidade básica dos roteadores para habilitá-los a participar de uma rede de serviços integrados. Ele inclui quatro componentes: o escalonador de pacotes, a rotina de controle de admissão, o classificador e o protocolo de reserva de recursos. Em princípio, a reserva de recursos pode ser executada por qualquer protocolo que seja compatível com o modelo, mas na prática o protocolo RSVP (*Resource Reservation Protocol*) (Braden et al. 1997, Mankin et al. 1997) é o padrão de fato, tanto que se refere com frequência à arquitetura IntServ/RSVP. Devido à sua importância, o RSVP é apresentado na Seção 2.1.4 e os demais componentes são apresentados a seguir:

- *Escalonador de pacotes*: gerencia o encaminhamento dos vários fluxos de dados, usando alguma política de filas (por exemplo, FIFO (*DropTail*), Fila com Prioridade Circular (*Round Robin*) e Fila Justa com Pesos (WFQ), dentre outras) e possivelmente também outro tipo de mecanismo. O escalonador deve ser implementado no local onde os pacotes são enfileirados e deve haver uma comunicação com a interface da camada de enlace de dados para controlar a alocação da largura de banda entre os fluxos. Outro componente importante que pode ser considerado como parte do escalonador de pacotes é o avaliador, que mede características de tráfego dos fluxos para auxiliar o escalonamento de pacotes e o controle de admissão.
- *Classificador*: mapeia pacotes que chegam em determinadas classes, onde todos os pacotes em uma classe recebem o mesmo tratamento. Classe aqui é uma abstração e cada roteador pode mapear um mesmo pacote para uma classe diferente. No entanto, a gra-

nulosidade de uma classe é bastante fina, ou seja, geralmente corresponde a um fluxo específico.

- *Controle de admissão*: implementa o algoritmo que um roteador usa para determinar se um novo fluxo pode ter seu pedido de QoS atendido sem interferir nas garantias feitas anteriormente. É algo semelhante ao que ocorre no sistema telefônico, onde nós ouvimos um "sinal de ocupado" quando o sistema não tem recursos disponíveis para atender a chamada que está sendo feita. Nesse caso, significa que alguns fluxos podem ter seus pedidos de recursos rejeitados por falta de recursos em algum dos roteadores.

## 2.1.2 O Serviço Garantido

O Serviço Garantido (Shenker et al. 1997) é uma classe de QoS proporcionado pelo modelo de serviços integrados que oferece um nível assegurado de largura de banda, um limite rígido de atraso fim a fim e uma proteção contra a perda de pacotes nas filas, para os pacotes que estiverem obedecendo o perfil de tráfego contratado. É direcionado para aplicações com requisitos rígidos de tempo real, como certas aplicações multimídia intolerantes (por exemplo, vídeo e áudio) que precisam de uma garantia firme de que um pacote não irá chegar no destino depois de um tempo maior que um limite especificado. Esse serviço não oferece garantia mínima da variação de atraso, ele simplesmente garante um atraso máximo gerado pelas filas.

A obtenção de um limite máximo para o atraso exige que todos os roteadores no caminho suportem o serviço garantido. Para uma rede de grande porte como a Internet, isso se torna quase que impossível. Nesse tipo de serviço, o comportamento fim a fim oferecido por uma série de roteadores que implementam o serviço garantido é um nível assegurado de largura de banda para um determinado fluxo que, quando utilizado por um fluxo que está sendo policiado, produz um serviço com atraso limitado para todos os pacotes que estejam dentro do perfil.

Para ter acesso a esse serviço, as aplicações controlam os seus fluxos através de um balde de fichas (*Token Bucket*) e a partir dos valores de taxa e rajada, cada roteador calcula vários parâmetros descrevendo como ele tem que tratar os pacotes desses fluxos. Combinando os parâmetros dos vários roteadores em um caminho, é possível calcular o atraso máximo que um pacote irá experimentar quando transmitido por aquele caminho. Uma vez que as aplicações podem controlar os valores de taxa e rajada dos fluxos, elas conseguem obter uma garantia provada matematicamente sobre o atraso máximo dos seus pacotes.

O serviço garantido necessita de controle de admissão para operar de acordo com as especificações. Teoricamente, ele pode ser utilizado com qualquer protocolo de reserva de recursos, mas apenas a sua utilização em conjunto com o RSVP foi especificada.

### 2.1.3 O Serviço de Carga Controlada

O Serviço de Carga Controlada (Wroclawski 1997) não oferece garantias quantitativas rígidas, como o Serviço Garantido. O comportamento fim a fim oferecido para uma aplicação por uma série de roteadores, os quais implementam esse serviço, se assemelha ao comportamento visto por aplicações que estão recebendo o serviço de melhor esforço em uma rede apenas levemente carregada (ou seja, sem nenhuma situação grave de congestionamento). As garantias que as aplicações têm são:

- Um percentual muito alto de pacotes transmitidos chegará com sucesso no receptor. Deve-se ter uma aproximação da taxa básica de erros do meio de transmissão, ou seja, pouquíssimos descartes em filas são permitidos aqui.
- O atraso sofrido por um alto percentual dos pacotes não deverá exceder muito o atraso mínimo sofrido por um pacote dentro de um fluxo. Ou seja, a maior parte dos pacotes deve ter um atraso muito próximo do atraso mínimo.

Para assegurar que essas condições serão válidas, aplicações que requisitam o serviço de carga controlada devem fornecer aos roteadores uma estimativa do tráfego de dados que elas irão gerar, chamada de TSpec, a qual é baseada em um balde de fichas. Como resposta, o serviço assegura que a aplicação terá a sua disposição recursos dos roteadores suficientes para processar adequadamente todos os pacotes que estiverem de acordo com a especificação contida no TSpec. Por outro lado, pacotes introduzidos na rede fora das especificações, poderão ser descartados, ou enfrentar um atraso mais significativo.

O objetivo do serviço de carga controlada é suportar uma ampla classe de aplicações que tem sido desenvolvida para a Internet atual, mas que não funcionam em situações de carga alta na rede. Alguns membros dessa classe são as aplicações de tempo real adaptáveis, atualmente sendo oferecidas inclusive comercialmente. Essas aplicações têm mostrado que funcionam bem com redes com carga leve, mas a qualidade se degrada rapidamente em condições de congestionamento. Um serviço que imita redes com carga leve é útil para essas aplicações.

As aplicações podem solicitar o serviço de carga controlada antes de iniciar as transmissões, ou então somente quando elas detectam que o serviço de melhor esforço não está oferecendo um desempenho aceitável. A primeira estratégia oferece uma maior garantia de que o nível de QoS não irá mudar enquanto durar a sessão. A segunda estratégia é mais flexível e barata, pois o serviço com tarifação mais alta não é utilizado durante todo o tempo de duração da sessão.

## 2.1.4 O Protocolo RSVP

O RSVP (*Resource Reservation Protocol*) (Braden et al. 1997, Mankin et al. 1997), é um protocolo desenvolvido para realizar reserva de recursos em uma rede de serviços integrados. O RSVP é utilizado por sistemas finais para requisitar à rede níveis específicos de QoS para as aplicações. Também é utilizado pelos roteadores para repassar as requisições de QoS para todos os outros roteadores que estiverem no caminho entre fonte e destino e para estabelecer e manter informações de estado que possibilitam oferecer o serviço desejado. As requisições RSVP geralmente terão como resultado a reserva de recursos feita em todos os roteadores no caminho dos dados.

Algumas características importantes do protocolo RSVP são:

- O RSVP faz reservas para aplicações tanto de unidifusão (*unicast*) como para multidifusão (*multicast*), se adaptando dinamicamente às alterações dos membros de um grupo ou de rotas.
- O RSVP faz reservas somente para fluxos unidirecionais. Para conseguir reservas duplex, deve-se solicitar duas reservas simplex distintas nos dois sentidos.
- No RSVP quem inicia e mantém reservas para os fluxos é o receptor dos dados (é chamado de *receiver-initiated*).
- O estado das reservas no RSVP é “leve”(*soft-state*), ou seja, tem um tempo máximo de validade depois do qual ele expira. O receptor deve ficar constantemente “atualizando” o estado das reservas. Isso permite que se ofereça facilmente suporte a mudanças nos membros dos grupos, como também se adapte automaticamente a alterações no roteamento.
- O RSVP não é um protocolo de roteamento. Ele usa as rotas escolhidas por qualquer protocolo de roteamento em uso atualmente ou que venha a ser utilizado no futuro.
- O RSVP transporta e mantém informações (de estado) sobre o controle de tráfego e controle de políticas que são tratadas por outros módulos. O controle de tráfego, no caso, é exercido pelos outros módulos do IntServ.
- O RSVP oferece vários estilos de reservas, para se adaptar a uma grande variedade de aplicações e usos.
- Roteadores que não implementam RSVP podem estar presentes no caminho que suas mensagens são encaminhadas transparentemente.
- O RSVP suporta tanto o Ipv4 (Comer 1998) quanto o IPv6 (Comer 1998).

As duas mensagens mais importantes do protocolo RSVP são PATH, que é originado no transmissor e RESV, que é originado no receptor. Os principais objetivos da mensagem PATH são informar o receptor sobre as características de tráfego da requisição do transmissor e sobre o caminho fim a fim entre eles. Além disso, a mensagem PATH instala informações de roteamento reverso em todos os nós por onde passa, para que a mensagem RESV possa percorrer o mesmo caminho. A PATH não faz a reserva, ela é feita pela mensagem RESV, enviada pelo receptor. As informações de roteamento reverso são necessárias porque é comum que a comunicação nos dois sentidos siga caminhos distintos. Sem essas informações, as reservas de recursos poderiam ser feitas em roteadores diferentes daqueles por onde os dados vão passar.

Após receber uma mensagem PATH, um receptor envia uma mensagem RESV de volta ao último roteador solicitando uma reserva de recursos de acordo com os parâmetros especificados em PATH. Essa mensagem é reencaminhada para todos os roteadores no caminho, até chegar no transmissor. Cada roteador pode aceitar ou rejeitar a reserva de recursos solicitada, de acordo com a quantidade de recursos disponíveis e a política de controle de admissão adotada. Se a requisição é rejeitada, o roteador envia uma mensagem de erro para o receptor e a sinalização é encerrada. Se a requisição é aceita, largura de banda do enlace e espaço em buffers são alocados para o fluxo e informações de estado relacionadas ao fluxo são instalados no roteador. A Figura 2.2 apresenta a sinalização realizada pelo protocolo RSVP.

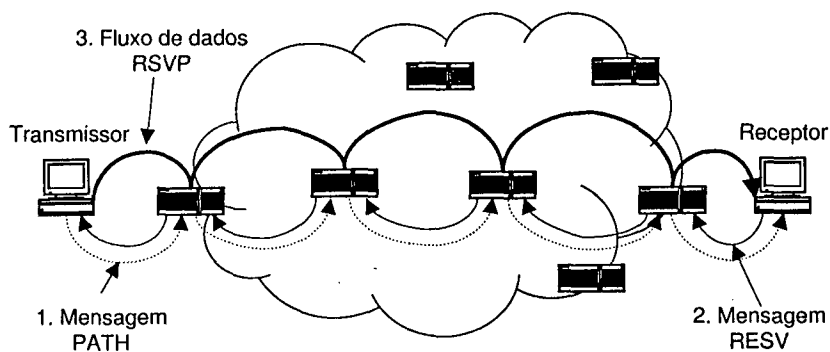


Figura 2.2: Sinalização RSVP

### 2.1.5 Problemas com IntServ

O modelo de serviço formado por IntServ e RSVP atualmente é considerado como não sendo escalável para ser utilizado em uma rede de grande dimensão, como a Internet pública. Os principais problemas são (Wroclawski 1997):

- A quantidade de informação de estado cresce proporcionalmente ao número de fluxos que um roteador tem que tratar. Isso impõe uma grande sobrecarga aos roteadores, em termos de capacidade de armazenamento e processamento considerando que roteadores de núcleo na Internet tratam simultaneamente milhares de fluxos.

- Para cada fluxo deve haver sinalização a cada nó (sistema final ou roteador). A troca de informações de sinalização é muito grande, inclusive porque o RSVP trabalha com estado leve (*soft-state*), ou seja, tem um tempo máximo de validade depois do qual ele expira. Isso prejudica a escalabilidade.
- As exigências para os roteadores são bastante altas. Todos têm que implementar RSVP, classificação, controle de admissão e escalonamento de pacotes.

## 2.2 Serviços Diferenciados

Baseado nas limitações encontradas no modelo Serviços Integrados, foi proposta a Arquitetura de Serviços Diferenciados (DiffServ) (Blake et al. 1998) que oferece QoS na Internet com escalabilidade: sem estado para cada fluxo e sinalização a cada nó. O tratamento oferecido por um nó compatível com essa abordagem, é efetuado sobre uma agregação de fluxos. Não mais para um fluxo individual. Os pacotes que não pertencem a nenhum fluxo contratado recebem o serviço padrão, o qual é equivalente ao serviço de melhor esforço, atualmente oferecido pela Internet. A classificação e marcação de pacotes ocorre nas bordas de um domínio de Serviço Diferenciados. Um Domínio de Serviços Diferenciados (DS) é composto por um grupo de nós contíguos que operam um conjunto comum de serviços. É válido lembrar que essa arquitetura traz as operações mais complexas para as bordas do domínio DS. Assim, temos um aumento na escalabilidade, uma vez que no interior do domínio ficam os mecanismos mais simples. Em síntese, podemos dizer que há um estado por fluxo nas bordas do domínio e um estado por classe de serviço nos nós internos ao domínio.

A abordagem Serviços Diferenciados também possui alguns problemas. Dentre eles, podemos citar a baixa granulosidade (não possui um bom mecanismo de distinção frente à diversidade de fluxos presentes) e o número limitado de classes. Uma descrição mais detalhada sobre essa abordagem se encontra no capítulo 3.

O encaminhamento das agregações é feito segundo uma política de Comportamento por Nó (*Per-Hop Behavior*, PHB). O resultado de uma transmissão, isto é, o serviço fim a fim é conseguido através da combinação dos PHBs entre os domínios ao longo do caminho. Uma questão importante a ressaltar é granulosidade de fluxos, ou seja, a distinção adotada por alguns dos modelos (IntServ, DiffServ ou mesmo Melhor Esforço) frente à diversidade de fluxos presentes. O serviço de melhor esforço tem granulosidade nula, uma vez que não há distinção entre os fluxos presentes. Já o modelo DiffServ, possui uma granulosidade média enquanto que IntServ é dotado de máxima granulosidade. A Figura 2.3 faz justamente essa comparação.

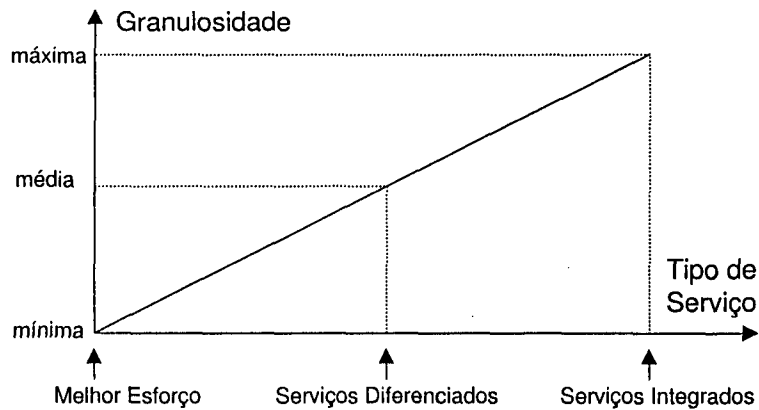


Figura 2.3: Comparação qualitativa entre modelos de serviço

## 2.3 Multiprotocol Label Switching (MPLS)

Na Internet, quando um roteador recebe um pacote, ele faz uma busca na sua tabela de roteamento e então, baseado no endereço IP do pacote, decide para onde enviá-lo. Essa busca pode levar bastante tempo, dependendo do tamanho da tabela de cada roteador. O MPLS (Faucher e et al. 2000) rompe com esse paradigma, usando um rótulo de tamanho fixo a partir do qual o roteador decide por onde enviar os pacotes. Neste caso, a busca na tabela de roteamento é feita apenas para os primeiros pacotes de um determinado fluxo. MPLS é na realidade a padronização de várias implementações existentes no mercado da técnica de encaminhamento baseado em rótulos (*label*). Essa forma de encaminhamento proporciona algumas vantagens em relação a maneira tradicional, como: 1) melhor desempenho no encaminhamento de pacotes; 2) criação de caminhos entre roteadores; e 3) possibilidade de associar requisitos de QoS baseados no rótulo.

MPLS é um esquema de encaminhamento que opera entre as camadas de rede (camada 3) e de enlace de dados (camada 2) do modelo RM-OSI/ISO. Cada pacote MPLS tem um cabeçalho específico, que contém um rótulo de 20 bits, um campo indicando a Classe de Serviço de 3 bits, um indicador de pilha de rótulos de 1 bit e um campo TTL (*Time de Live*) de 8 bits. O cabeçalho MPLS é encapsulado entre o cabeçalho de camada de rede e o cabeçalho de camada de enlace de dados, ou então sobreposto em algum campo específico da camada de enlace de dados, como o campo VPI/VCI do ATM. Um roteador MPLS, chamado de LSR (*Label Switching Router*), examina somente o rótulo para encaminhar os pacotes. MPLS é chamado de multiprotocolo porque qualquer protocolo de rede pode ser utilizado, embora esteja padronizado por enquanto apenas o protocolo IP. Os caminhos que os pacotes percorrem de um roteador a outro são chamados de LSPs (*Label Switching Paths*). Na entrada de um domínio MPLS, o roteador insere o cabeçalho nos pacotes, que são encaminhados através dele até um roteador de saída, que remove os cabeçalho e encaminha o pacote para o próximo domínio (que pode ou não ser um domínio MPLS).

O encaminhamento baseado em rótulos realizado pelo MPLS utiliza dois tipos de tabelas que fazem mapeamentos distintos, conforme ilustrado na Figura 2.4. Em ambas, cada entrada é chamada de NHLFE (*Next Hop Label Forwarding Entry*) e contém as seguintes informações:

- O roteador que é o próximo salto do pacote.
- Uma operação para ser realizada na pilha de rótulos do pacote, que pode ser no seu caso mais simples a troca do rótulo antigo pelo rótulo novo. Outras ações são o empilhamento de um novo rótulo (para construir um túnel) e a retirada do rótulo (realizada na saída da rede).
- Outras informações relevantes, como os níveis de QoS dispensados ao pacote.

Quando um roteador recebe um pacote que está sem um rótulo, ele utiliza a tabela FTN (FEC-to-NHLFE), para encaminhar cada FEC (*Forwarding Equivalence Classes*) para uma NHLFE. Quando o pacote chega no roteador com um rótulo, ele utiliza a ILM (*Incoming Label Map*), que encontra a NHLFE usando o rótulo como índice.

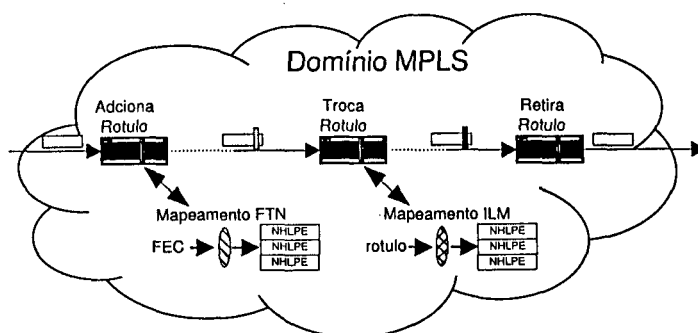


Figura 2.4: Encaminhamento de pacotes em um domínio MPLS

O processo de usar a FTN e a ILM para encaminhar pacotes é conhecido como "Troca de rótulos" (*label swapping*). Dependendo se o pacote vem rotulado ou não ele usa ou a ILM ou a FTN para fazer o mapeamento para uma NHLFE, mesmo que ele tenha a capacidade de realizar o encaminhamento IP convencional.

Sem dúvida, uma vantagem direta da utilização de MPLS é substituir o encaminhamento tradicional IP pelo encaminhamento baseado em rótulo, que é um processo consideravelmente mais rápido. Entretanto, somente isso não seria um motivo suficiente para a adoção de MPLS, uma vez que a tecnologia de roteadores de alta velocidade atual já permite fazer buscas muito rápidas na tabela. Uma das principais aplicações para MPLS hoje é a Engenharia de Tráfego, que será discutida mais adiante. Ela permite alterar o caminho normal que alguns pacotes seguiriam caso fossem encaminhados pelo esquema convencional, ou seja, pelo caminho mais curto, escolhido pelo protocolo de roteamento. O MPLS consegue forçar pacotes a seguirem



certas rotas preestabelecidas, o que é impossível no esquema convencional. Outra aplicação para essa abordagem é a emulação de redes orientadas a conexão, como *Frame Relay* ou ATM. Uma vez que MPLS é um mecanismo orientado a conexão, ele pode ser utilizado para emular qualquer serviço orientado a conexão não confiável com um LSP. Isso permite que uma rede integrada baseada em *datagramas* ofereça serviços ligados aos seus usuários usando uma única infra-estrutura. MPLS pode ser utilizado também para facilitar a integração de IP com ATM. A interligação de roteadores através de uma sub-rede ATM requer a configuração de vários circuitos virtuais, no pior caso  $N(N-1)/2$  circuitos virtuais, onde  $N$  é o número de roteadores. Usando roteadores MPLS e fazendo uma atualização dos *switches* ATM permite fácil integração entre ambas as tecnologias. *Switches* ATM podem se comportar como LSRs, fazendo o encaminhamento baseado em rótulos.

## 2.4 Roteamento baseado em QoS (QoSR)

O roteamento utilizado atualmente na Internet é direcionado para conectividade e tipicamente suporta somente o serviço de melhor esforço. Os protocolos atuais de roteamento usados na Internet, como OSPF e RIP, procuram sempre encontrar o menor caminho, baseados em uma única métrica, como peso administrativo ou quantidade de saltos. Esses protocolos são “oportunistas”, sempre procuram encontrar o menor caminho, mesmo quando ele não é o mais adequado, ou a troca freqüente de um caminho para outro pode gerar instabilidade. Caminhos alternativos com custos aceitáveis, mas não ótimos, não podem ser utilizados para rotear tráfego. No máximo, alguns protocolos aceitam manter rotas com custos exatamente iguais e distribuir o tráfego entre elas. O roteamento baseado em QoS (*QoS Routing*) (Crawley e et al. 1998) é um mecanismo de roteamento que seleciona o caminho percorrido pelos pacotes de um fluxo baseado no conhecimento da disponibilidade de recursos da rede, bem como nos requisitos de QoS dos fluxos, como largura de banda e atraso.

O QoSR pode ser considerado uma variação de Roteamento Baseado em Restrições (*Constraint-Based Routing*), onde as restrições são requisitos de QoS. Outras restrições utilizadas podem ser custo monetário e políticas de segurança. Uma vez que o interesse desse trabalho está em QoS, a discussão será direcionada especificamente para QoSR.

O QoSR deverá estender o paradigma de roteamento na Internet de três maneiras básicas. Em primeiro lugar, para suportar tráfego baseado em novos serviços, proporcionados por Int-Serv ou DiffServ, caminhos múltiplos entre roteadores devem ser encontrados, cada um com as suas características de QoS. Alguns desses serviços irão necessitar da distribuição de mais do que uma métrica, como largura de banda e atraso. Em segundo lugar, o roteamento oportunístico utilizado atualmente na Internet direciona o tráfego para uma nova rota tão logo que

um caminho "melhor" seja encontrado. O tráfego é redirecionado do caminho antigo, mesmo que ele esteja apto a satisfazer as necessidades dos fluxos. Além de gerar grande instabilidade, aumento da variação do atraso dos fluxos, as escolhas de roteamento nem sempre são adequadas. Em terceiro lugar, caminhos alternativos não são suportados, mesmo que possam atender às características de várias aplicações. Por isso, QoS pode encontrar um caminho mais longo, mas muito menos sobrecarregado que o caminho mais curto, que geralmente é o mais congestionado. O tráfego na rede, dessa forma, por ser distribuído mais igualmente.

Vale lembrar também que existe uma diferença entre QoS e reserva de recursos. Protocolos de reserva de recursos, como o RSVP, oferecem um método para requisitar e reservar recursos da rede, mas eles não proporcionam nenhum mecanismo para encontrar um caminho que tenha recursos suficientes para satisfazer os níveis de QoS requisitados. Por outro lado, QoS permite a determinação de um caminho com uma grande chance de acomodar a requisição de QoS, mas não inclui um mecanismo para reservar os recursos necessários. Na prática, as duas técnicas são complementares e geralmente implementadas em conjunto. Essa combinação permite exercer um controle significativo sobre rotas e recursos, ao custo de informações adicionais de estado e tempo de configuração. Por exemplo, um protocolo como o RSVP pode ser utilizado para disparar certas computações pelo mecanismo QoS que irão atender às necessidades específicas do fluxo.

Uma questão a se pensar também é o custo. Calcular rotas com base em restrições de QoS é um processo mais caro que o atual paradigma de roteamento da Internet. A questão é saber o ponto ótimo onde a melhoria em desempenho pela adoção de QoS vale a pena comparado com o aumento nos custos. O custo adicional gerado pelo QoS tem dois componentes principais (Apostolopoulos e et al. 1998): custo computacional e sobrecarga de protocolo.

O custo computacional é devido ao algoritmo mais sofisticado para escolha dos caminhos e à necessidade de processá-lo mais frequentemente. Dependendo da quantidade e do tipo de métricas que são utilizadas para o cálculo das rotas, o algoritmo pode se tornar NP completo (Wroclawski 1997). Em geral, existem soluções polinomiais que envolvem o uso de largura de banda e alguma outra métrica (atraso, variação do atraso, confiabilidade), mas assim mesmo o processamento é complexo. No entanto, os avanços tecnológicos permitem que se diminua a importância do custo computacional pela utilização de processadores mais rápidos e memórias maiores. Já a sobrecarga de protocolo ocorre devido à necessidade de distribuir atualizações sobre o estado dos recursos da rede entre os roteadores envolvidos no processamento das rotas. Essas atualizações se traduzem em aumento do tráfego e processamento na rede. Conforme a quantidade de informações transmitidas e a frequência com que isso é feito, essa distribuição das atualizações pode influenciar negativamente vários outros aspectos, como largura de banda e espaço de armazenamento. Então, o custo adicional proveniente da sobrecarga de protocolo é mais difícil de ser tratado e portanto representa um empecilho maior para a plena utilização de

Esse tipo de cenário se manifesta mesmo quando existem caminhos alternativos com capacidade ociosa suficiente para encaminhar o tráfego excedente. Ou seja, a utilização dos protocolos de roteamento baseados no menor caminho, tende a degradar o desempenho observado pelos fluxos de dados, mesmo quando existem recursos suficientes para tratar todo o tráfego.

A Figura 2.5 ilustra o tratamento que a Engenharia de Tráfego (ET) pode dar a certos tipos de tráfego, encaminhando pacotes por caminhos diferentes, dependendo das políticas adotadas. Em um domínio que não implementa ET, todos os pacotes devem ser encaminhados pelo caminho mais curto, no caso, o caminho do meio.

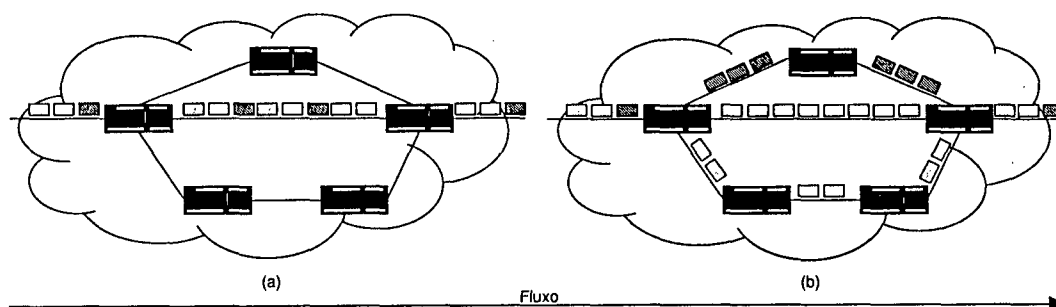


Figura 2.5: Encaminhamento de pacotes a) sem ET; b) com ET

## 2.6 Conclusões do capítulo

Foram apresentadas neste capítulo as principais abordagens que estão sendo discutidas atualmente para a provisão de Qualidade de Serviço (QoS) na Internet.

Como apresentado, Serviços Integrados e Serviços Diferenciados representam um rompimento com o modelo tradicional de melhor esforço utilizado na Internet, para oferecer serviços com garantias de QoS. IntServ é baseado em reserva de recursos e pode oferecer garantias rígidas a determinados fluxos de dados. O empecilho para sua utilização em larga escala para a construção de serviços fim a fim na Internet é sua falta de escalabilidade. DiffServ é uma arquitetura considerada escalável e, que oferece garantias para agregações de fluxos. Atualmente é considerada a abordagem que mais provavelmente será utilizada para QoS na Internet. Por isso, foi adotada para a implementação de nosso ambiente de teste.

Já MPLS é uma técnica de encaminhamento de pacotes, não necessariamente direcionada para o provimento de QoS. Pode ser utilizada para construir caminhos escolhidos explicitamente, que têm recursos suficientes para sustentar as necessidades de desempenho de certos fluxos de dados. Além disso, os pacotes que trafegam por esses caminhos podem receber tratamento diferenciado dentro dos roteadores. MPLS representa uma evolução com o mecanismo tradicional de encaminhamento de pacotes da Internet, chamado de salto a salto. (hop-by-hop).

Apresentou-se também o Roteamento baseado em Qualidade de Serviço (QoSR). Falou-se que esta é uma técnica de roteamento que encontra caminhos que atendem às necessidades de QoS de determinados fluxos. Não é uma técnica de oferecimento de QoS, apenas indica qual a rota mais adequada para que os níveis de QoS possam ser mantidos. Sem algum mecanismo de provimento de QoS, como IntServ ou DiffServ, é possível que, quando os pacotes forem roteados por um caminho escolhido por QoSR, ele já não atenda mais às suas necessidades. Além disso, ele não é um mecanismo de encaminhamento, ou seja, pode descobrir novas rotas, mas não tem como forçar pacotes de certos fluxos a seguirem obrigatoriamente essas rotas. Portanto, sua utilização faz mais sentido quando se modifica o mecanismo de encaminhamento básico, por exemplo, com MPLS.

E por fim, investiu-se resumidamente também sobre a Engenharia de Tráfego(ET). Vimos que a ET não é uma técnica específica, mas um processo de gerenciar o tráfego na Internet. Pode ser realizada manualmente, usando mecanismos das tecnologias de transmissão para desviar tráfego por caminhos alternativos (como por exemplo, usando caminhos virtuais ATM), ou utilizando alguma técnica automatizada que tem conhecimento das informações da camada IP da Internet, como as apresentadas anteriormente. Ou seja, dados os seus objetivos de desempenho, ET pode ser considerada como um processo que necessita de ferramentas as quais podem ser baseadas em IntServ/RSVP, DiffServ, MPLS e QoSR.

# Capítulo 3

## Serviços Diferenciados na Internet

A ARQUITETURA de Serviços Diferenciados foi desenvolvida pelo Grupo de Trabalho Diff-Serv (*Differentiated Services*) do IETF (*Internet Engineering Task Force*) (Group 2001), e tem como meta a provisão de QoS na Internet de forma escalável. Não há necessidade de se manter estado por fluxo nos roteadores, uma vez que existe uma fase de inicialização da conexão de uma forma explícita. Neste capítulo abordamos os conceitos introduzidos por essa proposta e descrevemos os mecanismos em discussão para implementá-la.

### 3.1 Serviços Diferenciados

A arquitetura de Serviços Diferenciados (*Differentiated Services - DS*) (Blake et al. 1998) busca o fornecimento de uma gama de serviços de forma escalável na Internet. Um serviço pode ser caracterizado por alguns aspectos significativos relacionados à transmissão de pacotes pela rede. Essas características definem as classes de serviço de acordo com algum parâmetro de desempenho, como vazão, atraso, *jitter* ou perda de pacotes. A arquitetura Serviços Diferenciados possibilita o atendimento aos requisitos de QoS de aplicações heterogêneas e de diferentes expectativas de usuários, além de permitir uma tarifação diferenciada para serviços providos na Internet. Os Serviços Diferenciados são oferecidos no interior de um domínio de Serviços Diferenciados (domínio DS). Um domínio como esse é composto por um conjunto de nós compatíveis com a proposta de Serviços Diferenciados que compartilham uma mesma política de provisão de serviços. A Figura 3.1 representa um domínio DS. No entorno do domínio DS estão localizados os nós de fronteira 1, 4 e 5. Nesses nós, os fluxos são agregados e estas agregações são encaminhadas pelo nós interiores 2 e 3. Esta arquitetura somente provê a diferenciação de serviços em uma única direção do fluxo de tráfego, sendo portanto assimétrica. Os nós de fronteira assumem, dependendo do sentido do tráfego tratado, o papel de nós de ingresso ou

egresso. Na ilustração da Figura 3.1, o nó 1 representa um roteador de ingresso e o nó 4 um roteador de egresso.

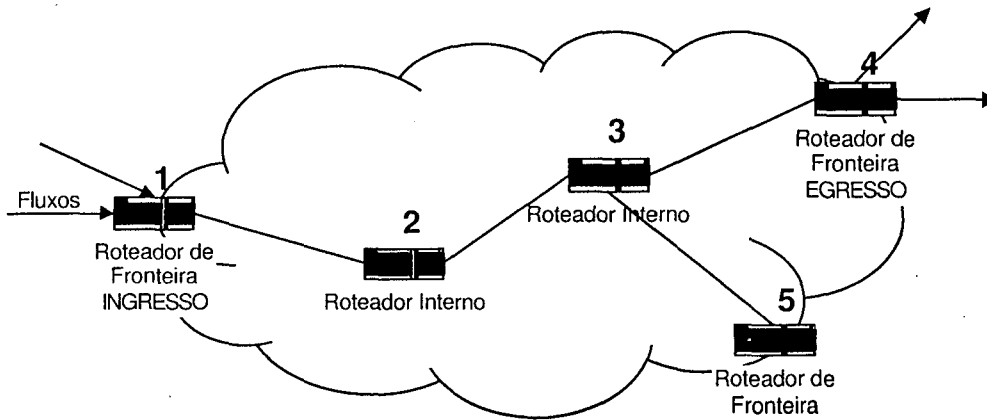


Figura 3.1: Domínio de Serviços Diferenciados - DS

### 3.1.1 Marcação do campo DS

A identificação das agregações de fluxos no interior de um domínio de Serviços Diferenciados é efetuada através da marcação de um novo campo chamado DS (*Differentiated Services*). O campo DS é obtido pela renomeação do campo TOS (*Type of Service*), no caso de IPv4, ou do campo *Traffic Class*, no caso de IPv6 (Nichols et al. 1998). A estrutura do campo DS é mostrada na Figura 3.2. Seis bits do campo DS formam o subcampo DSCP (*Differentiated Services Codepoint*), que identifica a agregação de fluxos. Os dois outros bits estão reservados para uso futuro. Em cada roteador compatível com a proposta de Serviços Diferenciados, o código (*Codepoint*) contido no subcampo DSCP é mapeado em um Comportamento por Nó (*Per-Hop Behavior - PHB*), que define o tratamento a ser recebido pelo pacote naquele roteador para o seu encaminhamento na rede.

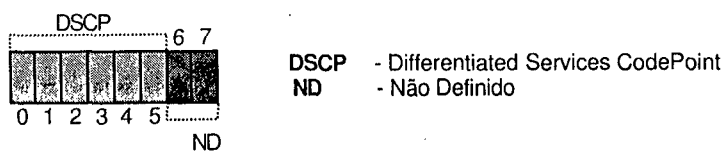


Figura 3.2: Estrutura do campo DS

Os roteadores de ingresso ao domínio de Serviços Diferenciados devem possuir informações que identifiquem um fluxo de dados individualmente. Desta forma, os pacotes pertencentes a um fluxo podem ser classificados e marcados com o código correspondente a uma determinada agregação de fluxos. Os classificadores que examinam a combinação de um ou mais campos do cabeçalho do pacote para a identificação de um fluxo, para que possam indicar a sua marcação como pertencentes a uma determinada agregação, são chamados de classificadores Multicampo

(*Multifield* - MF). Esses campos podem ser o endereço de origem, o endereço de destino, o número das portas na origem e no destino, o identificador de protocolo ou o campo DS. Os classificadores deste tipo encontram-se nos nós de ingresso de um domínio DS ou junto às fontes de tráfego.

No interior do domínio DS, a classificação dos pacotes é realizada somente pelo exame do subcampo DSCP. Os classificadores que realizam essa tarefa são chamados de classificadores de Comportamento Agregado (*Behavior Aggregate* - BA). A cada nó do domínio DS é associada uma tabela de mapeamento de fluxos para códigos, no caso de classificadores MF, ou de códigos para novos códigos, no caso de classificadores BA. Portanto, os classificadores BA permitem a remarcação do código associado aos pacotes. Resumidamente, como os códigos identificam as agregações, há um estado por fluxo nos nós de ingresso e um estado por agregação nos demais nós pertencentes ao domínio DS. O objetivo da agregação dentro do domínio DS é aumentar a escalabilidade pela manutenção de um menor número de estados.

### 3.1.2 Condicionamento de Tráfego

Além das funções de classificação de pacotes, a arquitetura DS prevê a adoção de um conjunto de elementos funcionais implementados em seus nós. Esses elementos são responsáveis pelo condicionamento do tráfego a ser atendido pela arquitetura DS. A política de condicionamento de tráfego integra o Acordo de Condicionamento de Tráfego (*Traffic Conditioning Agreement* - TCA) junto com as regras de classificação adotadas pelo domínio DS. O TCA também estabelece o perfil de tráfego contratado, que especifica as características do tráfego selecionado pelo classificador para compor uma agregação de fluxos.

Os elementos funcionais de condicionamento de tráfego incluem medidores, marcadores, suavizadores e policiadores. Os medidores são responsáveis por verificar a conformidade do tráfego ao perfil de tráfego contratado estabelecido no TCA. O seu resultado influencia as ações dos demais elementos. O marcador estabelece o código no campo DSCP do pacote, acrescentando este pacote a uma determinada agregação. Quando um marcador altera o código de um pacote, é dito que o pacote foi remarcado. O suavizador de tráfego retém um ou mais pacotes até que estes estejam em conformidade com o perfil contratado e possam ser encaminhados na rede. O *buffer* utilizado por um suavizador possui tamanho limitado. Portanto, eventualmente, pacotes podem ser descartados quando o tamanho do *buffer* é insuficiente. O policiador descarta os pacotes que excedem o perfil contratado e equivale a um suavizador cujo tamanho do *buffer* seja igual a zero. O relacionamento entre estes elementos está ilustrado na Figura 3.3. Os pacotes fora de perfil podem ser tratados de formas distintas. Eles podem ser descartados, suavizados ou remarcados para uma outra agregação que possua um serviço inferior. Essa decisão deve estar especificada no TCA.

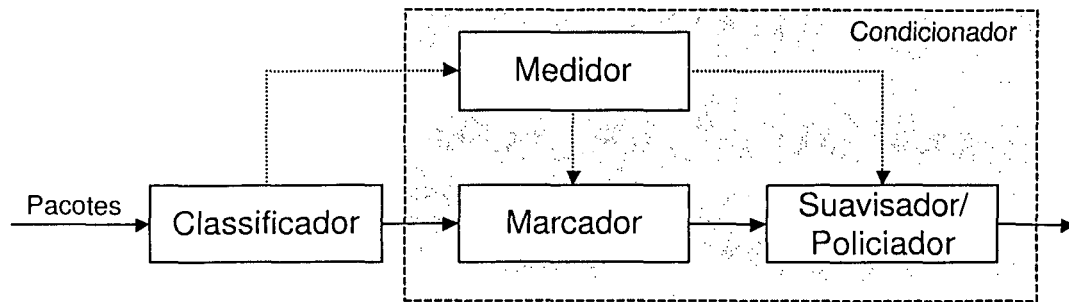


Figura 3.3: Funções de Condicionamento de Tráfego

As funções de condicionamento de tráfego, normalmente, estão localizadas nos nós de ingresso e egresso de um domínio DS. Contudo, essas funções podem também estar em nós interiores ao domínio DS, ou mesmo em nós não-DS. Nesses casos, os condicionadores podem estar no domínio da fonte de tráfego. Desta forma, essa fonte pode realizar uma marcação inicial ou pré-marcação em seus pacotes antes destes alcançarem o domínio DS. Essa alternativa possui algumas vantagens. A marcação torna-se mais simples, pois as regras de classificação são reduzidas. As aplicações que geram o tráfego possuem melhores condições de marcar adequadamente os pacotes de seus fluxos de acordo com as próprias características destes. Por desconhecer as características do tráfego, um mecanismo de condicionamento no roteador de ingresso pode marcar pacotes essenciais à qualidade do tráfego como fora do perfil e comprometer o resultado da transmissão. Assim, as fontes, cientes do conteúdo de seus fluxos de tráfego, podem pré-marcá-los e evitar a marcação indiscriminada dos roteadores de fronteira. Deve-se ressaltar que o condicionamento nas fontes não dispensa a monitoração do tráfego na fronteira do domínio DS. O serviço de encaminhamento oferecido a um cliente por um provedor de serviços deve estar estabelecido em um Acordo de Nível de Serviço (*Service Level Agreement - SLA*). Um cliente pode ser um indivíduo ou uma organização. O SLA especifica os detalhes do TCA e define o conjunto de critérios mensuráveis a serem utilizados para validar o recebimento do serviço contratado. Os SLAs tornaram-se necessários com o grande custo decorrente de falhas ou queda de desempenho na rede para as novas empresas que dependem dela para seus negócios. Dessa forma, um SLA estabelecido entre um cliente e um mantenedor de um domínio DS pode determinar que o atraso fim-a-fim máximo neste domínio para 99% dos pacotes oriundos deste cliente estarão limitados a um certo valor. Caso o mantenedor não cumpra essa expectativa de serviço, meios de compensação, também definidos no SLA, podem ser acionados pelo cliente. No caso de Serviços Diferenciados, o serviço de encaminhamento recebido é determinado pela seqüência de PHBs que atenderam a agregação em sua trajetória através do domínio.



### 3.1.3 PHB para implementação de Serviços Diferenciados

Atualmente, existem duas propostas de PHB (*Per-Hop Behavior*) para a implementação de Serviços Diferenciados: o Encaminhamento Expresso (*Expedited Forwarding* - EF) (Jacobson et al. 1999) e o Encaminhamento Assegurado (*Assured Forwarding* - AF) (Heinanen et al. 1999). O primeiro oferece a garantia de uma taxa mínima de transmissão pré-configurada em cada nó compatível com a proposta de Serviços Diferenciados que o suporte. Assim, o PHB EF pode ser utilizado para a obtenção de um serviço fim-a-fim com baixa perda, baixo retardo, baixa variação do retardo (*jitter*) e banda passante assegurada através de um domínio DS. Entre os pontos finais de um domínio DS, o Serviço de Encaminhamento Expresso apresenta-se como um caminho virtual dedicado que possui a taxa de transmissão pré-configurada. O PHB de Encaminhamento Assegurado provê quatro classes, cada uma com uma certa quantidade de recursos (memória e banda passante) alocada. Em cada classe, os pacotes podem ser marcados com uma de três preferências para descarte. Havendo congestionamento, a preferência de descarte determina a importância relativa entre os pacotes de uma mesma classe. Não há o reordenamento de pacotes de um mesmo fluxo quando estes pertencem a uma mesma classe.

## 3.2 Encaminhamento Expresso

O PHB de Encaminhamento Expresso (*Expedited Forwarding* - EF) (Jacobson et al. 1999) tem como objetivo principal definir garantias mais rígidas de QoS para aplicações muito sensíveis a variações de características temporais da rede. Ele pode ser utilizado para implementar um serviço com pouco atraso, pouca variação do atraso (*jitter*) e largura de banda garantida. Para os usuários, esse serviço, conhecido como Premium, parece com uma Linha Privativa Virtual. Para esse tipo de serviço, os pacotes que não estiverem dentro do perfil contratado, são descartados diretamente, não passando por uma reclassificação (Nichols et al. 1999) (Bennet et al. 2001) (Jacobson et al. 1999). O PHB EF é a versão de DiffServ para encaminhar tráfego de aplicações multimídia e de tempo real em geral.

As filas nas quais um pacote em trânsito permanece ao longo de sua rota pelos nós da rede são as maiores responsáveis pela perda, pelo retardo e pelo *jitter* apresentados por este pacote. O PHB de Encaminhamento Expresso objetiva diminuir a permanência dos pacotes pertencentes a uma determinada agregação de fluxos em filas no interior de um domínio DS. Para alcançar essa meta, o nó que oferece o PHB EF deve garantir que a agregação de fluxos contratante receba a taxa de serviço contratada, que deve ser maior que a taxa de chegada em qualquer instante. Essa garantia deve prevalecer independentemente da intensidade de outros tráfegos que cheguem ao nó.

Os roteadores de ingresso do domínio DS devem condicionar o tráfego EF de forma a assegurar que a sua taxa de chegada aos nós interiores esteja em conformidade com a taxa contratada. Essa medida protege o domínio DS de um uso abusivo e obriga as fontes de tráfego EF a suavizar o tráfego transmitido se desejarem evitar o condicionamento de seus pacotes no ingresso do domínio DS. Na realidade, conforme as agregações EF provenientes de diferentes roteadores de ingresso compõem novas agregações nos roteadores interiores ao domínio DS, estas novas agregações também precisam ser condicionadas para que a definição do PHB EF seja válida a qualquer momento por todo o trajeto. Portanto, a junção dessas agregações e também a diferença de capacidades entre os enlaces do interior de um domínio DS podem obrigar que um recondicionamento seja realizado em pontos no interior deste domínio. A implementação deste PHB é ilustrada na Figura 3.4.

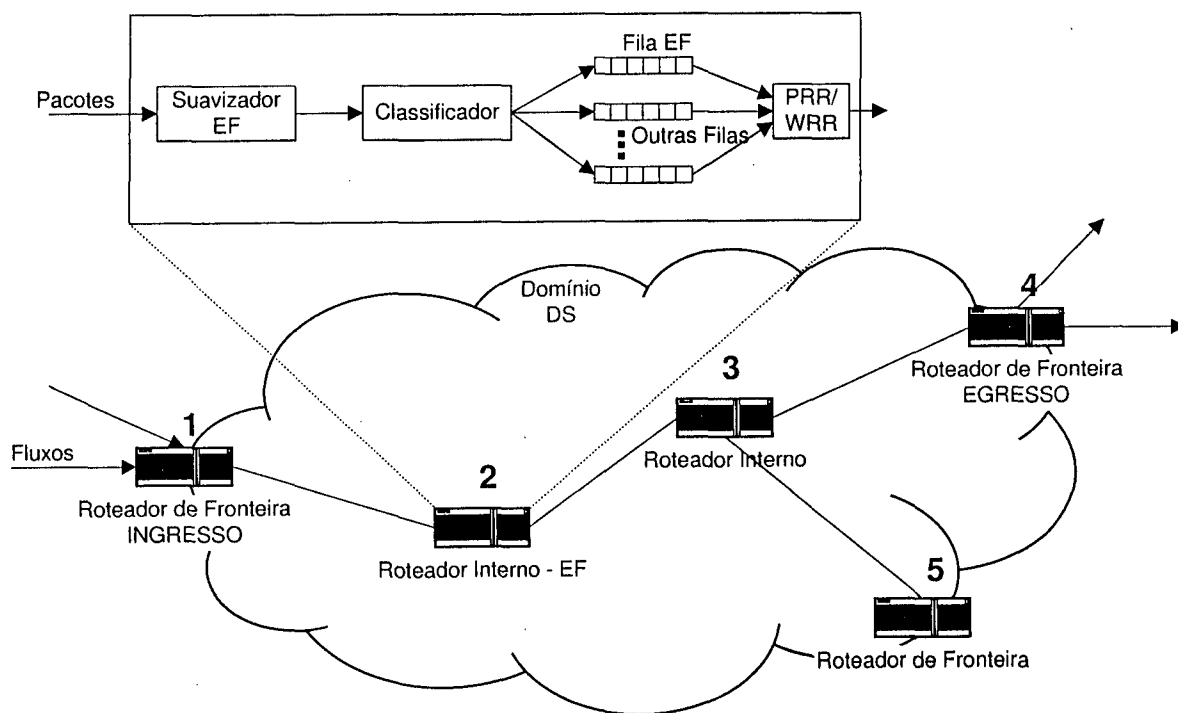


Figura 3.4: Implementação do PHB de Encaminhamento Expresso

A implementação do PHB EF pode ser realizada por diferentes mecanismos de escalonamento de filas. Um mecanismo com uma fila prioritária (*Priority Round Robin - PRR*) fornece o comportamento desejado. Nesse caso, torna-se especialmente importante o condicionamento de tráfego nas fronteiras do domínio DS. A ausência de tal condicionamento permitiria, em uma situação de abuso da agregação EF em relação à taxa de transmissão, a utilização de recursos não contratados em detrimento de tráfegos presentes nas demais filas. Outra possibilidade de implementação é a utilização de uma fila entre um conjunto de filas servidas por um mecanismo de *Round-Robin Ponderado (Weighted Round Robin - WRR)* (Kim 1993). A fatia de serviço alocada à fila que atende a agregação EF deve ser correspondente a, no mínimo, a taxa de serviço contratada.

Em (Jacobson et al. 1999), é realizada uma análise da variação do *jitter* apresentado por um tráfego CBR quando o PHB EF é implementado por PRR ou por WRR. A definição de *jitter* adotada é o valor absoluto da diferença entre os tempos de chegada ( $a_i$  e  $a_j$ ) de dois pacotes consecutivos menos a diferença entre seus tempos de partida ( $d_i$  e  $d_j$ ), ou seja:

$$jitter = |(a_j - d_j) - (a_i - d_i)| \quad (3.1)$$

A conclusão demonstra que a implementação por fila prioritária é o melhor caso e a implementação por WRR o pior com relação ao *jitter*. Mostra-se ainda que se a fatia WRR configurada for escolhida para igualar exatamente as taxas de chegada e saída em cada nó, os resultados não são estáveis. Então, foi definida uma razão serviço-chegada (*service-to-arrival ratio*). A definição da razão serviço-chegada  $r$  é apresentada na equação 3.2. O valor  $WRR_{fatia}$  representa a fração da banda passante do enlace de saída ( $BW_{saida}$ ) alocada à agregação EF e  $TP_{EF}$  é a taxa de pico solicitada pela agregação EF para o seu trânsito pelo domínio.

$$r = \frac{(WRR_{fatia} BW_{saida})}{TP_{EF}} = \frac{BandaPassanteReservada}{BandaPassanteSolicitada} \quad (3.2)$$

Como as fontes ou o domínio DS adjacente devem condicionar o tráfego para a sua entrada no domínio DS seguinte, o domínio posterior não deveria descartar nenhum pacote. Caso haja o descarte de algum pacote no nó de ingresso, recomenda-se que este descarte seja notificado como uma possível violação de segurança ou o efeito de uma configuração mal realizada (Jacobson et al. 1999).

### 3.3 Encaminhamento Assegurado

O PHB de Encaminhamento Assegurado (*Assured Forwarding - AF*) definido em (Heinanen et al. 1999) pode ser utilizado por um Domínio de Serviços Diferenciados (DS) provedor para oferecer níveis diferentes de garantias de encaminhamento para pacotes recebidos de um Domínio DS Cliente.

Foram definidas quatro Classes AF. Dentro de cada classe AF, um pacote IP pode ser marcado, ou pelo próprio usuário ou pelo domínio DS, com até três níveis de precedência para descarte. No caso de congestionamento dentro de uma classe AF, a precedência de descarte de

um pacote determina a importância relativa daquele pacote. Um nó DS em situação de congestionamento preferencialmente descarta pacotes com um maior valor de precedência de descarte, ao mesmo tempo em que evita descartar pacotes com um valor de precedência de descarte menor. Na realidade, o Serviço Assegurado aloca 12 códigos do espaço de 64 possíveis códigos para mapeamento em PHBs do subcampo DSCP. Assim, o AF define um grupo de PHBs onde são definidas até 12 níveis de serviços AF distintos, distribuídos em 4 filas (classes) com 3 precedências de descarte cada. A configuração desses níveis a cada nó DS pode ser diferente desde que se mantenha coerente em relação às prioridades relativas entre as filas e precedências de descarte.

Esse PHB é indicado para aplicações que requerem uma melhor confiabilidade que a oferecida pelo serviço *Best Effort*. Esse modelo de serviço, ao invés de fornecer uma garantia estrita, fornece uma melhor qualidade de transmissão que será obtida por um determinado tráfego quando existirem momentos de congestionamento. Um congestionamento ocorre quando um roteador recebe uma carga de tráfego maior do que pode tratar, situação que o leva a armazenar temporariamente os pacotes que chegam em uma fila. Eventualmente, um roteador necessita descartar pacotes devido à escassez de recursos para armazená-los. O controle de admissão exerce papel importante neste modelo, pois ele é encarregado de garantir a existência dos recursos necessários (*buffers* e banda passante) durante todo o percurso da origem ao destino. Um perfil associado a cada tráfego define o serviço esperado por este. Assim, um mecanismo de condicionamento de tráfego atua nas fronteiras do domínio DS de forma a marcar os pacotes de acordo com a sua conformidade com esse perfil. Os roteadores de fronteira são responsáveis pela manutenção da granulosidade dos serviços providos à agregação de fluxos.

A garantia existente no modelo de Serviço Assegurado é que pacotes marcados como conformes são entregues com alta probabilidade enquanto o tráfego agregado não exceder a taxa contratada definida no perfil de serviço. No entanto, é permitido ao usuário exceder as taxas contratadas. Ao fazê-lo, contudo, o usuário deve estar consciente de que o tráfego excedente não terá a mesma alta probabilidade de entrega. Conforme os pacotes são transportados pela rede e agregados a outros fluxos, roteadores nas fronteiras de outros domínios somente condicionam a agregação de fluxos.

O PHB AF permite ao mantenedor do domínio DS oferecer diferentes níveis de garantia no encaminhamento de pacotes IP para um usuário de seu domínio. O PHB AF provê a entrega de pacotes IP em uma de suas quatro classes independentes. Os pacotes IP devem pertencer a uma das classes AF de acordo com o contrato de serviço estabelecido de forma a usar os serviços oferecidos pelo PHB AF. A implementação das classes AF nos nós do domínio DS está ilustrada na Figura 3.5.

O nível de garantia para o encaminhamento depende da quantidade de recursos alocados para a classe AF, da carga atual de cada classe e em caso de congestionamento, do valor de

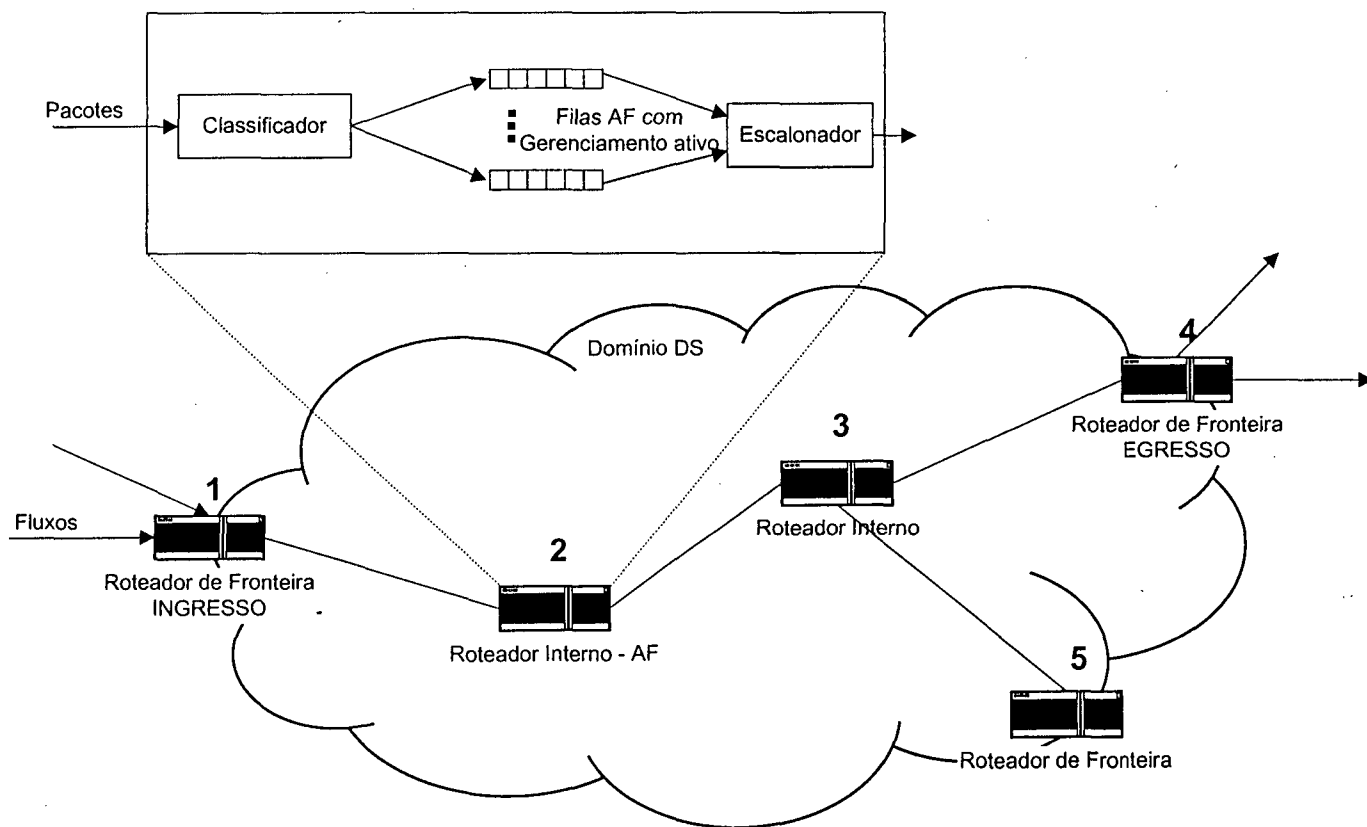


Figura 3.5: Implementação do PHB de Encaminhamento Assegurado

precedência de descarte do pacote IP. A implementação de um PHB AF procura minimizar congestionamentos de longa duração, porém permitindo congestionamentos de curta duração, como os resultantes de rajadas de tráfego (Rezende 1999b).

O PHB AF detecta e reage a congestionamentos de longa duração dentro de cada classe através do descarte de pacotes. Além disto, ele aceita em sua fila pacotes que causam congestionamento de curta duração. Para a realização desse procedimento é necessário um algoritmo para o gerenciamento ativo de filas.

### 3.4 Gerenciamento Ativo de Filas

Dentro de um Domínio de Serviços Diferenciados necessitamos de um gerenciamento de filas (classes) para a transmissão dos pacotes. Nessa seção, apresentamos alguns dos mecanismos de Gerencia de Filas utilizados atualmente.

Um mecanismo de gerenciamento ativo de filas deve conter uma função que monitore o nível de congestionamento instantâneo da fila e compute um nível de congestionamento suavizado. O algoritmo de descarte de pacotes utiliza o nível de congestionamento suavizado para determinar quando pacotes devem ser descartados.

Além disso, esse algoritmo deve ser insensível às características de fluxos de curta duração. Assim, fluxos com diferentes formas de rajadas curtas, mas com taxas de transmissão de pacotes de longo prazo idênticas, tem seus pacotes descartados basicamente com a mesma probabilidade. Esse comportamento pode ser obtido pela utilização de uma função aleatória para o descarte de pacotes. Um mecanismo de gerenciamento ativo de filas com essa característica é o RED (*Random Early Detection*) (Floyd e Jacobson 1993), que é o mecanismo recomendado como *default* para o gerenciamento ativo de filas na Internet (Branden et al. 1998).

### 3.4.1 RED

O mecanismo de controle de congestionamento do TCP permitiu à Internet a manutenção da oferta do serviço de melhor esforço mesmo em momentos de alta demanda. A idéia básica do mecanismo de controle de congestionamento do TCP é controlar a carga de tráfego da rede pela redução das taxas de transmissão das fontes de tráfego de acordo com o nível de congestionamento da rede. Ou seja, as fontes TCP devem colaborar e adaptar-se à intensidade do congestionamento da rede. Se uma fonte de tráfego TCP detecta ou observa uma perda de pacote baseada na informação dos pacotes que esta enviou, ela retrai sua taxa de transmissão para evitar perdas de mais pacotes e congestionamento. Portanto, a fonte de tráfego TCP utiliza a detecção da perda de pacotes como uma  *sinalização implícita de congestionamento*. Caso uma fonte de tráfego TCP verifique que todos os pacotes enviados estão sendo entregues pela rede, ela gradativamente aumenta a sua taxa de transmissão de forma a melhor utilizar a capacidade da rede. O crescimento vertiginoso na utilização da Internet que existe atualmente somente é possível devido à capacidade de adaptação das aplicações baseadas em TCP às condições de congestionamento da rede.

O mecanismo RED (*Random Early Detection*) (Floyd e Jacobson 1993) é implementado nos roteadores e projetado para atuar em conjunto com o protocolo TCP. Através do RED é realizada a monitoração do tamanho da fila FIFO (*First In, First Out*) no roteador e caso haja sinais de um congestionamento iminente, as fontes são notificadas de forma implícita pelo descarte de um de seus pacotes. Dessa maneira, a fonte é notificada com base no *timeout* correspondente ao pacote descartado. O TCP detecta possíveis congestionamentos através de *timeouts* ou por ACKs duplicados. Isso determina que a fonte deve ajustar a sua janela de congestionamento. Os roteadores RED descartam pacotes mais cedo do que os roteadores convencionais, então a fonte reduz sua janela de congestionamento mais cedo. Em outras palavras, o roteador RED descarta alguns pacotes, o que causa uma redução na taxa de transmissão de algumas fontes, com a esperança de que esta atitude evite o descarte de um número maior de pacotes posteriormente. Ele busca assim limitar um potencial congestionamento.

Quando um roteador RED utiliza uma fila FIFO, ao invés de esperar que a fila se torne com-

pletamente cheia, o que implica no descarte de todos os pacotes que chegam, ele decide descartar cada novo pacote com uma certa probabilidade sempre que o tamanho da fila exceda um determinado nível. O *Random Early* do nome do mecanismo vem dessa característica. Quando a fila não está vazia, o roteador computa o tamanho médio da fila (*avglen*) utilizando uma média ponderada por uma variável *qweight* de maneira similar àquela utilizada na computação dos temporizadores do TCP, como mostrado na equação 2.3 onde *curq* representa a ocupação instantânea da fila. A variável *qweight* determina a tolerância do roteador RED às rajadas de longa duração. Ou seja, quanto maior for o *qweight*, maior será o peso do tamanho instantâneo da fila no cálculo do novo valor do (*avglen*). Caso a fila esteja vazia, (*avglen*) é gradualmente diminuído em função do tempo de ociosidade da fila.

$$avglen = (1 - qweight)avglen + qweightcurq \quad (3.3)$$

O RED utiliza dois patamares para determinar a probabilidade de descarte do pacote recém-chegado à fila: *min\_thresh* e *max\_thresh*. Enquanto o tamanho médio da fila (*avglen*) permanecer abaixo de *min\_thresh*, todos os pacotes que chegam ao roteador são aceitos na fila. Quando *avglen* está entre os dois patamares, há uma probabilidade *p* do novo pacote ser descartado, onde *p* é uma função de *avglen* e do tempo decorrido desde o descarte do último pacote. Se *avglen* ultrapassa o limite *max\_thresh*, todos os próximos pacotes serão descartados até que a média de ocupação da fila seja reduzida. Devido à probabilidade do roteador RED descartar um pacote ser proporcional à parcela de ocupação na fila do roteador, quanto mais pacotes um fluxo enviar, maior será a probabilidade que um de seus pacotes seja descartado. A probabilidade de descarte correspondente a *max\_thresh* é determinada pelo parâmetro RED  $P_{max}$ . A agressividade de descarte de pacotes adotada pelo mecanismo RED é proporcional ao valor de  $P_{max}$ . O funcionamento do algoritmo RED em função dos parâmetros descritos é ilustrado pela Figura 3.6.

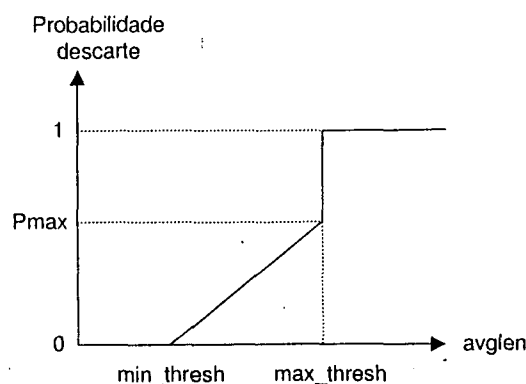


Figura 3.6: Parâmetros do algoritmo RED

A análise para escolha dos parâmetros deve considerar a caracterização da carga da rede que influencia diretamente o tamanho da fila. Por exemplo, se o tráfego possui muitas rajadas, *min\_thresh* deve ser suficientemente grande para manter uma alta utilização do enlace. Além disto, a diferença entre os dois limiares deve ser maior do que o incremento no tamanho médio da fila em um RTT. Caso contrário, não haveria tempo suficiente para as fontes detectarem o descarte com antecedência dos pacotes. Em (Rezende 1999a) é considerada como uma boa aproximação usar  $max\_thresh = 2 * min\_thresh$  dada a combinação de tráfego atualmente encontrada na Internet.

### 3.4.2 Extensões ao RED

O mecanismo RED para o gerenciamento ativo de filas apresenta algumas vantagens marcantes em relação à simples adoção de filas FIFO com a política de *Drop-Tail* nos roteadores, onde somente há descarte de pacotes quando a fila está completamente ocupada. Como já foi mencionado, as fontes TCP somente reduzem a sua taxa de transmissão após assumir que a rede está congestionada pela detecção da perda de um pacote. No entanto, um tempo considerável pode transcorrer desde o momento em que um pacote é descartado até o momento em que a fonte detecte este descarte. Como consequência, um grande número de pacotes pode ser descartado durante esse período pois as fontes estão enviando pacotes a uma taxa maior do que a rede pode suportar. Além disso, quando um pacote é descartado antes de alcançar o seu destino, todos os recursos que ele consumiu na sua transmissão desde a fonte até o ponto de descarte foram desperdiçados. O mecanismo RED procura reagir mais rapidamente ao detectar a iminência de um congestionamento e descarta alguns pacotes mais cedo evitando assim uma subutilização da rede. Com o transbordo de uma fila FIFO, todos os pacotes que chegarem à fila nos momentos seguintes serão imediatamente descartados. As fontes de todos esses pacotes se retraem ao detectar a perda de seus pacotes. Isto pode causar uma subutilização da rede. Esse problema é conhecido como sincronização global do TCP. Como o descarte antecipado no RED é aleatório, evita-se esse sincronismo na retração das fontes, melhorando a utilização da rede.

Apesar de ter sua contribuição de um melhor desempenho amplamente reconhecida, o RED também sofre algumas críticas e algumas extensões foram propostas para o seu aperfeiçoamento. Nesta subseção são apresentadas as motivações e as principais propostas de extensão ao RED atualmente em discussão.

#### 3.4.2.1 RED adaptativo

Para o funcionamento adequado do RED em redes congestionadas, a notificação de congestionamento deve ser enviada a um número mínimo de fontes de forma a reduzir suficientemente a



carga na rede e evitar a perda de pacotes devido ao transbordo da fila. A notificação de congestionamento para um número exagerado de fontes pode acarretar em uma subutilização da rede. O impacto na carga total da rede de uma indicação de congestionamento individual decresce conforme o número de conexões cresce. Caso o algoritmo RED não seja configurado para ser mais agressivo, a fila RED pode degenerar-se em uma simples fila *Drop-Tail*. Por outro lado, para um pequeno número de conexões, o impacto de uma notificação individual de congestionamento cresce. Neste caso, a menos que o algoritmo RED seja modificado de forma a ser menos agressivo, a rede pode ser subutilizada se fontes demais retraírem suas taxas de transmissão em resposta a um congestionamento iminente.

Uma configuração mais agressiva para o RED é necessária na presença de um grande número de fluxos para evitar uma maior perda de pacotes devido ao descarte determinístico quando *avglen* ultrapassa *max\_thresh* ou a fila FIFO está cheia. Por outro lado, uma menor agressividade quando poucos fluxos estão ativos previne a subutilização da rede. Assim, parâmetros adaptativos para o RED podem ser interessantes para o desempenho da rede. Esta é a motivação para a proposta do RED Adaptativo realizada em (Feng 1999).

O RED Adaptativo determina se o RED deve ser mais ou menos agressivo pela monitoração do comportamento do tamanho médio da fila (*avglen*). A Figura 3.7 ilustra o comportamento do RED Adaptativo. Caso *avglen* se mantenha no entorno de *min\_thresh*, o mecanismo está muito agressivo. Da mesma forma, se *avglen* continuamente cruza *max\_thresh*, o algoritmo não está suficientemente agressivo.

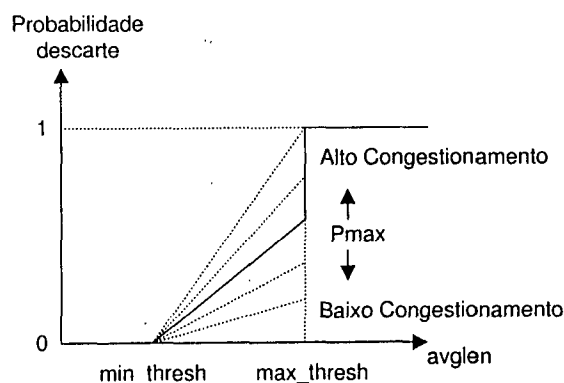


Figura 3.7: Comportamento do algoritmo RED Adaptativo

A partir do comportamento de *avglen*, o RED Adaptativo ajusta o parâmetro  $P_{max}$  adequadamente para tornar o algoritmo RED mais ou menos agressivo. Isso contrasta com o algoritmo RED original, apresentado na subseção 3.4.1, onde o parâmetro  $P_{max}$  permanece fixo. Durante um período de baixo congestionamento, a probabilidade de descarte se mantém bastante baixa até *avglen* alcançar *max\_thresh*. Em momentos de alto congestionamento, a probabilidade de descarte é aumentada rapidamente até que *avglen* seja inferior a *min\_thresh*. O algoritmo a seguir mostra o funcionamento do RED Adaptativo. Os fatores configuráveis  $\alpha$  e  $\beta$  controlam a

diminuição e o aumento de  $P_{max}$ , respectivamente.

```

if(min_thresh < avglen < max_thresh)then
    estado ← Intemediario
elseif(avglen < min_thresh ∧ estado <> abaixo)then
    estado ← abaixo
     $P_{max} \leftarrow P_{max}/\alpha$ 
elseif(avglen > max_thresh ∧ estado <> acima)then
    estado ← acima
     $P_{max} \leftarrow P_{max} \cdot \beta$ 
endif

```

### 3.4.2.2 FRED

Enquanto o RED Adaptativo procura uma melhor utilização da banda passante, o algoritmo FRED (*Flow Random Early Drop*) (Lin e Morris 1997) se propõe a fornecer uma divisão mais justa de banda passante entre os fluxos. O RED foi projetado considerando uma ação cooperativa entre as fontes de tráfego envolvidas. Ou seja, o algoritmo RED supõe que o descarte de um pacote indicará implicitamente à fonte a iminência de um congestionamento e ele espera que esta colabore reduzindo a sua taxa de transmissão. Isso é verdadeiro para fontes TCP que adaptam a sua taxa de transmissão às condições de congestionamento da rede. Fontes não adaptativas, como vídeo CBR, mantêm suas taxas de transmissão independentemente do nível de congestionamento da rede. Apesar do tráfego TCP formar a parcela predominante do tráfego presente na rede, diferentes versões e implementações do TCP reagem de forma distinta aos sinais de congestionamento. Inclusive, conexões TCP idênticas podem reagir de forma diferente a sinais de congestionamento semelhantes se possuírem tamanhos diferentes em suas janelas de congestionamento ou RTTs. Define-se que as conexões robustas possuem janelas de congestionamento maiores e RTTs menores do que as conexões frágeis. Em condições iguais, as conexões frágeis tendem a receber do RED fatias menores de banda passante do que as conexões robustas.

O descarte de pacotes realizado pelo algoritmo RED em proporção à banda passante dos fluxos pode levar a uma divisão injusta da banda passante entre os tráfegos envolvidos. Um fluxo pode ter um de seus pacotes descartado pelo algoritmo RED, mesmo que este fluxo não esteja colaborando para o aumento da probabilidade de descarte. A aceitação na fila de um pacote de uma conexão causa uma maior probabilidade de descarte para os pacotes pertencentes às demais conexões. Uma conexão não adaptativa pode fazer o RED aumentar a taxa de descarte de pacotes aplicada a todas as conexões.

O FRED armazena um estado por fluxo ativo, ou seja, fluxos que possuem pacotes armazenados na fila. Assim, o número de estados é limitado pela capacidade da fila, sendo o pior caso aquele onde cada fluxo possui somente um pacote na fila. O FRED também acrescenta alguns parâmetros próprios. Os valores  $min_q$  e  $max_q$  representam o número mínimo e máximo de pacotes que deve ser permitido a cada fluxo armazenar na fila. A variável  $avgcq$  estima a utilização média por fluxo da fila. O estado por fluxo do FRED armazena quantos pacotes cada fluxo possui na fila e uma variável *strike* para cada fluxo que conta o número de vezes que este fluxo não correspondeu à notificação de congestionamento. Através da monitoração destes parâmetros o FRED busca uma divisão mais justa da banda passante entre todos os fluxos envolvidos, sejam eles cooperantes ou não, robustos ou frágeis. Dessa forma, o FRED busca preservar a fatia justa para fluxos cooperantes na presença de tráfegos não cooperantes. Além disso, entre as conexões cooperantes, esse mecanismo preserva as conexões frágeis de uma divisão injusta de banda passante com as conexões robustas.

### 3.4.3 Generalização do RED

Nesse caso os pacotes em trânsito podem ser marcados como pertencentes a N classes distintas. Havendo pacotes de múltiplas classes, o tamanho médio da fila para cada classe pode ser calculado de maneira diferente ou ser adotado um tamanho médio da fila único considerando os pacotes de todas as classes. Para cada classe também é possível estabelecer um conjunto de parâmetros ( $min\_thresh$ ,  $max\_thresh$ ,  $P_{max}$ ) diferente (Kim et al. 1998). Assim, como proposto em (Goyal et al. 1999), qualquer política de RED adotada pode ser vista como uma variante de uma das quatro categorias genéricas seguintes:

- Única Média Único Limiar (Single Average Single Threshold - SAST);
- Única Média Múltiplos Limiares (Single Average Multiple Thresholds - SAMT);
- Múltiplas Médias Único Limiar (Multiple Average Single Threshold - MAST);
- Múltiplas Médias Múltiplos Limiares (Multiple Average Multiple Thresholds - MAMT).

A partir da generalização proposta, alguns exemplos podem ser citados. Um SAMT, onde há um número de conjuntos de parâmetros (limiares) adotados e apenas uma média calculada para todas as classes, pode ser exemplificado pelo WRED (*Weighted RED*) proposto pela Cisco (Systems 1997) e ilustrado na Figura 3.8 para o caso de três classes de serviço distintas. O algoritmo RED original é um exemplo para a categoria SAST.

O conceito de generalização do RED se aplica também às extensões do RED descritas. Essa generalização independe da variação de RED adotada. Por exemplo, pode-se ter um rotea-

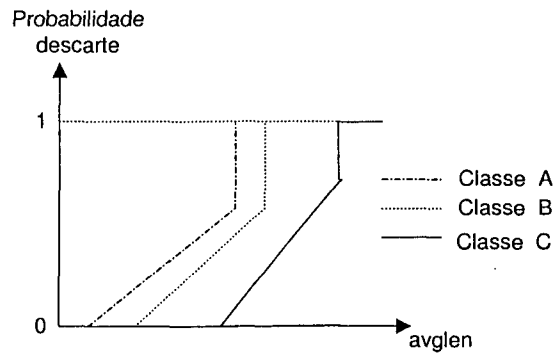


Figura 3.8: Exemplo de WRED com 03 classes

dor que implemente o RED Adaptativo com múltiplas médias e múltiplos limiares. De forma análoga, o mecanismo FRED pode ser também generalizado.

Na categoria MAMT, que é a mais genérica de todas, o cálculo do tamanho médio da fila deve seguir um procedimento de diferenciação entre as classes. A classe de maior prioridade tem sua média calculada considerando-se somente os pacotes pertencentes a esta classe. A segunda classe em prioridade considera os seus pacotes e os pacotes da classe de maior prioridade. Dessa forma, a classe menos prioritária considera todos os pacotes na fila para o cálculo do tamanho médio da fila. Esse procedimento pode ser melhor ilustrado pela descrição do mecanismo RIO (Clark e Fang 1998) realizada na subseção seguinte. Esse mecanismo é um exemplo de MAMT com dois conjuntos de parâmetros e duas médias de ocupação calculadas.

### 3.4.3.1 RIO

Ao contrário dos mecanismos anteriormente descritos, o mecanismo RIO procura também uma diferenciação de serviços através da identificação de classes de fluxos pelo campo DS. Como mencionado anteriormente, para cada classe AF, os roteadores na fronteira do domínio DS monitoram e marcam pacotes de fluxos individuais ou de agregações de fluxos de acordo com os diferentes níveis de precedência de descarte suportados pelo domínio. Se somente dois níveis de precedência de descarte são usados, pacotes de um fluxo que estejam em conformidade com o perfil de serviço contratado podem ser marcados como IN (*in-profile*) e pacotes que estejam fora do perfil de serviço podem ser marcados como OUT (*out-of-profile*). Nesse caso, o gerenciamento de fila nos roteadores internos ao domínio DS pode ser realizado pela adoção de dois algoritmos RED. Um algoritmo é associado aos pacotes que estejam em conformidade com o perfil, isto é, os pacotes marcados como IN, e outro associado aos pacotes OUT. Esse mecanismo é conhecido por RIO (RED com IN e OUT) (Clark e Fang 1998). O segundo mecanismo RED é configurado de forma mais agressiva que o primeiro, visando o descarte prioritário de pacotes OUT. O objetivo desse segundo mecanismo RED é reduzir os efeitos do congestionamento antes de tornar-se necessário o descarte de pacotes IN, permitindo um tratamento diferenciado

entre as classes IN e OUT.

O mecanismo RIO contabiliza o tamanho médio da fila para pacotes IN ( $avg_{IN}$ ) e também o tamanho médio da fila considerando-se tanto pacotes IN como pacotes OUT ( $avg_{TOTAL}$ ). A probabilidade de descarte de um pacote IN é função de  $avg_{IN}$ , enquanto a probabilidade de descarte de um pacote OUT é função de  $avg_{TOTAL}$ . Como ilustrado na Figura 3.9 existem três parâmetros para cada algoritmo RED. Os três parâmetros,  $min_{IN}$ ,  $max_{IN}$  e  $P_{maxIN}$ , definem a fase normal de operação  $[0; min_{IN}]$ , a fase que evita congestionamentos  $[min_{IN}; max_{IN}]$  e a fase de controle de congestionamento  $[max_{IN}; 1]$  para os pacotes IN. De forma análoga, os parâmetros  $min_{OUT}$ ,  $max_{OUT}$  e  $P_{maxOUT}$  definem as fases correspondentes para os pacotes OUT.

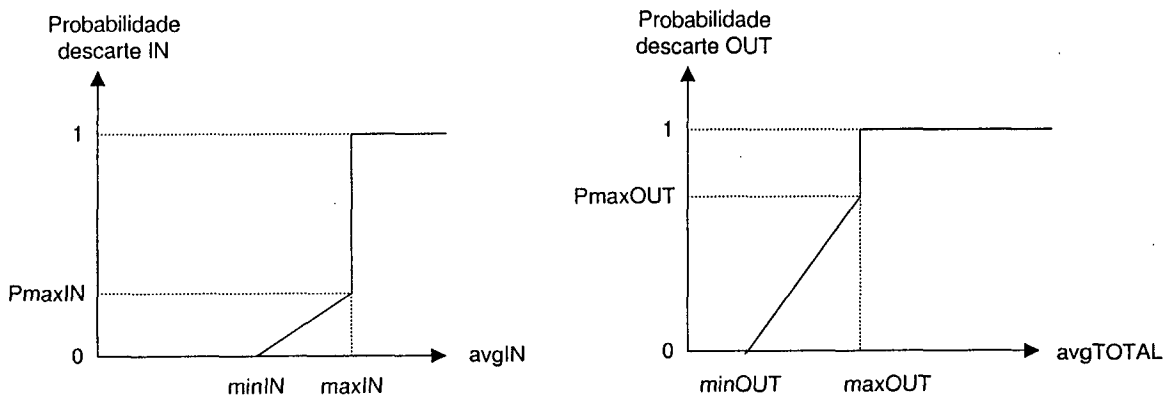


Figura 3.9: Parâmetros do Algoritmo RIO

A escolha adequada dos conjuntos de parâmetros ( $min_{IN}$ ,  $max_{IN}$ ,  $P_{maxIN}$ ) e ( $min_{OUT}$ ,  $max_{OUT}$ ,  $P_{maxOUT}$ ) provê uma discriminação contra os pacotes OUT. Um roteador RIO torna-se agressivo no descarte de pacotes OUT sob as três seguintes condições. Primeiro, ele descarta pacotes OUT mais cedo do que ele descarta pacotes IN se o valor de  $min_{OUT}$  é menor do que a configuração do parâmetro  $min_{IN}$ . Segundo, na fase onde evita-se o congestionamento, o roteador descarta pacotes OUT com maior probabilidade se o parâmetro  $P_{maxOUT}$  for maior do que  $P_{maxIN}$ . Terceiro, o roteador RIO entra na fase de controle de congestionamento para pacotes OUT antes do que para pacotes IN se a escolha do valor de  $max_{OUT}$  for menor do que  $max_{IN}$ .

Essencialmente, há o descarte inicial de pacotes OUT quando um congestionamento iminente é detectado e, caso o congestionamento persista, todos os pacotes OUT são descartados. Contudo, se uma inundação de pacotes marcados como IN ocorrer, o roteador RIO deve descartar também pacotes IN na esperança de controlar o congestionamento, no entanto, em uma rede adequadamente dimensionada isto não deve ocorrer. Quando um roteador estiver persistentemente operando na fase de controle de congestionamento, descartando pacotes IN, há uma indicação de que a rede está mal provisionada. Através deste mecanismo, o roteador espera obter uma melhor vazão para pacotes IN do que para pacotes OUT. Com a apropriada provisão

de recursos na rede, este mecanismo deve ser capaz de oferecer garantias de banda passante à agregação IN, desde que esse tráfego não ultrapasse a capacidade da rede.

### 3.5 Aplicabilidade de Serviços Diferenciados

Uma empresa que mantenha um domínio de Serviços Diferenciados pode oferecer a seus clientes SLAs que determinem uma garantia estatística de previsão de serviços no interior de seu domínio. Para que essa garantia seja estendida, totalmente ou em parte, para diversos pontos da rede, são necessários SLAs entre os diferentes domínios de Serviços Diferenciados. A Figura 3.10 apresenta um exemplo de dessa relação interdomínios. Para que o cliente A do domínio DS 1 possa transmitir tráfego de forma diferenciada ao nó D, que é cliente do domínio DS 2, um SLA interdomínio é necessário entre os domínios DS 1 e 2. Esse SLA pode determinar a expectativa de serviço da agregação de tráfego oriunda do domínio DS 1 com relação aos serviços oferecidos ao domínio DS 2. O SLA do cliente A com o domínio DS 1 deve estabelecer a expectativa de serviço deste cliente em relação inclusive aos SLAs que seu domínio possua com outros domínios DS.

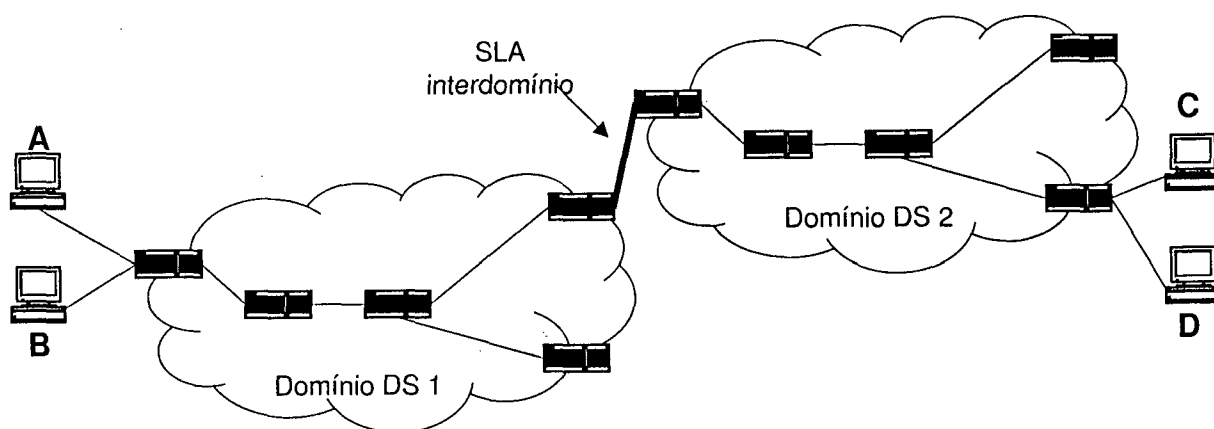


Figura 3.10: Serviços Diferenciados providos por mais de um domínio

Mecanismos de condicionamento de tráfego presentes nos nós de egresso dos domínios DS garantem a conformidade das agregações de saída ao SLA interdomínio. Da mesma forma, outros mecanismos condicionamento localizados no nó de ingresso do domínio seguinte fiscalizam essa conformidade. Assim, ambos os domínios têm condições de se manter dentro do acordo estabelecido no SLA.

Para fornecer expectativas de serviços aos seus clientes, os mantenedores de domínios de Serviços Diferenciados podem valer-se dos mecanismos discutidos ao longo deste capítulo. O PHBs oferecem uma garantia estatística às agregação de fluxos. Cabe ao domínio DS garantir que os recursos alocados sejam suficientes para atender aos fluxos individuais de seus clientes dentro dessa agregação. O PHB EF realiza uma alocação explícita de recursos para a sua

agregação de fluxos e, por isso, deve ter um maior custo final para o cliente. O PHB AF oferece garantias estatísticas de banda passante e seus mecanismos de gerenciamento ativo de filas visam controlar fluxos adaptativos. Esse serviço pode ser oferecido com um custo menor aos clientes.

A convivência de ambos os PHBs em um mesmo domínio é possível e pode ser realizada como na Figura 3.11.

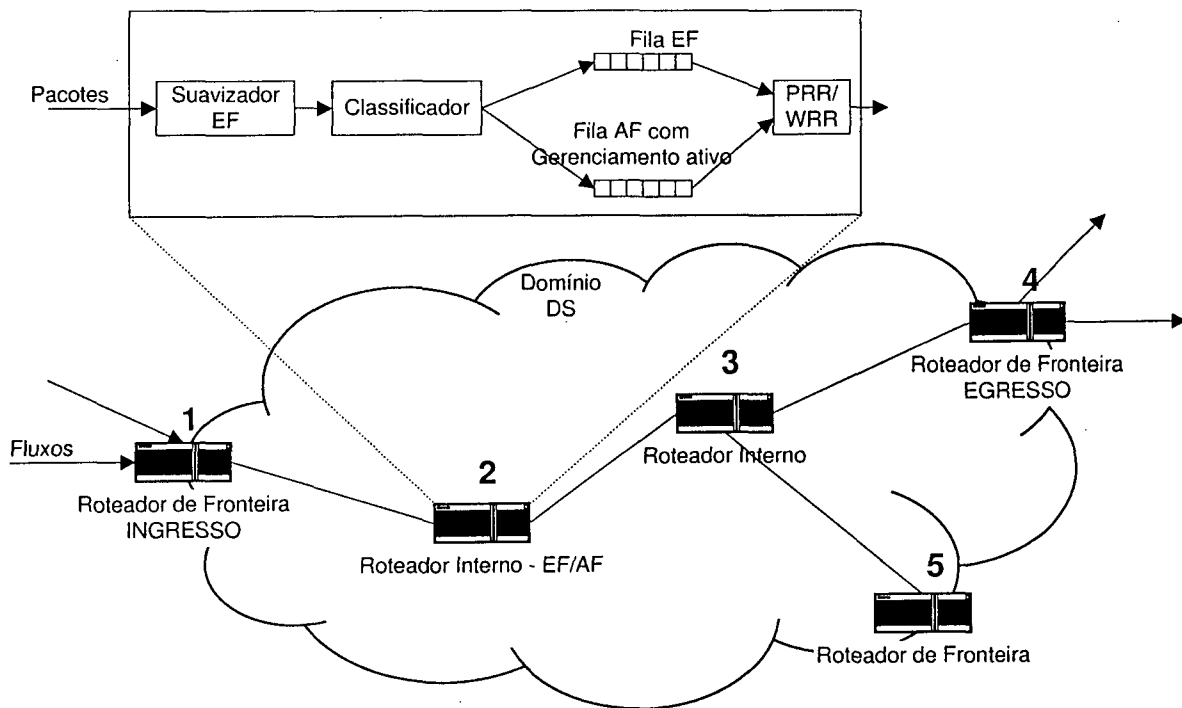


Figura 3.11: Implementação conjunta dos PHBs EF e AF

A fila dedicada à agregação EF recebe tratamento prioritário, no caso de escalonamento por PRR, ou uma fatia do enlace de saída que não viole a definição do PHB EF, no caso de escalonamento por WRR. No caso de WRR, a fila AF recebe o restante do enlace de saída. A diferenciação na fila AF é realizada pelo mecanismo de gerenciamento ativo de filas empregado, através de níveis diferentes de precedência para descarte e, possivelmente, cálculo diferenciado de médias de ocupação de fila para cada classe. O tráfego de melhor esforço, que não é atendido por nenhum PHB em especial e não possui qualquer expectativa de serviço, segue pela fila AF com o maior valor de precedência de descarte possível.

### 3.6 Conclusões do capítulo

Apresentou-se nesse capítulo, de forma detalhada a abordagem Serviços Diferenciados. Mostrou-se que hoje para essa abordagem estão sendo padronizados dois tipos de PHB: Encaminhamento Expresso (EF) (Jacobson et al. 1999) e Encaminhamento Assegurado (AF) (Heinanen

et al. 1999). Foram definidos também um PHB Padrão, ou PHB BE (*Best Effort*), para o comportamento de encaminhamento de tráfego de melhor esforço e PHB CSC (*Class Selector Compliant*) para compatibilidade com implementações existentes de IP Precedente (um padrão antigo para especificar precedência de Pacotes IP). Vimos também um pouco sobre os mecanismos e algoritmos de gerência de filas utilizados hoje em conjunto com a abordagem DiffServ.

No princípio tinha-se como meta desse trabalho implementar as duas proposta de PHB (EF e AF) em nosso ambiente. Porém, a alta complexidade dessa integração fez com fosse optado, para esse trabalho, apenas pelo PHB EF. O capítulo seguinte, apresenta a arquitetura de nosso ambiente de testes, assim como as ferramentas utilizadas.



# Capítulo 4

## Plataforma de Testes

APRESENTAMOS nesse capítulo a estrutura e a arquitetura da plataforma de testes - *testbed* - montada e utilizada durante esse projeto. O *testbed* é um ambiente condicionado para transmitir aplicações de mídias contínuas, tais como áudio e vídeo, sobre redes IP com Qualidade de Serviço, e que adota a abordagem DiffServ (*Differentiated Service*)(Blake et al. 1998) como centro de sua estrutura.

Procuramos nesse projeto acompanhar os emergentes padrões da IETF (*Internet Engineering Task Force*) para DiffServ, onde tem sido escritos em um número recente de Drafts, além de RFCs. O objetivo deste *testbed* é prover um interdomínio de serviços diferenciados - DiffServ - onde as conseqüências de engenharia, comportamento e policiamento de novos serviços IP possam ser explorados. As mais demandadas e avançadas aplicações de rede requerem absolutamente, serviço assegurado por-fluxo (Internet2 1998) e este é o objetivo desse ambiente de testes, com finalidade de explorar esta classe de novos serviços.

Nas próximas seções apresentamos a ferramenta utilizada para a geração e monitoração de tráfego, a ferramenta utilizada para configurar as interfaces de rede para que se obtenha uma qualidade de serviço para determinadas classes de serviços e, finalmente a estrutura e a arquitetura do ambiente de teste.

## 4.1 A Ferramenta TMG

### 4.1.1 Descrição da Ferramenta

O TMG (*Traffic Monitor Generator*) é uma ferramenta para geração de tráfego e medição em ambiente de alto desempenho (Leocádio e Rodrigues 2000). Estes programas, desenvolvidos dentro do projeto Almadem pelo LAND (*Laboratory for Modeling, Analysis and Development of Networks and Computer Systems*) (LAND/UFRJ 2001) na Universidade Federal do Rio de Janeiro, visam, sobretudo, possibilitar a coleta precisa e confiável de estatísticas para arquiteturas de alto desempenho, como Fast ou Giga Ethernet comutados e ATM. Num ambiente de rede de alto desempenho, podemos estar interessados em determinar o desempenho de equipamentos isolados, como, por exemplo, a latência de comutadores, ou determinar o desempenho de serviços de rede, como o tempo de processamento de um servidor em um ambiente LANE (emulação de rede local), ou o tempo de resposta de um servidor em um ambiente CLIP (IP clássico). O tempo de resposta de um serviço pode ser inferido a partir do atraso ida e volta das requisições/respostas e das latências específicas dos equipamentos intermediários, existentes no trajeto dos pacotes. Em ambos os casos, medidas de baixa ordem de grandeza e com precisão devem ser feitas.

Na construção da ferramenta, os autores se interessaram, particularmente, em medir atrasos da ida e volta de pacotes, bem como detectar possíveis variações estatísticas do retardo entre pacotes. Medidas de latência de comutadores são da ordem de poucos  $\mu s$  e o atraso pode também ser desta ordem, se o ambiente for de rede local ou metropolitana. Obter estatísticas fim a fim (em uma única direção) com este nível de precisão requer uma sincronização perfeita dos relógios dos equipamentos envolvidos, na transmissão e na recepção. Uma forma possível de atingir essa sincronização é o uso de uma interface especial de relógio baseada em GPS, que forneceria um padrão de tempo global único, com precisão. Este tipo de hardware não é comum e envolve custos adicionais. Entretanto, no cálculo de estatísticas ida e volta, preferiu-se adotar o esquema monitor/refletor, onde o tráfego é gerado em uma estação, enviado até a estação refletora, refletido e retornado à estação origem. Nesse caso, temos um tráfego bidirecional. A comparação entre as marcações de tempo (*timestamps*) nos pacotes entre a transmissão e a recepção permite o cálculo do atraso ida e volta de cada pacote, bem como a medida do atraso entre pacotes. No caso do ATM nativo, a ação da estação refletora pode ser dispensada e a reflexão pode ser feita, passivamente, estabelecendo uma conexão virtual em *loop*, ou pelo uso de cabos externos, interconectando portas de transmissão e recepção. No caso de geração de tráfego, como usamos apenas uma estação transmissora, este problema não se aplica.

Seguindo este modelo monitor/refletor, outras ferramentas existem notadamente TTCP (TTCP 2001) e NetPerf (NetPerf 2001). Estas ferramentas não foram otimizadas para medi-

das de desempenho de alta precisão, e, em especial, a saída de estatísticas destas ferramentas não apresenta os resultados com intervalos de confiança para os valores medidos. Em ambientes de alta velocidade, onde a precisão dos resultados é crítica e a ordem de grandeza das medidas é muito pequena, a apresentação de resultados com intervalo de confiança é fundamental. A ferramenta TMG - Almadem - apresenta o intervalo de confiança de 90% para todas as estatísticas pertinentes e segue o modelo monitor/refletor. Estas foram razões suficientes para sua escolha em nosso ambiente de testes.

Com relação à geração de tráfego, as ferramentas existentes permitem a especificação de uma taxa a ser gerada ao nível de aplicação, mas não fornecem a taxa efetivamente gerada no meio físico, que, em geral, é o que desejamos determinar. Por exemplo, se contratamos ao serviço ATM de uma operadora um caminho virtual permanente com banda constante (PVP CBR) e igual a 20 Mbps, e sabendo que a operadora não admite tráfego além do contratado, se quisermos comprovar a garantia de banda do serviço contratado, temos que enviar no meio físico o equivalente a 20 Mbps e verificar descarte zero de células. Se especificarmos simplesmente 20 Mbps ao nível de aplicação, a consideração de todos os overheads envolvidos irá gerar uma taxa acima de 20 Mbps no meio físico e descartes ocorrerão. Redes locais de alto desempenho trabalhando com Ethernet comutado *full duplex*, permitem o cálculo preciso do *overhead* no meio físico, pela ausência de colisões. Evidentemente, em redes compartilhadas como Ethernet normal, o tráfego gerado no meio físico por uma determinada aplicação é acrescido de colisões, fazendo com que a taxa efetivamente transmitida seja maior. Neste caso, somente podemos determinar o nível mínimo da taxa no meio físico. A ferramenta TMG - Almadem - operando com TCP/IP, UDP/IP ou nativamente sobre ATM, inclui o cálculo das taxas efetivas de transmissão e recepção no meio físico, o que facilita a programação e configuração de testes.

#### 4.1.2 A Composição da Ferramenta

A ferramenta TMG é composta basicamente de quatro programas principais, cada qual possuindo suas variantes (UDP, TCP, ATM, etc): Gerador, Monitor, Refletor e SendWait. Cada programa tem sua finalidade própria, podendo ou devendo ser utilizado de forma conjunta com algum outro. O uso dos programas refletores, por exemplo, está diretamente vinculado ao uso dos programas monitores, com exceção do ATM nativo, onde a reflexão das informações pode ser feita passivamente, através de configuração do próprio comutador ATM ou cabo externo entre interfaces, dispensando a utilização de um programa refletor.

Os programas monitores são constituídos por dois processos: um dedicado à geração de pacotes em instantes de tempo pré-determinados, outro dedicado à recepção dos pacotes previamente gerados e cômputo de suas estatísticas. Cada variante do programa possui seus parâmetros de entrada que, entre outras coisas, permitem a determinação da duração da sessão

e do intervalo de geração de pacotes. O processo transmissor é o processo pai e dá origem ao processo coletor (filho), imediatamente antes de iniciar a transmissão de pacotes. A partir desse momento, o processo transmissor gerará um pacote contendo, em geral, número de seqüência e *timestamp* de transmissão a cada  $E \cdot s$ , onde  $E$  é um parâmetro fornecido pelo usuário. O processo coletor fará a leitura dos pacotes refletidos, computando as estatísticas. Terminada a transmissão, o processo pai fica à espera do término do processo filho, que, após o recebimento do último pacote ou *timeout*, fará o processamento final das estatísticas coletadas. Os resultados finais são impressos na tela ou gravados em arquivo.

Cada variante do programa monitor, exceto a variante TCP, apresenta, dois tipos de chamada: *monitor* e *monitor2*. A primeira chamada é mais adequada a sessões de pequena duração, quando desejamos uma caracterização rápida dos parâmetros de latência do ambiente de rede sendo avaliado. A segunda chamada é otimizada para sessões de maior duração, ideal para situações onde se deseja determinar possíveis variações estatísticas no comportamento da rede, durante um período mais longo de observação.

Os programas refletores têm como única função a leitura e subsequente retransmissão das PDU's enviadas pelos programas monitores equivalentes, sem qualquer processamento extra de informação. Ao contrário dos programas monitores, os refletores são constituídos por apenas um processo, que realiza tanto leitura como escrita.

Os programas *sendwait* foram realizados com o intuito de permitir medida precisa de latência bastante pequena. Uma vez que nos programas monitores temos dois processos rodando simultaneamente, transmissor e coletor, há uma competição pelo processador da estação, gerando imprecisão. Essa competição poderá afetar as medições da ferramenta em função, principalmente, da capacidade de processamento da estação, do tamanho de pacote utilizado, e do intervalo de geração de pacotes, especificado pelo usuário. Para contornar o problema, o programa só envia um novo pacote após o recebimento do anterior ou na ocorrência de um *timeout*. Esta melhoria é mais significativa em ambiente de rede local, onde o atraso de propagação é desprezível ou muito pequeno e latências são apenas de alguns microssegundos.

Os programas geradores têm por objetivo a geração de tráfego de fundo durante a realização dos experimentos. Estes programas permitem a geração de grandes volumes de tráfego (dependendo da capacidade da estação sendo utilizada), sendo capazes de congestionar ambientes de rede local como Fast Ethernet. Nos experimentos realizados pelos pesquisadores do projeto Almadem, uma estação Sun UltraSparc 170 Mhz foi capaz de gerar 120 Mbps. A implementação desses programas determina a geração de tráfego em intervalos fixos de tempo, sendo portanto do tipo determinístico (ao nível de aplicação). O volume de tráfego gerado a cada transmissão é função da taxa solicitada pelo usuário. Dependendo da variante do programa, o usuário tem também acesso ao tamanho da unidade de transmissão a ser utilizada.

Os programas geradores de tráfego utilizam a pilha UDP/IP. Nos programas de monitoração, o usuário pode optar entre pilhas UDP/IP, TCP/IP ou ATM nativo para execução. No caso específico do ATM nativo para o sistema operacional Solaris, a API de programação utilizada foi a API/XTI disponibilizada pela FORE Systems Inc., não sendo, portanto, o programa compatível com outras APIs ATM. A API citada é disponibilizada juntamente com o driver da interface de rede ATM, podendo este ser obtido diretamente no site da empresa <http://www.fore.com>. Os programas em ATM nativo para Linux foram implementados com a API desenvolvida por Werner Almesberger, que pode ser obtida em <http://icawww1.epfl.ch/linux-atm>, sendo independente de fabricantes. Os programas desenvolvidos para as pilhas UDP/IP e TCP/IP fazem uso de programação *socket* BSD padrão, não possuindo maiores requisitos quanto a sua utilização. Esses programas encontram-se disponíveis para os sistemas operacionais Solaris, Linux e FreeBSD. Todos os programas possuem ajuda na linha de comando, facilitando enormemente a especificação dos parâmetros necessários.

Na versão 1.1 da ferramenta, a única informação relativa à qualidade de serviço disponível ao usuário é a PCR (*Peak Cell Rate*) desejada. Se o valor especificado for igual a zero, o descritor será associado a um PVC do tipo UBR (*Unspecified Bit Rate*). Caso o valor especificado seja positivo, o descritor será associado a um PVC CBR, com taxa de pico igual a PCR solicitada. Atualmente a ferramenta encontra-se disponível para uso apenas sobre PVC's, sendo toda a execução feita em modo síncrono (bloqueante). O tráfego gerado pela ferramenta é estritamente aquele especificado pelo usuário.

Os programas monitores e refletores são interoperáveis ao nível de sistema operacional, ou seja, podemos, por exemplo, utilizar uma variante de programa monitor em uma estação rodando Linux, com seu refletor equivalente em uma estação rodando Solaris. Isto é verdadeiro também para os programas em ATM nativo. Quando tratamos de ATM nativo, no entanto, o processo de reflexão pode ser alcançado pela configuração de um *loopback* no próprio *switch* ATM (dispensando assim o uso de um refletor).

Os programas geradores operam somente sobre a pilha UDP/IP. Como estes programas se prestam apenas à geração de tráfego de fundo, como perturbação ao ambiente sendo testado, a geração de tráfego em ATM pode ser alcançada indiretamente através de um ambiente de IP clássico ou LANE, utilizando-se os próprios geradores UDP. Uma alternativa, caso tenhamos apenas PVC's puros, é o uso dos programas monitor/refletor nativos para geração de tráfego.

Quando utilizamos IP sobre PVC CBR nas estações, a taxa de pico a ser suportada pelo PVC é fornecida ao se configurar a interface lógica IP/ATM na estação. Observe que se o PVC for configurado como CBR com determinada taxa de pico PCR, o *driver* IP da estação limitará o tráfego a ser gerado nesta interface lógica IP/ATM a este valor PCR, ainda que a aplicação, usando programação *socket*, tente gerar um tráfego acima de PCR.

Em nosso caso específico, estamos interessados em medir o desempenho dos serviços de rede com relação a aplicações que exigem certa garantia, principalmente temporal, com relação a entrega de seus dados. Em nossos testes, consideramos apenas os parâmetros seguintes: taxa de perda de pacotes, atraso sofrido por pacote e o atraso entre os pacotes (*jitter*). Para isso, achamos a ferramenta TMG - Almadem - adequada para com nosso ambiente de teste.

## 4.2 Controlador de Tráfego

Os Kernels mais recente do Linux oferecem uma grande variedade de funções de controle de tráfego, as quais podem ser combinadas de uma forma modular. Esta seção trata-se de descrever a aplicação TC (*Traffic Controller*) (Almesberger et al. 1999, Radhakrishnan 1999) para dar suporte a Serviços Diferenciados baseado nos elementos de controle de tráfego existentes, além dos novos componentes implementados para esse fim.

A figura 4.1 mostra, de uma forma superficial, como o kernel processa os dados recebidos da rede e como ele gera novos dados para serem enviados para a rede.

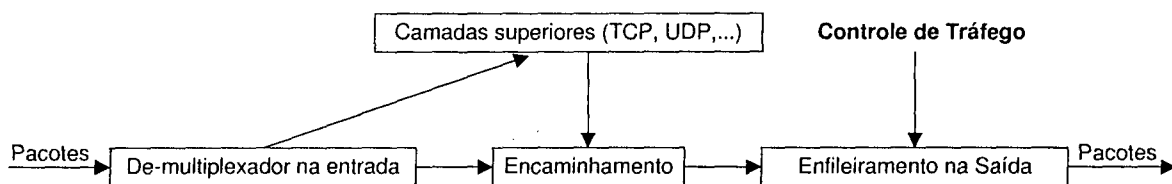


Figura 4.1: Controle de Tráfego no Linux

“Encaminhamento” inclui a escolha da interface de saída, a escolha do próximo *hop*, encapsulamento, etc. Uma vez tudo isso feito, os pacotes são enfileirados na respectiva interface de saída. Nesse ponto que o Controle de Tráfego entra em ação. O Controle de Tráfego pode, entre outras coisas, decidir se os pacotes são enfileirados ou descartados (por exemplo se a fila tem preestabelecido algum limite de tamanho, ou se o tráfego excede algum limite de taxa), ele pode decidir qual a ordem em que os pacotes são enviados (por exemplo para obter prioridade para certos fluxos), ele pode temporizar o envio dos pacotes (por exemplo para limitar a taxa de tráfego fora do limite), etc.

Uma vez que o Controle de Tráfego tenha liberado um pacote para ser enviado, o driver do dispositivo o seleciona e o transmite pela rede.

## 4.2.1 Componentes

O Código do controle de tráfego no Kernel Linux consiste, principalmente, dos seguintes componentes:

- Fila
- Classes (associada a uma disciplina de fila)
- Filtros
- Policiadores

Inicialmente definiremos as disciplinas de fila possíveis.

### 4.2.1.1 Disciplinas de Fila

Cada dispositivo de rede tem uma fila associada a ele. Existem 11 tipos de disciplinas de fila que são atualmente suportadas pelo Linux:

- Class Based Queue (CBQ)
- Token Bucket Flow (TBF)
- Clark Shenker Zhang (CSZ)
- First In First Out (FIFO)
- Priority
- Traffic Equalizer (TEQL)
- Stochastic Fair Queuing (SFQ)
- Asynchronous Transfer Mode (ATM)
- Random Early Detection (RED)
- Generalized RED (GRED)
- DiffServ Marker (DS-MARK)

As filas são identificadas por um handle (número maior : número menor), onde o número menor é zero para as filas. Os handles são usados para associar classes às disciplinas de fila. As classes são discutidas mais adiante.

Com dito antes, cada dispositivo de rede tem uma disciplina de fila associada a ele, a qual controla como os pacotes enfileirados lidam com o mesmo. Uma disciplina de fila simples pode consistir de uma única fila, na qual todos os pacotes são armazenados na ordem em que eles têm sido enfileirados.

As disciplinas de fila mais elaboradas podem usar filtros para distinguir amostras de classes diferentes de pacotes e processar cada classe em um caminho diferente, por exemplo, para obter uma maior prioridade para uma classe com relação a outra.

A figura 4.2 mostra um exemplo de tal disciplina de fila. Perceba que múltiplos filtros podem ser associados a uma classe.

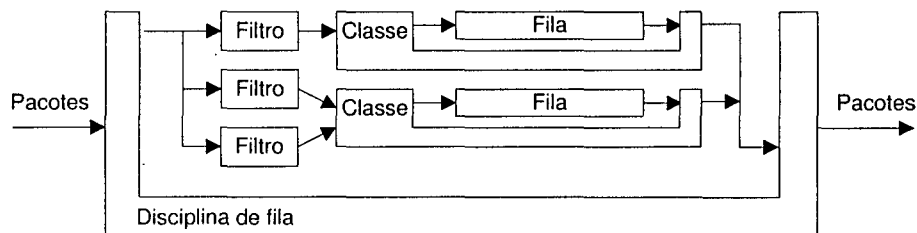


Figura 4.2: Disciplina de Fila simples com múltiplas classes

As disciplinas de fila e as classes estão intimamente ligadas: a presença de classes e suas semânticas são propriedades fundamentais da disciplina de fila. Ao contrário, os filtros podem ser combinados arbitrariamente com as disciplinas de fila e as classes desde que as disciplinas de fila possuam classes para mapear os pacotes para si.

A figura 4.3 mostra um exemplo de tal pilha: primeiro, existe uma disciplina de fila com duas prioridades de tempo. Pacotes que são selecionados pelo filtro para seguir para a classe de alta prioridade e pacotes que seguem para a classe de baixa prioridade. Sempre que existem pacotes na fila de alta prioridade, eles são enviados antes dos pacotes na fila de baixa prioridade. Com a intenção de providenciar um tráfego de alta prioridade sobre um tráfego de baixa prioridade, usou-se nesse exemplo, um *Token Bucket Filter* (TBF), o qual força a taxa para a fila de 1Mbps para aquela fila. Finalmente, o enfileiramento da fila de baixa prioridade é feito por meio de uma disciplina de fila FIFO. Perceba que existem outras possibilidades para implementar essas prioridades para classes, como por exemplo usando *Class Based Queuing* (CBQ).



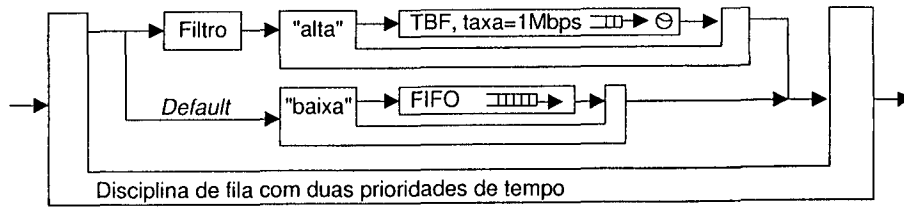


Figura 4.3: Combinação de prioridade, TBF e disciplinas de fila FIFO

#### 4.2.1.2 Classes

Como antes mencionado, filas e classes estão interligados. Cada classe possui uma fila, a qual por *default* é uma fila FIFO. Quando a função *enqueue* da disciplina de fila é chamada, a disciplina de fila aplica os filtros para determinar a classe para a qual o pacote deve seguir. Ele então chama a função *enqueue* da disciplina de fila dessa classe.

Existem duas formas de se identificar uma classe. Uma é via o identificador de classe, o qual é especificado pelo usuário. E o outro identificador, o qual é usado junto ao kernel para identificar uma classe, é referenciado como identificador interno. Esse ID é único e é assinalado pela disciplina de fila. O ID de classe é um tipo de dado u32 (classificador que considera quatro parâmetros durante a classificação: Porta e IP Origem, Porta e IP Destino), enquanto o ID interno é um inteiro longo não assinalado. Muitas das funções nas classes usam o ID interno para identificar a classe. Entretanto, existem umas poucas funções que usam o ID de classe também.

Múltiplos IDs de classe podem mapear para algum ID interno, entretanto, o ID de classe enviará alguma informação adicional do classificador para a disciplina de fila ou classe.

O ID de uma classe, assim como o identificador de disciplina de fila, é estruturado na forma de (número maior:número menor). O número maior corresponde a instância da disciplina de fila à qual a classe pertence, enquanto o número menor identifica a classe dentro de sua própria instância.

Nem todas as disciplinas de fila suportam classes. Dentre as que suportam classes estão a CBQ, DS-MARK, CSZ e a p-FIFO. As demais disciplinas de fila não suportam classes.

As operações permitidas para manipulação das classes junto a várias disciplinas de fila que suportam essas classes são as seguintes:

- *Graft*: usada para anexar uma nova disciplina de fila a uma classe. Como mencionado na seção anterior, a disciplina de fila *default* anexada quando a classe é criada é a FIFO. Para trocar essa disciplina de fila, a operação *graft* é executada sobre a classe.
- *Get*: usada para retornar o ID interno da classe. Essa função incrementa o uso do contador

de classe.

- *Put*: essa função é chamada quando um classe anteriormente referenciada usando a função *get* não é está mais sob referência. Ela decrementa o uso do contador de classe.
- *Change*: usada para trocar as propriedades associadas às classes. Entretanto, a função *change* é também usada para criar classes em alguns casos.
- *Delete*: usada para remover uma classe. Ela determina o uso da classe, através do contador referente; no caso de ser igual a zero, desativa e remove a classe.
- *Walk*: usada para se ter uma interação sobre todas as classes de uma disciplina de fila. Ela é geralmente usada para se obter um diagnóstico dos dados para todas as classes de uma disciplina de fila.
- *Tcf-chain*: usada para retornar o ponteiro para a lista de filtros que está associada a uma classe. Cada classe está associada a um lista de filtros, a qual contem a listagem de filtros que são usados para identificar os pacotes que pertencem a uma classe em particular.
- *Bind-tcf*: usada para anexar uma instância de um filtro a uma classe. Esta função é similar a função *get*. A diferença é que uma classe que está apontada pelo filtro não pode ser removida sem a remoção dos filtros.
- *Unbind-tcf*: usada para remover uma instância de um filtro anexada a uma classe. Aqui o contador do número de filtros anexados a classe é decrementado. Para um classe ser removida, esse contador deve ser zero.
- *Dump-class*: usada para diagnosticar dados sobre as classes. Existe um conjunto de dados que está principalmente sobre as classes e a função *dump* e é usado para observar esses valores.

#### 4.2.1.3 Filtros

Filtros são usados para classificar os pacotes baseado em algumas propriedades do pacote, por exemplo, o byte TOS no cabeçalho IP, endereço IP, números de porta, etc. Ele é acionado quando a função *enqueue* da disciplina de fila é chamada. As disciplinas de fila usam filtros para assinalar os pacotes que estão chegando para uma de suas classes.

Os filtros podem também ter uma estrutura interna: ele pode controlar elementos internos, os quais são referenciados por um manuseador (*handle*). Esses handles não são divididos em números maior e menor como os IDs das classes. O handle igual a 0 se refere ao próprio filtro. Como nas classes, filtros também possuem ID interno, o qual pode ser obtido com a ajuda da função *get*. A estrutura básica dos filtros está representada na figura 4.4.

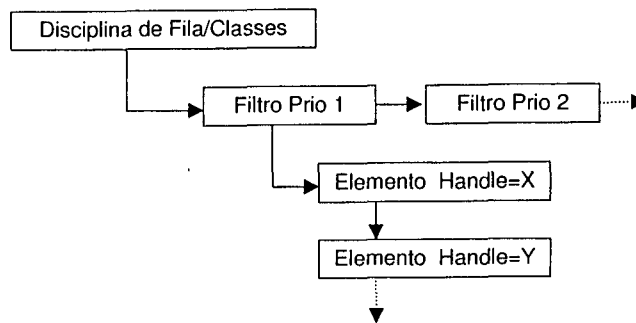


Figura 4.4: Estrutura do Filtros

A seguir, falaremos de forma bem rápida sobre as funções que podem ser executadas junto aos filtros:

- *Classify*: usada para associar um pacote a uma classe tendo como base certas propriedades do pacote. Por exemplo, o classificador *route*, classifica os pacotes levando-se em consideração o endereço IP de destino.
- *Init*: usada para inicializar os parâmetros para um filtro.
- *Destroy*: usada para remover um filtro. Se o filtro ou alguns de seus elementos estão registrados com classes, a função *destroy* sobre o filtro, chama uma outra função específica para desfazer tais registros. Para filtros mais complicados tais como *u32*, essa função também se torna mais complexa.
- *Get*: como antes mencionado, todo filtro tem um ID interno correspondente a um handle. Este mapeamento pode ser obtido com a ajuda da função *get* no filtro. Esta função, por sua vez é similar a *get* nas classes.
- *Put*: usada para desfazer a referência a um filtro. Referência essa feita pela função *get*. Porém em geral, a função *put* nunca é chamada.
- *Change*: usada para trocar as propriedades de um filtro. Esta é semelhante a função *change* nas classes e disciplinas de fila. A configuração dos parâmetros é passada usando um mecanismo que é similar ao utilizado para passar parâmetros para classes e disciplinas de fila. Se um políciador está anexado a um filtro, então suas propriedades também são alteradas.
- *Delete*: usada para remover um elemento particular de um filtro. Como discutido antes, para remover um filtro inteiro, a função *destroy* é chamada.
- *Walk*: usada para interagir sobre todos os elementos de um filtro e chamar uma função *callback* para cada um desses elementos. Geralmente usada para se obter um diagnóstico dos elementos de um filtro.

- *Dump*: usada para se obter uma informação estatística sobre o filtro.

## 4.2.2 A Ferramenta TC

Diante dos conceitos vistos até então com relação ao tratamento que o sistema operacional Linux oferece para se obter qualidade de serviço numa transmissão, podemos falar aqui um pouquinho da ferramenta TC (*Traffic Controller*). Essa ferramenta é um programa ao nível do usuário que pode ser usada para se criar e associar filas aos dispositivos de saída de rede. Ela é usada para inicializar vários parâmetros de fila e associar classes com esses parâmetros de fila. O TC pode ser usado também para inicializar filtros com base numa tabela de roteamento, classificadores u32, classificadores tcindex e classificadores RSVP. Esse programa usa de *netlink sockets* como um mecanismo para comunicação com as funções de rede do kernel. A figura 4.5 ilustra, de forma geral, essa comunicação.

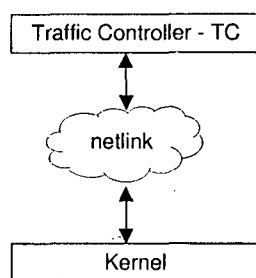


Figura 4.5: Espaço do usuário para comunicação com o kernel usando o TC

A implementação do protótipo de suporte a DiffServ requer o acréscimo de mais três novos componentes para o controle de tráfego no kernel: da disciplina de fila *sch-dsmark* para extrair e setar o DSCP (*Differentiated Service Codpoint*), o classificador *cls-tcindex* o qual usa essas informações, e a disciplina de fila *sch-gred* a qual suporta múltiplas prioridades de descarte e compartilhamento de *buffer*.

Somente a disciplina de fila para extrair e setar o DSCP é verdadeiramente específica para a arquitetura DiffServ. Os outros dois elementos podem ser usados em outros contextos.

## 4.3 Arquitetura da Plataforma de testes

Apresentamos nessa seção a arquitetura do *testbed*. Este é um ambiente ou plataforma de teste para aplicações de mídias contínuas, sobre redes IP com Qualidade de Serviço, e que adota a abordagem DiffServ (*Differentiated Service*) (Blake et al. 1998) como centro de sua estrutura.

O objetivo principal deste *testbed* é prover um interdomínio de serviços diferenciados - Diff-Serv - onde as conseqüências de engenharia, comportamento e policiamento de novos serviços IP possam ser explorados. Adotamos para nosso ambiente de testes somente o tipo de serviço Encaminhamento Expresso (EF). Isso porque consideramos a integração dessa proposta de PHB com a proposta Encaminhamento Assegurado é um tanto complexo.

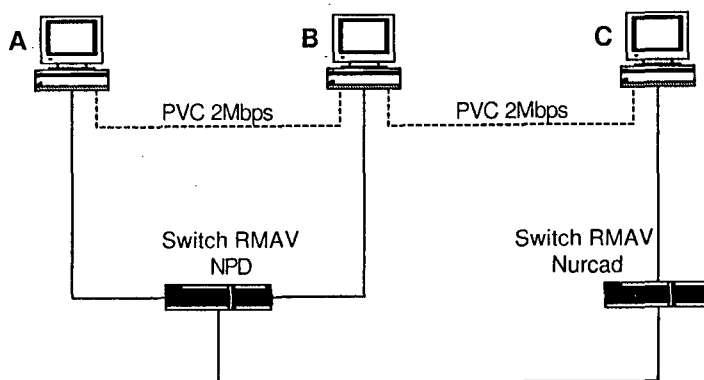
Como descrito em (Internet2 1998) as mais demandadas e avançadas aplicações de rede requerem absolutamente, serviço assegurado por-fluxo e este é o objetivo de nosso *testbed*, com finalidade de explorar esta classe de novos serviços. A *Draft Internet* (Bernet e et al. 1998) chama esses de "serviços quantitativos".

Procuramos na arquitetura do ambiente de teste acompanhar os emergentes padrões da IETF (*Internet Engineering Task Force*) para DiffServ, onde tem sido escritos em um número recente de Drafts, além de RFCs.

Esse texto está consistente com a terminologia definida em (Blake et al. 1998), e cada rede participante do *testbed* será considerada um "Domínio DS" e a união desses domínios, o *testbed* em si, uma "Região DS". No ambiente de teste, cada participante pode cooperar para prover um ou mais serviços interdomínios, além do padrão, isto é, o tradicional modelo serviço "melhor esforço - BE (*Best Effort*)". O primeiro destes serviços a ser implementado é o Serviço *Virtual Leased Line* (VLL), ou Premium, descrito em (Nichols et al. 1999). Todo domínio de DS *testbed* pode suportar o comportamento per-hop (PHB) do encaminhamento expresso (EF) (Jacobson et al. 1999) e configurar seus classificadores de tráfego e condicionadores (medidores, marcadores, formatadores e descartadores) para prover um serviço VLL para agregados EF. Um próximo passo é assegurar que o PHB do encaminhamento assegurado seja também suportado.

Nosso ambiente de teste é formado por três PCs. Cada qual com o sistema operacional Linux devidamente instalado. A versão do kernel utilizada é a 2.4.0 test12. Essa versão já trás em sua forma nativa os módulos relacionados a QoS. Um dos PCs possui duas interfaces de rede, fazendo com que este atue como roteador em nosso ambiente de testes. Os outros dois, os quais atuam com origem e destino do tráfego, possuem apenas uma interface de rede. Entre os PCs, foram estabelecidos dois PVC (*Permanent Virtual Circuit*) com um limite de 2Mbps cada. Com isso conseguimos, através do CLIP (Classic IP), configurar o ambiente para simular duas redes locais interligadas por meio de uma máquina, a qual atua como roteador. A Figura 4.6 ilustra essa arquitetura. A linha contínua corresponde às conexões que formam a rede física, enquanto que a linha tracejada corresponde às conexões que formam a rede virtual.

Na Universidade Federal de Santa Catarina, a máquina C está conectada a um *switch* ATM da RMAV-FLN (Rede Metropolitana de Alta Velocidade de Florianópolis) no NURCAD (Núcleo de Redes de Alta Velocidade e Computação de Alto Desempenho), enquanto que as

Figura 4.6: Arquitetura do *testbed*

máquinas A e B estão conectadas a outro *switch* ATM da RMAV situado no NPD (Núcleo de Processamento de Dados).

Implementamos como parte deste trabalho *Scripts* os quais rodam o programa TC (*Traffic Controller*) com finalidade de criar nas interfaces de rede das máquinas as disciplinas de fila, as classes e os filtros. Esses *scripts* podem ser usados pelo administrador de uma rede para configurar os serviços diferenciados num roteador Linux.

Na máquina A, o *script* nela implementado tem a função de torná-la um roteador de borda (*edge router*) em nosso domínio DS. Nessa máquina, além da geração e medição do tráfego, ocorre a marcação (remarcação) dos pacotes como sendo pertencentes ao tráfego EF (*Expedited Express*) ou BE (*Best Effort*). Nela acontece também o primeiro policiamento do tráfego.

A máquina B possui duas interfaces de rede. Nessa máquina, implementou-se o *script* capaz de, por meio do programa TC, fazer com que essa máquina atue como roteador interior (*core router*) ao nosso domínio DS. Nas interfaces de rede dessa máquina, ocorre simplesmente o policiamento do tráfego através da análise do campo TOS (Type of Service) no cabeçalho dos pacotes. Isso cumpre uma das regras da abordagem DiffServ a qual "joga" todo o processamento mais pesado, com relação a marcação por exemplo, para as bordas do domínio DS.

Em nosso caso específico, como estamos usando uma ferramenta de geração e medição de tráfego com o esquema monitor/refletor (seção 4.1), a máquina C, que inicialmente é destino do tráfego proveniente da máquina A, atua também como máquina de origem do tráfego ao devolver os dados para a máquina A. Logo, a essa máquina roda também um *script* que a torna um roteador de borda (*edge router*) em nosso domínio DS.

A seguir, apresentamos as configurações dos roteadores *edge* e *core* de uma forma mais detalhada, levando em conta a posição da interface de rede da cada máquina pertencente ao *testbed*. Como já foi mencionado antes, estamos trabalhando apenas com dois tipos de serviços. O Serviço EF e o Serviço BE.

### 4.3.1 Configuração do roteador de borda (*Edge*)

Os fluxos provenientes de um cliente no ponto final entram em um domínio DS ainda no Roteador *Edge*. Com isso, a arquitetura no roteador *edge* tem que prover a marcação do DSCP (*Differentiated Service Code Point*) e também o policiamento dos pacotes. Em nossos testes, o cliente é o próprio roteador *edge*, uma vez que essa máquina estará rodando o aplicativo TMG (*Traffic Monitor Generator*) gerando e medindo assim, o tráfego. A arquitetura é como mostra a figura 4.7.

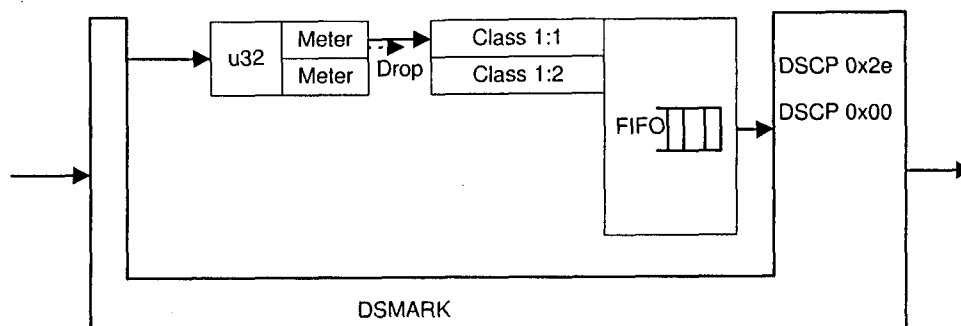


Figura 4.7: Arquitetura do roteador de borda

O primeiro nível de fila na arquitetura é a disciplina de fila `ds_mark`, a qual está diretamente associada com o dispositivo de saída, ou seja, a interface de saída da placa de rede. A disciplina de fila `ds_mark` seta o DSCP para o *codepoint* requerido.

A partir da disciplina de fila `ds_mark` um filtro `u32` é criado. Os filtros `u32` são criados para um fluxo em particular. Cada fluxo pode ser unicamente identificado por uma tupla composta de quatro parâmetros (IP de origem, IP de destino, porta de origem, porta de destino) ou por uma tupla composta por três parâmetros (IP origem, IP destino, porta destino). Cada filtro é associado a um medidor o qual policia o fluxo com base em pacotes dentro do perfil (*profile*) ou fora do perfil (*out of profile*) de tráfego. Um DSCP do fluxo EF dentro do perfil é marcado como `0x2e` e os pacotes fora do perfil são descartados. No encaminhamento assegurado (AF) os pacotes dentro do perfil são marcados com o *codepoint* assinalado e os pacotes fora do perfil são reclassificados como *best effort*.

Os pacotes ao saírem do Roteador *Edge*, têm seus DSCPs marcados e são também policiados de acordo com a largura de banda alocada pelo cliente junto ao controlador de Largura de Banda (*Bandwidth Broker*).

### 4.3.2 Configuração do roteador interno (*Core*)

A arquitetura *DiffServ Two Bit* nos roteadores *core* permitem a coexistência do Encaminhamento Assegurado e Encaminhamento Expresso. Para nosso *testbed*, consideraremos apenas o

tráfego EF. A figura 4.8 mostra a arquitetura do roteador interno pertencente ao nosso domínio DS.

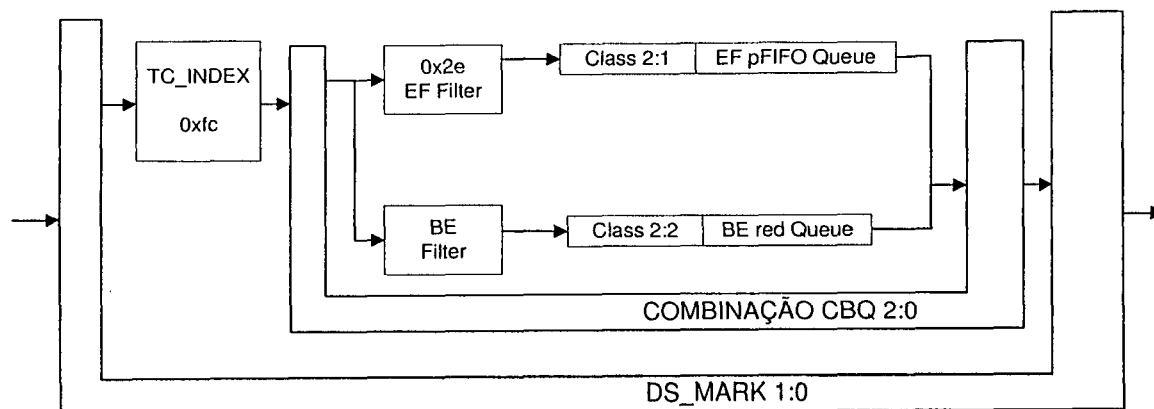


Figura 4.8: Arquitetura do roteador interno

O primeiro nível de fila nessa arquitetura é a disciplina de fila `ds_mark`, a qual será diretamente associada ao dispositivo de saída. Esta disciplina de fila extrairá o *byte* TOS do pacote, quando este entrar e o passará para um classificador `tc_index`. O primeiro nível do classificador `tc_index` extrai o DSCP do byte TOS o mascarando com `0xfc` e então retira dois bits da direita e acrescenta dois zeros à esquerda. Por exemplo, se o pacote pertence ao tráfego EF, ou seja, se o TOS é `0xb8`,

TOS `0xb8` 10111000

Mask `0xfc` 10111000

Shift 2 00101110 (0x2e, que corresponde ao “*code point*” para EF)

Então o classificador armazena o DSCP no `skb-tc_index` e o passa para o próximo nível de filtros `tc_index`. Uma vez a classificação feita, o pacote é então manuseado sobre a fila baseada em classes (CBQ) 2:0, uma vez que estamos utilizando disciplinas de fila do tipo CBQ (*Class Based Queue*). Esta CBQ será usada para diferenciar entre serviços EF e BE. Quando os pacotes entram nessa CBQ, os pacotes EF são direcionados para a classe 2:1 por meio do filtro `0x2e` e todos os demais pacotes, são encaminhados para a classe 2:2 por meio do filtro `0x00`. Para EF, uma `pfifo` é associada com a classe 2:1. A classe EF é configurada para ser *bounded* e *isolated* para assegurar que ela receba a banda passante a ela associada. O parâmetro *bounded* indica que essa classe de serviço não poderá solicitar largura de banda “emprestada” para outra classe. Já o parâmetro *isolated*, indica que essa classe não poderá “emprestar” largura de banda para outra classe. A classe BE tem uma disciplina de fila do tipo RED (*Random Early Detection*) (Floyd e Jacobson 1993). Essa classe, ao contrário da EF usa o parâmetro *borrow*. Esse parâmetro permite que essa classe compartilhe recursos na rede com outros fluxos.



## **4.4 Conclusões do capítulo**

Vimos nesse capítulo detalhes sobre a arquitetura de nosso ambiente de testes. Aqui foram apresentadas as ferramentas utilizadas para geração/monitoração de tráfego e para configuração das interfaces de rede num ambiente Linux. O próximo capítulo apresenta os experimentos realizados sobre esse ambiente de testes, assim como os resultados obtidos.

# Capítulo 5

## Experimentos e Resultados

COM o objetivo de investigar o desempenho do tráfego de dados quando atravessando um domínio de Serviços Diferenciados (DS), vários ensaios foram realizados sobre o nosso ambiente de teste. O comportamento desses tráfegos é avaliado para verificar o impacto dos mecanismos da arquitetura de Serviços Diferenciados no atendimento dos requisitos desses tráfegos. Conforme visto no capítulo 3, estão em discussão (Group 2001) (Blake et al. 1998), atualmente, duas propostas de encaminhamento por nó (*Per-Hop Behavior* - PHB) para a implementação de Serviços Diferenciados: o Encaminhamento Assegurado - AF - (Heinanen et al. 1999) e o Encaminhamento Expresso - EF - (Jacobson et al. 1999). Esse trabalho porém, se limitou a investigar o serviço oferecido ao tráfego de mídias contínuas, modelado por fonte CBR (*Constant Bit Rate*), pelo PHB EF no interior de um domínio DS.

A complexidade contida na implementação conjunta das duas propostas de encaminhamento de pacotes por nó da abordagem DiffServ num ambiente de rede real, fez com esse trabalho abordasse apenas o PHB EF. No entanto, a implementação desse PHB foi suficiente para a investigação do comportamento das aplicações envolvendo mídias contínuas ao atravessarem um domínio DS.

Esse PHB pode ser utilizado para a obtenção de um serviço fim-a-fim com baixa perda, baixo retardo, baixa variação do retardo (*jitter*) e largura de banda garantida através de um domínio DS. As filas nas quais os pacotes em trânsito permanecem nos nós ao longo de sua rota na rede são as maiores responsáveis pela perda, pelo retardo e pelo *jitter* apresentado por esse pacote. O PHB EF objetiva diminuir a permanência dos pacotes pertencentes a uma determinada agregação de fluxos em filas no interior de um domínio de Serviços Diferenciados (DS). Para alcançar esta meta, os nós que ofereçam o PHB EF devem garantir que a agregação de fluxos contratante receba a taxa de serviço contratada, que deve ser maior que a taxa de chegada em qualquer instante.

A seção 5.1 apresenta a arquitetura e configurações do ambiente de teste. Os experimentos, os vários casos e os resultados obtidos são apresentados na seção 5.2. Por fim uma análise geral é feita na seção 5.3.

## 5.1 Topologia do ambiente de teste

### 5.1.1 Arquitetura do *testbed*

Para realização dos testes, foi utilizado o ambiente de teste mostrado na seção 4.3 o qual é constituído de três máquinas PC rodando o sistema operacional Linux, versão do kernel 2.4.0 test12. Essa versão fornece suporte a QoS já trazendo, em sua forma nativa, os módulos relacionados a QoS. Entre os PC foi estabelecido um PVC (*Permanent Virtual Circuit*) com largura de banda igual a 2Mbps. A classe do PVC é CBR. Duas das máquinas utilizadas em nosso ambiente de teste estão configuradas como roteadores de borda. Essas duas máquinas, além de realizarem a marcação e o policiamento dos pacotes, atuam como fonte e destino do tráfego. Já a terceira máquina está configurada como roteador interior. Essa máquina atua como única rota entre as outras duas. Nela, acontece apenas a classificação e o policiamento dos pacotes.

Nos ensaios realizados, a reserva de banda passante em todos os enlaces para a agregação de fluxos EF é mantida à 500Kbps. Esta alocação representa 25% do enlace de 2Mbps do domínio DS. O restante é destinado ao tráfego de melhor esforço. A tabela 5.1 mostra as reservas e as prioridades relacionadas às classes.

<i>Classe</i>	<i>Prioridade</i>	<i>Largura de banda (Kbps)</i>
PHB EF	1	500
BE	7	1500

Tabela 5.1: Características das Classes de Serviço para um PVC com largura de banda igual a 2Mbps

### 5.1.2 Ferramentas de medição

Para geração e medição de tráfego, foi utilizado o programa *monitor\_udp* em conjunto com o programa *refletor\_udp*, os quais pertencem ao conjunto de programas TMG (*Traffic Monitor Generator*) do projeto Almadem - UFRJ. Foi utilizado também o programa *gerador\_udp* para transmitir um fluxo de fundo, com finalidade de preencher o *link*. Essa ferramenta mede a performance de uma rede levando em conta a largura de banda gerada, a taxa de perda de pacotes, o atraso sofrido por um pacote num caminho fim a fim e o atraso entre os pacotes.

Mais detalhes sobre essa ferramenta de geração e medição de tráfego podem ser encontrados na seção 4.1.

Para configurar as interfaces de rede das máquinas envolvidas no ambiente, foi utilizada a ferramenta TC (*Traffic Controller*). Essa ferramenta é um programa ao nível do usuário que pode ser usada para criar e associar filas aos dispositivos de saída de rede. Ela é usada para inicializar vários parâmetros de fila e associar classes com esses parâmetros de fila. O TC pode ser usado também para inicializar filtros com base numa tabela de roteamento, classificadores u32, classificadores tcindex e classificadores RSVP. Esse programa usa de *netlink sockets* como um mecanismo para comunicação com as funções de rede do kernel. Uma descrição mais detalhada desse controlador de tráfego, pode ser encontrada na seção 4.2.

### 5.1.3 Estratégia para os testes

Para os testes realizados, foram considerados quatro casos distintos. Primeiro, foi transmitido na rede somente tráfego do tipo EF considerando apenas um fluxo. O objetivo desse caso é investigar o comportamento dos mecanismos de reservas de recursos adotados pela ferramenta TC (*Traffic Controller*). No segundo caso, foi transmitido na rede somente tráfego do tipo EF, considerando dois fluxos. O objetivo aqui é o mesmo do caso anterior, acrescido do desejo de investigar o comportamento de um fluxo do tipo EF na presença de outro fluxo desse mesmo tipo. Em um terceiro caso, foram transmitidos na rede tráfego EF, considerando apenas um fluxo, e tráfego BE, com a finalidade de investigar o comportamento do tráfego EF na presença do tráfego melhor esforço. E por fim foram transmitidos na rede tráfego do tipo EF, agora considerando dois fluxos, e tráfego do tipo Melhor Esforço (*Best Effort* - BE). O objetivo aqui é investigar o comportamento de dois fluxos EF na presença do tráfego do tipo BE. Esses dois últimos casos e suas análises caracterizam os principais resultados desse trabalho. As seções a seguir mostram os resultados para esses quatro casos.

Os ensaios realizados envolvem fontes CBR (*Constant Bit Rate*) para a modelagem do tráfego de mídias contínuas. Esse fato limita nosso ambiente de testes porque várias aplicações envolvendo mídias contínuas utilizam fontes VBR (*Variant Bit Rate*). Utilizou-se fontes desse tipo porque as ferramentas do ambiente utilizadas para gerar o tráfego só trabalham com esse tipo de fonte. Em nossos experimentos, essas fontes são configuradas de acordo com o tipo de classe de serviço que está sendo transmitido, conforme veremos nas seções a seguir.

É válido ressaltar também que para a manutenção de aplicações de tempo-real interativas, por exemplo áudio e vídeo numa video-conferência, requer um atraso máximo de 200ms para que se tenha uma satisfação do usuário no nível da aplicação. Já as aplicações de tempo-real não-interativas, tais como fluxos contendo áudio e vídeo (*streaming*), requerem um atraso as

vezes até de alguns poucos segundos. Uma baixa variação neste retardo (*jitter*) também é importante, pois diminui a quantidade de *buffers* nos receptores, necessários para compensar estas variações (Xue e et al. 1999).

## 5.2 Experimentos e Resultados Obtidos

Os resultados apresentados nesta seção são medidos de uma fonte EF e/ou BE de referência localizada no nó A (Edge - figura 4.6) que transmite para o destino no nó C (Edge - figura 4.6).

Em todos os ensaios, os dados para as duas classes de serviço foram informados dentro do TC (*Traffic Controller*). Os parâmetros do TC foram modificados com o objetivo de marcar o *byte* TOS no cabeçalho dos pacotes. Com isso, cada pacote ao sair pela interface de rede da máquina Roteador de Borda (*Edge*) tem seu *byte* TOS assinalado como 0xb8 ou 0x00 para indicar tráfego EF ou BE, respectivamente.

Apesar de uma determinada taxa de tráfego ser gerada no nível da aplicação, observamos nos gráficos que há sempre um deslocamento inferior com relação ao ponto analisado. Isso ocorre porque a taxa gerada no nível da aplicação nunca coincide com a taxa de transmissão efetiva, uma vez que temos que considerar o desempenho do Processador e do Sistema Operacional. Por exemplo quando geramos no nível da aplicação uma largura de banda de 500Kbps, temos efetivamente uma taxa gerada no nível da transmissão, aproximadamente igual a 450Kbps. No entanto, para facilitar a análise com relação aos gráficos de resultado, vamos nos referir sempre à taxa gerada no nível da aplicação.

### 5.2.1 Caso tráfego EF - Um Fluxo

Inicialmente foi investigado o serviço contratado como Serviço EF. Para isso, na rede reservada, todo tráfego gerado foi marcado como EF. O tráfego gerado foi marcado como EF na interface de rede de saída na própria máquina fonte. Levando em conta que foram contratados e reservados 500Kbps para esse tipo de tráfego, o processo de geração e medição de tráfego foi repetido para três níveis de largura de banda diferentes conforme mostra a tabela 5.2: primeiro abaixo do contratado, ou seja, uma taxa menor que 500Kbps; segundo, igual ao contratado e por fim, uma taxa igual a 1000Kbps, isto é, o dobro do contratado. Para cada nível de largura de banda, rodou-se cinco vezes o *monitor\_udp* e o *refletor\_udp*, escolhendo o resultado que indicava a melhor performance da rede. Para cada um dos cinco testes, foram gerados 20000 (vinte mil) pacotes, do tipo UDP, variando os demais parâmetros como intervalo entre as gerações de pacote, tamanho de cada pacote, endereços IP, portas, etc. A escolha de cinco rodadas serve como

uma garantia adicional, já que medidas da ordem de microssegundos podem ser afetadas por execuções adversas dos sistemas operacionais dos equipamentos.

Fluxos	Reservado (Kbps)	Gerado (Kbps)
EF	500	250
EF	500	500
EF	500	1000

Tabela 5.2: Largura de banda reservada e gerada para o fluxo EF

A figura 5.1 mostra o resultado quando foi gerado o tráfego para os três níveis de largura de banda para o serviço EF. Nota-se que a perda de pacotes é próxima de zero no ponto *a* onde a taxa de tráfego esta abaixo do contratado, ou seja, a taxa gerada e medida é menor que 500Kbps. No ponto *b* onde temos uma taxa gerada muito próxima da taxa contratada, a taxa de descarte de pacotes aproximadamente 2%. Esse valor apesar de ser próximo de zero, para algumas aplicações não é aceitável. Entretanto, temos no ponto *c* onde a taxa gerada é igual ao dobro da taxa contratada, isto é, 1000Kbps, uma grande perda de pacotes. Cerca de 52% dos pacotes transmitidos, são perdidos. Isso se explica pelo fato do serviço EF utilizar apenas os recursos contratados. Esse tipo de serviço de maneira alguma tenta utilizar de mais recursos da rede, além do contratado. Mesmo que haja largura de banda ociosa na rede.

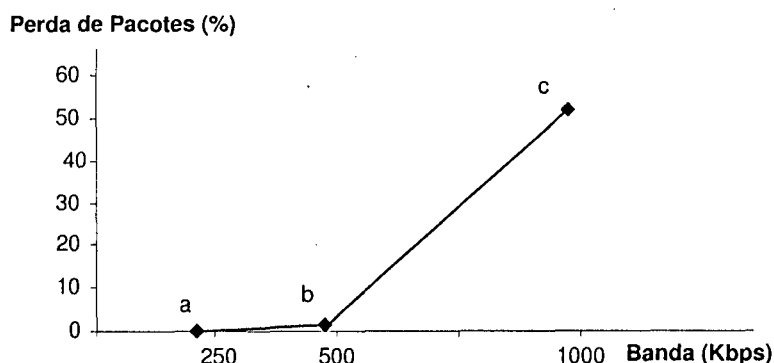


Figura 5.1: Perda de pacotes com relação ao tráfego gerado

O atraso médio sofrido por pacote e o atraso médio entre pacotes, como se observa na figura 5.2 nos pontos *a* e *d*, é aproximadamente 1 milissegundo para o nível abaixo do contratado. Valor fortemente inferior ao exigido numa transmissão de áudio e/ou vídeo. Nota-se também um bom comportamento com relação ao atraso médio por pacote e ao atraso médio entre pacotes para o tráfego EF igual ao contratado. Por outro lado, percebe-se que o atraso por pacote é considerável, assim como o atraso entre pacotes quando temos uma taxa de tráfego igual ao dobro do contratado. Aqui o atraso por pacote chega a praticamente dobrar com relação a quando temos tráfego igual ao contratado. Porém mesmo assim, o atraso tanto por pacote quanto entre pacotes é considerado baixo. Isso porque no momento em que ocorre uma saturação do enlace reservado para o fluxo EF, os pacotes sofrem um pequeno atraso, sendo logo descartados. Esses parâmetros estão relacionados à quantidade de tráfego que ultrapassa o contratado.

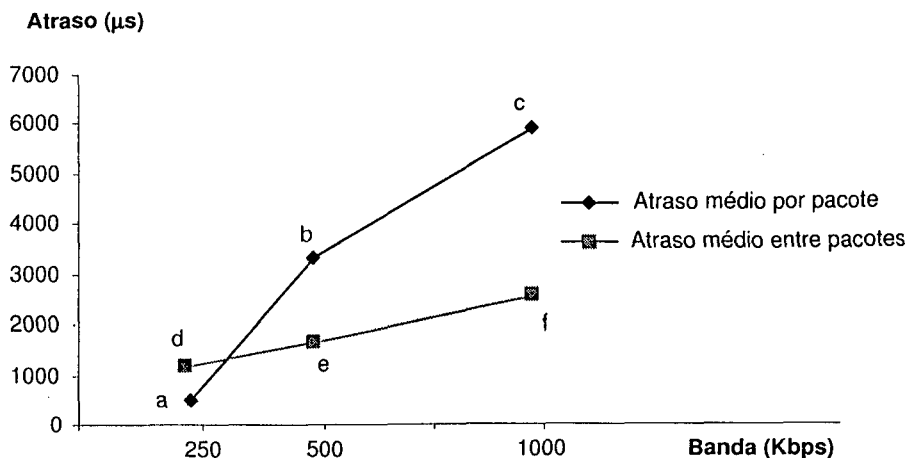


Figura 5.2: Atrasos sofrido pelo fluxo EF com relação ao tráfego gerado

### 5.2.2 Caso tráfego EF - Dois Fluxos

Outro caso considerado e analisado em nossos testes foi a geração de dois fluxos classificados como EF. Foram estabelecidas prioridades diferentes para cada fluxo. O fluxo EF1 recebeu prioridade igual a 1. Isso o tornou mais prioritário que o fluxo EF2, o qual possui prioridade igual a 3. O objetivo desse caso, onde temos dois fluxos com pacotes recebendo a mesma marcação, é investigar, além do comportamento de um fluxo EF na presença de outro fluxo do mesmo tipo, o tratamento por fluxo no roteador de borda e o tratamento agregado no roteador interior. Vale lembrar aqui que já no roteador de borda do domínio DS, os pacotes recebem um primeiro policiamento. Esse policiamento atua como controle de admissão. Porém esse controle de admissão trabalha com fluxos agregados. Esse tratamento agregado é conseguido nos nós interiores porque no interior do domínio DiffServ, são feitos o policiamento e o encaminhamento considerando apenas o conteúdo do *byte* TOS no cabeçalho do pacote. E no roteador interior, ambos os fluxos estão marcados como EF. Então, para que se tenha um encaminhamento dos fluxos EF, temos que saber, a priori, qual a quantidade de recursos da rede a soma de todos os serviços do tipo EF irá ocupar. O papel do SLA (*Service Level Agreement*) é fundamental para isto, estabelecendo um Acordo de Nível de Serviço para o serviço de encaminhamento oferecido a um cliente por um provedor.

Nesse segundo caso nós conservamos a quantidade de largura de banda contratada anteriormente para o tráfego EF, isto é, 500Kbps e variamos as fontes geradoras de tráfego. Primeiro foram gerados em paralelo dois tráfegos classificados como EF numa taxa igual a 125Kbps cada. Isso fez com que os fluxos EF trafegassem na rede utilizando uma largura de banda igual a 250Kbps, o que corresponde à metade da capacidade do enlace reservado para esse fluxo. Num segundo instante, foi gerado um tráfego de 250Kbps para cada um dos dois fluxos EF, ou seja, foi ocupado todo o enlace de 500Kbps reservado para esse tipo de serviço. E, por fim foram gerados, em paralelo, dois fluxos EF com uma taxa de transmissão igual a 500Kbps para cada.

Isso fez com que o limite contratado fosse ultrapassado em 50%. A tabela 5.3 mostra esses valores para os fluxos EF1 e EF2. Esses valores foram estabelecidos para serem similares aos valores do caso de um fluxo.

Fluxos	Reservado (Kbps)	Gerado (Kbps)
EF1	250	125
EF2	250	125
EF1	250	250
EF2	250	250
EF1	250	500
EF2	250	500

Tabela 5.3: Largura de banda reservada e gerada para os fluxos EF1 e EF2

A figura 5.3 mostra justamente essa competição por recurso pelos fluxos EF. Percebe-se através do resultado apresentado nessa figura, que quando temos uma taxa de transmissão, somando os dois fluxos, igual a 250Kbps, a perda de pacotes é igual a zero para os dois fluxos (pontos *a* e *d*). Porém quando temos uma taxa de 250Kbps para cada tráfego EF, a perda ou descarte de pacotes permanece zero (ponto *b*) para o fluxo EF1 e chega a aproximadamente 2% para o fluxo EF2 (ponto *e*). No entanto quando geramos um tráfego de 500Kbps para cada fluxo, totalizando na rede um tráfego de 1000Kbps, isto é, o dobro do contratado, temos uma alta taxa de perda de pacotes para ambos os fluxos, sendo que bem maior para o fluxo EF2 conforme observarmos na figura 5.3 (pontos *c* e *f*).

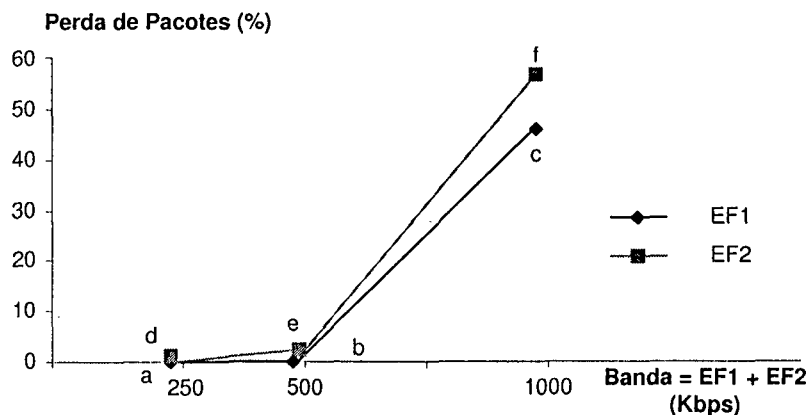


Figura 5.3: Perda de pacotes numa transmissão envolvendo dois fluxos EF

Com relação ao atraso médio sofrido pelos pacotes nesse segundo caso de testes, podemos observar o gráfico de resultado na figura 5.4. Como se percebe, o atraso é pequeno nos pontos *a* e *d* onde temos um tráfego total dos fluxos dentro do limite de 500Kbps. Porém o atraso médio chega a 10ms (ponto *f*) para o tráfego de menor prioridade quando temos na rede uma carga equivalente ao dobro da carga contratada.

Finalmente, a figura 5.5 apresenta o atraso médio entre os pacotes para esse segundo caso de testes. Nota-se, observando os pontos *a* e *d*, que quando a carga do tráfego está dentro do total



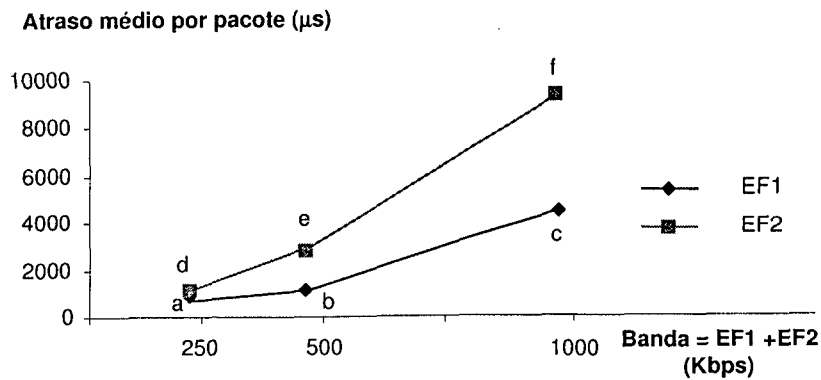


Figura 5.4: Atraso médio por pacote numa transmissão envolvendo dois fluxos EF

contratado para o tráfego EF, o tráfego de menor prioridade chega a ter um atraso médio entre pacotes próximo ao valor do atraso médio entre pacotes para o fluxo mais prioritário. Isso se repete para os pontos *b* e *e* onde temos a soma da taxa de transmissão para os dois fluxos igual ao contratado. No entanto, a partir do momento que o tráfego ultrapassa o contratado, como era de se esperar, o fluxo menos prioritário volta a sofrer maior atraso entre pacotes.

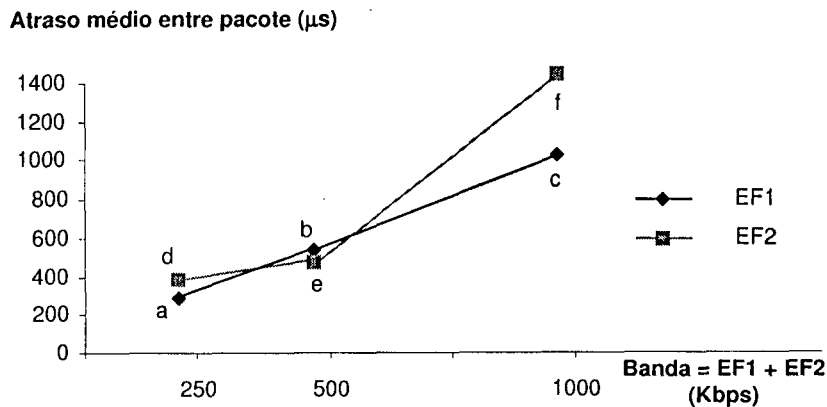


Figura 5.5: Atraso médio entre pacotes antes e depois da alocação de mais recursos na rede

### 5.2.3 Caso Tráfego EF (um fluxo) e Tráfego BE

Apresentados os devidos testes com relação ao tráfego EF individualmente, vamos agora investigar como esse tipo de tráfego se comporta, com relação a taxa de perda, ao atraso por pacote e ao atraso entre pacotes, na presença do tráfego *Best Effort* - BE. Vale dizer que a avaliação desse caso, assim como do próximo, onde temos tráfego do tipo EF e tráfego do tipo BE competindo por recursos da rede, é um dos objetivos principais da análise nesse trabalho. Para gerar e medir dois tipos de tráfegos diferentes, executou-se em paralelo, dois *monitor\_udp* e dois *refletor\_udp*. Esse é o mesmo procedimento para a geração dos fluxos EF no caso anterior. Porém aqui, um fluxo é marcado como EF e o outro como BE. Vale lembrar que a remarcação do tráfego BE é para que tenhamos um melhor controle também sobre esse tipo de fluxo, pois por padrão, os

pacotes já estão marcados como melhor esforço.

Consideramos para nosso terceiro caso três situações diferentes para o tráfego EF: primeiro com a taxa de transmissão igual a 250Kbps. Segundo, com a taxa de transmissão igual a 500Kbps, isto é, igual a taxa contratada para esse tipo de serviço. E terceiro, uma taxa de transmissão igual a 550Kbps (10% acima do contratado). Para cada situação dessa, variamos o valor da banda gerada para o tráfego BE com a intenção de manter a taxa de banda total abaixo da capacidade do enlace (1500Kbps) num primeiro instante, depois igual a 2Mbps num segundo instante e, por fim, acima da capacidade total do enlace (2500Kbps). A tabela 5.4 mostra os valores das taxas reservadas e geradas para os fluxos EF e BE.

Primeira situação			Segunda situação			Terceira situação		
Fluxos	Reservado (Kbps)	Gerado (Kbps)	Fluxos	Reservado (Kbps)	Gerado (Kbps)	Fluxos	Reservado (Kbps)	Gerado (Kbps)
EF	500	250	EF	500	500	EF	500	550
BE	1500*	1250	BE	1500*	1000	BE	1500*	950
EF	500	250	EF	500	500	EF	500	550
BE	1500*	1750	BE	1500*	1500	BE	1500*	1450
EF	500	250	EF	500	500	EF	500	550
BE	1500*	2250	BE	1500*	2000	BE	1500*	1950

\* Limite igual ao restante do enlace

Tabela 5.4: Largura de banda reservada e gerada para os fluxos EF e BE

As figuras 5.6, 5.7 e 5.8 mostram os resultados com relação a taxa de perda tanto para o tráfego EF quanto para o tráfego BE nas situações descritas anteriormente, respectivamente para uma taxa de 250Kbps, 500Kbps e 550Kbps para o tráfego EF.

A figura 5.6 apresenta a situação onde o tráfego EF é igual a 250Kbps, enquanto que nos pontos *a* e *d* o tráfego BE é igual a 1250Kbps; a perda é zero para ambos os tipos de tráfegos. O mesmo acontece nos pontos *b* e *e*, onde a taxa do tráfego BE é igual a 1750Kbps. Isso se explica pelo fato da taxa de transmissão do tráfego EF estar dentro do contratado, assim como a taxa total não ultrapassar o limite do enlace nesses pontos. Entretanto, nos pontos *c* e *f* onde taxa de tráfego BE é igual a 2250, temos um aumento no descarte de pacotes apenas com relação ao tráfego BE. Isso já era de se esperar, uma vez que o limite do enlace foi ultrapassado em 25%. O fato da taxa de tráfego EF estar muito abaixo do contratado, explica o descarte igual a zero para esse tipo de tráfego em todos os pontos da figura.

Na figura 5.7 para uma taxa constante de 500Kbps para o tráfego EF, temos nos pontos *a* e *d* uma taxa de 1000Kbps para o tráfego BE. Nos pontos *b* e *e* temos uma taxa de transmissão 1500Kbps para o tráfego BE. E, por fim, temos nos pontos *c* e *f* uma taxa de 2000Kbps para o tráfego BE. Notamos aqui que o comportamento da taxa de perda para os fluxos EF e BE é semelhante ao apresentado na figura 5.6. Porém quando foi gerado um tráfego acima da capacidade do enlace (pontos *c* e *f*), notamos que o tráfego EF sofre um aumento de perda de 2,3% com relação aos pontos anteriores onde tínhamos uma taxa total de 1500 e 2000Kbps. Isso pode

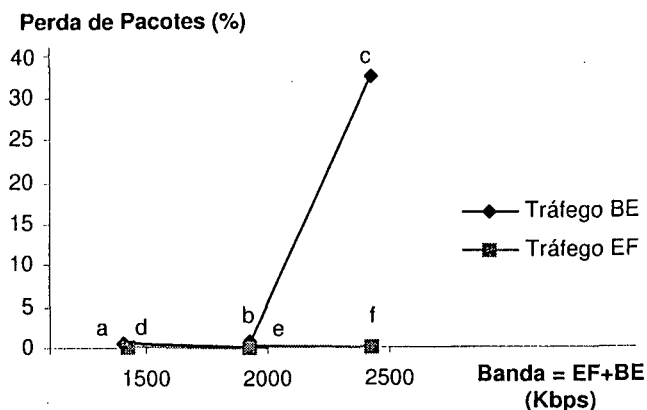


Figura 5.6: Perda de pacotes - tráfego EF (250Kbps) e BE

ser explicado pelo fato de nos pontos *c* e *f*, o limite do enlace de 2Mbps ser ultrapassado e de a taxa reservada para o tráfego EF (500Kbps) ser totalmente utilizada. Sendo assim, mesmo tendo uma taxa reservada de 500Kbps para o tráfego EF, alguns pacotes desses acabam permanecendo em fila ou sendo descartado.

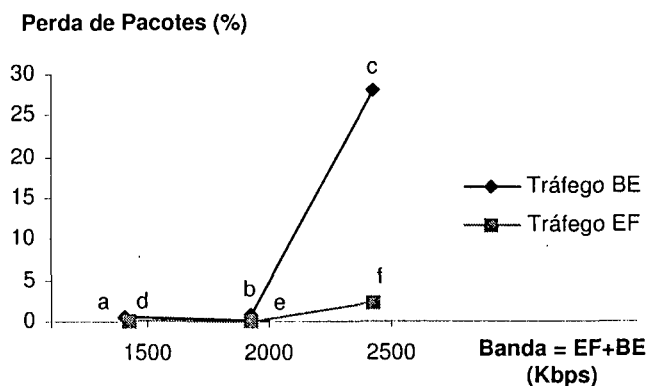


Figura 5.7: Perda de pacotes - tráfego EF (500Kbps) e BE

Finalmente ao analisarmos os resultados apresentados na figura 5.8 onde temos uma taxa constante igual a 550Kbps para o tráfego EF, percebemos uma taxa de descarte para o tráfego EF em todos os pontos. Isso pode ser explicado porque nesse gráfico, em todos os pontos, a taxa de transmissão para esse tipo de tráfego foi de 10% acima do contratado. Já o tráfego BE teve taxas gerada de 950Kbps, 1450Kbps e 1950Kbps para os pontos (*a* e *d*), (*b* e *e*) e (*c* e *f*), respectivamente. Isso fez com que para o tráfego BE, tivéssemos um descarte maior apenas no ponto *f*, quando ultrapassou-se a banda reservada para o PVC.

Quanto ao atraso médio por pacote durante a transmissão, podemos observar os resultados nas figuras 5.9, 5.10 e 5.11. Nos pontos *a* e *d* da figura 5.9, temos uma taxa de transmissão para o tráfego EF igual a 250Kbps e uma taxa de 1250Kbps para o tráfego BE. Nesses pontos percebemos que o atraso está com um valor próximo de 1000µs. No entanto quando a carga da rede se iguala a capacidade total do enlace (pontos *b* e *e*) o atraso para o tráfego EF permanece o mesmo enquanto que o atraso para o tráfego BE chega a aumentar 80%. Quando há a saturação

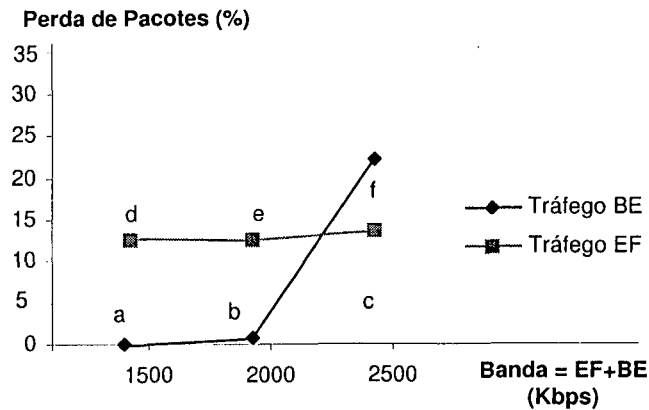


Figura 5.8: Perda de pacotes - tráfego EF (550Kbps) e BE

da rede (pontos *c* e *f*) ficando a taxa de transmissão em torno de 2500Kbps, o valor do atraso para o tráfego EF sofre um aumento de 65%, e o atraso para BE tem aumento de aproximadamente 100%.

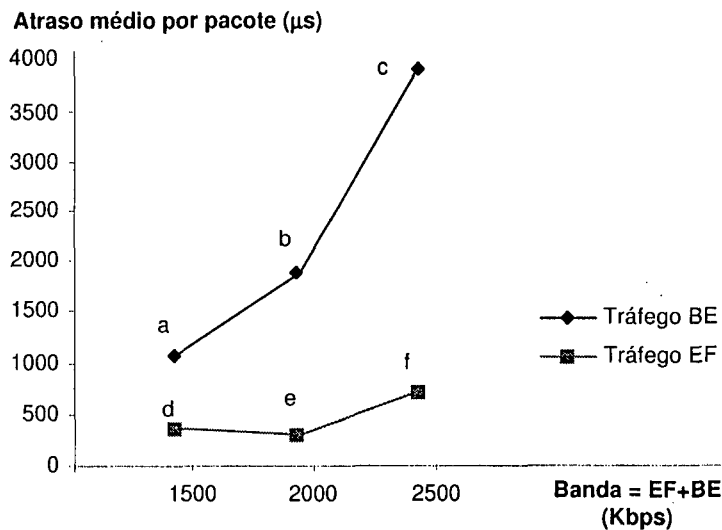


Figura 5.9: Atraso médio por pacote - tráfego EF (250Kbps) e BE

Porém, mesmo assim esse é um atraso médio muito baixo. A explicação para um atraso tão baixo é o fato de termos uma taxa de transmissão para o tráfego EF igual a metade do contratado.

Já a figura 5.10 nos mostra o resultado do teste na situação onde a taxa para o tráfego EF é igual a 500Kbps e a taxa para o tráfego BE é 1000Kbps nos pontos *a* e *d*, 500Kbps para o tráfego EF e 1500Kbps para o tráfego BE nos pontos *b* e *e* e, por fim uma taxa ainda igual a 500Kbps para o tráfego EF contra uma taxa de 2000Kbps para o tráfego BE nos pontos *c* e *f*. Observando os resultados do gráfico, nota-se um aumento no atraso para o tráfego BE próximo a 60% entre os pontos *a* e *b* contra um aumento próximo a 75% entre os pontos *b* e *c*. Enquanto isso, o aumento no valor do atraso para o tráfego EF é de aproximadamente 18% entre os pontos *d* e *e* e de 13% entre os pontos *e* e *f*. O aumento no atraso ser menor para o tráfego EF se explica pelo fato desse tráfego permanecer dentro da taxa contratada a priori para esse tipo de tráfego,

em todos os pontos do gráfico. Sendo assim, o tráfego BE que sofre com a saturação do enlace.

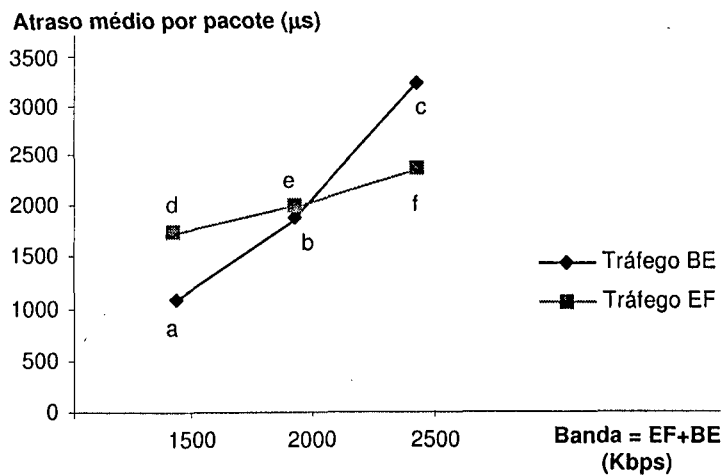


Figura 5.10: Atraso médio por pacote - tráfego EF (500Kbps) e BE

E, ainda com relação ao atraso sofrido por pacotes na rede, temos a figura 5.11, a qual mostra os resultados dos testes na rede quando temos uma taxa de tráfego EF igual 550Kbps; nesta situação temos para o tráfego BE uma taxa gerada igual a 950Kbps nos pontos *a* e *d*, 1450Kbps nos pontos *b* e *e*, finalmente uma taxa igual a 1950Kbps nos pontos *c* e *f*. Notamos um atraso médio por pacotes para o tráfego EF maior do que para o tráfego BE em todos os pontos do gráfico, com exceção do ponto *f*, onde temos a saturação do enlace. Isso era de se esperar, porque o tráfego EF ultrapassou o valor contratado desde o início da transmissão.

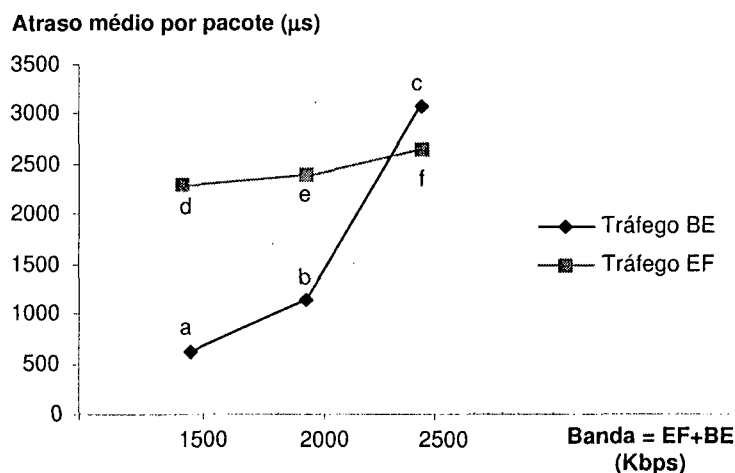


Figura 5.11: Atraso médio por pacote - tráfego EF (550Kbps) e BE

As figuras 5.12, 5.13 e 5.14 nos apresentam os resultados com relação ao atraso entre pacotes na transmissão para as taxas respectivas de 250Kbps, 500Kbps e 55Kbps para o tráfego EF. Nos pontos *a* e *d* da figura 5.12 para o valor da taxa de transmissão para o tráfego EF igual a 250Kbps temos uma taxa igual a 1250Kbps para o tráfego BE. Temos nos pontos *b* e *e* 1750Kbps para o tráfego BE. Por fim, temos nos pontos *c* e *f* uma taxa de transmissão igual a 2250Kbps para o tráfego BE.

Quando observarmos os resultados apresentados por essa figura, percebemos que o atraso médio entre pacotes permanece praticamente o mesmo em todos os pontos, com um valor próximo de  $1500\mu s$ , para o tráfego classificado como EF. Já para o tráfego do tipo BE, o aumento do atraso entre os pontos *b* e *c* é duas vezes maior entre os pontos *a* e *b*. Isso já era de se esperar porque como a taxa de tráfego para o tráfego EF está fixada abaixo da taxa contratada, ao carregarmos o enlace, o tráfego mais atingido é o de menor prioridade, ou seja, o de melhor esforço.

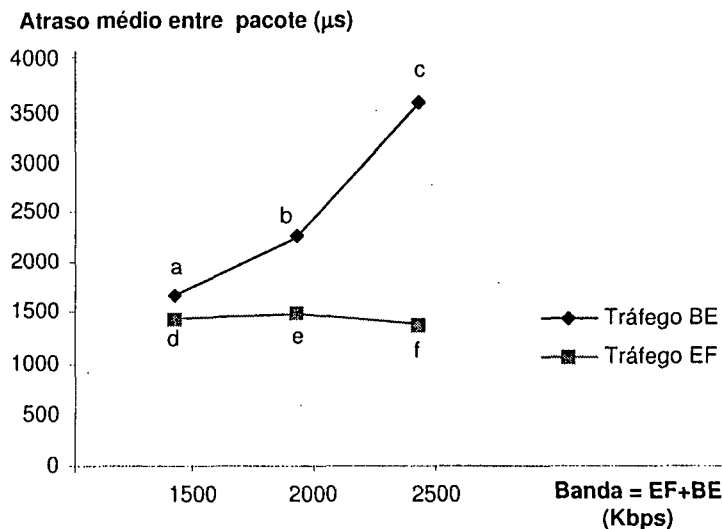


Figura 5.12: Atraso médio entre pacotes - tráfego EF (250Kbps) e BE

A figura 5.13 apresenta os resultados para uma taxa de transmissão igual a 500Kbps para o tráfego EF em todos os pontos. Notamos que esse tráfego sofre um pequeno aumento no valor do atraso médio entre pacotes de aproximadamente  $1000\mu s$  entre os pontos *d* e *e*, chegando ao valor de  $2000\mu s$ . E praticamente, permanece com esse valor, mesmo ao saturarmos a rede. O desvio aqui é pequeno porque, mesmo o enlace estando saturado, o tráfego EF permanece dentro do limite contratado.

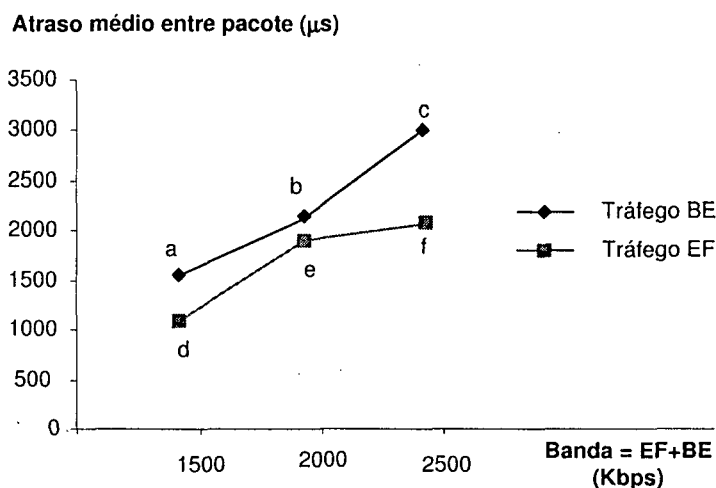


Figura 5.13: Atraso médio entre pacotes - tráfego EF (500Kbps) e BE

Por outro lado, conforme podemos apreciar no gráfico da figura 5.14 para um tráfego de 550Kbps para EF, esse tráfego possui um atraso médio entre pacotes maior que o do tráfego BE em todos os pontos, com exceção do ponto *f* onde temos a saturação do enlace. Isso porque nesse gráfico estão sendo mostrados os resultados dos testes feitos quando se tinha uma taxa de transmissão para o tráfego EF 10% acima do contratado. Observa-se também nessa figura que o atraso médio entre pacotes para o tráfego EF sobe 38% entre os pontos onde a carga total da rede é igual a 1500 e 2000Kbps contra 10% entre os pontos *e* e *f*. No entanto, o grande aumento constatado com relação ao atraso médio por pacote foi para o tráfego BE entre os pontos *b* e *c* (aproximadamente 60%). Isso porque entre esses pontos, temos a saturação do enlace.

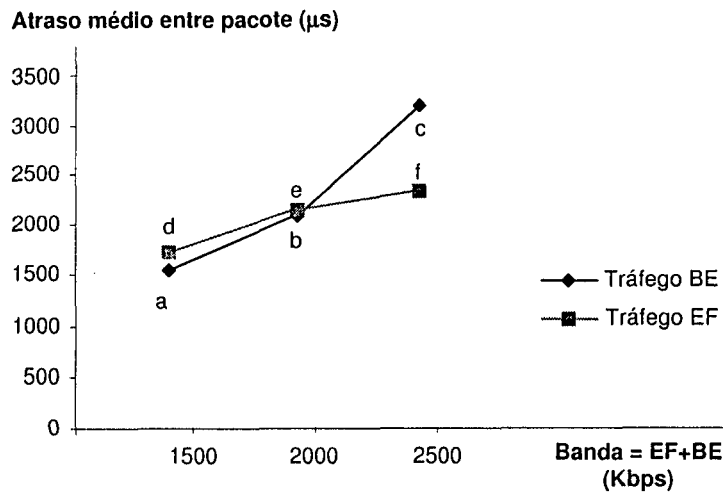


Figura 5.14: Atraso médio entre pacotes - tráfego EF (550Kbps) e BE

#### 5.2.4 Caso Tráfego EF (dois fluxos) e Tráfego BE

Nesse caso procuramos investigar o comportamento de dois fluxos do tipo EF na presença do tráfego BE. Os fluxos EF foram denominados EF1 e EF2. Estabelecemos para este caso que EF1 é mais prioritário que EF2. Para gerar e medir três fluxos, nós executamos de forma simultânea três *monitor\_udp* e três *refletor\_udp*. Aqui, dois fluxos são marcados como EF com suas devidas prioridades (1 e 3 respectivamente), conforme já dito, e um como BE.

Seguindo o cenário de testes apresentado no caso anterior, consideramos para este caso três situações diferentes para o tráfego EF: taxa de transmissão igual a 250Kbps, 500Kbps e 550Kbps, sendo essas taxas divididas igualmente entre os dois fluxos EF1 e EF2. Para cada situação dessa, variamos o valor da banda gerada para o tráfego BE com a intenção de manter a taxa de banda total abaixo da capacidade do enlace (1500Kbps) num primeiro instante, depois igual a 2000Kbps num segundo instante e, por fim, acima da capacidade total do enlace (2500Kbps). A tabela 5.5 mostra os valores das taxas reservadas e geradas para os fluxos EF1, EF2 e BE.

Primeira situação			Segunda situação			Terceira situação		
Fluxos	Reservado (Kbps)	Gerado (Kbps)	Fluxos	Reservado (Kbps)	Gerado (Kbps)	Fluxos	Reservado (Kbps)	Gerado (Kbps)
EF1	250	125	EF1	250	250	EF1	250	275
EF2	250	125	EF2	250	250	EF2	250	275
BE	1500*	1250	BE	1500*	1000	BE	1500*	950
EF1	250	125	EF1	250	250	EF1	250	275
EF2	250	125	EF2	250	250	EF2	250	275
BE	1500*	1750	BE	1500*	1500	BE	1500*	1450
EF1	250	125	EF1	250	250	EF1	250	275
EF2	250	125	EF2	250	250	EF2	250	275
BE	1500*	2250	BE	1500*	2000	BE	1500*	1950

\* Limite igual ao restante do enlace

Tabela 5.5: Largura de banda reservada e gerada para os fluxos EF1, EF2 e BE

A figuras 5.15, 5.16 e 5.17 mostram os resultados com relação a taxa de perda para o tráfego EF1, para o tráfego EF2 e para o tráfego BE nas situações descritas anteriormente.

Observando a figura 5.15 onde a taxa gerada tanto para o tráfego EF1 quanto para o tráfego EF2 é 125Kbps, notamos que nos pontos *a*, *d* e *g*, quando temos uma taxa para o tráfego BE é igual a 1250Kbps, a perda é zero para os três tráfegos. O mesmo ocorre nos pontos *b*, *e* e *h*, onde a taxa do tráfego BE é igual a 1750Kbps. Isso se explica pelo fato da taxa de transmissão do tráfego EF1 somada a taxa do tráfego EF2 estar dentro do contratado, assim como a taxa total não ultrapassar o limite do enlace nesses pontos. Entretanto, nos pontos *c*, *f* e *i* onde a taxa de tráfego BE é igual a 2250, temos um aumento no descarte de pacotes apenas com relação ao tráfego BE. Isso já era de se esperar, uma vez que o limite do enlace foi ultrapassado em 25%. O fato da soma das taxas dos tráfegos EF estar muito abaixo do contratado, explica o descarte igual a zero para esse tipo de tráfego em todos os pontos da figura.

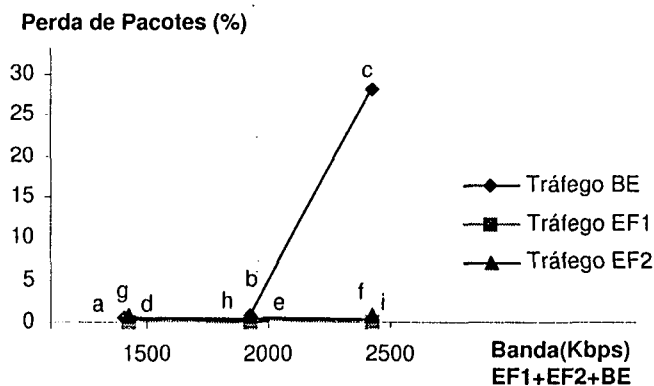


Figura 5.15: Perda de pacotes - tráfegos EF1 (125Kbps), EF2 (125Kbps) e BE

Na figura 5.16 para a taxa de transmissão igual a 250Kbps tanto para EF1 quanto para EF2, temos nos pontos (*a*, *d* e *g*), (*b*, *e* e *h*) e (*c*, *f* e *i*) uma taxa gerada para o tráfego BE igual a 1000Kbps, 1500Kbps e 2000Kbps respectivamente. Notamos aqui que o comportamento da taxa de perda para os fluxos EF1, EF2 e BE é semelhante ao apresentado na figura 5.15. Porém



quando foi gerado um tráfego acima da capacidade do enlace (pontos *c*, *f* e *i*), notamos que o tráfego EF1 sofre um aumento de perda de 1,8% e o tráfego EF2 sobre um aumento de perda de 2,6% com relação aos pontos anteriores onde temos uma taxa total de 1500 e 2000Kbps. Isso pode ser explicado pelo fato de nos pontos *c*, *f* e *i*, o limite do enlace de 2Mbps ser ultrapassado. Isso faz com que, mesmo os pacotes EF, permaneçam algum tempo na fila ou sejam descartados.

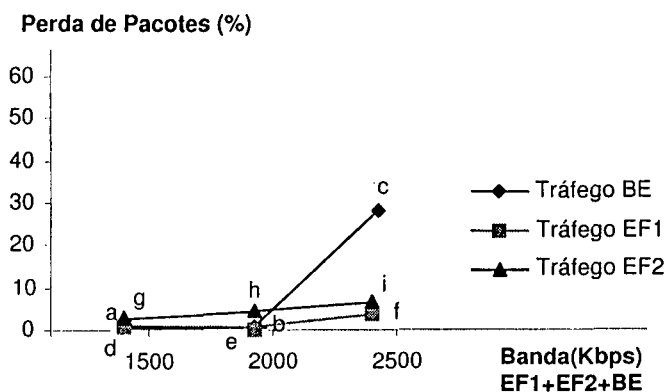


Figura 5.16: Perda de pacotes - tráfegos EF1 (250Kbps), EF2 (250Kbps) e BE

Por fim ao analisarmos os resultados apresentados na figura 5.17 onde temos uma taxa gerada, tanto para EF1 quanto para EF2, igual a 275Kbps, notamos uma taxa de descarte para o tráfego EF2 em todos os pontos do gráfico, assim como para o tráfego EF1. Isso pode ser explicado porque nesse gráfico, em todos os pontos, a taxa de transmissão para o tipo de tráfego EF (EF1 +EF2) foi 10% acima do contratado. No entanto, notamos um descarte um pouco maior para o tráfego EF2. A questão é que no momento da admissão dos fluxos no roteador de borda, o tráfego de menor prioridade sofre uma maior rejeição. Já com relação ao tráfego BE, somente no ponto *c* tivemos uma taxa de descarte maior. Isso pode ser explicado pelo fato de nesse ponto, termos a saturação do enlace.

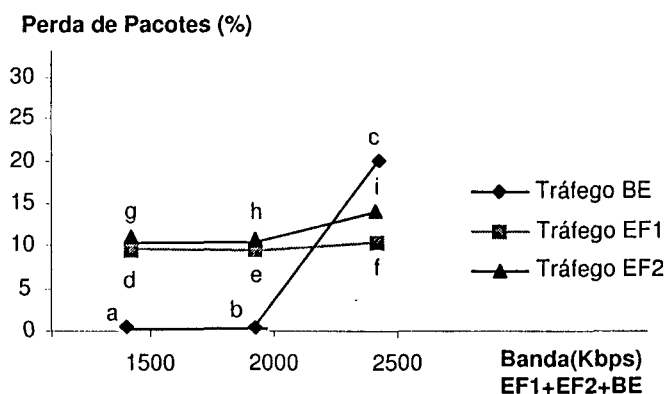


Figura 5.17: Perda de pacotes - tráfegos EF1 (275Kbps), EF2 (275Kbps) e BE

Quanto ao atraso médio por pacote durante a transmissão, podemos observar os resultados nas figuras 5.18, 5.19 e 5.20 onde temos uma taxa gerada igual a 250Kbps, 500Kbps e 550Kbps para o tráfego EF respectivamente, sendo que estas divididas igualmente entre o tráfego EF1 e EF2. Nos pontos *a*, *d* e *h* da figura 5.18, temos uma taxa de transmissão igual a 1250Kbps para

o tráfego BE. Nesses pontos percebemos que o atraso está com um valor próximo de  $1000\mu s$  para o tráfego BE e próximo de  $500\mu s$  para os tráfegos EF1 e EF2. No entanto quando a carga da rede se iguala a capacidade total do enlace (pontos *b*, *e* e *h*) o atraso para o tráfego EF1 permanece o mesmo enquanto que o atraso para o tráfego EF2 dobra de valor. Para o tráfego BE nesse momento, o atraso médio por pacote chega a aumentar cerca de 50%.

Quando há a saturação da rede (pontos *c*, *f* e *i*) ficando a taxa de transmissão total em torno de 2500Kbps, o atraso médio para o tráfego EF1 continua o mesmo. Porém para o tráfego EF2, cuja prioridade é menor, sofre um aumento de aproximadamente 15% com relação ao ponto anterior onde a taxa total era 2000Kbps. Porém, mesmo assim esse é um atraso médio muito baixo. A explicação para um atraso tão baixo está no fato de termos uma taxa de transmissão para o tráfego EF1 e EF2 igual a metade do contratado.

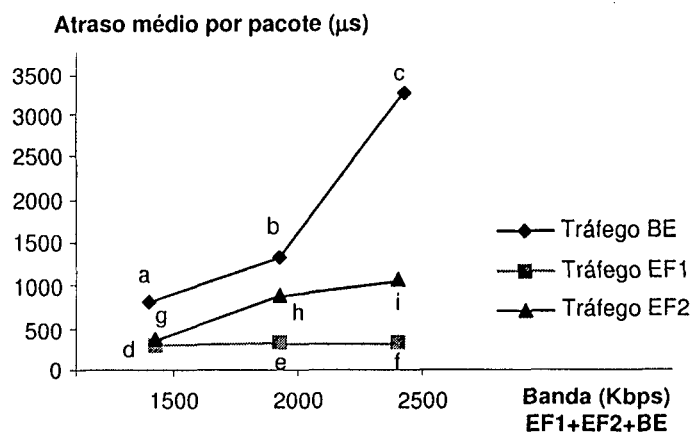


Figura 5.18: Atraso médio por pacote - tráfego EF1 (125Kbps), EF2 (125Kbps) e BE

Já a figura 5.19 nos mostra o resultado do teste na situação onde a taxa gerada para o tráfego BE é 1000Kbps nos pontos *a*, *d* e *g*, 1500Kbps nos pontos *b*, *e* e *h* e, 2000Kbps nos pontos *c*, *f* e *i*. Observando os resultados do gráfico, nota-se que o atraso médio para o tráfego BE praticamente não mudou entre os pontos *a* e *b* contra um aumento de duas vezes mais entre os pontos *b* e *c*. Enquanto isso, o aumento no valor do atraso médio por pacote para o tráfego EF2 é de aproximadamente 12% entre os pontos *g* e *h* e de 8% entre os pontos *h* e *i* com relação ao seu valor quando a taxa de transmissão total era de 2000Kbps. O aumento menor no atraso para o tráfego EF2 se explica pelo fato desse tráfego permanecer dentro da taxa contratada a priori para esse tipo de tráfego, em todos os pontos do gráfico. Sendo assim, o tráfego BE que sofre com a saturação do enlace.

E, ainda com relação ao atraso médio sofrido por pacotes na rede, temos a figura 5.20, com taxa de transmissão para o tráfego BE igual a 950Kbps para tráfego BE nos pontos *a*, *d* e *g*, 1450Kbps nos pontos *b*, *e* e *h* e, 1950Kbps nos pontos *c*, *f* e *i*. Vendo os resultados, notamos que o atraso médio por pacote foi maior para o tráfego EF2 em todos os pontos do gráfico, com exceção do ponto *i*, onde o atraso foi maior para o tráfego BE. Quando houve a sobrecarga do enlace (pontos *c*, *f* e *i*) o atraso aumentou cerca de 10% para EF2. Nesse momento, o atraso

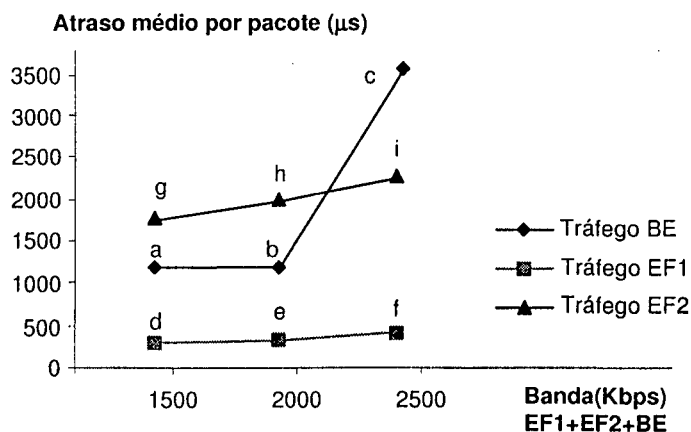


Figura 5.19: Atraso médio por pacote - tráfego EF1 (250Kbps), EF2 (250Kbps) e BE

para o tráfego BE sofreu um aumento igual ao triplo do valor anterior, onde tínhamos uma taxa total de 2000Kbps no enlace. O fato do tráfego do tipo BE sofrer um aumento no atraso médio por pacote tão grande no ponto *c* pode ser explicado pelo fato de termos nesse ponto uma taxa transmitida para esse tráfego muito alta (1950Kbps). Por questão de prioridade, o tráfego EF1 não sofreu quase nada na transmissão.

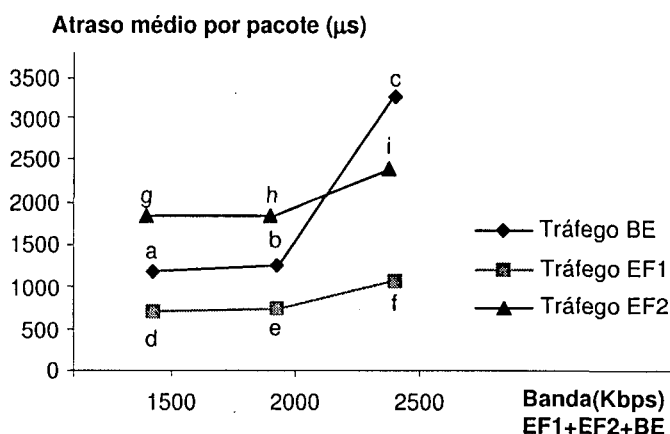


Figura 5.20: Atraso médio por pacote - tráfego EF1 (275Kbps), EF2 (275Kbps) e BE

Nas figuras 5.21, 5.22 e 5.23 temos uma taxa igual a 250Kbps, 500Kbps e 550Kbps, respectivamente, para o tráfego EF, sendo que esses valores divididos igualmente entre os dois fluxos. Elas nos apresentam os resultados com relação ao atraso médio entre pacotes na transmissão. Para o tráfego BE, temos nos pontos *a*, *d* e *g* da figura 5.21 uma taxa gerada igual a 1250Kbps, nos pontos *b*, *e* e *h* um valor de taxa de transmissão igual a 1750Kbps e, por fim, nos pontos *c*, *f* e *i* uma taxa de transmissão igual a 2250Kbps. Quando observarmos os resultados apresentados por essa figura, percebemos que o atraso médio entre pacotes permanece praticamente o mesmo em todos os pontos, com um valor próximo de 1000µs para o tráfego classificado como EF (EF1 e EF2). Já para o tráfego do tipo BE, o aumento do atraso entre os pontos *b* e *c* é quase três vezes maior do que entre os pontos *a* e *b*. Isso já era de se esperar porque como a taxa de tráfego para os tráfegos EF1 e EF2 está fixada abaixo da taxa contratada, ao carregarmos o enlace o tráfego mais atingido é o de menor prioridade dentre os três, ou seja, o de melhor esforço.

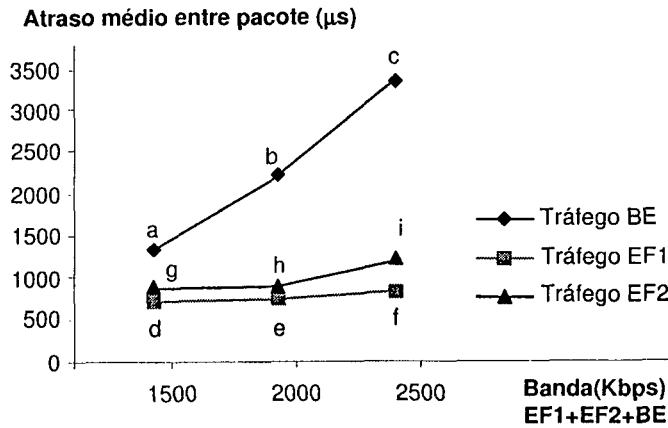


Figura 5.21: Atraso médio entre pacotes - tráfego EF1 (125Kbps), EF2 (125Kbps) e BE

Na figura 5.22, a qual apresenta os resultados quando temos uma taxa de transmissão igual a 250Kbps para cada um dos tráfegos EF1 e EF2 em todos os pontos, podemos notar que o tráfego EF2 sofre um aumento no valor do atraso médio entre pacotes de aproximadamente 50% entre os pontos *g* e *h*, chegando ao valor de 1500µs. Em seguida, esse fluxo sofre um aumento no atraso médio entre pacotes de aproximadamente 30% entre os pontos *h* e *i*. O desvio aqui é pequeno porque, mesmo o enlace estando saturado, os tráfegos EF1 e EF2 permanecem dentro do limite contratado.

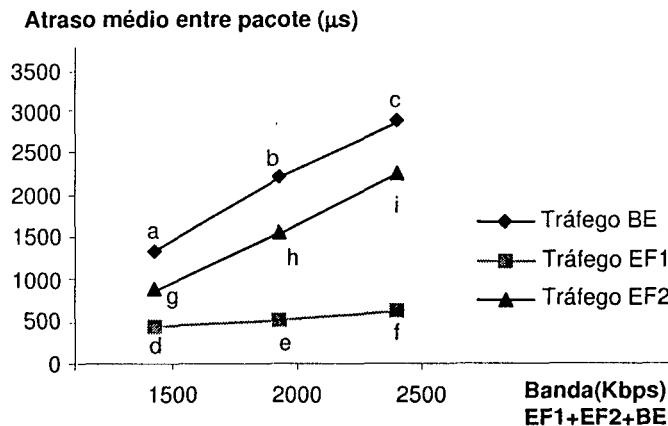


Figura 5.22: Atraso médio entre pacotes - tráfego EF1 (250Kbps), EF2 (250Kbps) e BE

Por outro lado, conforme podemos apreciar no gráfico da figura 5.23, com exceção dos pontos *c*, *f* e *i*, o tráfego EF2 possui um atraso médio entre pacotes próximo do tráfego BE em todos os pontos. Isso porque nesse gráfico estão sendo mostrados os resultados dos testes feitos quando se tinha uma taxa de transmissão para o tráfego EF1 junto com o EF2 10% acima do valor contratado, ou seja, igual 550Kbps. Observa-se também nessa figura que o atraso médio entre pacotes permanece praticamente o mesmo para esse tráfego EF2 (1450µs) em todos os pontos. Porém com relação ao tráfego BE, ao sobrecarregar o enlace, seu atraso médio entre pacotes sofre um aumento de aproximadamente 50% com relação ao ponto *b*. Já o atraso médio entre os pacotes para o tráfego EF1 permanece baixo, em torno de 650µs em todos os pontos. Isso pode ser explicado pelo fato do tráfego EF1 possuir uma maior prioridade com relação ao

tráfego EF2 e ao tráfego BE.

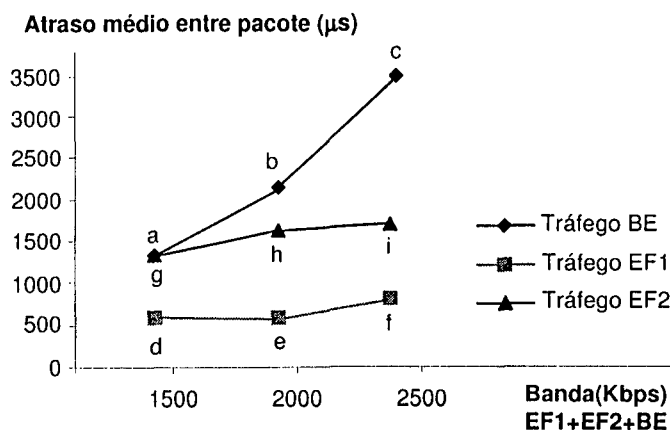


Figura 5.23: Atraso médio entre pacotes - tráfego EF1 (275Kbps), EF2 (275Kbps) e BE

### 5.3 Comentários sobre os Experimentos

Vários ensaios foram realizados sobre o ambiente de teste montado nesse trabalho. Esses ensaios tiveram como meta investigar o comportamento e desempenho de um tráfego ao atravessar por um domínio Serviços Diferenciados.

Para os ensaios, quatro casos foram considerados. Um primeiro onde tivemos apenas um fluxo do tipo Encaminhamento Expresso (EF). Um segundo onde trabalhou-se com dois fluxos do tipo EF, com prioridades diferentes, na rede. O terceiro caso tratou-se de um ambiente onde tivemos um fluxo do tipo EF e outro do tipo Best Effort (BE). Por fim, foi analisado num quarto caso, um ambiente onde se tinha dois fluxos do tipo EF de prioridade diferente e um do tipo BE. Para os dois primeiros casos, considerou-se três cenários onde tínhamos respectivamente tráfego EF abaixo do contratado (250Kbps), tráfego EF igual ao contratado e tráfego EF igual ao dobro contratado (1000Kbps). Já para os dois últimos casos, considerou-se cenários onde se trabalhou respectivamente com um tráfego total abaixo da capacidade do enlace (1500Kbps), tráfego total igual a capacidade do enlace (2000Kbps) e tráfego excedendo a capacidade do enlace em 25% (2500Kbps).

Em todos os quatro casos, como vimos através dos resultados, o tráfego EF tem uma boa performance de encaminhamento na rede, tanto com relação a perdas quanto a atrasos, desde que sua carga não ultrapasse a capacidade contratada. Caso contrário, o descarte é inevitável.

Através do segundo e do quarto caso, foi possível verificar que quando se tem dois tráfegos do tipo EF na rede, no caso de um eventual congestionamento, o tráfego mais atingido, como se era de esperar, foi o tráfego de menor prioridade. Isso porque logo no roteador de borda (*edge*), já ocorre um controle de admissão o qual encaminha de forma privilegiada o tráfego

mais prioritário.

Observando os casos três e quatro, percebe-se como é o comportamento do tráfego do tipo EF na presença do tráfego BE. Nas três situações consideradas para esses dois casos, foi possível observar que o tráfego do tipo EF sofreu maiores perturbações apenas quando sua própria taxa ultrapassou a taxa contratada. E não quando a rede estava sobrecarregada.

A tabela 5.6 faz uma breve comparação entre o caso 1 e o caso 2, considerando apenas a situação onde temos um tráfego EF igual ao dobro do contratado. Na tabela, os fluxos no caso 2 foram transmitidos de forma simultânea. Observando essa tabela, nota-se que há uma perda maior de pacotes quando se tem dois tráfegos EF concorrendo por recursos na rede. Mesmo tendo nas duas situações um tráfego total igual a 1000Kbps, o descarte para um fluxo EF sozinho é igual a 52% contra um descarte de 48% para o fluxo EF1 e um descarte de 53,5% referente ao tráfego EF2. Fica claro aqui então que quando se tem recursos na rede insuficientes para o tráfego EF, tanto no roteador de borda quanto no roteador interior, ocorre uma disputa entre os fluxos desse tipo. Essa disputa faz com que haja perda de pacotes para todos os fluxos do tipo EF, e não apenas para o fluxo de menor prioridade.

Com relação ao atraso médio por pacote nessa situação onde se tem o dobro de tráfego EF sendo transmitido na rede, nota-se uma diferença entre os dois casos. No caso 1 o atraso médio por pacote é de 6000 $\mu$ s. Já para o caso 2 ao se realizar a soma o atraso médio por pacote para os dois fluxos, obtém-se o valor 14000 $\mu$ s. Isso pode ser explicado pelo fato de haver, tanto no roteador de borda quanto no roteador interior, uma concorrência entre os dois fluxos EF por recursos da rede.

	Fluxo	Reservado (Kbps)	Gerado (Kbps)	Taxa de Perda - %	Atraso por pacote - $\mu$ s	Atraso entre pacotes - $\mu$ s
<b>Caso 1</b>	EF	500	1000	52	6000	2550
<b>Caso 2</b>	EF1	250	500	48	4200	1050
	EF2	250	500	53,5	9800	1400

Tabela 5.6: Resultados referentes ao caso 1 e caso 2 para o tráfego total igual a 1000Kbps

Por fim, a tabela 5.7 compara os resultados do caso 3 e do caso 4, considerando a situação onde temos o enlace ultrapassado em 25%, ou seja, um tráfego total igual a 2500Kbps. Esses dois casos têm em comum o fato de trafegar na rede fluxo(s) EF e fluxo BE. Quando se tem um tráfego total para o fluxo EF igual a metade do contratado, em ambos os casos, mesmo tendo a saturação do enlace, o descarte de pacotes é zero. Isso pode ser explicado pelo fato de não haver compartilhamento da banda contratada para o(s) fluxo(s) EF com outros tipos de tráfego, como por exemplo tráfego BE. Porém no momento em que gerou-se um tráfego EF com taxa igual ou superior a contratada o descarte é inevitável, chegando até a 13,5% para o fluxo EF no caso 3 quando se tem uma taxa de 550Kbps para esse fluxo, e 13% para o tráfego EF2 no caso 4 quando se tem uma taxa de transmissão igual a 275Kbps para cada fluxo EF. Aliás o que já

era de se esperar, pois, como foi observado no caso 1 e no caso 2, a rede fornece garantias na transmissão para o tráfego EF desde que este esteja dentro do contratado.

	Fluxo	Reservado (Kbps)	Gerado (Kbps)	Taxa de Perda - %	Atraso por pacote - $\mu$ s	Atraso entre pacotes - $\mu$ s
<b>Caso 3</b>	EF	500	250	0	750	1375
	BE	1500*	2250	34	3950	3670
	EF	500	500	2,3	2400	2050
	BE	1500*	2000	28,5	3250	3000
	EF	500	550	13,5	2600	2350
	BE	1500*	1950	22,4	3000	3250
<b>Caso 4</b>	EF1	250	125	0	300	820
	EF2	250	125	0	1050	1200
	BE	1500*	2250	28	3300	3480
	EF1	250	250	2,2	480	550
	EF2	250	250	5,1	2100	2250
	BE	1500*	2000	27	3520	2800
	EF1	250	275	10,1	1000	750
	EF2	250	275	13	2420	1700
	BE	1500*	1950	20,5	3450	3530

\* Limite igual ao restante do enlace

Tabela 5.7: Resultados referentes ao caso 3 e caso 4 para o tráfego total igual a 2500Kbps

## 5.4 Comentários sobre o ambiente de teste

O fato do ambiente de teste implementado nesse trabalho possuir apenas um roteador interior, não permitiu que se avaliasse o comportamento de um tráfego ao ter que cruzar um domínio de Serviços Diferenciados passando por diversos nós. Esse pode ser considerado um dos motivos pelos quais não se obteve atrasos que possam ser comparados aos atrasos máximos permitidos para tráfego de voz e vídeo numa conversação interativa (200ms) (Xue e et al. 1999). Porém, através dos resultados apresentados foi possível comparar a performance do tráfego do tipo EF individualmente, na presença de outro tráfego EF e, quando esse tráfego compartilha o enlace com um tráfego do tipo melhor esforço.

O ambiente de teste aqui implementado, como visto antes, possui apenas três máquinas. Duas atuando como roteador de borda e uma como roteador interior. Isso fez com que as máquinas de borda tivessem um processamento pesado, pois essas tinham que gerar, marcar, policiar e medir o tráfego. Uma melhor performance sobre as máquinas poderia ter sido obtida caso o ambiente tivesse máquinas exclusivamente para geração e medição de tráfego.

Quanto a ferramenta TMG (*Traffic Monitor Generator*) aqui utilizada para gerar e medir tráfego, pode-se destacar a simplicidade de utilização, assim como o fato de possuir um boa documentação. Essa ferramenta atendeu as necessidades desse trabalho, uma vez que trás como

estatísticas próprias, medidas como tráfego gerado, taxa de perda de pacotes, atraso por pacote e atraso entre pacotes. Parâmetros esses que eram de total interesse para a realização desse trabalho.

Os scripts implementados no âmbito desse trabalho possuem como objetivo principal configurar as interfaces de redes das máquinas envolvidas. Com esses scripts o administrador da rede consegue definir uma máquina como roteador de borda ou roteador interior. As linhas de comando desses scripts executam o aplicativo TC (*Traffic Controller*). Uma dificuldade encontrada aqui é o fato do aplicativo TC não possuir uma boa documentação.

Outros testes podem ser realizados sobre a plataforma de teste. Dentre estes, pode-se destacar a transmissão de dados de vídeo e áudio reais, que por falta de tempo não puderam ser realizados. Além do mais, é possível, apesar de não ter sido implementado nesse trabalho, usar esta plataforma para Encaminhamento Assegurado (AF) e para integração entre o serviço dos tipos Encaminhamento Expresso, Encaminhamento Assegurado e Melhor Esforço (Blake et al. 1998). Nesse caso seria possível ter os serviços classificados como EF que trafegando na rede dentro do perfil contratado, obteriam total garantia de entrega dos pacotes, dentro do prazo exigido. Os pacotes marcados como AF que teria uma prioridade de encaminhamento acima do tráfego BE. Vale lembrar que para cada classe AF, se o tráfego estiver acima do perfil contratado, seus pacotes são remarcados com uma prioridade mais baixa, sendo trocado de classe até chegar na condição de ser do tipo melhor esforço.

Por fim, se esse ambiente de testes fosse dotado de vários roteadores interiores, e não apenas um, seria possível se ter uma avaliação melhor dos parâmetros atraso por pacotes e atraso entre pacotes, pois aumentaria a probabilidade do tráfego na rede permanecer em filas. No entanto, por falta de estrutura física, não foi possível implementar nesse trabalho atual, tal ambiente.



# Capítulo 6

## Conclusão

ESTE trabalho teve como objetivo o estudo mais aprofundado da abordagem Serviços Diferenciados e a verificação de suas propriedades, para a implementação de um ambiente de testes de baixo custo, assim como a validação de seu uso. Aqui procurou-se falar sobre os elementos para controle de tráfego no Linux de uma forma geral. Nesse propósito, foi apresentado como a infra-estrutura existente pode ser estendida com o objetivo de suportar Serviços Diferenciados.

Neste trabalho foi apresentado um estudo sobre a utilização da arquitetura de Serviços Diferenciados para o suporte de tráfegos de mídias contínuas, por exemplo áudio e vídeo. Nos ensaios realizados enfocou-se a utilização do Encaminhamento Expresso para transporte de dados em um ambiente de Serviços Diferenciados numa rede IP. Foram utilizadas fontes de dados do tipo CBR, uma vez que a ferramenta geradora de tráfego aqui utilizada, trabalhava com esse tipo de tráfego.

Essa dissertação trabalhou com aplicações em um ambiente real, apesar das fontes de dados simularem, através de uma ferramenta de geração de tráfego, tráfego de áudio e vídeo. Trabalhar com um protótipo de rede real ao invés de um simulador é uma vantagem, por um lado, porque o desenvolvimento do projeto gira em torno de dados verídicos. Entretanto, por outro lado as dificuldades a serem enfrentadas são maiores devido ao fato de nem sempre termos software e hardware que, de forma adequada, implementem os Serviços.

Nesse trabalho considerou-se quatro casos diferentes. Em todos os casos, foi reservado 25% do enlace para o tráfego EF e o restante ficou para o tráfego BE. Com o caso 1 e o caso 2, onde tinha-se somente tráfego do tipo EF trafegando pela rede, percebeu-se que esse tipo de tráfego se comporta muito bem desde que cumpra o acordo feito pelo SLA. Caso seja transmitido um tráfego com largura de banda superior a contratado, ocorre logo o descarte de pacotes e/ou atraso por pacote na transmissão. Isso porque o tráfego do tipo EF não compartilhar recursos

com o restante da rede.

Com os casos 3 e 4, notou-se após a obtenção dos resultados, que o tráfego do tipo EF não sofre tanto com a presença exagerado do tráfego BE. Mesmo nos piores casos onde tinha-se uma taxa de tráfego total 25% maior que a capacidade do enlace, se o tráfego EF respeitou os limites contratados, tanto o descarte quanto o atraso, permanecem baixo. Entretanto, quando a taxa de transmissão para o tráfego EF está acima do contratado, o descarte é inevitável, pois não há reclassificação de pacotes para esse tipo de tráfego.

Os resultados apresentados na seção 5.2 indicam que é possível obter grandes melhorias com relação a transmissão de áudio e vídeo. Isso porque se consegue fazer com que os pacotes não esperem muito tempo em filas, em uma rede implementando a proposta PHB EF da abordagem de Serviços Diferenciados. Viu-se também que a reserva de recursos para o tráfego EF um pouco maior que a desejada se faz necessária, uma vez que é preciso levar em conta os cabeçalhos dos pacotes (Hassan e Atiquzzaman 2000). Porém nos demonstra que se o tráfego cumprir com seu limite, o atraso permanece baixo e não há perda de informação durante a transmissão para o Encaminhamento Expresso (apesar que se a taxa de transmissão for “igual” ao limite - seção 5.2.1 - o descarte é pequeno - 2% -, mas não desprezível). Por outro lado, ao ultrapassar o limite reservado, o descarte assim como o atraso é certo.

O grande problema com relação ao Encaminhamento Expresso é que este requer uma alocação explícita de recursos da rede para a manutenção de sua definição. Logo, este é um serviço de alto custo que oferece alta qualidade. Uma das metas futuras de estudo é sem dúvida a implementação também do Encaminhamento Assegurado. Isso permitiria realizar uma comparação entre os dois PHB, assim como usá-los de forma cooperativa entre si.

Um outro teste que deveria ser feito em nosso ambiente de teste é a transmissão de vídeo e áudio reais. Assim, seria possível avaliar, além das estatísticas da transmissão, a satisfação do usuário final na estação destino.

Por fim, como perspectivas futuras, deve-se incluir também como trabalhos futuros nessa área, a utilização de uma topologia mais complexa onde se tivéssemos a interação entre vários domínios DS.

# Referências Bibliográficas

- Almesberger, W., Salim, J. H. e Kuznetsov, A. (1999). Differentiated services on linux.
- Apostolopoulos, G. e et al. (1998). Quality of service based routing: A performance perspective, *ACM SIGCOMM'98* .
- Awduche, D. e et al. (2000). A framework for internet traffic engineering, *Internet Draft draft-ietf-te-framework-00.txt*.
- Bennet, J., Benson, K., Chiu, A., Davari, S., Kalmanek, C., Stiliadis, D., Davie, B., Charny, A., Baker, F., Boudec, J. Y. L., Courtney, W., Firoiu, V. e Ramakrishnam, K. K. (2001). An expedited forwarding phb, *Internet Draft draft-ietf-diffserv-rfc2598bis-00.txt*.
- Bernet, Y. e et al. (1998). A framework for differentiated services, *Internet Draft* .
- Bernet, Y. e et al. (1999). A framework for integrated services operation over diffserv networks, *Internet Draft draft-ietf-issll-diffserv-rsvp-03.txt*.
- Blake, S., Black, D. L., Carlson, M., Davies, E., Wang, Z. e Weiss, W. (1998). An architecture for differentiated services, *Internet RFC 2475* .
- Bonaventure, O. e Cnodder, S. D. (1999). A rate adaptative shaper for differentiated services, *Internet Draft draft-bonaventure-diffserv-rashaper-00*.
- Braden, R., Clark, D. e Shenker, S. (1994). Integrated services in the internet architecture: na overview, *Internet RFC 1633* .
- Braden, R., Zhang, L., Berson, S., Herzog, S. e Jamin, S. (1997). Resource reservation protocol (rsvp), *Internet RFC 2205 Version 1 Functional Specification*.
- Branden, B., Clark, D. D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K. K., Wroclawski, J. e Zhang, L. (1998). Recommendations on queue management and congestion avoidance in the internet, *Internet RFC 2309* .

- Clark, D. D. e Fang, W. (1998). Explicit allocation of best effort packet delivery service, *IEEE/ACM Transactions on Networking* **6**(4): 362–373.
- Comer, D. E. (1998). *Interligação em Rede com TCP/IP*, V. 1, 3 ed., Campus - Rio de Janeiro.
- Crawley, E. e et al. (1998). A framework for qos-based routing in the internet, *Internet RFC* 2386 .
- Faucher, F. L. e et al. (2000). Mpls support of differentiated services, *Internet Draft draft-ietf-MPLS-diff-ext-03.txt*.
- Feng, W. C. (1999). Improving internet congestion control and queue management algorithms, *PhD thesis, University of Michigan* .
- Floyd, S. e Jacobson, V. (1993). Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking* **1**(4): 397–413.
- Goyal, M., Duresi, A., Jain, R. e Liu, C. (1999). Effect of number of drop precedences in assured forwarding, *Internet Draft draft-goyal-diffserv-dpstdy-02*.
- Group, D. S. D. I. W. (2001). <http://www.ietf.org/html.charters/diffserv-charter.html>.
- Hassan, M. e Atiquzzaman, M. (2000). *Performance of TCP/IP Over ATM Networks*, V. 1, 1 ed., Artech House - Boston - London.
- Heinanen, J., Baker, F., Weiss, W. e Wroclawski, J. (1999). Assured forwarding phb group, *Internet RFC* 2597 .
- Heinanen, J. e Guerin, R. (1999a). A single rate three color marker, *Internet Draft draft-heinanen-diffserv-srtcm-01*.
- Heinanen, J. e Guerin, R. (1999b). A three color marker, *Internet Draft draft-heinanen-diffserv-tcm-01*.
- Heinanen, J. e Guerin, R. (1999c). A two rate three color marker, *Internet Draft draft-heinanen-diffserv-trtcm-01*.
- Internet2 (1998). Report from the first internet2 joint applications/engineering qos workshop, <http://www.internet2.edu/qos/may98Workshop> .
- Jacobson, V., Nichols, K. e Poduri, K. (1999). An expedited forwarding phb, *Internet RFC* 2598 .
- Kim, B. K. (1993). Simulation study of weighted round-robin queueing policy, *Tese Mestrado, Depto of Computer Science, University of Massachusetts, Lowell, MA* .
- Kim, H. (1999). A fair marker, *Internet Draft draft-kim-fairmarker-diffserv-00*.

- Kim, H., Leland, W. E. e Thomson, S. E. (1998). Evaluation of band width assurance service using red for internet service differentiation, *Relatório Técnico, Bell Communication Research* .
- LAND/UFRJ (2001). Laboratory for modeling, analysis and development of networks and computer systems, <http://www.land.ufrj.br> .
- Leocádio, M. A. P. e Rodrigues, P. H. A. (2000). Uma ferramenta para geração de tráfego e medição em ambiente de alto desempenho, *SBRC2000* .
- Lin, D. e Morris, R. (1997). Dynamics of random early detection, *In Proceedings of ACM, SIGCOMM'97* .
- Mankin, A., Baker, F., Braden, B., Bradner, S., O'Dell, M., Romanow, A., Weinrib, A. e Zhang, L. (1997). Resource reservation protocol (rsvp), *Internet RFC 2208 Version 1 Applicability Statement - Some Guidelines on Deployment*.
- Naser, H., Leon-Garcia, A. e Aboul-Magd, O. (1998). Voice over differentiated services, *Internet Draft draft-naser-voice-diffserv-eval-00*.
- NetPerf (2001). <http://www.netperf.org>.
- Nichols, K., Blake, S., Baker, F. e Black, D. L. (1998). Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers, *Internet RFC 2474* .
- Nichols, K., Jacobson, V. e Zhang, L. (1999). A two-bit differentiated services architecture for the internet, *Internet RFC 2638 draft-naser-voice-diffserv-eval-00*.
- Radhakrishnan, S. (1999). Linux - advanced networking overview, **1**.
- Rezende, J. F. (1999a). Assured service evaluation, *In IEEE Global Telecommunications Conference - GLOBECOM99, Rio de Janeiro, Brazil* .
- Rezende, J. F. (1999b). Avaliação do serviço assegurado para a diferenciação de serviços na internet, *In Anais do XVII Simpósio Brasileiro de Redes de Computadores, SBRC99, Salvador, BA* p. 339–353.
- Semret, N., Liao, R. R. F., Campbell, A. T. e Lazer, A. A. (1999). Market pricing of differentiated internet services, *In Proceedings of the 7th IEEE/IFIP International Workshop on Quality of Service - IWQoS'99* .
- Shenker, S., Partridge, C. e Guerin, R. (1997). Specification of guaranteed quality of service, *Internet RFC 2212* .
- Systems, C. (1997). Advanced qos services for the intelligent internet, *White paper* .

- Teitelbaum, B. (1999). Draft qbone architecture, internet2 qos workin gruoup draft, *Internet Draft draft-i2-qbone-arch-06*.
- TTCP (2001). <http://www.ccci.com/tools/ttcp/>.
- Wroclawski, J. (1997). Specification of the controlled-load network element service, *Internet RFC 2211*.
- Xiao, X. e Ni, L. (1999). Internet qos: the big picture, *IEEE Network* **13**(2): 1–13.
- Xue, L. e et al. (1999). Video multicast over the internet, *IEEE Network*.