Aldir Lopes dos Santos

Controle de Banda em Redes TCP/IP Utilizando o Linux

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Esp. Samuel Pereira Dias

Lavras
Minas Gerais - Brasil
2010

Aldir Lopes dos Santos

Controle de Banda em Redes TCP/IP Utilizando o Linux

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *24 de abril de 2010*

Prof. DSc. Joaquim Quinteiro Uchoa

Prof. MSc. Herlon Ayres Camargo

Prof. Esp. Samuel Pereira Dias
(Orientador)

Lavras
Minas Gerais - Brasil
2010

"Bendito seja o Senhor, minha rocha, que adestra as minhas mãos para a peleja e os meus dedos para a guerra" (Salmos 144:1). Ao Deus que sempre foi minha força e minha luz, que nunca me abandonou e nem se envergonhou de mim: À Ele toda honra e toda glória.

Agradecimentos

Agradeço aos meus familiares e amigos pelo incondicional apoio e compreensão, principalmente nos momentos de ausência e mau humor, oriundos das incumbências e pressões cotidianas. A esses eu me sinto extremamente devedor, principalmente por deixar frequentemente de verbalizar o sentimento que aqui dentro reside, de falar sobre o quanto são importantes para mim.

Aos tais agora falo abertamente que sempre foram como um norte confiável para minha bússola, permitindo que eu tivesse um lugar seguro para retornar nos dias mais tempestuosos de minha humilde existência. De todo meu coração eu lhes digo: Obrigado!

Sumário

1 Introdução **1**

1.1 Ojetivos 2

1.1.1 Objetivo Geral 2

1.1.2 Objetivos Específicos 2

1.2 Metodologia 2

1.3 Organização do Trabalho 3

2 Elementos Gerais de Qualidade de Serviço **5**

2.1 Qualidade de Serviço 5

2.2 Arquiteturas de *QoS* 6

2.2.1 Melhor Esforço 7

2.2.2 Serviços Integrados - *IntServ* 7

2.2.2.1 Classes de Serviço 8

2.2.3 Serviços Diferenciados - *DiffServ* 8

2.2.3.1 Classes de QoS em *DiffServ* 10

2.3 Controle de Banda 11

2.3.1 Relação de Controle de Banda com *QoS* 12

2.3.2 Controle *versus* Aumento de Banda 12

3 Controle de Tráfego **15**

3.1 Estrutura Básica de um Roteador 15

3.1.1 Portas de Entrada e Saída 16

3.1.2 Processador de Roteamento 17

3.1.3 Circuitos de Comutação 17

3.2 Priorização de Tráfego 17

3.2.1 Classificação e Marcação 18

3.2.2 As Filas 19

3.2.3 Mecanismos de Escalonamento 19

3.2.3.1 *FIFO - First In First Out* 20

3.2.3.2 *PQ - Priority Queueing* 20

3.2.3.3 *FQ - Fair Queueing* 21

3.2.3.4 *CBQ - Class Based Queueing* 22

3.2.3.5 *HTB - Hierarchical Token Bucket* 22

3.3 Limitando o Uso de Recursos 24

3.3.1 *Policiamento* 24

3.3.2 *Suavização* 24

4 Controle de Tráfego no *Linux* **27**

4.1 Subsistema de Rede 27

4.2 Pontos para Inserção de Elementos de Controle 29

4.3 Elementos de Controle de Tráfego no *Linux* 30

4.3.1 Escalonadores 31

4.3.2 Classes 32

4.3.3 Filtros 33

4.3.4 Policiamento e Suavização no *Linux* 34

4.3.5 Identificando os Elementos na Estrutura 35
4.4 A Ferramenta *tc* 36

5 Estudo de Caso: Controle de Banda Utilizando *Linux* **41**
5.1 A Instituição 41
5.2 Redes e Sistemas Envolvidos 42
5.3 Histórico do Problema na Unidade 43
5.4 Descrição Objetiva do Problema 45
5.5 Proposta de Solução para o Problema 45
5.6 Medições Iniciais 45
5.6.1 A Capacidade do Link de Dados 46
5.6.2 Demanda de Banda dos Sistemas Essenciais 47
5.7 Reservando a Largura de Banda Necessária 49
5.8 Resultados da Implementação 51

6 Conclusão **53**

Referências Bibliográficas **60**

A Estratégia de Medição do Tráfego **61**
A.1 Pontos de Medição 61
A.2 Equipamento Utilizado nas Medições 62
A.3 *Software* de Medição de Tráfego 63
A.3.1 Medindo a Capacidade Total de *Download* da Rede 64
A.3.2 Medindo a Capacidade Total de *Upload* da Rede 64
A.3.3 Medindo a Demanda de Banda dos Sistemas Utilizados . . 65

Lista de Figuras

2.1 A Arquitetura *DiffServ*. Fonte: Adaptado de (KAMIENSKI; SADOK, 2000, p. 18). 9

2.2 Domínios *DiffServ*. Fonte: Adaptado de (FILHO, 2006, p. 32). . . 10

3.1 Estrutura Básica de um Roteador. Fonte: Adaptado de (FOROUZAN, 2006, p. 771). 16

3.2 Roteamento com Priorização de Tráfego - Fonte: Adaptado de (FILHO, 2006, p. 39). 18

3.3 *PQ - Priority Queueing*. Fonte: Adaptado de (SANTANA, 2006, p. 70). 20

3.4 *FQ - Fair Queueing*. Fonte: Adaptado de (SANTANA, 2006, p. 72). 21

3.5 *CBQ - Class Based Queueing*. Fonte: Adaptado de (SANTANA, 2006, p. 71). 22

3.6 *HTB - Hierarchical Token Bucket*. Fonte: Adaptado de (GOMES; SATURNINO, 2006, p. 67). 23

3.7 Controle no Uso de Recursos - Fonte: Adaptado de (SANTANA, 2006, p. 75). 25

4.1 Roteamento no *Linux*. Fonte: Adaptado de (ALMESBERGER, 1999, p. 48). 28

4.2 Pontos para Adição de Elementos Controle. Adaptado de (FILHO, 2006, p. 44) e (BLESS; WEHRLE, 1989, p. 3). 29

4.3 Elementos de Controle de Tráfego no *Linux*. Fonte: Adaptado de (ALMESBERGER, 1999, p. 49). 30
4.4 Identificando os Elementos. Fonte: Adaptado de (HUBERT *et al.*, 2003, p. 53). 35
4.5 Exemplo de Estrutura de Controle de Tráfego 37
4.6 Lista Completa dos Comandos Utilizados 40
4.7 Utilizando a Opção *show* 40

5.1 Detalhamento dos Elementos e Serviços Envolvidos na Estrutura . 43
5.2 Capacidade Total do *Link* de Dados - *Download* 46
5.3 Capacidade Total do *Link* de Dados - *Upload* 46
5.4 Terminal de Acesso ao *mainframe* - *Download* 47
5.5 Terminal de Acesso ao *mainframe* - *Upload* 47
5.6 Sistema Acessando o Banco de Dados Remoto - *Download* 48
5.7 Sistema Acessando o Banco de Dados Remoto - *Upload* 48
5.8 *Script* de Reserva da Largura Banda 50
5.9 Comprovando a Reserva de Banda 52

A.1 Pontos Onde Serão Feitas as Medições de Tráfego 62
A.2 Script de Configuração da *bridge* 63

Resumo

As redes *TCP/IP* não foram criadas com base nos requisitos demandados por muitas das aplicações que atualmente as utilizam e por isso não trazem nativamente componentes para tratar de problemas relacionados à oferta de qualidade de serviço. O sistema operacional *Linux*, por outro lado, traz em seu *kernel* um moderno subsistema de rede capaz de fornecer todos os elementos necessários para a implementação de um robusto e eficiente controle de tráfego. O presente trabalho aborda alguns dos principais conceitos de qualidade de serviço em redes *TCP/IP*, além de apresentar uma proposta que preconiza a adoção do *Linux* como solução viável na provisão de controle de banda e priorização de tráfego, sendo dada uma ênfase especial às redes de dados presentes em ambientes de pequeno porte.

Palavras-Chave: QoS; TCP/IP; Linux; Controle de Banda

Capítulo 1

Introdução

Os computadores não surgiram como atualmente são conhecidos, nem em capacidade computacional, nem em objetivos e aplicações. De fato, as poucas unidades que eram construídas tinham destinação bem definida, uma vez que seu preço proibitivo, até mesmo para a maioria das grandes empresas da época, os fadava quase sempre ao uso em aplicações do próprio governo.

Um dos pontos decisivos na evolução desse equipamento foi a estratégia de interconectá-los em um ambientes de redes, que iniciando em laboratórios de universidades e repartições militares do governo, haveriam de se expandir até fazer parte do cotidiano do cidadão comum de nossa época. Com a disseminação das redes de dados, cresceu também a sua importância dentro das estratégias de negócios das empresas, até o ponto de várias atividades não poderem mais ser executadas em um equipamento sem conexão com algum tipo de rede.

O próprio sucesso da estratégia de oferecer cada vez mais serviços nos ambientes de rede transformou-se, ironicamente, num dos responsáveis pelo aumento da distância entre a capacidade física da infraestrutura de redes existente e o fluxo de dados demandado por esses novos serviços.

Em decorrência de tal escassez na capacidade de transmissão de uma rede, situações de concorrência podem acontecer tanto entre computadores, no caso de compartilharem uma mesma conexão, quanto numa única máquina, onde diversas aplicações disputam entre si pela banda destinada ao computador. É neste sentido que o presente trabalho constitui-se numa proposta onde se utilize ferramentas desenvolvidas como *software* livre para controlar o fluxo de dados envolvido nos casos de compartilhamento da capacidade de transmissão de uma rede.

A verificação da aplicabilidade dos conceitos apresentados neste trabalho se dará por meio de um estudo de caso conduzido em uma instituição pública estadual atendida por um *link* de 256 kbps, que necessita de racionalizar a distribuição dessa capacidade para evitar que suas atividades administrativas continuem sendo comprometidas.

1.1 Ojetivos

1.1.1 Objetivo Geral

Investigar a possibilidade de se utilizar elementos de controle de tráfego do sistema operacional *Linux* para implementar soluções de controle de banda em redes *TCP/IP*, com enfoque a ambientes de pequeno porte.

1.1.2 Objetivos Específicos

- Apresentar os conceitos básicos de qualidade de serviço e as principais arquiteturas propostas para lidar com esta questão;
- abordar os principais elementos de controle de tráfego presentes no sistema operacional *Linux*, exemplificando sua utilização;
- implementar os conceitos apresentados neste trabalho em uma instituição estadual coma finalidade de promover a racionalização do compartilhamento de um *link* de dados de baixa capacidade.

1.2 Metodologia

No decorrer deste trabalho são utilizadas, basicamente, duas estratégias metodológicas, sendo a primeira delas a pesquisa bibliográfica, utilizada para apresentar toda a parte conceitual do documento, compreendida entre os capítulos 2 e 4, numa compilação das abordagens de diversos autores sobre os temas tratados. A segunda estratégia metodológica utilizada é a experimentação, empregada no estudo de caso do Capítulo 5, para verificar a eficácia da utilização de alguns dos conceitos abordados neste trabalho na solução de um problema real.

1.3 Organização do Trabalho

Os temas abordados ao longo deste trabalho estão distribuídos da seguinte forma:

- O **Capítulo 2** apresenta noções gerais do tema Qualidade de Serviço;
- O **Capítulo 3** aborda os principais elementos de controle de tráfego para arquiteturas de serviços diferenciados, enquanto que o **Capítulo 4** detalha algumas particularidades da implementação desses elementos no sistema operacional *Linux*;
- O **Capítulo 5** apresenta um caso básico de controle de banda utilizando o elementos de controle de tráfego do *Linux*;
- No **Capítulo 6** estão presentes as considerações finais sobre este trabalho.

Capítulo 2

Elementos Gerais de Qualidade de Serviço

Qualidade de serviço não é um conceito presente desde as especificações originais das redes de dados *TCP/IP*, mas sim uma estratégia que começou a ser utilizada como solução para demandas que posteriormente surgiram, algumas das quais foram elencadas no Capítulo 1. Serão apresentados no decorrer deste capítulo alguns dos principais fundamentos de qualidade de serviço sobre redes *TCP/IP*.

2.1 Qualidade de Serviço

Qualidade de Serviço ou *QoS*[1] é um termo normalmente utilizado para definir o grau de satisfação do usuário em relação a um serviço que lhe é oferecido. Todavia, quando aplicado dessa forma, *QoS* se torna um conceito muito abstrato e árduo de ser conseguido, o que decorre da dificuldade de se entregar, com exatidão, algo que não possa ser detalhadamente descrito, medido ou avaliado.

Além disso, *QoS* não é um termo restrito ao universo das redes de computadores, o que contribui para que uma definição que atenda satisfatoriamente uma área não seja aplicável a outra. Vários autores comentam sobre tal dificuldade de se chegar a uma definição única e precisa de *QoS*, incluindo Ferguson e Huston (1998), que discorrem mais minuciosamente sobre este problema.

[1] *Quality of Service*.

O Termo *QoS* será mencionado ao longo deste trabalho como um conjunto de indicadores específicos e mensuráveis que possam ser mantidos dentro de certos limites durante o fornecimento de um serviço. Isto permitirá que os parâmetros a serem levados em consideração e os níveis aceitáveis de cada um deles sejam definidos individualmente para cada aplicação em particular.

Por mais simples que seja uma aplicação, caso seu funcionamento dependa da transmissão de dados através de uma rede, alguns requisitos estarão naturalmente envolvidos no processo. Em consonância com Forouzan (2006), tomaremos como parâmetros de *QoS* apenas alguns deles, os quais afetam mais comumente o desempenho das aplicações e, consequentemente, a satisfação de seus usuários:

- **Confiabilidade:** Também mencionada como perda de pacotes por alguns autores, representa o número de pacotes que foram transmitidos, mas que não chegaram ao seu destino em um determinado período de tempo, ou cujas respostas de confirmação foram perdidas;
- **Atraso:** É a diferença de tempo registrada entre o momento que um pacote deixa a sua origem e chega ao seu destino;
- ***Jitter*:** É a variação dos valores de atraso registrados entre pacotes distintos de uma transmissão. Vários fatores podem influir nesta variação, como congestionamentos e rotas diferentes, ambas decorrentes do uso da comutação de pacotes, que compartilha o meio de acesso e pode encaminhar dados de uma mesma transmissão usando caminhos de rede distintos;
- **Largura de Banda:** A capacidade máxima de transferência de dados que pode ser comportada por uma conexão. A largura de banda disponível para uma aplicação está diretamente ligada à quantidade do parâmetro disponível na rede e à existência de outras máquinas e/ou processos compartilhando esta capacidade.

2.2 Arquiteturas de *QoS*

Segundo Sousa (2005), se uma aplicação tivesse ao seu dispor todos os recursos das estruturas de redes por onde seus dados trafegam, dificilmente a mesma enfrentaria problemas com *QoS*. A questão é que as redes de dados possuem suas origens fundamentadas no compartilhamento de recursos e a própria técnica de comutação de pacotes, utilizada em redes como *TCP/IP*, foi adotada para favorecer esta funcionalidade.

Neste sentido, prover *QoS* implica diretamente em alterar as regras de concorrência de uma rede, de tal maneira que alguns pacotes sejam privilegiados pelo controle de tráfego. O tipo de tratamento destinado a cada pacote em uma rede dependerá da arquitetura de *QoS* adotada, sendo algumas delas apresentadas nas próximas subseções.

2.2.1 Melhor Esforço

Este é o tipo de serviço padrão oferecido nas redes *TCP/IP*, que oferece apenas conectividade básica sem quaisquer garantias. Apesar de não prover *QoS* ele é o modelo utilizado atualmente pela Internet e atende as exigências de grande parte das aplicações convencionais, como correio eletrônico e transferência de arquivos, uma vez que o protocolo TCP, que foi desenvolvido para trabalhar sobre redes não confiáveis, possui mecanismos que controlam o sequenciamento e a perda de pacotes.(MELO, 2001).

2.2.2 Serviços Integrados - *IntServ*

Especificada pela comunidade internacional conhecida como *IETF*[2], através da *RFC 1633* (BRADEN; CLARK; SHENKER, 1994) esta arquitetura se baseia na implementação de algumas modificações em redes baseadas no protocolo *IP*, como forma de permitir garantias no fornecimento de paramentos rígidos de *QoS*.

Segundo Leal (2004) os Serviços Integrados foram desenvolvidos para otimizar a utilização de redes e recursos para novos aplicativos, como multimídia em tempo real, que sofrem com os retardos de roteamento e perdas devido a congestionamentos. Os parâmetros rígidos demandados por tais aplicações podem ser fornecidos graças a um mecanismo de reserva prévia de recurso.

Tais reservas acontecem da seguinte forma: os recursos são solicitados dinamicamente pela aplicação e devem ser repassados a cada dispositivo da rede existente entre a origem e o destino. Se todos os dispositivos dispõem dos recursos solicitados, eles são reservados para tal aplicação e assim permanecerão enquanto houver demanda para este fluxo (SANTANA, 2006, p. 25).

[2] *The Internet Engineering Task Force*: `http://www.ietf.org/`.

2.2.2.1 Classes de Serviço

Os serviços presentes na arquitetura *IntServ* são definidos em classes, conforme os níveis de garantias que podem ser oferecidos para os parâmetros de *QoS* demandados pelas aplicações, estando disponíveis as seguintes classes de serviços:

- **Serviço Garantido:** Classe direcionada principalmente a aplicações muito sensíveis a atrasos, como conversas telefônicas, uma vez que oferece garantias matematicamente prováveis de um limite rígido para este parâmetro. Especificada pela *RFC 2212* (SHENKER; PARTRIDGE; GUERIN, 1997), esta classe aceita reservas de largura de banda, oferecendo inclusive uma proteção contra a perda dos pacotes que estiverem dentro dos limites do tráfego reservado;
- **Serviço de Carga Controlada:** Esta classe é especificada pela *RFC 2211* (WROCLAWSKI, 1997) e não garante um baixo valor de atraso, mas oferece uma alta confiabilidade (baixo descarte de pacotes) e um baixo *jitter* (pequena variação do atraso). O nível do serviço fornecido por esta classe pode ser considerado "intermediário", semelhante ao que seria oferecido pela arquitetura de melhor esforço em uma rede não congestionada, sendo indicada para aplicações que apresentem um funcionamento satisfatório nessas condições;
- **Serviço de Melhor Esforço:** Apesar de não ser um nível de serviço fornecido diretamente pela arquitetura *IntServ*, o fato de estar naturalmente presente nas redes baseadas no protocolo *IP* o torna a classe de serviço padrão, aplicada ao tráfego não associado a nenhuma das outras classes. Quando utilizado em redes *IntServ*, o serviço de melhor esforço pode apresentar um desempenho muito pior que o oferecido por ele mesmo em redes sem *QoS*. Isso se deve ao fato deste serviço constantemente ter que trabalhar apenas com as "sobras" de recursos da rede.

2.2.3 Serviços Diferenciados - *DiffServ*

Além da arquitetura de serviços integrados a *IETF* também formou outro grupo de trabalho sobre a mesma temática da oferta de qualidade de serviço na Internet, propondo uma nova abordagem, através da arquitetura denominada de Serviços Diferenciados - *DiffServ*[3] (Figura 2.1).

[3] *Differentiated Services*.

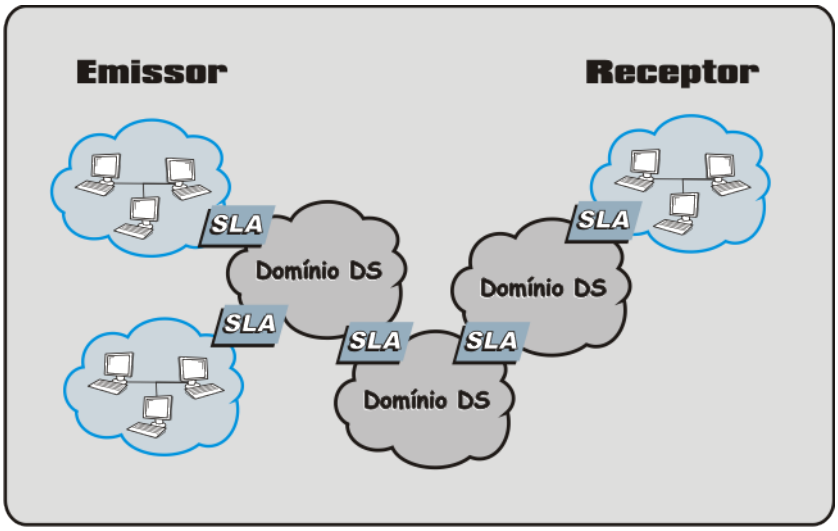

Figura 2.1: A Arquitetura *DiffServ*. Fonte: Adaptado de (KAMIENSKI; SADOK, 2000, p. 18).

Em comparação com o mecanismo de reserva prévia de recursos presente em *IntServ*, a arquitetura *DiffServ*, descrita na *RFC 2475* (BLAKE *et al.*, 1998), representa um modelo bem mais simplificado de oferta de QoS, que, resumidamente, contempla os seguintes componentes:

- Um cliente que necessite de garantias de *QoS* em suas transmissões de dados, deve contratar um Serviço Diferenciado - *DS*[4] de seu provedor;
- Os níveis dos parâmetros de qualidade deste *DS* oferecido ao cliente estarão expressamente descritos num Contrato de Nível de Serviço – *SLA*[5];
- Uma rede que oferece *DS*, mediante um *SLA*, aos seus clientes é chamada de Domínio *DS*;
- um Domínio *DS* deve implementar todos os mecanismos necessários para garantir a qualidade dos *DS* que acordou com seus clientes, podendo para isso, inclusive, se tornar um cliente de outro Domínio *DS*.

[4] *Differentiated Service*.
[5] *Service Level Agreements*.

2.2.3.1 Classes de QoS em *DiffServ*

Na arquitetura *DiffServ* as garantias de parâmetros de QoS não são fornecidas individualmente para cada fluxo de dados que se inicia, mas são aplicadas em agrupamentos de pacotes chamados de classe. Os pacotes que forem definidos como integrantes de uma classe serão tratados como um único fluxo de dados, de forma que receberão um mesmo *DS*, exatamente como se fossem uma única conexão.

Para que um pacote possa receber um *DS* na rede, ele precisa primeiramente ser identificado dentre os outros inúmeros pacotes que não terão este tratamento. Em *DiffServ* esta identificação é feita através de marcações em um dos campos do cabeçalho *IP* (nas redes *IPv4* é utilizado o campo conhecido como *TOS*[6]), de modo que o pacote possa ser reconhecido como integrante de uma determinada classe (SOUSA, 2005, p. 24-25). A Figura 2.2 permite um visualização dos pontos da rede onde esse processo de marcação efetivamente acontece.

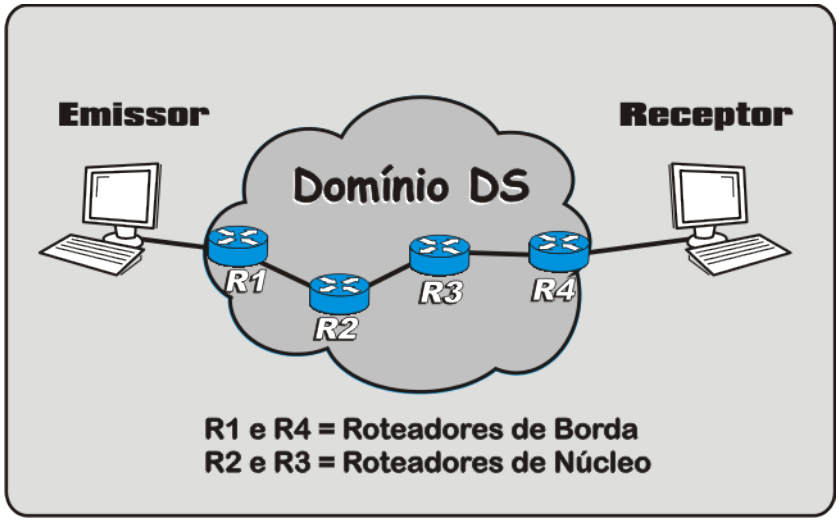

Figura 2.2: Domínios *DiffServ*. Fonte: Adaptado de (FILHO, 2006, p. 32).

Em um domínio *DS*, os roteadores responsáveis pelo interfaceamento com redes externas são chamados de roteadores de borda. Os demais roteadores, que tratam apenas do tráfego interno de um domínio *DS*, são chamados de roteadores de núcleo. Toda a marcação de pacotes acontece exclusivamente na entrada do

[6] *Type of Service*.

domínio, de modo que apenas os roteadores de borda participam deste processo (AGOSTINHO *et al.*, 2006).

Além dessa função de novas marcações nos cabeçalhos dos pacotes, os roteadores de borda também são os responsáveis por corrigir as marcações que tenham chegado incorretas ao domínio e pelo controle dos fluxos de dados provenientes de cada um dos seus clientes, de modo que eles não ultrapassem os limites dos parâmetros de *QoS* contratados, os quais são expressamente descritos no *SLA*.

Embora todos os roteadores de um domínio *DS* estejam programados para separar pacotes em filas distintas e encaminhá-los, conforme a prioridade definida para suas respectivas classes, apenas os roteadores de borda são incumbidos de fazer as marcações e o controle rígido do uso de recursos nas portas de entrada do domínio *DS*. Isso permite que as tarefas de maior complexidade sejam concentradas apenas nos roteadores de borda, mantendo a simplicidade nos equipamentos do núcleo da rede (MELO, 2001, p. 29).

Dentro do domínio *DS* e devidamente identificados, os pacotes estarão prontos para receberem o tratamento correspondente à sua classe. Uma vez que *Diff-Serv* não trabalha com reserva prévia de recursos, um *DS* é sempre oferecido com base na técnica de priorização de tráfego, exigindo, portando, a existência de filas distintas para cada classe de *QoS*, em todos os roteadores do domínio *DS*.

Uma limitação ao se trabalhar apenas com o esquema de prioridade relativa, é que mesmo garantindo que uma aplicação gerando tráfego de determinada prioridade terá melhor desempenho que outra gerando tráfego de menor prioridade, não impede que ambas as aplicações recebam um atendimento com recursos muito aquém de suas reais necessidades (SOUSA, 2005, p. 29).

2.3 Controle de Banda

Uma das implicações de se concentrar a abordagem da qualidade de serviço apenas no estudo das arquiteturas propostas pela *IETF* é a possível falsa impressão de que seu uso não é factível em ambientes compostos por redes de pequeno porte. Isso é plenamente compreensível, uma vez que a *IETF* tem como referencial para suas pesquisas uma rede cujas as dimensões são as da própria Internet.

Por outro lado, o cotidiano de pequenos escritórios é melhor retratado em um cenário formado por redes internas, onde todos os computadores compartilham uma mesma conexão com a Internet, a qual é fornecida por um provedor que tem

como única especificação do serviço oferecido apenas uma vaga referência do parâmetro largura de banda.

Algo que parece agravar a situação nesses pequenos ambientes de redes é a atual tendência de modernização das rotinas de trabalho através da adição de elementos tecnológicos, como sistemas *online* e consultas a bases de dados remotas, os quais se tornaram tão essenciais, ao ponto de algumas atividades não poderem mais ser executadas sem eles. Em muitos casos, o tempo necessário para a realização de determinadas tarefas está intrinsecamente ligado ao desempenho oferecido pela rede às aplicações utilizadas durante a realização das mesmas.

2.3.1 Relação de Controle de Banda com *QoS*

Em linhas gerais, controlar a distribuição da largura de banda em uma rede, mesmo implicando em atuar sobre um único parâmetro, não deixa de ser uma abordagem de *QoS*. De fato, é possível encontrar várias semelhanças entre um roteador implementando controle de banda e um típico roteador de borda da arquitetura *DiffServ*, o que se deve principalmente ao fato de algumas das técnicas e mecanismos utilizados serem os mesmos em ambos os casos.

Além disso, aplicar controle de banda em uma rede não deixa de ser uma forma de se oferecer um serviço diferenciado a um determinado fluxo de dados. Para que esse serviço de oferta diferenciada de banda aconteça, cada pacote pertencente a um fluxo privilegiado precisa ser reconhecido e, portanto, também faz uso de alguma forma de classificação. Em ambos os casos é feito o uso de classes para agrupar pacotes que recebem um tratamento semelhante e os algoritmos de escalonamento do tráfego também são basicamente os mesmos.

2.3.2 Controle *versus* Aumento de Banda

Um dos grandes desafios enfrentados ao se propor melhorias em redes de pequenos ambientes, é a limitação de se trabalhar apenas com um dos parâmetros de *QoS*: a largura de banda, e mesmo esta costuma ser oferecida sem muita rigidez. Em tais situações, onde o desempenho da aplicação é insatisfatório e apenas um dos parâmetros é passível de ser alterado, a solução trivial recai invariavelmente sobre a busca de um aumento na largura de banda da conexão, junto ao provedor.

O primeiro aspecto dessa decisão a ser levado em consideração é o econômico, pois o custo de um produto é, normalmente, inversamente proporcional à

sua oferta. Essa máxima se torna evidente nas localidades com escassez de oferta de banda, onde os valores cobrados, mesmo por serviços básicos de conexão, costumam ser bem elevados para residências e pequenas empresas.

Além dos custos desses serviços, ainda existem situações onde os próprios provedores não conseguem elevar a largura de banda entregue a seus clientes além de certo ponto, o que se deve por vezes às limitações da tecnologia empregada, do meio físico utilizado na conexão, ou pelo fato do próprio provedor ser dependente de outras empresas que lhe fornecem um serviço já limitado.

Mesmo supondo que o aumento na largura de banda seja tecnicamente possível e economicamente comportado pelo orçamento do cliente, isso ainda não garante uma solução definitiva para o problema, visto que algumas aplicações simplesmente procuram utilizar o máximo de banda disponível na rede (SOUSA, 2005, p. 15) e que, independente de quanta banda houver, novas aplicações serão rapidamente criadas para consumi-la (KAMIENSKI; SADOK, 2000, p. 3).

Deve se considerar que, mesmo que a contratação de um serviço de conexão com a Internet tenha sido motivada pela necessidade de aplicações específicas, é muito difícil pensar que seu uso ficará restrito apenas a elas. Na prática, devido a infinidade de opções oferecidas na rede mundial, a banda acaba sendo compartilhada com diversas outras aplicações como transferência de arquivos, correio eletrônico e acesso a páginas *HTTP*[7].

Esse aproveitamento da conexão para fins diversos costuma ser desejável até um ponto bem específico: o máximo possível de uso que não interfira no motivo central da aquisição. Para que a demarcação deste ponto de corte seja feita com certa precisão e sem depender da boa vontade dos usuários, torna-se obrigatório o uso de alguma tecnologia que controle essa distribuição de banda na rede.

Todavia, é oportuno ressaltar que, assim como o simples aumento da banda oferecida pela conexão pode não solucionar o problema do desempenho das aplicações de uma rede, qualquer proposta de um controle de banda também possui suas próprias limitações:

- O controle de banda não aumenta a capacidade de transmissão de uma rede, mas apenas permite interferir na forma como esse recurso é distribuído;
- O controle de banda não supre a falta de outros parâmetros de *QoS*: se a rede oferece a largura de banda necessária para uma aplicação e mesmo assim

[7] *HyperText Transfer Protocol*.

ela não conseguir um desempenho satisfatório quando executada sozinha na rede, o controle de banda também não irá resolver esse problema;

- Reservar banda para um grupo específico de aplicações pode fazer com que as demais tenham seu desempenho tão seriamente afetado ao ponto da sua execução se tornar inviável;

Dessa forma, é possível perceber que o aumento na largura de banda contratada de uma conexão e a adoção de um controle na distribuição da mesma, não são abordagens mutuamente exclusivas, ao invés disso, podem ser perfeitamente combinadas para se conseguir o objetivo principal de manter um desempenho satisfatório no maior número possível de aplicações.

Capítulo 3

Controle de Tráfego

Provisão de *QoS* e controle de banda são técnicas que implicam diretamente em inibir a concorrência "justa" por recursos em uma rede, de forma que determinados fluxos de dados sejam privilegiados em detrimento de outros. Essa interferência na democracia da rede é implementada nos roteadores através de mecanismos de controle de tráfego.

Esse capítulo trata do funcionamento básico dos roteadores e de alguns dos elementos que a eles podem ser adicionados com o intuito de se conseguir mecanismos mais eficientes na tarefa de controlar o fluxo de pacotes em uma rede.

3.1 Estrutura Básica de um Roteador

Esta seção apresenta uma compilação dos escritos de (FOROUZAN, 2006, p. 771-779) e (FILHO, 2006, p. 19-20) sobre a estrutura e o funcionamento de um roteador, além de contribuições de outros autores, citados diretamente no texto.

Os roteadores são dispositivos responsáveis pelo encaminhamento de pacotes entre seguimentos distintos de rede. Isto é feito mediante a verificação do cabeçalho de cada um desses pacotes para determinar a interface de saída que deverá encaminhá-los ao próximo nó do caminho.

Mesmo considerando que existe várias formas de se implementar mecanismos de comutação de pacotes nos roteadores (SANTOS, 2006, p. 8), a lógica dessa operação não sofre muitas modificações entre as diversas arquiteturas utilizadas, permanecendo em conformidade com o modelo exposto na Figura 3.1:

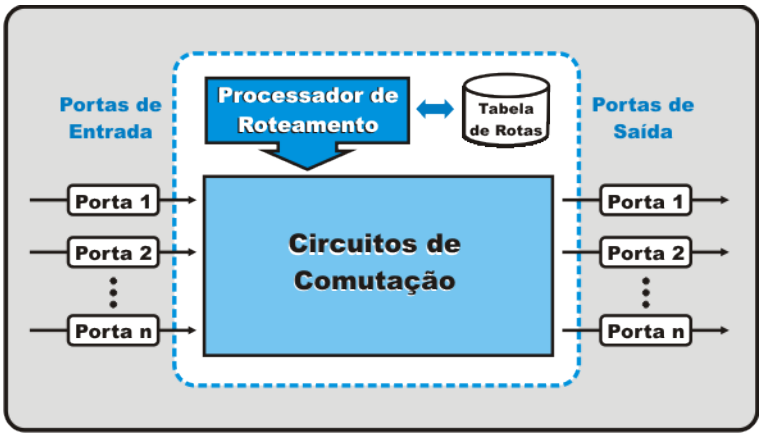

Figura 3.1: Estrutura Básica de um Roteador. Fonte: Adaptado de (FOROUZAN, 2006, p. 771).

3.1.1 Portas de Entrada e Saída

Os componentes do circuito de comutação em um roteador trabalham com pacotes, a unidade de dados da camada Internet (*TCP/IP*), ou camada de Rede (*OSI*), de maneira que as portas representam os circuitos responsáveis por trabalharem com todas as demais unidades de dados das camadas inferiores. Isso inclui as interfaces de rede e demais componentes relacionados com esse processamento.

Uma porta de entrada trabalha desde a recepção dos dados eletricamente codificados até a entrega de um pacote ao circuito de comutação, enquanto que uma porta de saída trabalha de modo inverso, recebendo pacotes que devem ser conduzidos ao meio de transmissão. Na prática, uma mesma porta realiza funções de entrada e saída, conforme a direção do fluxo de dados a ela direcionado.

É possível que em algumas situações dados cheguem à uma porta de entrada sem que consigam ser imediatamente encaminhados por uma porta de saída. Nesses casos, até que possam ser transmitidos, eles serão armazenados em uma região de memória chamada comumente de "*buffer*", também mencionada por alguns autores como uma "fila de *hardware*" (SANTANA, 2006, p. 70).

Cada porta de entrada e de saída está associada a um *buffer*, de maneira que dados que chegam pelas interfaces de entrada, são convertidos em pacotes e direcionados para seus respectivos *buffers*, onde serão processados pelos circuitos de comutação. Uma vez determinada a porta de saída para um pacote, ele será direcionado ao *buffer* da mesma, onde aguardará até ser reconvertido em sinais elétricos e enviado pela interface de rede correspondente.

3.1.2 Processador de Roteamento

Os pacotes são reconstruídos a partir dos dados recebidos nas portas de entrada e são encaminhado aos seus respectivos *buffers*, onde se tornam acessíveis ao processador de roteamento, o qual faz a leitura do endereço de destino no cabeçalho, para determinar, mediante consulta à tabela de rotas, qual será o próximo destino do pacote e por qual porta de saída ele deverá ser encaminhado para lá.

3.1.3 Circuitos de Comutação

Um dos desafios de um roteador é mover eficientemente os pacotes dos *buffers* de entrada para os *buffers* de saída, pois a velocidade como isso é feito afeta o tamanho dos *buffers* e, consequentemente, o desempenho geral do processo e o atraso registrado na entrega dos pacotes. Essa etapa no repasse dos pacotes acontece nos circuitos de comutação, os quais são os responsáveis pela execução das regras de encaminhamento determinadas pelo processador de roteamento.

Em dispositivos específicos de roteamento existem componentes de *hardware* que implementam essa tarefa de comutação de várias formas, dependendo da arquitetura utilizada pelo dispositivo. Em computadores que realizam a função de roteador, os *buffers* de entrada e saída são implementados em regiões da própria memória principal e a comutação acontece pela movimentação de dados entre essas regiões, realizada por processos específicos do sistema operacional utilizado.

3.2 Priorização de Tráfego

A estrutura básica do roteador apresentado anteriormente possui algumas limitações, sendo a mais visível delas a falta de mecanismos de distinção de tráfego. Um roteador desse tipo apenas se ocupa de encaminhar os pacotes o mais rápido possível, observando para isso unicamente a ordem de chegada dos mesmos.

Um dos elementos que podem ser alterados nesses rotadores é a ordem em que os pacotes são encaminhados entre as interfaces de entrada e de saída, uma vez que a oferta de serviços diferenciados, apresentada na subseção 2.2.3, se baseia na priorização do atendimento a determinados fluxos de dados, em relação ao restante do tráfego. A Figura 3.2 apresenta algumas modificações na estrutura básica de um roteador (Figura 3.1), para que o mesmo possa trabalhar com priorização de fluxos.

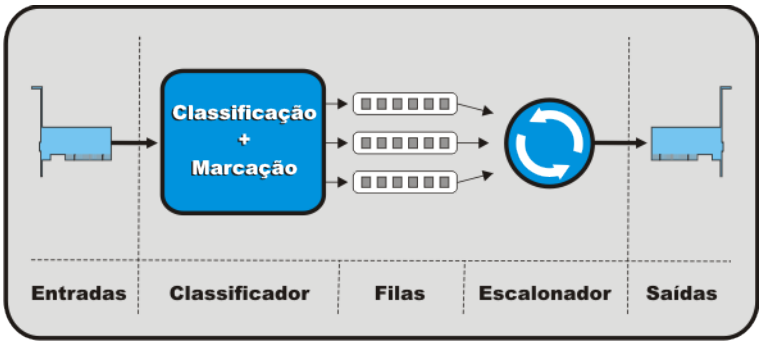

Figura 3.2: Roteamento com Priorização de Tráfego - Fonte: Adaptado de (FILHO, 2006, p. 39).

3.2.1 Classificação e Marcação

Um ponto básico na provisão de *QoS* é o exame dos fluxos de dados que chegam ao roteador, cujo objetivo não é oferecer um tratamento único a cada um dos pacotes da rede, mas sim identificá-los e agrupá-los com outros que receberão o mesmo tipo de tratamento. Como esses agrupamentos são normalmente chamados de "classes" essa etapa é conhecida como "classificação".

O ponto chave da classificação são os critérios: um pacote é reconhecido como pertencente a uma classe a partir da análise de um conjunto de características combinadas, as quais podem estar espalhadas entre as diversas camadas do modelo. Como exemplo de algumas dessas características que podem ser utilizadas na tarefa de classificação de um pacote, Santana (2006) cita:

- Interface física e endereço *MAC*;
- Endereço *IP* de origem e/ou destino, campo *TOS*;
- Portas *TCP* ou *UDP*;
- Assinatura das aplicações e informações contidas nos cabeçalhos.

Tal processo de desempacotar várias camadas para examinar seu conteúdo em busca de características peculiares, tem um custo computacional considerável e para que esse complexo processo não se repita em todas as etapas seguintes, adota-se o procedimento chamado de marcação, onde códigos são inseridos em campos do cabeçalho *IP*, permitindo que um pacote possa ser reconhecido facilmente como pertencente a uma classe simplesmente pela leitura do campo marcado.

Na arquitetura *DiffServ* essa classificação acontece conforme a função do dispositivo: nos roteadores de borda o processo é completo, examinando os dados, fazendo e corrigindo marcações e direcionando os pacotes para as filas correspondentes, enquanto que nos equipamentos do núcleo o direcionamento dos pacotes se dá apenas mediante a leitura das marcações feitas anteriormente nas bordas (SANTANA, 2006, p. 31).

3.2.2 As Filas

As filas são regiões de memória alocadas pelos componentes do mecanismo de controle de tráfego para auxiliarem nas tarefas de classificação e controle da vazão de um determinado do fluxo de dados. Em roteadores que implementam a diferenciação de tráfego, as filas são essencialmente importantes, uma vez que cada um dos fluxos identificados durante a classificação é direcionado para uma fila em particular, correspondendo à sua classe.

Um parâmetro que deve ser levado em conta com relação a uma fila é o seu tamanho. Uma fila muito grande pode implicar em altas taxas de atraso e *jitter*, enquanto que uma fila muito pequena pode transbordar facilmente durante as tentativas de enviar dados para a rede de forma mais rápida do que ela pode aceitar, resultando em excesso de perda de pacotes (MELO, 2001, p. 32).

3.2.3 Mecanismos de Escalonamento

Examinar o tráfego que entra no roteador para classificá-lo, marcá-lo e separá-lo em filas não faria muito sentido, se depois de todo esse processo ele apenas fosse conduzido às interfaces de saída sem qualquer tratamento. A peça central na priorização é o escalonador: decide que pacote será atendido primeiro em um sistema de filas múltiplas (NETO, 2006, p. 17).

O principal recurso disputado pelos pacotes que chegam a um roteador é o imediato acesso à interface de saída e o escalonador é o componente responsável por gerenciar a distribuição deste recurso. A maneira como o escalonador desempenha sua tarefa depende do mecanismo utilizado, dentre os quais serão apresentados os mais comuns nas próximas subseções.

3.2.3.1 *FIFO - First In First Out*

O *FIFO* (primeiro a entrar, primeiro a sair) é o mecanismo de escalonamento trivial, oferecendo apenas a funcionalidade de armazenar e encaminhar os pacotes conforme a disponibilidade da rede. A ordem de chegada dos pacotes ao roteador é que determina a distribuição dos recursos, de maneira que o primeiro a chegar será transmitido assim que o caminho na rede estiver disponível.

Sendo a implementação padrão da maioria dos roteadores, este mecanismo não provê qualquer diferenciação no tratamento oferecido a fluxos de dados e, nessas situações, também dispensa as fases de classificação e marcação dos pacotes. Isto não quer dizer que o *FIFO* seja inútil na implementação de serviços diferenciados, uma vez que ele pode ser usado no tratamento de pacotes pertencentes a um mesmo fluxo ou classe de dados (PRIOR, 2001, p. 31).

3.2.3.2 *PQ - Priority Queueing*

Com uma abordagem totalmente oposta ao *FIFO*, que não provê qualquer diferenciação no tratamento oferecido a fluxos de dados, o mecanismo *PQ* (enfileiramento prioritário) trabalha com um esquema rígido de priorização do acesso à interface de saída (Figura 3.3).

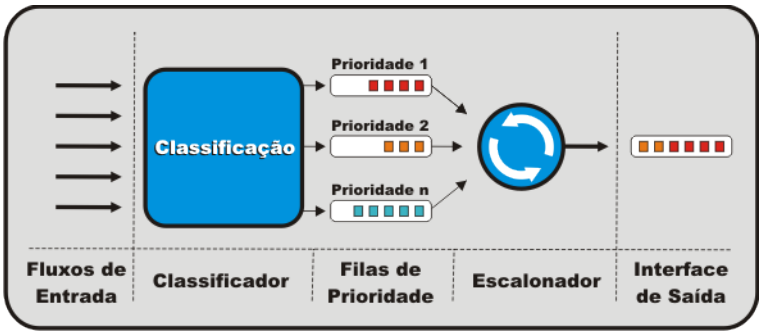

Figura 3.3: *PQ - Priority Queueing*. Fonte: Adaptado de (SANTANA, 2006, p. 70).

Este mecanismo consiste na definição de filas com prioridades diferentes (alta, média, normal e baixa, por exemplo), as quais são rigidamente servidas pelo escalonador, que só começa a atender uma fila quando todas as outras com prioridade maior que a dela estiverem completamente vazias (COUTO, 2007, p. 34).

Para aplicações de alta prioridade o mecanismo funciona de forma satisfatória, porém, as filas de prioridade inferior tende a sofrer diversos problemas com a falta do recurso, como altas taxas de atraso e *jitter* e, em casos extremos, até perder completamente o acesso ao recurso, quando seus pacotes deixam de ser transmitidos (SANTANA, 2006, p. 70).

3.2.3.3 *FQ - Fair Queueing*

A família de algoritmos *Fair Queueing - FQ* (enfileiramento justo) se baseia nos conceitos propostos por Nagle (1985) e possui como meta garantir que todos os fluxos de dados tenham acesso à largura de banda disponível, numa forma de distribuição "justa" do recurso, conforme ilustrado na Figura 3.4.

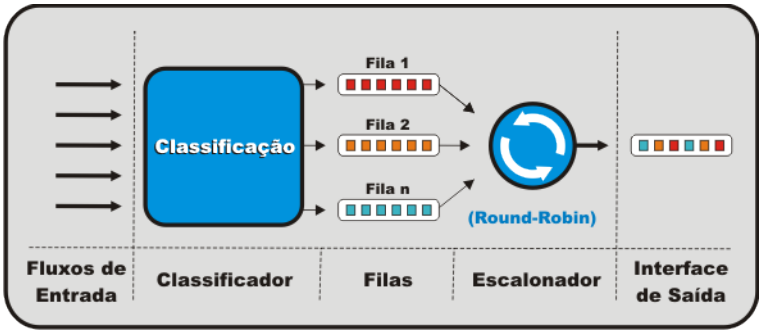

Figura 3.4: *FQ - Fair Queueing*. Fonte: Adaptado de (SANTANA, 2006, p. 72).

Tal distribuição justa acontece mediante a identificação dos pacotes provenientes de comunicações distintas, o que no *Weighted Fair Queueing - WFQ* (enfileiramento justo ponderado), é feito dinamicamente, com base apenas em dados como os endereços de origem e destino, tipo de protocolo, número de porta e valores do campo *TOS* (DAVIDSON *et al.*, 2008, p. 190).

Todavia, pode ser inviável criar filas individuais para todos os fluxos de pacotes, de maneira que o algoritmo *SFQ - Stochastic Fairness Queueing* (Enfileiramento Estocástico Justo) utiliza funções de *hash* para distribuir os pacotes do tráfego entre as filas existentes. Na ocorrência de fluxos diferentes serem classificados em uma mesma fila, eles compartilharão a banda que deveria ser destinada a apenas um deles, de maneira que para minimizar os impactos de tais ocorrências, a função *hash* responsável pela distribuição dos pacotes é substituída periodicamente (GOMES; SATURNINO, 2006, p. 58).

3.2.3.4 *CBQ - Class Based Queueing*

O *Class Based Queueing - CBQ* (enfileiramento baseado em classes) é uma variante do *PQ*, criada para contemplar priorização de tráfego, mas sem o inconivente de deixar fluxos de menor prioridade sem acesso ao recurso. Cada fila é servida de um percentual fixo da banda por um escalonador *round-robin*, numa sequência pré-definida (COUTO, 2007, p. 35), como ilustrado na Figura 3.5 (SANTANA, 2006, p. 71).

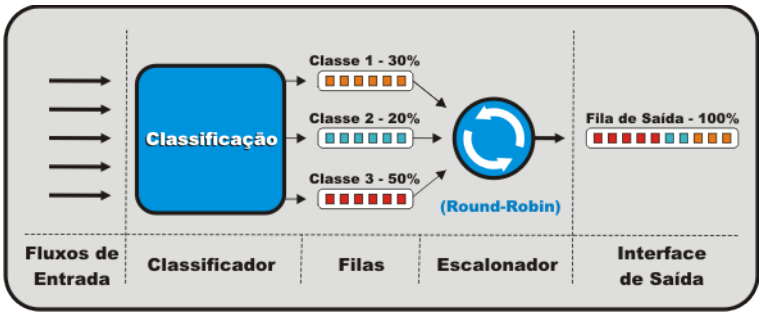

Figura 3.5: *CBQ - Class Based Queueing*. Fonte: Adaptado de (SANTANA, 2006, p. 71).

O *CBQ* foi o primeiro mecanismo a fornecer garantias efetivas de banda (SANTANA, 2006, p. 71), característica que permite sua utilização nos pontos de uma rede mais suscetíveis a situações de congestionamento, numa estratégia de assegurar a entrega de uma porção fixa da largura de banda a um tráfego selecionado sem, contudo, impedir que a capacidade remanescente seja utilizada pelos demais fluxos (MELO, 2001, p. 36).

3.2.3.5 *HTB - Hierarchical Token Bucket*

Não obstante o *CBQ* tenha surgido como uma alternativa ao *PQ*, a maneira como ele implementa suas funcionalidades também tem sido alvo de várias críticas, principalmente com relação à complexidade e imprecisão dos cálculos utilizados na definição das fatias de banda (HUBERT, 2002, p. 219).

Adotando a mesma abordagem do *CBQ*, de criar classes de tráfego e fracionar a largura de banda disponível entre elas, mas utilizando uma estratégia diferente para implementar esta divisão, surgiu o mecanismo de escalonamento conhecido como *HTB - Hierarchical Token Bucket* (balde de fichas hierárquico), definido por

seus autores como um substituto mais compreensível, intuitivo e rápido para o *CBQ* (DEVERA; COHEN, 2002).

O *HTB* permite o uso de um enlace físico para criar classes que funcionem como enlaces lógicos mais lentos, conforme quantidade de banda a ela destinada (SILVA; ALBUQUERQUE, 2009, p. 3). Cada classe recebe uma fração pré-definida da banda total disponível e pode ofertá-la a outras classes, fracionando-a novamente, criando então a estrutura hierárquica a que o nome do mecanismo faz referência (Figura 3.6).

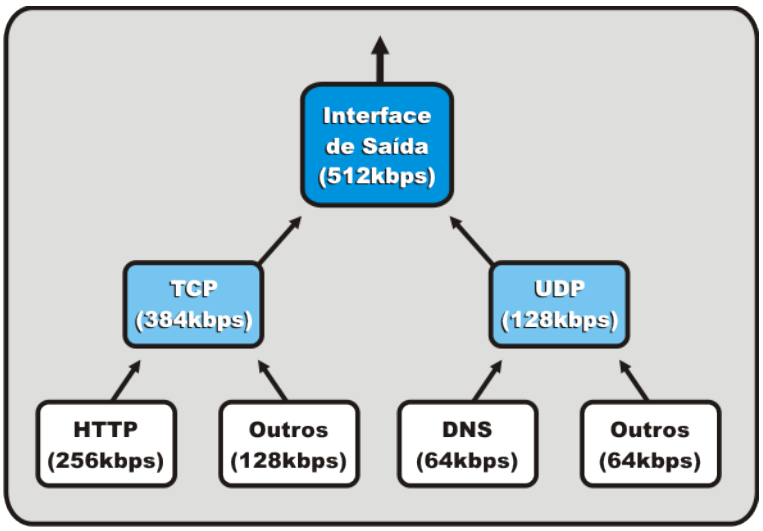

Figura 3.6: *HTB - Hierarchical Token Bucket*. Fonte: Adaptado de (GOMES; SATURNINO, 2006, p. 67).

Uma vez que a divisão de banda do *HTB* é baseado no conceito de "balde de fichas" (SANTANA, 2006, p.78-79), é possível fazer a seguinte analogia: cada classe recebe uma certa quantidade de fichas em intervalos regulares de tempo e cada ficha dá o direito de transmitir uma quantidade predefinida de dados. A cada transmissão de dados a classe devolve uma quantidade dessas fichas, de acordo com o montante de tráfego encaminhado e no caso das mesmas se acabarem, ela precisará aguardar até que receba mais de sua classe pai.

Além disso, uma classe pode repassar à outra as fichas que não utilizou, pelo processo conhecido como "*borrowing*" (empréstimo) (SILVA; ALBUQUERQUE, 2009, p. 3). Ao se definir a banda que um enlace lógico receberá e o valor máximo a que ela poderá se estender, o *ceil* (teto), também fica implícita a quantidade

desse recurso que poderá ser obtida através de empréstimos. Definir a banda de um enlace igual ao *ceil* corresponde a impedi-lo de fazer empréstimos.

3.3 Limitando o Uso de Recursos

Limitar o uso de recurso é uma estratégia útil em varias situações, como quando duas redes de velocidades diferentes estão conectadas, por exemplo. Nesse caso será necessário limitar a banda que chegará a rede de menor velocidade para que essa não fique congestionada (COUTO, 2007, p. 38). Serão abordadas nesta seção duas técnicas utilizadas para controlar o uso de recursos em uma rede: o policiamento e a suavização de tráfego.

3.3.1 *Policiamento*

Esta é uma das técnicas utilizadas para monitorar se o consumo dos recursos disponíveis em uma rede está de acordo com o conjunto de parâmetros que definem as condições em que o tráfego encaminhado por um cliente será aceito pelo fornecedor de um serviço de conexão. Tais parâmetros estão normalmente especificado no Contrato de Nível de Serviço - *SLA* (FILHO, 2006, p. 22).

O policiamento faz a verificação instantânea do tráfego e imediatamente executa uma ação pré-configurada, caso ocorra uma violação. O mecanismo é baseado na definição de um limite de tráfego que uma interface pode receber ou transmitir e, caso esse limite seja ultrapassado, por padrão, os pacotes excedentes são descartados.

Um recurso passível de ser implementado em domínios *DiffServ*, como alternativa ao descarte imediato de pacotes, é a remarcação do tráfego excedente com uma codificação de prioridade mais baixa, de forma que a ação sobre esses fluxos fique a cargo do próximo roteador do domínio (SANTANA, 2006, p. 74-75).

3.3.2 *Suavização*

A suavização, também conhecida como *shaping* (modelagem), é uma técnica alternativa ao policiamento, na tarefa de controlar os excessos no consumo de banda, sendo sua principal característica o uso de *buffers* para armazenar o tráfego excedente, evitando assim a adoção do descarte de pacotes como ação padrão.

A suavização pode otimizar a quantidade de dados transmitidos, simplesmente armazenando os pacotes excedentes para encaminhá-los quando houver disponibilidade de banda e isso implica em um comportamento mais linear e suave do tráfego, como exemplificado na Figura 3.7.

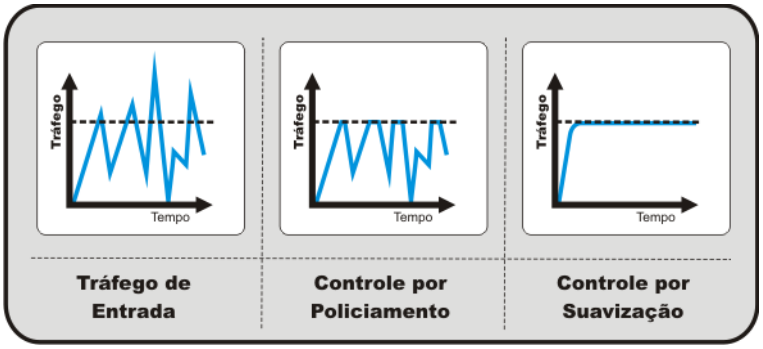

Figura 3.7: Controle no Uso de Recursos - Fonte: Adaptado de (SANTANA, 2006, p. 75).

É importante ressaltar que o descarte de pacotes ainda pode ocorrer quando se utiliza a suavização para controlar o tráfego excedente, bastando para isso que as situações de congestionamento se prolonguem por um intervalo de tempo superior à capacidade de acumulação dos *buffers* utilizados (SANTANA, 2006, p. 75).

Capítulo 4

Controle de Tráfego no *Linux*

O *Linux* possui desde a versão 2.2 do seu *kernel* um subsistema de rede completamente redesenhado, cujo código proporciona desempenho e quantidade de recursos capaz de posicioná-lo em um grupo com poucos competidores entre os sistemas operacionais. Os recursos de roteamento, filtragem e classificação disponíveis neste novo código são mais numerosos até que os fornecido por muitos dos roteadores dedicados (HUBERT *et al.*, 2003, p. 6).

O *Linux* disponibiliza, através de seu *kernel* e de algumas ferramentas que o auxiliam no controle de tráfego, um vasto conjunto de recursos ligados à provisão de *QoS*, incluindo neste grupo todos os elementos necessários para implementar as arquiteturas *IntServ* e *DiffServ* (ALMESBERGER *et al.*, 2000, p. 440). Nos tópicos seguintes serão abordados alguns desses recursos, mantendo o foco nos elementos necessários para prover controle de banda por meio de *DiffServ*.

4.1 Subsistema de Rede

O subsistema de rede do *Linux* apresenta um notável desempenho ao lidar com o tratamento e encaminhamento de fluxos de dados, o que se deve, principalmente, ao fato dessas tarefas serem realizadas por funções nativas do próprio *kernel* do sistema. Quando considerada a configuração padrão do *Linux*, no entanto, os recursos oferecidos pelo sistema não diferem muito daqueles que são encontrados em um típico roteador, contendo basicamente a estrutura apresentada na Figura 4.1 (ALMESBERGER, 1999, p. 48).

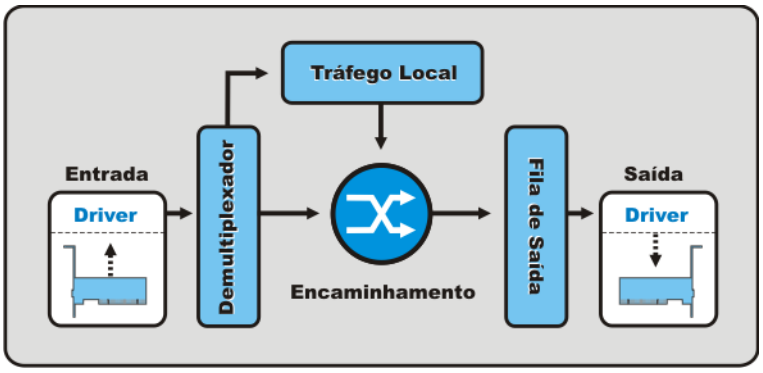

Figura 4.1: Roteamento no *Linux*. Fonte: Adaptado de (ALMESBERGER, 1999, p. 48).

Em termos de funcionamento, temos a seguinte descrição (FILHO, 2006, p. 38-43) (BLESS; WEHRLE, 1989, p. 2-3): os dados recebidos eletricamente na interface de entrada são copiados pelo *driver* deste dispositivo paras as estruturas internas do *kernel*, onde serão reconvertidos em pacotes pelas rotinas do componente demultiplexador.

Cada pacote recuperado da rede será processado de acordo com seu endereço de destino, de maneira que se um pacote está destinado à máquina local, ele terá seu conteúdo devidamente entregue às rotinas do sistema que implementam os protocolos da camada de transporte, conforme o tipo dos dados que estão sendo transportados pelo pacote (*TCP, UDP*, etc.).

Caso o sistema local esteja configurado para atuar como um roteador e o pacote recebido possui como destino outra máquina da rede, então ele será repassado para o bloco responsável pelo encaminhamento de pacotes, o qual examinará a tabela de roteamento para determinar o próximo ponto no destino e a interface que será utilizada no envio dos dados.

Além das funções de roteamento, realizadas através do repasse de pacotes entre as interfaces de rede, o bloco de encaminhamento também é utilizado para dar destinação ao tráfego gerado na máquina local, encapsulando em pacotes e destinando corretamente os dados provenientes das camadas superiores.

A fase final do encaminhamento de pacotes é justamente a reorganização do tráfego em fluxos individuais, conforme o segmento de rede a que serão destinados. Cada grupo de pacotes destinados a um mesmo segmento é ordenado e direcionado à fila de saída da interface correspondente, para dali serem coletados e convertidos em sinais elétricos pelo conjunto *driver*/interface de saída.

4.2 Pontos para Inserção de Elementos de Controle

A estrutura anteriormente apresentada corresponde ao funcionamento do *Linux* como um roteador básico, sem tratamento diferenciado para qualquer dos fluxos que por ele trafega. Uma representação gráfica alternativa para a mesma estrutura é exibida na Figura 4.2, que detalhando melhor o modelo anterior, revela a existência de uma fila unificada para todos os fluxos de entrada e de filas individuais para cada uma das interfaces de saída (BLESS; WEHRLE, 1989, p. 3).

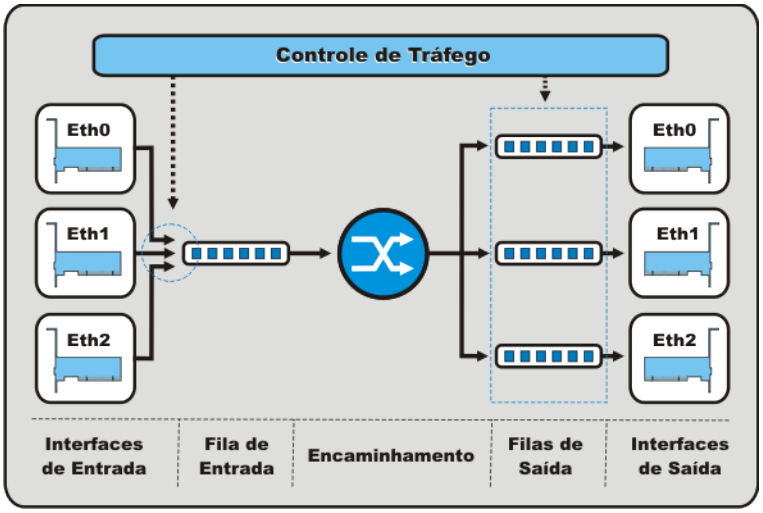

Figura 4.2: Pontos para Adição de Elementos Controle. Adaptado de (FILHO, 2006, p. 44) e (BLESS; WEHRLE, 1989, p. 3).

Além de omitir a representação do tráfego gerado na máquina local, por sua irrelevância na análise em questão, a figura destaca alguns dos pontos onde podem ocorrer as principais atuações dos componentes do controle de tráfego do *Linux*. O primeiro desses pontos representa justamente o momento em que os pacotes recuperados a partir das sequências de *bits* recebidas nas interfaces de entrada são enfileirados para ingressarem no bloco de encaminhamento (FILHO, 2006, p. 44).

Este ponto, conhecido como *Ingress Qdisc*[1] (disciplina de enfileiramento de entrada), é especialmente interessante para aplicação de policiamento, uma vez que os pacotes recém recuperados já podem ser examinados, antes mesmo de demandarem processamento do bloco de encaminhamento.

[1] *Qdisc: Queueing Discipline* (Disciplina de Enfileiramento).

O segundo ponto destacado na figura anterior é conhecido como *Egress Qdisc* (disciplina de enfileiramento de saída) e corresponde ao local onde atuam os mais importantes componentes do controle de tráfego do *Linux*. É justamente neste ponto que acontece o enfileiramento dos pacotes destinados a uma interface de saída, quando então o sistema permite a inclusão dos principais elementos para a provisão de *QoS*, conforme exibido na Figura 4.3 (ALMESBERGER, 1999, p. 49).

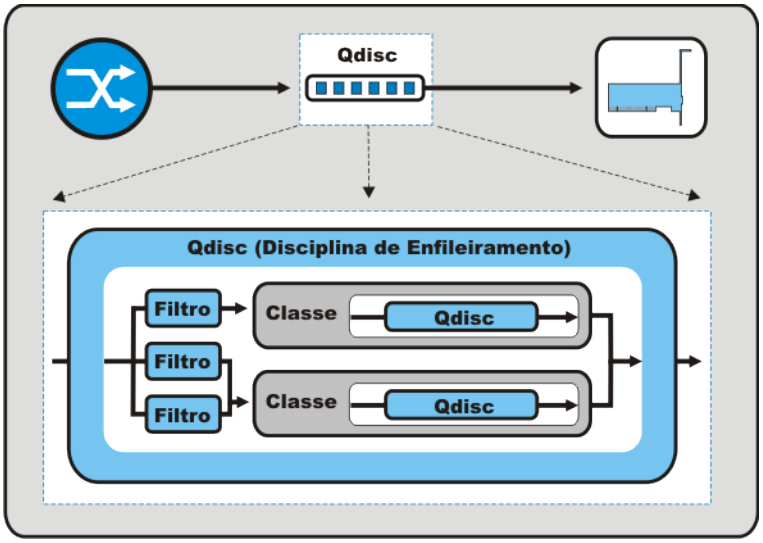

Figura 4.3: Elementos de Controle de Tráfego no *Linux*. Fonte: Adaptado de (ALMESBERGER, 1999, p. 49).

4.3 Elementos de Controle de Tráfego no *Linux*

O *Linux* oferece suporte a arquitetura *DiffServ* desde a versão 2.2 de seu *kernel* (através da aplicação de um *patch*), sendo que todas versões posteriores à 2.4 contemplam nativamente em seu código esta facilidade (CHOWDHURY, 2005, p. 35), enquanto que o mecanismo de escalonamento *HTB* foi incluído apenas a partir da versão 2.4.20 (DEVERA; COHEN, 2002).

Tal suporte a serviços diferenciados no *Linux* não se baseia na instalação ou execução de um simples aplicativo, mas na utilização de alguns dos elementos de controle de tráfego disponibilizados pelo sistema, como escalonadores, classes e filtros. A descrição de tais elementos no restante desta seção, excetuando menções expressas a outros autores, será baseada em Hubert *et al.* (2003).

4.3.1 Escalonadores

Os escalonadores são os principais componentes do controle de tráfego do *Linux*, onde são conhecidos como *qdisc* (*queueing discipline*), ou disciplina de enfileiramento, cujo próprio nome faz menção à sua funcionalidade de tratar a ordem em que os pacotes em sua entrada devem aparecer em sua saída.

O *Linux* possui dois grupos distintos de escalonadores: os "classificados" (*classful*), capazes de receber internamente outros elementos, como classes, filtros e até outros escalonadores, e os "não classificados" (*classless*), que não contemplam tal adição de outros componentes. Segundo Hubert *et al.* (2003) e Brown (2006), o código fonte do *kernel* do *Linux* contempla os seguintes escalonadores *Classless*:

- *First In First Out - FIFO* (*PFIFO e BFIFO*)
- *Packet FIFO Fast* (*PFIFO-FAST*)
- *Stochastic Fairness Queuing* (*SFQ*)
- *Random Early Detection* (*RED*)
- *Token Bucket Filter* (*TBF*)

E também os seguintes escalonadores *classful*:

- *Class Based Queuing* (*CBQ*)
- *Hierarchical Token Bucket* (*HTB*)
- *Hierarchical Fair Service Curve* (*HFSC*)
- *Multi Band Priority Queueing* (*PRIO*)
- *Generic Random Early Detection* (*GRED*)
- *Differentiated Service Marker* (*DSMARK*)

Hubert *et al.* (2003) esclarece ainda, que os termos *Ingress Qdisc* e *Egress Qdisc*, utilizados na seção 4.2, não se referem propriamente a escalonadores, mas aos locais destinados à inserção dos elementos de controle de tráfego responsáveis pelo tratamento dos fluxos que entram (*ingress*) e saem (*egress*) da máquina.

Enquanto que a *Ingress Qdisc* contempla apenas a adição de seletos elementos policiadores de tráfego, a *Egress Qdisc*, que interliga o bloco de encaminhamento a uma interface de saída, representa o componente central do controle de tráfego no sistema. Por sua notável versatilidade, permite configurações tão simples quanto um único mecanismo *FIFO*, ou tão complexas quanto uma estrutura *HTB* contendo várias classes e filtros.

Tal versatilidade se deve principalmente ao fato do *Linux* permitir que o escalonador padrão *classless* associado à *Egress Qdisc* seja substituído por uma opção *classful*, capaz de suportar a adição de outros componentes que podem ser utilizados na composição de estruturas de controle de tráfego mais complexas, exatamente como apresentado, anteriormente, na Figura 4.3.

Este conjunto de funcionalidades faz com que tal escalonador associado à *Egress Qdisc* para oferecer suporte aos demais elementos do controle de tráfego, seja conhecido normalmente como "escalonador principal". Quanto ao escalonador *classless* mencionado como padrão para a *Egress Qdisc*, ele é uma variante do mecanismo *FIFO* chamada de *PFIFO-FAST* que, apesar de disponibilizar uma estrutura simples de priorização baseada apenas em marcações do campo *TOS*, é pouco configurável e funciona exatamente como o *FIFO* na maioria dos casos.

4.3.2 Classes

As classes são os elementos que respondem pelo tratamento diferenciado oferecido a fluxos de dados específico. Um escalonador *classful* suporta a inclusão de outros elementos, estando entre eles as classes, que por sua vez podem se ramificar em outras classes, denominadas "filhas". Quando uma classe é a última de sua hierarquia, não possuindo outras como filhas, então ela é chamada de "folha".

Quando uma nova classe é criada, um escalonador do tipo *FIFO* é imediatamente associado a ela, mas se para essa classe for criada uma outra classe como filha, esse escalonador será automaticamente removido, passando a existir apenas na nova classe "folha", o ponto final da hierarquia.

Apesar do escalonador padrão anexado a uma classe recém criada ser do tipo *FIFO*, ele poderá ser substituído por outro que seja considerado mais adequado, incluindo os que estejam entre as opções *classful* e permitam a inclusão de novas classes, o que expande consideravelmente as combinações de estruturas que podem ser implementadas no sistema (HUBERT *et al.*, 2003, p. 50).

4.3.3 Filtros

Não faria muito sentido criar uma estrutura complexa dentro de um escalonador principal, contendo várias classes e até outros escalonadores, se não houvesse um meio de identificar e direcionar seletivamente os fluxos de dados para esses elementos. Esse conceito é o mesmo que define a já explanada classificação, que no *Linux* acontece através do uso de filtros.

Os filtros são componentes bastante flexíveis, de maneira que a classificação corresponde a apenas uma de suas funcionalidades. Em linhas gerais, os filtros são compostos de dois elementos: um mecanismo destinado a identificar padrões nos pacotes e uma ação predefinida, que deve ser aplicada aos pacotes onde tais padrões forem encontrados.

Uma particularidade desse tipo de estrutura, é que cada elemento adicionado ao escalonador principal possui um identificador único, de maneira que a ação definida para um filtro de classificação de tráfego é simplesmente encaminhar os pacotes identificados pelo padrão para o componente que possui um identificador predefinido. Quanto aos filtros disponíveis para o controle de tráfego no *Linux*, Hubert *et al.* (2003) cita:

- **FW:** Este filtro baseia suas decisões em marcações prévias do *netfilter*[2], o que faz dele uma opção interessante para aqueles que já estão familiarizados com a estrutura de comandos do *iptables*[3], por exemplo.

- **U32:** Este é considerado o principal filtro do *Linux*, já que permite a identificação de padrões em qualquer parte de um pacote *IP*, embora seja normalmente usado para pesquisas em campos do cabeçalho, como protocolos, endereços e portas de origem e/ou destino. Além de pesquisas em campos predefinidos, ele também pode examinar um pacote como uma sequência de *bits*, buscando padrões, a partir de um deslocamento em relação ao início do cabeçalho ou de outra parte conhecida do pacote.

- **Route:** Este filtro se baseia nas rotas de origem e destino de um pacote para definir a ação que a ele deve ser aplicada.

- **TCINDEX:** Baseia suas ações a partir da identificação de marcações prévias de um escalonador *DSMARK*.

[2] http://www.netfilter.org/.
[3] http://www.netfilter.org/.

4.3.4 Policiamento e Suavização no *Linux*

Além de classificadores de pacotes, alguns filtros também podem atuar como policiadores de tráfego no *Linux*, o que pode ser feito mediante a definição de um volume máximo de tráfego a ser aceito, considerando apenas os pacotes que se enquadrarem nos padrões de classificação do filtro, e uma ação a ser aplicada, caso esse limite seja excedido.

O primeiro fator a ser considerado nesses casos, é a necessidade de um mecanismo capaz de medir a quantidade de tráfego que atravessa o filtro, o que pode ser feito, basicamente, de duas formas no *Linux*. A primeira delas é por meio de um cálculo estimativo, baseado em medições sucessivas que são feitas pelo *kernel*, as quais se repetem 25 vezes a cada segundo (HUBERT *et al.*, 2003, p. 83).

A outra forma de adicionar métricas ao tráfego é utilizando um escalonador *classless* do tipo *TBF - Token Bucket Filter*, que, como o próprio nome expressa, controla a entrega de pacotes com base no mecanismo do balde de fichas, cujo conceito foi apresentado na subseção 3.3.2, que aborda a técnica de suavização.

Tais situações, onde um filtro é submetido um volume de tráfego superior à quantidade máxima previamente definida, são conhecidas como "*overlimit*". Com relação às ações a serem adotadas quando um filtro entra em "*overlimit*", Hubert *et al.* (2003) cita quatro delas:

- **Continue:** Impede que os pacotes do tráfego excedente sejam classificados pelo filtro em questão, mas permite que eles continuem sendo testados pelos demais filtros, mantendo a possibilidade de que este tráfego ainda seja classificado em algum deles.
- **Drop:** Essa é a ação que representa o comportamento padrão que seria adotado no policiamento: os pacotes do tráfego excedente são simplesmente descartados.
- **Pass/OK:** Essa ação faz com que o filtro simplesmente ignore as situações de "*overlimit*" e continue a classificar os pacotes normalmente. Esse tipo de ação pode ser utilizado para desativar temporariamente o policiamento em um filtro, sem precisar substituí-lo por outro.
- **Reclassify:** Esta é a ação padrão definida para as situações de "*overlimit*" em um filtro, onde os pacotes excedentes são simplesmente direcionados para o atendimento reservado ao tráfego não classificado (*default*), o que corresponde normalmente ao serviço de melhor esforço.

4.3.5 Identificando os Elementos na Estrutura

Visualizar os elementos de uma estrutura de controle de tráfico, bem como identificar as relações existentes entre eles, podem ser tarefas relativamente simples, se considerado o uso de figuras para tais fins, uma vez que as representações gráficas se valem de artifícios como linhas e setas para indicar a hierarquia e as ligações existentes entre os componentes.

Na prática, porém, tais artifícios inexistem, de maneira que para reproduzir computacionalmente uma estrutura graficamente representada, é necessário primeiro traduzi-la em comandos aceitos pelo sistema em questão. A Figura 4.4 será utilizada para exemplificar algumas das convenções que devem ser adotadas durante a criação de uma estrutura de controle de tráfego no *Linux*.

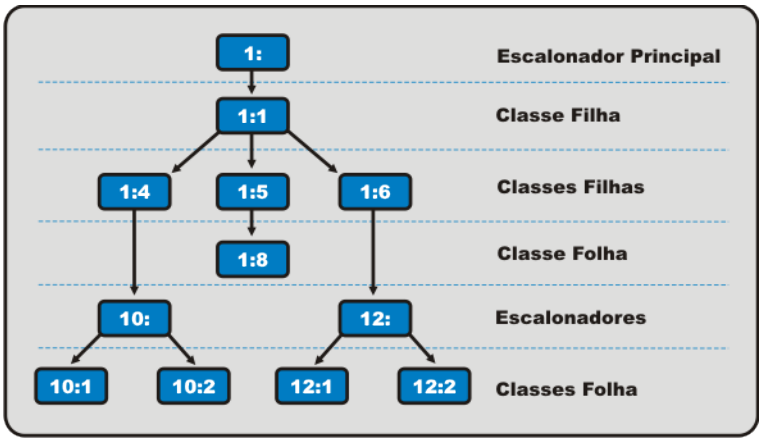

Figura 4.4: Identificando os Elementos. Fonte: Adaptado de (HUBERT *et al.*, 2003, p. 53).

O primeiro aspecto a ser considerado é que os elementos não são criados ligados uns aos outros, como aparecem nas figuras, ao invés disso, eles são individualmente inseridos no sistema, mesmo quando criados pela execução de um único arquivo em lote. A ligação lógica entre eles só é possível pela existência dos identificadores chamados de *handle* (nos escalonadores) e *classid* (nas classes).

Com exceção do escalonador principal, todos os demais componentes ao serem criados precisam informar o Identificador de seu elemento "pai", o que permite recriar logicamente as relações de hierarquia. Os filtros, apesar de não possuírem um identificador próprio, precisam declarar, durante a sua criação, tanto o identificador do seu elemento "pai", de onde procederá o tráfego a ser filtrado, como o identificador do elemento que receberá o tráfego por ele classificado (*flowid*).

Cada identificador é composto por dois números de 16 *bits*, dispostos da seguinte forma: 0xFFFF:0xFFFF (número-maior:número-menor) (ALMESBERGER, 1999, p. 56). O "número maior" precisa ser único em cada um dos escalonadores da estrutura e, caso não tenha sido escolhido pelo usuário, será gerado automaticamente pelo sistema. O mesmo não acontece com as classes, que devem adotar obrigatoriamente o "número maior" de seu elemento pai (*parent*).

Quanto ao "número menor", ele deve ser omitido ou adotado como 0 (zero) em elementos do tipo escalonador, sendo que o próprio sistema irá ignorar outros valores que sejam informados para este campo. Deve se observar ainda a obrigatoriedade na adoção de "números menores" exclusivos entre classes de um mesmo escalonador, devido à necessidade de se manter a existência de identificadores únicos na estrutura, uma vez que elas já compartilham entre si o mesmo "número maior", herdado compulsoriamente de seu elemento pai.

Outro aspecto comumente observado nos exemplos literários, é a adoção do *handle* "1:" (que também equivale a "1:0") como identificador padrão para o escalonador principal. Tal prática possui efeito apenas didático, uma vez que o próprio sistema costuma adotar outros identificadores para este elemento, quando um valor não é expressamente informado no comando de criação.

4.4 A Ferramenta *tc*

Não obstante o caráter revolucionário das funcionalidades adicionadas ao subsistema de rede do *Linux*, a partir a versão 2.2 de seu *kernel*, a maioria delas sequer poderiam ser utilizadas pelos usuários do sistema, se não surgissem também ferramentas apropriadas para a configuração das mesmas. Neste sentido foi desenvolvido o pacote *iproute2*[4], que consiste em um conjunto de utilitários de linha de comando, destinados a configurar as funcionalidades ligadas às redes *IP*, através da manipulação de estruturas do *kernel* (BROWN, 2006).

Dentre tais utilitários que fazem parte do pacote *iproute2*, se encontra também o binário "*tc*", que funciona como uma interface de configuração das funcionalidades de controle tráfego que estão presentes no *kernel* do *Linux*. A estrutura de comandos do *tc* é bastante flexível, aceitando uma infinidade de parâmetros, de maneira que serão apresentados nesta seção apenas alguns exemplos básicos dos comandos aceitos pela ferramenta.

[4]Disponível em `http://www.linux-foundation.org/en/Net:Iproute2`.

A Figura 4.5, que apresenta uma estrutura básica de controle de tráfego, será utilizada como exemplo, com o objetivo de ter sua representação gráfica recriada no sistema, a partir de comandos executados pela ferramenta "*tc*". A seguinte sintaxe básica, que representa uma pequena fração das opções disponíveis, será utilizada nos comandos a serem executados:

```
# tc [ elemento ] [ opção ] [ dev dispositivo ] [ parâmetros ]
```

Onde:

- **tc:** a ferramenta;
- **elemento:** qdisc | class | filter (escalonador, classe, filtro);
- **opção:** add | del | show (adicionar, remover, exibir);
- **dispositivo:** a interface utilizada: eth0, ppp0, etc.

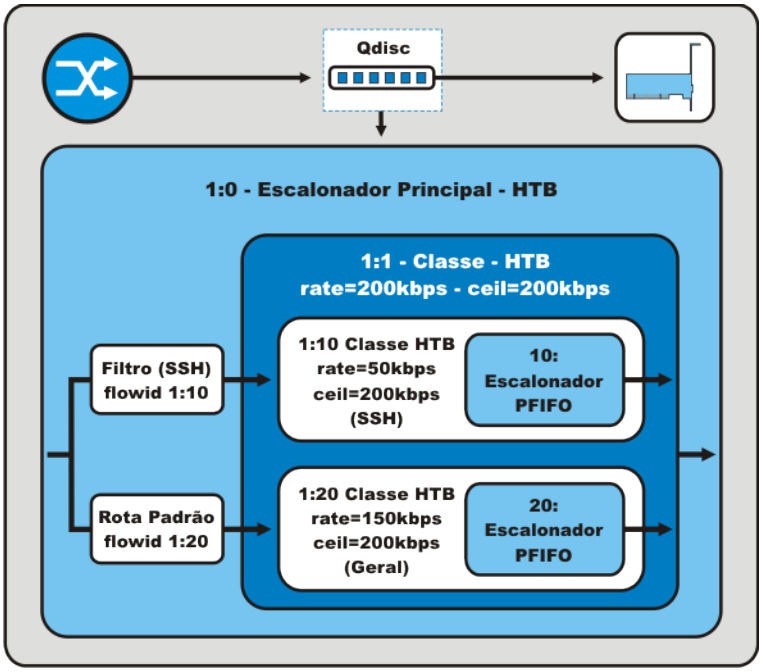

Figura 4.5: Exemplo de Estrutura de Controle de Tráfego

A proposta da estrutura apresentada na Figura 4.5 é utilizar uma mecanismo baseado no escalonador *HTB* para limitar o tráfego da interface de saída a 200 kbps, distribuídos da seguinte forma: 50 kbps (expansíveis até 200 kbps) para pacotes provenientes de fluxos *SSH* e 150 kbps (expansíveis até 200 kbps) para os demais pacotes do tráfego.

Adicionando o escalonador principal (1:0):

```
# tc qdisc add dev eth0 root handle 1:0 htb default 20
```

- **tc qdisc add:** adicionar um elemento do tipo escalonador;
- **dev eth0:** a interface na qual o elemento será adicionado;
- **root:** informa que se trata do elemento "raiz" (principal);
- **handle 1:0:** o identificador do elemento;
- **htb:** o tipo do elemento;
- **default 20:** informa que o elemento filho com "número menor" 20 receberá todo fluxo que não foi classificado pelos filtros.

Adicionando a classe 1:1:

```
# tc class add dev eth0 parent 1:0 classid 1:1 htb \
     rate 200kbit ceil 200kbit burst 15k
```

- **tc class add:** adicionar um elemento do tipo classe;
- **dev eth0:** a interface na qual o elemento será adicionado;
- **parent 1:0:** informa o identificador do elemento pai;
- **classid 1:1** informa o identificador deste elemento;
- **htb:** o tipo do elemento;
- **rate 200kbit:** a banda padrão reservada para o elemento;
- **ceil 200kbit:** a banda máxima para o elemento (incluindo empréstimos);
- **burst 15k:** a quantidade máxima de tráfego a ser transmitida numa "rajada".

Adicionando as classes 1:10 e 1:20 (procedimento análogo ao anterior):

```
# tc class add dev eth0 parent 1:1 classid 1:10 htb \
     rate 50kbit ceil 200kbit burst 15k
# tc class add dev eth0 parent 1:1 classid 1:20 htb \
     rate 150kbit ceil 200kbit burst 15k
```

Adicionando os escalonadores 10: (10:0) e 20: (10:0). Este procedimento tem fins apenas didáticos, uma vez que, segundo Hubert *et al.* (2003), uma classe folha utiliza automaticamente um enfileiramento baseado no mecanismo *FIFO*, caso nenhum escalonador tenha sido a ela adicionado:

```
# tc qdisc add dev eth0 parent 1:10 handle 10: pfifo
# tc qdisc add dev eth0 parent 1:20 handle 20: pfifo
```

Adicionando o filtro que direciona o tráfego *SSH* para a classe 1:10:

```
# tc filter add dev eth0 parent 1: protocol ip  prio 1 \
     u32 match ip sport 22 0xffff flowid 1:10
```

- **tc filter add:** adicionar um elemento do tipo filtro;
- **parent 1:** informa o elemento pai deste filtro, cujo tráfego será filtrado;
- **protocol ip:** informa o protocolo do filtro;
- **prio 1:** a ordem em que este filtro será consultado na lista de filtros;
- **u32:** o tipo do filtro;
- **match ip sport 22 0xffff:** a pesquisa do filtro será feita no cabeçalho *IP*, no campo "porta de origem" e o valor procurado será 22 (*SSH*);
- **flowid 1:10:** o destino dos pacotes que coincidirem com o padrão do filtro.

Uma estrutura pode conter vários filtros e vários deles podem apontar para uma mesma classe. Todavia, o exemplo proposto adota apenas um filtro, sendo que o tráfego que não for classificado por ele será direcionado para a classe 1:20, declarada como destino padrão (*default*) durante a criação do elemento ao qual o filtro foi anexado, o escalonador principal (1:0).

A Figura 4.6 exibe listagem completa dos comandos utilizados, enquanto que um exemplo de utilização da opção *show* é exibido na Figura 4.7. Uma descrição completa das opções aceitas pela ferramenta *tc* pode ser consultada em sua própria *man page* no sistema.

```
# tc qdisc add dev eth0 root handle 1:0 htb default 20
# tc class add dev eth0 parent 1:0 classid 1:1 htb \
     rate 200kbit ceil 200kbit burst 15k
# tc class add dev eth0 parent 1:1 classid 1:10 htb \
     rate 50kbit ceil 200kbit burst 15k
# tc class add dev eth0 parent 1:1 classid 1:20 htb \
     rate 150kbit ceil 200kbit burst 15k
# tc qdisc add dev eth0 parent 1:10 handle 10: pfifo
# tc qdisc add dev eth0 parent 1:20 handle 20: pfifo
# tc filter add dev eth0 parent 1: protocol ip  prio 1 \
     u32 match ip sport 22 0xffff flowid 1:10
```

Figura 4.6: Lista Completa dos Comandos Utilizados

```
# tc qdisc show dev eth0
  qdisc htb 1: root r2q 10 default 20 direct_packets_stat 0
  qdisc pfifo 10: parent 1:10 limit 1000p
  qdisc pfifo 20: parent 1:20 limit 1000p

# tc class show dev eth0
  class htb 1:10 parent 1:1 leaf 10: prio 0 rate 50000bit ceil 200000bit \
  burst 15Kb cburst 1599b
  class htb 1:1 root rate 200000bit ceil 200000bit burst 15Kb cburst 1599b
  class htb 1:20 parent 1:1 leaf 20: prio 0 rate 150000bit ceil 200000bit \
  burst 15Kb cburst 1599b
```

Figura 4.7: Utilizando a Opção *show*

Capítulo 5

Estudo de Caso: Controle de Banda Utilizando *Linux*

No capítulo anterior foram apresentadas algumas das características do subsistema de rede do *Linux*, juntamente com alguns dos elementos por ele disponibilizados para as tarefas de controle de tráfego. Apesar dos inúmeros recursos presentes no sistema, os quais permitem seu uso em projetos extremamente complexos de controle de tráfego, estruturas que contemplem a provisão de funcionalidades básicas também podem ser implementadas com relativa simplicidade.

Em conformidade com esta afirmação, este capítulo apresenta um caso de implementação de controle de banda, utilizando os elementos de controle de tráfego do *Linux*, como solução para um problema de compartilhamento da capacidade de transmissão da conexão de dados de um escritório de médio porte do serviço público estadual de Minas Gerais.

5.1 A Instituição

A Secretaria de Estado de Meio Ambiente e Desenvolvimento Sustentável de Minas Gerais - SEMAD[1] é responsável pela coordenação do Sistema Estadual do Meio Ambiente e Recursos Hídricos - SISEMA, que possui entre seus componentes os seguintes órgãos: Fundação Estadual do Meio Ambiente - FEAM, Instituto Estadual de Florestas - IEF e Instituto Mineiro de Gestão das Águas - IGAM.

[1] `http://www.semad.mg.gov.br/`

Numa estratégia de manter suas unidades de atendimento o mais próximo possível da população que depende de seus serviços, alguns desses órgãos adotam um sistema de gestão hierárquico descentralizado, dispondo de unidades administrativas regionais, as quais são responsáveis pelos escritórios de atendimento instalados nos municípios de sua jurisdição. O estudo de caso aqui exposto tem como cenário uma das unidades regionais de um dos órgãos do SISEMA.

Por questões de segurança e ética profissional serão propositalmente omitidas as informações detalhadas de localização geográfica, dados de usuários e outras que não forem estritamente relevantes para a compreensão do problema e da proposta de solução expressa neste documento. Pelos mesmos motivos, todos os endereços de rede mencionados neste relato também serão meramente fictícios.

5.2 Redes e Sistemas Envolvidos

A Companhia de Tecnologia da Informação do Estado de Minas Gerais - Prodemge[2], além de oferecer inúmeros outros serviços, representa o nó central de uma rede *intranet* acessada por diversas unidades administrativas do Estado. Sua política interna de segurança permite que apenas unidades mantendo conexões dedicadas com seus servidores possam ter acesso a alguns desses serviços.

Entre tais serviços disponibilizados exclusivamente para as unidades conectadas à rede *intranet* da companhia, está o acesso aos sistemas administrativos e financeiros da Secretaria de Estado de Planejamento e Gestão - SEPLAG, mantidos numa arquitetura do tipo *mainframe*, e acessados por meio de terminais remotos sobre o protocolo *telnet*.

O acesso a tais sistemas administrativos e financeiros da SEPLAG não é apenas desejável, mas um requisito imprescindível para que as unidades administrativas estaduais ordenadoras de despesas e geradoras de documentos de receita possam desempenhar suas atividades cotidianas de planejamento e gestão em conformidade com os padrões estabelecidos para a administração pública estadual.

Além de suas atividades que desempenha como unidade administrativa estadual, o escritório adotado como objeto do estudo de caso em questão é primordialmente uma repartição pública do SISEMA, e como tal depende de alguns serviços de rede para quase todas as suas atividades de atendimento ao público.

[2] `http://www.prodemge.gov.br/`

Tais sistemas voltados às atividades de meio ambiente desenvolvidas pela unidade são baseados na arquitetura cliente-servidor, onde o programa cliente é instalado localmente nas máquinas do escritório e o servidor de banco de dados permanece hospedado nos domínios de rede da Prodemge, sendo acessado remotamente pelas máquinas que o utilizam.

Por fim, a Prodemge disponibiliza ainda um servidor *proxy*, utilizado como porta de acesso à Internet para tais redes conectadas a seus servidores pelos *links* dedicados. A Figura 5.1 reúne em um diagrama todos os elementos citados até este ponto e os demais componentes da estrutura de rede do escritório, sendo este o ambiente tecnológico envolvido no estudo de caso.

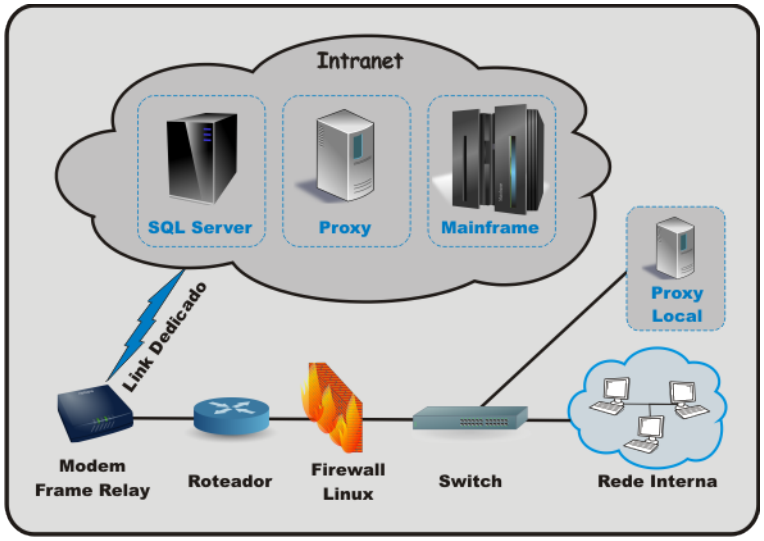

Figura 5.1: Detalhamento dos Elementos e Serviços Envolvidos na Estrutura

5.3 Histórico do Problema na Unidade

O escritório em questão foi criado em 1996, quando então possuía 3 computadores e se conectava com a *intranet* da Prodemge através de um *link* dedicado de 256 kbps, utilizando a tecnologia *Frame Relay*. Nesta ocasião, apenas um desses computadores podia se conectar ao *mainframe* da Prodemge e os demais sistemas de gestão utilizados eram baseados em um banco de dados local que executava, semanalmente, rotinas de sincronização com o servidor de banco de dados remoto.

Em 2006, a estrutura que já possuía 12 computadores, dos quais 2 acessavam o *mainframe* da Prodemge, passou por mais uma importante modificação, substituindo os sistemas baseados no banco de dados local por versões que se conectavam diretamente ao servidor de banco de dados remoto. Nessa ocasião, apenas um entre os 12 computadores da unidade acessava esse sistema "*online*".

Tal solução eliminou o atraso na atualização das informações lançadas no sistema, mas também tornou a realização de determinadas tarefas totalmente dependente da conexão com as redes da Prodemge, onde se encontrava o banco de dados central utilizado. Apesar disso, a largura de banda do *link* permaneceu inalterada, sob a justificativa de que o planejamento financeiro do órgão não contemplava investimentos em melhoria de infraestrutura, impedindo alterações nos contratos de prestação de serviços já assumidos.

Em 2007, quando todos os sistemas instalados na estrutura já sofriam com as paralisações constantes, oriundas das sobrecargas do *link* com a Prodemge, foi instalado nas dependências do escritório uma máquina com o sistema operacional *Linux*, executando o servidor *proxy Squid*[3], numa estratégia de minimizar a concorrência do tráfego *web* com as conexões de dados utilizadas pelos sistemas.

A solução implementada com o *Squid* adotava além de seu trivial sistema de *cache* a liberação de acesso apenas para algumas páginas nos horários de maior utilização dos sistemas, numa estratégia que diminuiu sensivelmente os problemas, mas que não evitou que as paralisações nos sistemas continuassem a acontecer, ainda que em uma frequência menor.

No inicio de 2009, contemplado por um projeto de reestruturação de unidades regionais, o escritório seu mudou para uma nova sede e teve todos os seus computadores e equipamentos de rede substituídos, além de receber uma nova rede interna com cabeamento estruturado. O projeto previa ainda o aumento da largura de banda do *link* de dados para 2 Mbps, mas a operadora que fornecia o serviço alegou que não podia comportar novas ofertas de banda para os clientes da região.

Ao final do projeto de reestruturação, em agosto de 2009, a estrutura da unidade era composta por 20 computadores, dos quais 4 acessavam o *mainframe* da Prodemge e 3 possuíam os sistemas que utilizavam o banco de dados remoto do órgão. O *link* de dados permanecia em 256 kbps, não mais por limitações financeiras, mas por falta de oferta de banda nas operadoras da região, enquanto todos os sistemas continuavam a apresentar instabilidades em momentos imprevisíveis, mesmo que o serviço estivesse normal nas demais unidades do estado.

[3]Disponível em `http://www.squid-cache.org/`.

5.4 Descrição Objetiva do Problema

Além dos acessos a páginas de Internet, que são considerados como desejáveis no ambiente, existem outras fontes secundárias geradoras de tráfego, como atualizações de programas e de definições de vírus, as quais podem ocorrer muitas vezes em momentos não programados pelo usuário.

Uma vez que os sistemas que acessam o banco de dados do órgão e os terminais de acesso remoto ao *mainframe* da Prodemge são considerados essenciais para a realização das atribuições da unidade, o problema atual da estrutura é a falta de instrumentos para controlar a distribuição da escassa largura de banda, de tal maneira que apenas a capacidade não demandada pelos sistemas definidos como essenciais possa ser dividida entre os demais acessos registrados na rede.

5.5 Proposta de Solução para o Problema

Utilizar os elementos de controle de tráfego presentes no *Linux*, numa forma de construir uma arquitetura baseada em priorização, capaz de garantir a reserva da largura de banda exigida pelos sistemas considerados essenciais. O restante da capacidade de transmissão do *link* de dados deverá ser dividido da forma mais justa possível entre os demais fluxos presentes na rede.

O mecanismo de controle de banda proposto como solução para os problemas levantados neste estudo de caso deverá ser implementado no equipamento anteriormente descrito na Figura 5.1 como "Firewall *Linux*", cujas as configurações de *hardware* e de *software* são idênticas às do equipamento utilizado nas medições de tráfego, descrito na Seção A.2 do Apêndice A.

5.6 Medições Iniciais

Não faria sentido falar sobre a implantação de qualquer solução de controle de banda sem que primeiramente fossem tomadas algumas medições com relação à estrutura onde se pretende implementar tal solução. Para este estudo de caso em particular serão levantadas informações como: a capacidade total da rede, se esta capacidade se mantém estável com o passar do tempo e quanto de banda deverá ser reservada para cada fluxo. Tais medições, apresentadas nas próximas subseções, são baseadas na estratégia de medição apresentada no Apêndice A.

5.6.1 A Capacidade do Link de Dados

Uma vez que este é o recurso cujo compartilhamento será alvo do processo de controle, as medições consideradas imprescindíveis serão aquelas relativas à capacidade total do parâmetro que se deseja controlar: a largura de banda. As Figuras 5.2 e 5.3 apresentam, respectivamente, os gráficos referentes às medições de *download* e *upload* do *link* de dados, os quais expressam as seguintes medições:

- ***Download* (Mínimo / Máximo / Médio):** 232.1k / 257.3k / 249.8k
- ***Upload* (Mínimo / Máximo / Médio):** 233.4k / 258.4k / 246.9k

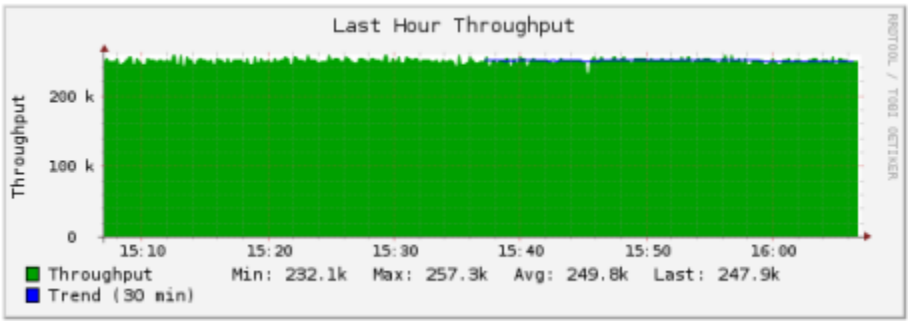

Figura 5.2: Capacidade Total do *Link* de Dados - *Download*

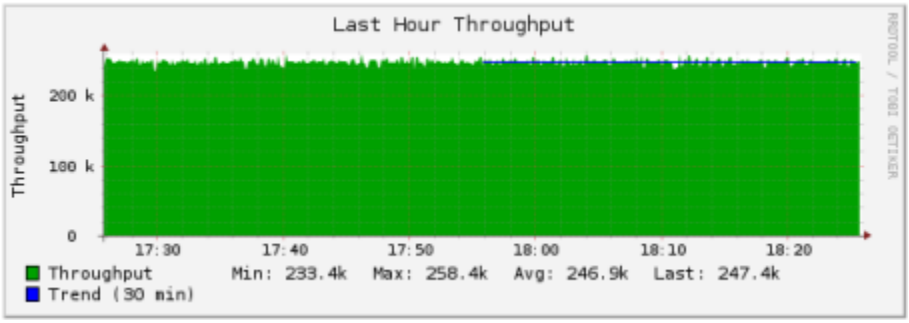

Figura 5.3: Capacidade Total do *Link* de Dados - *Upload*

Uma vez que os valores de *download* e *upload* de tais medições são muito semelhantes, inclusive apresentando poucas variações nos gráficos ao longo do tempo, assumiremos que o *link* de dados possui a mesma capacidade para ambos sentidos do tráfego, de maneira que o valor exato de 250 kbps, por aproximação com médias registradas, será assumido como a capacidade máxima do *link*.

5.6.2 Demanda de Banda dos Sistemas Essenciais

Tão importante quando conhecer a capacidade de transmissão do *link* de dados compartilhado, é saber a largura de banda demandada por cada uma das aplicações que terão uma fatia do recuso reservada na rede. Para este estudo de caso em particular, dois sistemas serão definidos como preferenciais, com relação à utilização da largura de banda disponível.

O primeiro desses sistemas corresponde aos terminais de acesso ao servidor *mainframe* da Prodemge, cujos gráficos contendo os resultados das medições de consumo de banda para a execução de apenas um desses terminais são apresentados nas Figuras 5.4 e 5.5.

A reserva de banda para este sistema será feita com base nos valores de pico registrados: 9,6 kbps para *download* e 3,4 kbps para *upload*. Considerando que 4 (quatro) desses terminais são atualmente executados no ambiente, teremos, com arredondamento dos valores, 40 kbps de *download* e 15 kbps de *upload*.

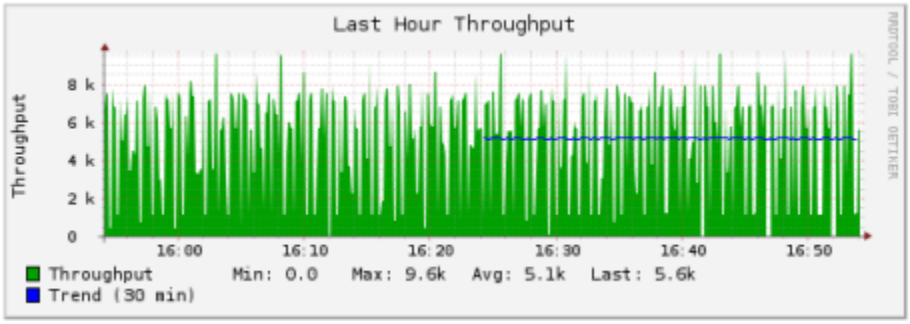

Figura 5.4: Terminal de Acesso ao *mainframe* - *Download*

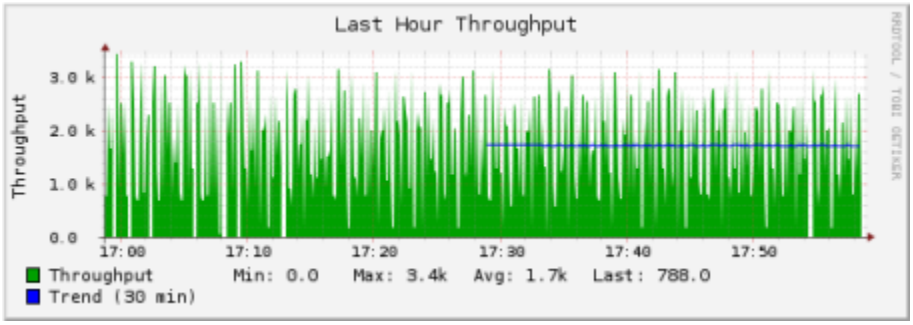

Figura 5.5: Terminal de Acesso ao *mainframe* - *Upload*

Outro aspecto a ser levado em conta com relação a esses terminais de acesso remoto é sua natureza de sistema interativo, o que implica em uma atenção especial quanto ao tempo de resposta para as requisições do usuário. Dessa forma, além de reservar a fatia de banda necessária, pode ser também interessante garantir que seus pacotes tenham alguma prioridade no atendimento, com relação aos demais fluxos de dados presentes na rede.

O segundo sistema definido como preferencial é um *software* concebido pela própria instituição que acessa um banco de dados remoto, sendo essa conexão a responsável pelo consumo de banda da aplicação. As Figuras 5.6 e 5.7 apresentam os gráficos de consumo de banda medidos para uma única instância do sistema. Considerando que até 3 (três) instâncias podem ser executadas simultaneamente, assume-se, com arredondamento dos valores, uma demanda de 80 kbps de *download* e 40 kbps de *upload*.

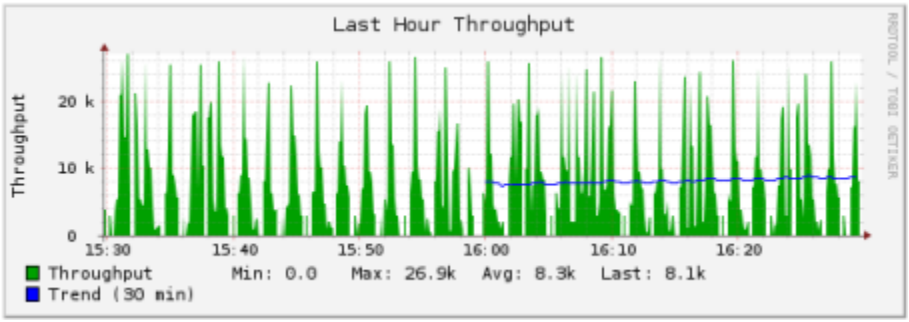

Figura 5.6: Sistema Acessando o Banco de Dados Remoto - *Download*

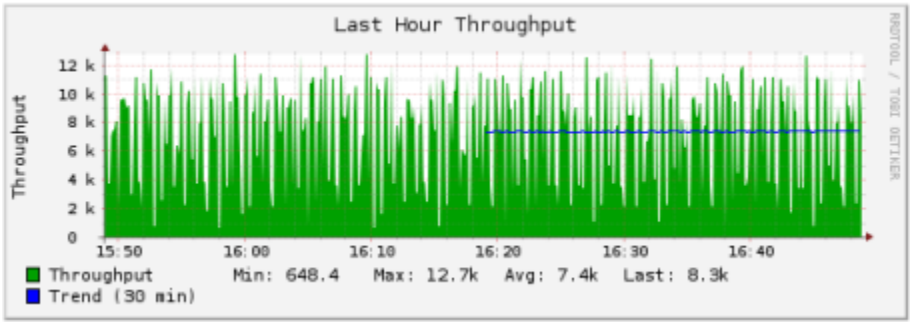

Figura 5.7: Sistema Acessando o Banco de Dados Remoto - *Upload*

5.7 Reservando a Largura de Banda Necessária

Em linhas gerais, o *script* apresentado na Figura 5.8, que será utilizado para fazer as reservas de largura de banda neste estudo de caso, não difere muito daquele que foi utilizado na Seção 4.4, para exemplificar os comandos da ferramenta *tc*. Em decorrência disso, apenas serão detalhados os elementos do atual *script* que não estavam presentes em tal exemplo anterior.

O primeiro desses elementos é a cláusula "prio", presente nos comandos de criação dos escalonadores *htb*, a qual se refere à prioridade de atendimento das ramificações de uma classe. Valores mais baixos indicam uma prioridade maior, de maneira que o valor "0" (zero) foi atribuído à classe responsável pela reserva de banda para os terminais interativos de acesso ao *mainframe* da Prodemge.

Os escalonadores do tipo *SFQ - Stochastic Fairness Queueing* (descrito na Subseção 3.2.3.3) representam outro elemento que não estava presente nos exemplos anteriores, mas que aqui foi adicionado com a finalidade de garantir uma distribuição de banda mais uniforme entre os fluxos que fazem parte de uma mesma classe. Ele será especialmente importante para a classe que atende o tráfego geral, uma vez que o número de fluxos que compartilharão essa fatia de banda será, indiscutivelmente, o maior da estrutura. Com relação aos principais parâmetros utilizados na criação desses escalonadores, temos:

- ***quantum*:** A quantidade de *bits* que deve ser servida a uma classe antes de atender uma outra. Deve ser, no mínimo, o tamanho de um pacote.
- ***perturb*:** O tempo, em segundos, entre os recálculos do *hash* para o *sqf*.

Uma questão digna de explicação é a ausência da tradicional lista de requisitos do *kernel* para que se possa trabalhar com controle de tráfego no *Linux*. O autor optou por não fazer a costumeira inclusão desse artefato por alguns motivos. O primeiros deles é o fato de quase todos os trabalhos a incluírem exatamente como aparece nos documentos referentes às versões 2.2 e 2.4 do *kernel*, sem ao menos consultar sua validade para as versões atuais.

Quanto a isso, após uma consulta básica ao código-fonte da versão 2.6.33 do *kernel*, na seção "*Networking support -> Networking options -> QoS and/or fair queueing*", o autor concluiu que muitas das opções presentes em tais listagens de requisitos sequer existem atualmente e que as demais, o que inclui as que permaneceram válidas e as novas que surgiram, já estão automaticamente habilitadas para a compilação padrão do referido código-fonte.

```
#!/bin/sh
#-----------------------------------------------------------------------------------------
# Script de reserva da largura banda usando o HTB
#-----------------------------------------------------------------------------------------
# Definindo variaveis

  IF_INT=eth0            # Interface ligada  a rede interna
  IF_EXT=eth1            # Interface ligada  a rede externa
  LINK_DLD=250           # Capacidade de download do link - em kbps
  LINK_ULD=250           # Capacidade de upload do link - em kbps

  SYS1_DLD=40            # Sistema01: demanda de download - em kbps
  SYS1_ULD=15            # Sistema01: demanda de upload - em kbps
  SRV1_IP=192.168.0.10   # Sistema01: IP do servidor utilizado
  SRV1_PORT=23           # Sistema01: Porta do servico utilizado

  SYS2_DLD=80            # Sistema02: demanda de download - em kbps
  SYS2_ULD=40            # Sistema02: demanda de upload - em kbps
  SRV2_IP=192.168.0.20   # Sistema02: IP do servidor utilizado
  SRV2_PORT=1433         # Sistema02: Porta do servico utilizado

#-----------------------------------------------------------------------------------------
# Ativando o controle de banda

# uso: bw_control [iface] [link_bw] [sys01_bw] [sys02_bw]
bw_control(){

  GERAL=$(echo "`expr $2 - $3 - $4`kbit")
  tc qdisc del dev $1 root
  tc qdisc add dev $1 root handle 1:0 htb default 30
  tc class add dev $1 parent 1:0 classid 1:1 htb rate $2kbit ceil $2kbit burst 15k
  tc class add dev $1 parent 1:1 classid 1:10 htb rate $3kbit ceil $2kbit burst 15k prio 0
  tc class add dev $1 parent 1:1 classid 1:20 htb rate $4kbit ceil $2kbit burst 15k prio 1
  tc class add dev $1 parent 1:1 classid 1:30 htb rate $GERAL ceil $2kbit burst 15k prio 2
  tc qdisc add dev $1 parent 1:10 handle 10:0 sfq quantum 1514 perturb 10
  tc qdisc add dev $1 parent 1:20 handle 20:0 sfq quantum 1514 perturb 10
  tc qdisc add dev $1 parent 1:30 handle 30:0 sfq quantum 1514 perturb 10
  tc filter add dev $1 parent 1:0 prio 1 $SYS1_FILTER flowid 1:10
  tc filter add dev $1 parent 1:0 prio 2 $SYS2_FILTER flowid 1:20
}

# Reservando banda para o trafego de download
SYS1_FILTER="protocol ip u32 match ip src $SRV1_IP match ip sport $SRV1_PORT 0xffff"
SYS2_FILTER="protocol ip u32 match ip src $SRV2_IP match ip sport $SRV2_PORT 0xffff"
bw_control $IF_INT $LINK_DLD $SYS1_DLD $SYS2_DLD

# Reservando banda para o trafego de upload
SYS1_FILTER="protocol ip u32 match ip dst $SRV1_IP match ip dport $SRV1_PORT 0xffff"
SYS2_FILTER="protocol ip u32 match ip dst $SRV2_IP match ip dport $SRV2_PORT 0xffff"
bw_control $IF_EXT $LINK_ULD $SYS1_ULD $SYS2_ULD
```

Figura 5.8: *Script* de Reserva da Largura Banda

Essa idéia de que as novas versões do *kernel* do *Linux* já contemplam, por padrão, os elementos necessários para a implementação de controle de tráfego, também foi observada na distribuição utilizada neste estudo de caso, de maneira que a versão 5.0.4 do *Debian*, com *kernel* 2.6.26-2, não precisou de qualquer alteração para operacionalizar a estrutura de controle de banda proposta. Mesmo não fazendo parte do *kernel*, é importante lembrar que nem mesmo o programa *iproute2* precisou ser instalado nesta ocasião.

5.8 Resultados da Implementação

Do ponto de vista do usuário de uma rede de dados, a principal comprovação da eficiência de uma implementação de controle de banda é, evidentemente, o funcionamento satisfatório de suas aplicações. Considerando que a intenção da implementação apresentada neste estudo de caso é justamente garantir a reserva de uma fatia de banda para os sistemas definidos como essenciais, a melhor maneira de comprovar a validade da solução proposta é certamente a demonstração de que tais aplicações não sentirão mais os efeitos negativos da existência de outros fluxos de dados concorrentes na rede.

Algumas particularidades da aplicação baseada no banco de dados remoto fazem dela a melhor candidata para ser utilizada neste tipo de demonstração, sendo a primeira delas o fato de seu consumo de banda ser nitidamente o maior registrado nos gráficos. Além disso, ela também se mostrou muito mais sensível ao problema da falta de largura de banda que as demais aplicações.

Neste sentido, enquanto que os terminais de acesso remoto ao *mainframe* da Prodemge traduzem a questão da falta de largura banda por meio de uma lentidão que prejudica bastante sua interatividade, os sistemas baseados no banco de dados remotos simplesmente perdem a conexão e se tornam inoperantes, provocando a perda dos dados não salvos e obrigando a reabertura do programa.

Durante os testes realizados após a implementação do controle de banda proposto, no entanto, tais situações de perda de conexão não foram mais observadas, mesmo considerando os momentos de total sobrecarga do *link* de dados avaliado. Tal informação pode ser visualizada na Figura 5.9, que apresenta gráficos diferentes para medições realizadas durante a mesma execução do sistema em questão.

Enquanto o gráfico superior mostra a banda consumida durante uma das execuções do sistema avaliado, o gráfico inferior apresenta o consumo total da rede no mesmo instante de tempo. Após algum tempo de medição foram disparadas,

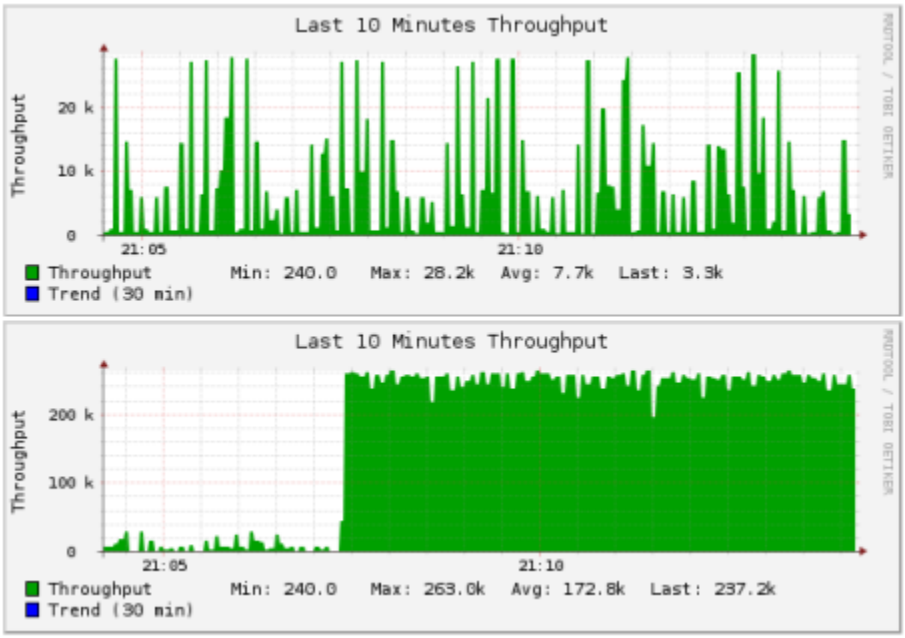

Figura 5.9: Comprovando a Reserva de Banda

em outra máquina da rede, algumas tarefas de *download* e abertura de páginas na *web*, numa nítida tentativa de sobrecarregar o *link* de dados.

O resultado observado é que não houve qualquer alteração significativa no gráfico que apresenta o consumo de banda para o sistema testado, comprovando a efetividade da solução de reserva de banda proposta. Além disso, outro efeito benéfico observado, mas não planejado pelo autor, foi que as páginas na *web* foram acessadas com relativa desenvoltura, mesmo considerando que a rede já estava funcionando em sua carga total.

Este aspecto positivo foi atribuído ao escalonador *SQF* anexado à classe responsável pelo tratamento do tráfego geral não classificado, de maneira que sendo um escalonador da família de enfileiramento justo evitou que os fluxos de *download* monopolizassem a estreita largura de banda que sobrara após as reservas.

Capítulo 6

Conclusão

A escassez de largura de banda para determinadas aplicações é um problema muito comum em ambientes que compartilham uma conexão de dados entre diversas máquinas. Neste sentido, foram apresentadas ao longo deste documento algumas considerações sobre as origens das redes *TCP/IP*, evidenciando que elas não foram criadas com base nos requisitos demandados por muitas das aplicações que atualmente a utilizam.

Em consonância com o objetivo geral mencionado no capítulo introdutório, a linha de trabalho adotada durante a criação deste documento foi a utilização das técnicas de pesquisa bibliográfica e experimentação com os seguintes objetivos:

- levantar material bibliográfico sobre a utilização dos elementos de controle de tráfego do sistema operacional *Linux* como alternativa viável na provisão de controle de banda em redes *TCP/IP*;
- verificar a aplicabilidade do conhecimento compilado a partir da pesquisa bibliográfica em um estudo de caso.

Os resultados obtidos pela aplicação de tal linha metodológica foram eficientes em satisfazer os objetivos propostos para este trabalho, uma vez que:

- pela pesquisa bibliográfica foi possível levantar as concepções ideológicas de um grupo diversificado de autores sobre a temática da qualidade de serviço e compilá-las numa apresentação dos conceitos básicos e das principais arquiteturas propostas para lidar com esta questão;

- ainda pela utilização da mesma linha metodológica, foi possível apresentar as funcionalidades oferecidas pelo subsistema de rede do *Linux*, através de seus elementos de controle de tráfego, destinadas à implementação de tais conceitos de qualidade de serviço;
- não obstante tal embasamento bibliográfico, uma comprovação prática da eficácia de se utilizar o *Linux* como ferramenta de provisão de controle de banda foi evidenciada através dos resultados observados no estudo de caso conduzido, cujas metas propostas foram integralmente alcançadas.

Outro resultado positivo obtido durante o desenvolvimento do trabalho foi a possibilidade de registrar a temática da qualidade de serviço e do controle de banda fazendo sempre um contraponto ao discurso voltado apenas para redes de grande porte. Tal discurso, presente na maioria dos trabalhos, tem sido, na opinião do autor, um dos grandes responsáveis pela falsa impressão de complexidade e inaplicabilidade da abordagem em ambientes menores.

Além de explícita no texto, tal forma preferencial de abordagem sobre o tema também foi mantida no desenvolvimento do estudo de caso, evidenciada na escolha de uma implementação trivial como solução para um problema sério, enfrentado por um ambiente real, caracterizado por um escritório de pequeno porte. Os resultados de tal experiência credenciam o autor a afirmar que uma solução não precisa necessariamente ser complexa para ser eficiente.

Exatamente por isso que a conclusão obtida na finalização deste trabalho é que o *Linux*, além de muito robusto e abundante em recursos, é também muito flexível. Tal característica permite que implementações de controle de tráfego possam contemplar os mais diversos ambientes, independente do número de usuários e de serviços oferecidos.

A abordagem sobre qualidade de serviço e controle de banda presente neste trabalho não resume de forma alguma as possibilidades de apresentação para esses temas, ao contrário disso, cria novas oportunidades a serem exploradas. Neste sentido, como perspectiva de trabalhos futuros, seria bastante útil a proposta de uma solução integrada de provisão de *QoS*, voltada para um maior aproveitamento da largura de banda nos ambientes de pequeno porte.

Além de utilizar os elementos de controle de tráfego do *Linux* para racionalizar a distribuição de banda, tal ferramenta deveria contemplar a instalação e configuração de aplicações voltadas ao *cache* de páginas e de endereços de rede. Caso uma solução desse tipo viesse a ser desenvolvida, considerando o público alvo a que se destina, seria especialmente importante que os esforços estivessem

focados na concepção de algum tipo de interface unificada e simplificada para a instalação e configuração das aplicações utilizadas.

Uma vez que também são comuns trabalhos voltados para a mesma temática de qualidade de serviço, mas utilizando as ferramentas disponíveis para os sistemas da família *Unix*, também seria igualmente interessante um estudo comparativo do desempenho obtido ao se implementar soluções semelhantes no *Linux* e em algum desses sistemas.

Referências Bibliográficas

AGOSTINHO, L.; SILVEIRA, D. S.; SOUSA, R. G.; FAINA, L. F. Reconfiguração Dinâmica de Classes Diffserv no Suporte a QoS face à Dinâmica da Rede. *Revista Hífen*, PUCRS Uruguaiana, v. 30, n. 58, p. 129–136, II Semestre 2006. Disponível em: <http://revistaseletronicas.pucrs.br/ojs/index.php/hifen-/article/view/3803/2898>.

ALMESBERGER, W. *Scalable Resource Reservation for the Internet*. 173 p. Tese (Doutorado) — Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1999. Disponível em: <http://www.almesberger.net/cv/papers-/thesis.pdf>.

ALMESBERGER, W.; GIORDANO, S.; MAMELI, R.; SALSANO, S.; SALVATORE, F. A Prototype Implementation for the Intserv Operation over Diffserv Networks. In: *Global Telecommunications Conference, 2000. GLOBECOM '00*. San Francisco, California, United States: IEEE, 2000. v. 1, p. 439–444.

BLAKE, S.; BLACK, D. L.; CARLSON, M. A.; DAVIES, E.; WANG, Z.; WEISS, W. *An Architecture for Differentiated Services*. RFC 2475, dec 1998. 34 p. Disponível em: <http://www.rfc-editor.org/rfc/rfc2475.txt>.

BLESS, R.; WEHRLE, K. Evaluation of differentiated services using an implementation under linux. In: *IWQoS '99: Seventh International Workshop on Quality of Service*. London, UK: IEEE, 1989. p. 10. Disponível em: <http://www.cs.ucl.ac.uk/research/iwqos99/papers/046-final.ps.gz>.

BRADEN, R.; CLARK, D.; SHENKER, S. *Integrated Services in the Internet Architecture*: an Overview. RFC 1633, jun 1994. 33 p. Disponível em: <http://www.rfc-editor.org/rfc/rfc1633.txt>.

BROWN, M. A. *Traffic Control HOWTO*. Guide to IP Layer Network Administration with Linux (LinuxIP), oct 2006. Version 1.0.2. Disponível em: <http://linux-ip.net/articles/Traffic-Control-HOWTO/>. Acesso em: 16 nov. 2009.

CHOWDHURY, M. A. I. *Priority Queuing for Real Time Services in Diffserv IPv6 Network*. 57 f. Dissertação (M Eng. Electrical-Electronics and Telecommunication) — Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia, 2005. Disponível em: <http://eprints.utm.my-/5323/>.

COUTO, A. R. *Controle de uso de banda garantindo QoS baseado em Diffserv usando FreeBSD*. 61 p. Monografia (Bacharelado em Ciência da Computação) — Curso de Ciência da Computação, Universidade Federal da Bahia, Salvador, 2007. Disponível em: <http://www.disciplinas.dcc.ufba.br/pub/MATA67-/TrabalhosSemestre20071/monografia-AndreCouto.pdf>.

DAVIDSON, J.; PETERS, J.; BATHIA, M.; KALIDINDI, S.; MUKHERJEE, S. *Fundamentos de VoIP*: Uma abordagem sistemica para a compreensao dos fundamentos de voz sobre ip. 2. ed. Porto Alegre: Bookman, 2008. 392 p.

DEVERA, M.; COHEN, D. *HTB Linux queuing discipline manual - user guide*. 2002. Updated in 5.5.2002. Disponível em: <http://luxik.cdi.cz/˜devik/qos/htb-/manual/userg.htm>. Acesso em: 13 aug. 2009.

FERGUSON, P.; HUSTON, G. *Quality of Service*: delivering QoS on the Internet and in corporate networks. New York: John Wiley & Sons Inc, 1998. 266 p.

FILHO, J. L. d. O. *Tecnologias DiffServ como Suporte para a Qualidade de Serviço (QoS) de Aplicações Multimídia*: Aspectos de configuração e integração. 122 f. Dissertação (Mestrado Profissional em Redes de Computadores) — Universidade Salvador - UNIFACS, Salvador, 2006. Disponível em: <http://tede.unifacs.br/tde_busca/arquivo.php?codArquivo=75>.

FOROUZAN, B. A. *Cominucação de Dados e Redes de Computadores*. 3. ed. São Paulo: Bookman, 2006. 840 p.

GOMES, N. M. V.; SATURNINO, R. A. N. Relatório Técnico, *QoS na Vídeo-Difusão sobre IPv6*. Leiria, Portugal: [s.n.], sep 2006. 233 p. Disponível em: <http://linux-ip.net/articles/Traffic-Control-HOWTO/>.

HUBERT, B. Linux Advanced Routing & Traffic Control. In: *Ottawa Linux Symposium*. Ottawa, Canada: [s.n.], 2002. p. 213–222. Disponível em:

<http://www.linuxsymposium.org/archives/OLS/Reprints-2002/Proceedings-OLS2002.pdf>.

HUBERT, B.; GRAF, T.; MAXWELL, G.; MOOK, R. v.; OOSTERHOUT, M. v.; SCHROEDER, P. B.; SPAANS, J.; LARROY, P. *Linux Advanced Routing & Traffic Control HOWTO*. DocBook Edition, oct 2003. Revision: 1.43. Disponível em: <http://lartc.org/lartc.pdf>. Acesso em: 6 feb. 2009.

KAMIENSKI, C. A.; SADOK, D. Qualidade de Serviço na Internet. In: SBC'2000. *Anais da XIX Jornada de Atualização em Informática*. Curitiba: Universitária Champagnat, 2000. II, p. 326. Disponível em: <http://www.cin-.ufpe.br/~cak/publications/jai2000.pdf>.

LEAL, M. A. A. *QoS - Qualidade de Serviço em TCP/IP*. 69 p. Monografia (Pós-Graduação em Administração de Redes Linux) — Departamento de Ciencia da Computação, Universidade Federal de Lavras, Lavras, 2004. Disponível em: <www.ginux.ufla.br/node/69>.

MELO, E. T. L. *Qualidade de Serviço em Redes IP com DiffServ*: Avaliação através de medições. 153 p. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Santa Catarina, Florianópolis, 2001.

NAGLE, J. *On Packet Switches With Infinite Storage*. RFC 970, dc 1985. 9 p. Disponível em: <http://www.rfc-editor.org/rfc/rfc970.txt>.

NETO, E. M. S. *Especificação de uma Rede MPLS Fim-a-Fim com Diferenciação de Serviços*. 173 p. Tese (Doutorado em Engenharia Elétrica) — Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, Natal, 2006. Disponível em: <http://bdtd.bczm.ufrn.br/tedesimplificado%-/tde_busca/processaArquivo.php?codArquivo=607>.

PRIOR, R. P. d. M. C. *Qualidade de Serviço em Redes de Comutação de Pacotes*. 146 p. Dissertação (Mestrado em Engenharia Electrotécnica e de Computadores) — Faculdade de Engenharia, Universidade do Porto, Porto, 2001. Disponível em: <http://purl.pt/5471/1/>.

QUINTERO, N. F. M. *Mecanismo do Kernel para o Controle de Tráfego e Variação da Taxa Fim a Fim - E2E*. 79 p. Dissertação (Mestrado em Engenharia de Telecomunicações) — Curso de Pós-Graduação em Engenharia de Telecomunicações, Universidade Federal Fluminense, Niterói, 2007. Disponível em: <http://www.bdtd.ndc.uff.br/tde_busca/arquivo.php?codArquivo=2243>.

SANTANA, S. F. L. *Proposta de Refência para Projetos de Qualidade de Serviço (QoS) em Redes Corporativas*. 185 f. Dissertação (Mestrado em Redes de Computadores) — Universidade Salvador - UNIFACS, Salvador, 2006. Disponível em: <http://tede.unifacs.br/tde_busca/arquivo.php?codArquivo=73>.

SANTOS, C. R. d. *Proposta e Análise de Desempenho de um Comutador de Pacotes com Enfileiramentos na Entrada e na Saída*. 83 p. Tese (Doutorado em Engenharia Elétrica) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2006. Disponível em: <http://libdigi.unicamp.br/document/?code=vtls000386349>.

SHENKER, S.; PARTRIDGE, C.; GUERIN, R. *Specification of Guaranteed Quality of Service*. RFC 2212, sep 1997. 20 p. Disponível em: <http://www.rfc-editor.org/rfc/rfc2212.txt>.

SILVA, C. R. d.; ALBUQUERQUE, C. V. N. d. Priorização de chamadas de voz em redes em malha sem fio. In: *XVII Simpósio Brasileiro de Telecomunicações (SBrT 2009)*. Blumenau, SC, Brasil: [s.n.], 2009. p. 6. Disponível em: <http://www.ic.uff.br/˜celio/papers/sbrt09.pdf>.

SOUSA, N. F. S. d. *Implantação de Controle de Banda para otimização do uso de rede de computadores através de QoS*. 69 p. Monografia (Bacharelado em Ciência da Computação) — Curso de Bacharelado em Ciência da Computação, Universidade Federal do Piauí, Teresina, 2005. Disponível em: <http://www.pop-pi.rnp.br/artigos/Nathan% underline MonografiaFinal.pdf>.

WROCLAWSKI, J. *Specification of the Controlled-Load Network Element Service*. RFC 2211, sep 1997. 19 p. Disponível em: <http://www.rfc-editor.org-/rfc/rfc2211.txt>.

Apêndice A

Estratégia de Medição do Tráfego

Seria incoerente falar sobre melhoria na qualidade de um serviço, sem apresentar quaisquer métricas que comprovassem a efetividade da solução adotada. Em conformidade com o estudo de caso proposto no Capítulo 5, são abordadas aqui questões relativas a medições do parâmetro largura de banda na rede avaliada.

Segundo Quintero (2007), existem duas modalidades de medição de tráfego: a ativa e a passiva. Ferramentas baseadas em medição ativa simulam as duas extremidades de uma conexão, gerando pacotes específicos para que os valores de parâmetros como perda de pacotes, atraso e *jitter* sejam obtidos diretamente da análise desses fluxos previamente gerados.

As ferramentas de medição passiva, por outro lado, apenas capturam e analisam os pacotes presentes no segmento de rede avaliado, os quais são gerados pelas aplicações reais ali executadas. Todas as medições apresentadas durante esse trabalho foram feitas utilizando esta abordagem, baseando-se unicamente no tráfego gerado pelos sistemas e aplicações utilizados no cotidiano da rede analisada.

A.1 Pontos de Medição

O equipamento responsável pela captura de pacotes do tráfego analisado foi instalado nos pontos destacados na Figura A.1. O Ponto marcado como “A” foi utilizado nas medições da capacidade total da rede, enquanto que os pontos marcados como “B” foram utilizados para medir as necessidades de largura de banda de cada instância de uma aplicação em particular.

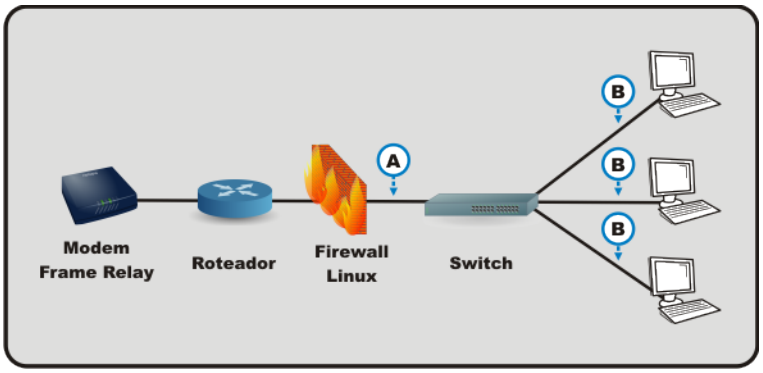

Figura A.1: Pontos Onde Serão Feitas as Medições de Tráfego

A.2 Equipamento Utilizado nas Medições

A ferramenta utilizada nas tarefas de medição foi instalada em um computador com duas interfaces de rede, configuradas em modo "*bridge*". Este equipamento foi conectado entre as interfaces de origem e de destino do segmento de rede onde aconteceu a medição. A seguinte descrição apresenta os principais componentes de *hardware* presente no equipamento utilizado:

- **Processador:** AMD Athlon^{TM} 64 X2 - 2.1GHz
- **Memória:** 2 GB DDR2-800 SDRAM
- **Disco Rígido:** 160GB - 7.200 rpm
- **Interfaces de Rede:** D-Link DGE-530T Gigabit

Quanto às configurações de *software*, o sistema operacional utilizado nesta máquina foi o *Linux*, na versão 5.0.4 da distribuição *Debian*[1] e como requisito para a criação da *bridge* foi instalado o pacote *bridge-utils*, através do comando:

```
# aptitude install bridge-utils
```

Após instalado o pacote *bridge-utils* no sistema, o *script* apresentado na Figura A.2 foi utilizado na criação de uma *bridge* com base nas duas interfaces de rede da máquina. O mesmo *script* possibilitou a adição de um endereço *IP* à interface criada, de modo que a máquina fosse localizada na rede.

[1] Disponível em `http://www.debian.org/`.

```
#!/bin/sh
#-----------------------------------------------------------------
# Script de configuracao das interfaces de rede
#-----------------------------------------------------------------
# Definindo variaveis
  IFCONFIG=/sbin/ifconfig       # Path do comando ifconfig
  BRCTL=/usr/sbin/brctl         # Path do comando brctl
  IF_INT=eth0                   # Interface ligada  a rede interna
  IF_EXT=eth1                   # Interface ligada  a rede externa
  IF_BRIDGE=br0                 # Interface bridge a ser criada
  IP_BRIDGE=192.168.0.100       # IP da bridge a ser criada
  MASK_BRIDGE=255.255.255.0     # Mascara da bridge a ser criada

#-----------------------------------------------------------------
# Criando a bridge
  $BRCTL addbr $IF_BRIDGE
  $BRCTL addif $IF_BRIDGE $IF_INT
  $BRCTL addif $IF_BRIDGE $IF_EXT
  $IFCONFIG $IF_INT 0.0.0.0 up
  $IFCONFIG $IF_EXT 0.0.0.0 up

#-----------------------------------------------------------------
# Definindo um IP para a bridge
  $IFCONFIG $IF_BRIDGE $IP_BRIDGE netmask $MASK_BRIDGE

#-----------------------------------------------------------------
# Habilitando forwarding no kernel
  echo "1" >/proc/sys/net/ipv4/ip_forward
```

Figura A.2: Script de Configuração da *bridge*

A.3 *Software* de Medição de Tráfego

Existe uma infinidade de *software* destinados à medição de tráfego em uma rede, sendo a maioria deles igualmente capaz de registrar medidas para o parâmetro largura de banda. Não sendo esse um aspecto crítico para o desenvolvimento do estudo de caso, o autor optou pela ferramenta de medição de tráfego *ntop*[2], que possui como uma de suas características a capacidade de isolar fluxos específicos do tráfego para serem medidos separadamente dos demais.

[2]Disponível em http://www.ntop.org/.

Essa capacidade é decorrente da possibilidade de se utilizar filtros para a tarefa de captura dos pacotes na rede, o que permite definir o tipo de tráfego que irá compor os resultados apresentados pelo *software*. A versão do *ntop* utilizada nos testes foi a 3.3, instalada automaticamente com a execução do seguinte comando:

```
# aptitude install ntop
```

Os gráficos e resultados gerados pelo *ntop* são disponibilizados por meio de um servidor *web* interno, escutando por padrão na porta 3000 da máquina onde o mesmo está sendo executado. Além dos gráficos e estatísticas do tráfego, essas páginas também permitem o acessos às inúmeras configurações do *software*, como a inserção dinâmica dos filtros de captura de pacotes.

A.3.1 Medindo a Capacidade Total de *Download* da Rede

A capacidade máxima de download da rede foi medida fora de um horário de expediente, para que o atendimento ao público não fosse prejudicado. A estratégia consistiu em disparar várias instâncias de *download* e abertura de páginas *web*, de maneira a sobrecarregar a rede durante todo o processo de medição.

O equipamento de medição foi instalado entre o *firewall* e o *switch*, no ponto anteriormente definido como "A" e o endereçamento da rede interna assumido como 192.168.0.0/24. Para que apenas o tráfego de *download* fosse isolado no processo de medição adotou-se o seguinte filtro de captura de pacotes no *ntop*:

```
(not src net 192.168.0.0/24) and (dst net 192.168.0.0/24)
```

A.3.2 Medindo a Capacidade Total de *Upload* da Rede

Uma vez que alguns dos sistemas utilizados no estudo de caso são do tipo "interativo", onde a velocidade no envio de dados também afeta o desempenho das aplicações, a capacidade total de transmissão da rede para o tráfego de *upload* também foi aferida. Como na estratégia anterior, a medição foi realizada do horário de expediente, sendo a rede sobrecarregada com o envio de vários arquivos e o seguinte filtro de captura de pacotes empregado no *ntop*:

```
(src net 192.168.0.0/24) and (not dst net 192.168.0.0/24)
```

A.3.3 Medindo a Demanda de Banda dos Sistemas Utilizados

Com relação à demanda de largura de banda dos sistemas locais que acessam o *mainframe* e o servidor de banco de dados, ambos localizados na rede *intranet* da Prodemge, ela foi aferida durante a execução dos mesmos em condições consideradas ideais, num ambiente com capacidade ociosa de transmissão de dados.

Para que tais condições fossem possíveis, os testes com cada um desses sistemas foram feitos fora do horário de expediente, sem nenhuma outra máquina cliente ligada na rede. Além disso, foram propositalmente realizadas as tarefas que normalmente enfrentavam maior lentidão, quando executadas no horário normal de expediente.

Outra estratégia a empregada foi a adoção de um *software* gerador de macros para gravar uma das sessões de utilização dos sistemas e repeti-la posteriormente uma grande quantidade de vezes, sem a intervenção do usuário, quando então foi feita a medição da banda consumida. Uma vez que as máquinas cliente da rede utilizam o sistema operacional *Microsoft Windows XP*[3], o *software* adotado para essa tarefa foi o *AutoHotkey*[4], o qual, além de livre, é também gratuito.

Durante a execução desses testes o equipamento de medição foi instalado entre o *switch* e a máquina cliente geradora do tráfego, no ponto anteriormente definido como "B". Os filtros de captura de pacotes do *ntop* adotados para isolar o tráfego de *download* e de *upload* nos testes foram, respectivamente:

```
(not src net 192.168.0.0/24) and (dst host IP) and (src port PORTA)
(src host IP) and (not dst net 192.168.0.0/24) and (dst port PORTA)
```

A variável "IP" foi substituída pelo endereço de rede da máquina cliente utilizada nos testes, enquanto que "PORTA" representa a porta do servidor remoto que identificava o tipo de serviço testado. A demanda dos sistemas testados foram assumidas como os valores de pico registrados por uma hora nos gráficos do *ntop*.

[3] http://www.microsoft.com/windows/windows-xp/default.aspx.
[4] Disponível em http://www.autohotkey.com/.