

# QoS para VoIP através do HTB

**Israel Dias de J. Júnior**

Graduando em Ciência da Computação pela Universidade Tiradentes (UNIT)  
Co-fundador e membro da comunidade Arch Linux Brasil

israeljunior@gmail.com, zael@archlinux-br.org

## Introdução

Quando o kernel do Linux tem que enviar para a rede vários pacotes através de um dispositivo de rede qualquer ele precisa decidir quais enviará primeiro, quais retardará e quais descartará. Utilizando diversos algoritmos o escalonador de pacotes do kernel procura executar esta tarefa da maneira mais equilibrada possível. A regra padrão para o escalonador de pacotes é a FIFO (First In, First Out) - o primeiro a chegar é o primeiro a ser enviado. Você pode alterar este comportamento padrão fazendo com que o escalonador envie pacotes da maneira que lhe for mais útil [7].

## Quality of Service (QoS)

Há algum tempo atrás não havia uma preocupação excessiva com a maneira que os pacotes eram transmitidos. Pouco importava se eles chegavam ao seu destino. Atualmente, com o surgimento de aplicações como voz sobre IP e videoconferência, tais preocupações existem e são necessárias, pois elas necessitam de uma garantia de que os pacotes serão transferidos de forma correta aos seus destinatários.

Quality of Service (em português, Qualidade de Serviço), como já diz o nome, refere-se à garantia de um serviço de qualidade. É um requisito da(s) aplicação(ões) para a(s) qual(is) exige-se que determinados parâmetros (atrasos, vazão, perdas, ...) estejam dentro de limites bem definidos (valor mínimo, valor máximo) [1]. O termo é aplicado com frequência às aplicações de telecomunicações e redes de computadores, recebendo uma definição específica para cada. Nas telecomunicações, Qualidade de Serviço significa oferecer uma boa qualidade de voz, reduzir os ecos, etc. Nas aplicações de informática, Qualidade de Serviço significa garantir uma transmissão eficiente de pacotes através da implantação de sistemas de prioridade ou controle de banda.

Alguns dos problemas que podem acontecer na transmissão de pacotes são:

- Atraso ou latência. É a medida do tempo decorrido entre o início de uma atividade e a sua conclusão. Os atrasos podem acontecer quando pacotes tomam caminhos alternativos para evitar congestionamentos ou quando enfrentam filas de espera. Pode-se, por exemplo, dizer que o tempo de latência de um satélite VSAT é de 300ms, o que significa que um caractere enviado a partir de um ponto leva 300ms a chegar a outro, passando pelo satélite.

- Jitter. É a variação na latência com que os pacotes atingem o seu destino final. Sua causa é o processamento com tempo variável nos equipamentos de rede [1].
- Perda de pacotes. Pode acontecer de um, vários ou todos os pacotes serem descartados quando há muito congestionamento na rede ou por algum problema físico de cabeamento ou hardware. O uso de VoIP não é tolerante a perda de pacotes: perdas mínimas como 1%, por exemplo, podem degradar uma chamada, provocando erros audíveis nesta.

Existem duas arquiteturas padronizadas pelo IEEE para implantação de QoS em uma rede IP: IntServ (Integrated Services ou Serviços Integrados) e DiffServ (Differentiated Services ou Serviços Diferenciados). Seus padrões são definidos no RFC 1633 e RFC 2475, respectivamente.

## **Integrated Services ( IntServ – Serviços Integrados)**

O IntServ é caracterizado pela alocação de recursos para dois novos tipos de serviços, que são os serviços garantidos para aplicações que necessitam de um atraso constante e serviços de carga controlada para aplicações que requerem segurança e destacam o serviço de melhor esforço [2].

Uma aplicação qualquer, ao solicitar a reserva de recursos na rede, deve, primeiramente, enviar as especificações desejadas - como a QoS específica (níveis de latência, jitter, perda de pacotes, etc) e o tipo de tráfego que passará pela rede [5]. Esta solicitação passa por todos os roteadores da rede, que analisam as especificações e determinam se têm recursos suficientes para atendê-las (sem que outras sessões de QoS já aceitas sejam prejudicadas). Todo esse processo de sinalização é feito através do Resource Reservation Protocol (RSVP). Além disso, um escalonador de pacotes, classificador e uma rotina de controle de admissão devem ser implementados para garantir QoS via IntServ.

Se todos os roteadores da rede tiverem recursos suficientes para garantir a sessão esta é aceita e o tráfego da aplicação passa a fluir por rotas especiais definidas no momento da aceitação.

Segundo [3], com o crescimento excessivo da Internet, a utilização de arquiteturas IntServ para oferecer Qualidade de Serviço foi descartada. Em uma rede pequena os roteadores garantem que todas as requisições sejam atendidas. Mas e na Internet? Vamos considerar algo menor: um Provedor de serviços. Será que os roteadores suportariam a quantidade de requisições? Obviamente, não. Assim sendo, esta arquitetura não é recomendada para redes nas quais o tráfego e a quantidade de requisições são grandes.

## **Differentiated Services (DiffServ – Serviços Diferenciados)**

O DiffServ é uma alternativa ao IntServ, porém bem mais simples. Ele possui uma escalabilidade maior, já que os fluxos com requisitos de QoS similares são agregados em uma mesma classe - que pode receber um serviço melhor ou pior que outras definidas [6].

A arquitetura DiffServ também oferece uma maneira mais simples de categorização. Nela, os pacotes não precisam percorrer todos os roteadores de uma rede para que a concessão de

serviços seja garantida: as funções mais complexas de marcação, classificação e condicionamento de tráfego (policiamento e moldagem) ocorrem nos roteadores de borda, enquanto os roteadores do núcleo preocupam-se apenas em como dar o tratamento de expedição às classes de tráfego. A este tratamento é dado o nome de per-hop-behavior, ou PHB [6].

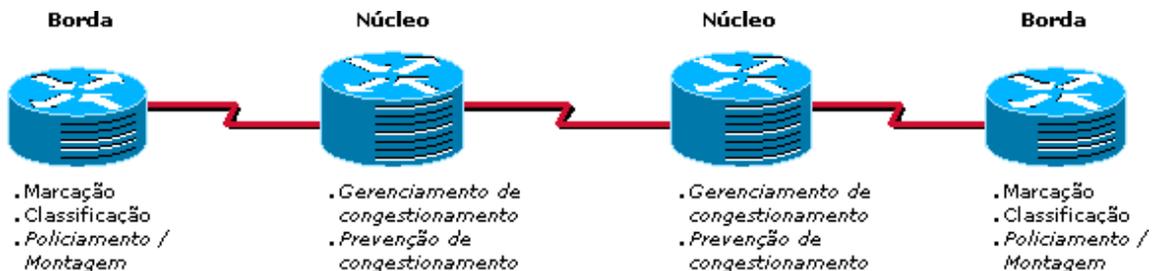


FIGURA 1. ESQUEMA DOS ROTEADORES DE BORDA E NÚCLEO.

A classificação dos pacotes é baseada em um campo do cabeçalho IP denominado DS-field, que pode ser considerado como uma redefinição do campo Type of Service, presente no cabeçalho IPv4, ou do campo Class of Traffic, do IPv6. São utilizados seis bits do DS-field para identificar o valor do DSCP (differentiated services codepoint), usado na seleção do PHB. Os dois últimos bits (CU - currently unused), como o nome já diz, não são utilizados atualmente e, conseqüentemente, são desconsiderados pela arquitetura DiffServ [6].

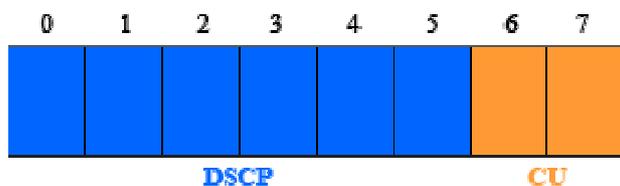


FIGURA 2. REPRESENTAÇÃO DO DS-FIELD.

## Definições

### Queues (Filas de Espera)

Um queue ou fila de espera é uma lista de itens que estão esperando por uma ação. É um conceito semelhante à uma fila de um banco: a pessoa que está na primeira posição da fila aguarda por um caixa vazio para que seja atendida; quando esta pessoa se direciona a um caixa a próxima fica no aguardo. Analogamente, podemos trazer este conceito para a área de redes de computadores. Sendo assim, uma fila de espera contém vários pacotes que aguardam por uma ação (seja ela de redirecionamento, de descarte, de atraso, etc.). O modelo mais simples de fila de espera existente em redes é o FIFO.

Uma fila de espera pode, também, usar sub-filas, que permitem uma complexidade maior em uma operação de agendamento [8].

### Flows (Fluxos)

Um fluxo acontece quando um (ou vários) pacote(s) atravessa(m) um caminho entre a origem e o destino. Um exemplo de fluxo pode ser uma conexão entre dois computadores. Nela acontece um fluxo de pacotes, seja através do protocolo TCP, UDP ou outro qualquer.

## **Tokens e Buckets (Baldes)**

### **Componentes**

#### **Queing Discipline (qdisc)**

O qdisc é um agendador de pacotes. Em português, queing discipline significa disciplinador de enfileiramento. Ou seja, um agendador de pacotes rearranja a ordem dos pacotes para que estes obedeçam às regras de enfileiramento especificadas.

Segundo [8], o qdisc no Linux é o bloco principal onde o controle de tráfego é efetuado.

Existem dois tipos de qdisc: classless e classful. Um qdisc classless, como o nome já diz, não é dotado de classes. Conseqüentemente, não pode oferecer filtros específicos. Um exemplo de qdisc classless é o FIFO. Já um qdisc classful é dotado de classes e oferece suporte a manipuladores aos quais podem ser anexados filtros. Exemplos de qdiscs classful são o CBQ e o próprio HTB.

#### **Classes**

As classes só existem em qdiscs classful e são formas de oferecer tratamento diferenciado a diferentes tipos de tráfego. Dentro das classes podem existir uma ou várias subclasses e até mesmo qdiscs.

#### **Filtros**

Segundo [9], é um dos componentes necessários ao controle de tráfego - e o mais complexo. Um filtro exerce a função de filtrar os pacotes pertencentes a uma classe ou qdisc e pode exercer a função de policiador, executando uma ação baseada em um critério específico.

## **Hierarchic Token Buckets (HTB)**

### **Introdução**

Hierarchic Token Buckets (HTB) é um agendador de pacotes para o Linux e uma fácil e eficaz alternativa ao CBQ. Um agendador, simplesmente falando, organiza/reorganiza pacotes para um tráfego de saída, através da simulação de conexões lógicas em uma conexão física.

O HTB está incluso no kernel do Linux desde a versão 2.4.20 e a sua utilização pode requerer uma recompilação - caso não esteja ativado por padrão. No nosso caso, adotamos a distribuição Slackware Linux 10.2 – instalada no modo FULL – e ela já vem com o HTB compilado como módulo no kernel padrão (2.4.31).

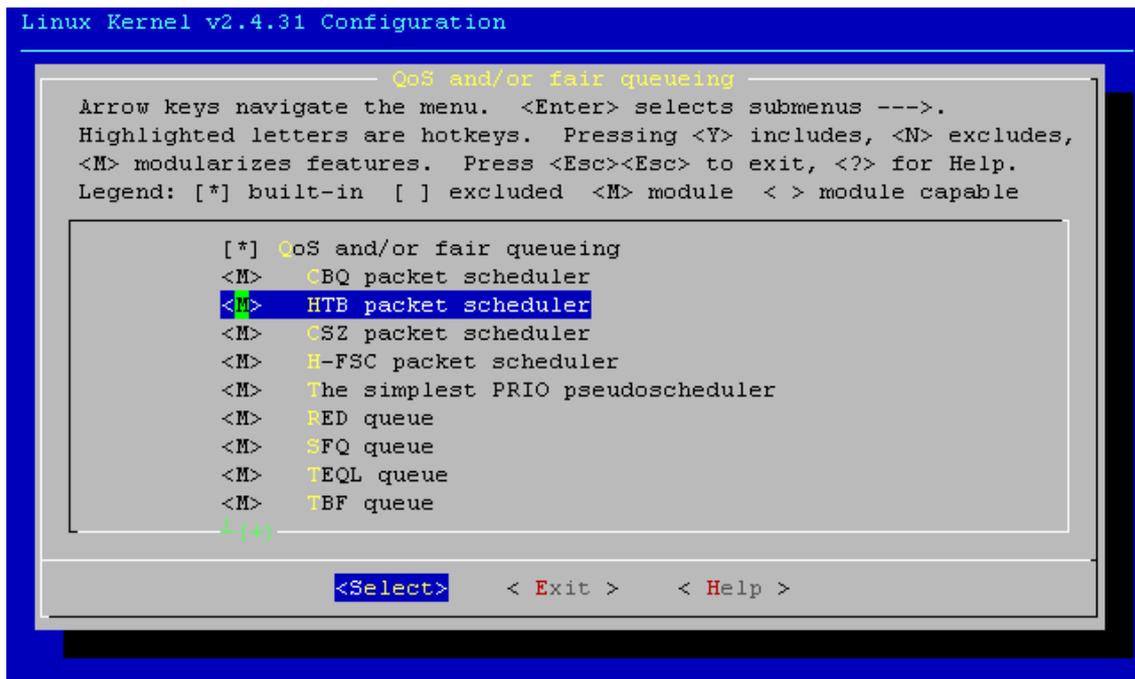


FIGURA 3. MAKE MENUCONFIG.

## Carregando o módulo no Kernel

O módulo se chama `sch_htb` e pode ser carregado através do comando:

```
# modprobe sch_htb
```

Porém, ao reiniciar o sistema o módulo não será carregado automaticamente. Uma das soluções é adicionar a linha acima ao arquivo `/etc/rc.d/rc.local` ou então ao arquivo `/etc/rc.d/rc.modules` - caso ela já exista neste último basta descomentar. Observe que apenas adicionar a linha a um dos arquivos não irá carregar o módulo no presente momento, ainda sendo necessário, desta vez, carregá-lo através do `modprobe`.

Para listar todos os módulos carregados no kernel utilize o comando:

```
# lsmod
```

A saída do comando será parecida com esta:

```
root@ldg001:~# lsmod
Module                Size  Used by    Not tainted
sch_htb               20064  0  (unused)
pcnet32               17028  1
mi                    2272   0  [pcnet32]
crc32                 2880   0  [pcnet32]
ide-scsi              9392   0
agpgart               45508  0  (unused)
root@ldg001:~# █
```

FIGURA 4. LSMOD.

Note que o módulo `sch_htb` está carregado nesta máquina.

## Instalando e configurando ferramentas

Além do suporte ao HTB ativado, precisamos que a ferramenta `tc` esteja instalada no sistema. Ela faz parte do pacote `iproute2` e já vem instalada no Slackware 10.2, então não precisaremos instalar nenhum pacote adicional. Sua utilidade é dar instruções ao kernel de como tratar o tráfego da rede.

Para verificar se o `tc` está instalado utilize o comando:

```
# whereis tc
```

A saída obtida deve ser semelhante a esta:

```
root@ldg001:~# whereis tc
tc: /sbin/tc /usr/lib/tc /usr/man/man8/tc.8.gz /usr/share/man/man8/tc.8.gz
```

Caso a saída seja vazia, provavelmente o `tc` não está instalado. Assim será necessário baixar o arquivo `htb3.6-020525.tgz`, relativo ao HTB3 - e mais recente. Utilizando o `wget` para baixar:

```
# wget http://luxik.cdi.cz/~devik/qos/htb/v3/htb3.6-020525.tgz
```

Em seguida crie um diretório, extraia o conteúdo do arquivo para ele e copie o binário `tc` para `/sbin`:

```
# mkdir htb
# tar -zxf htb3.6-020525.tgz -C ./htb
# cp ./htb/tc /sbin/tc
```

# Definindo um cenário-exemplo e regras para o controle de tráfego

## Cenário-exemplo

Uma empresa de médio porte qualquer tem implantado na sua Intranet servidores que garantem a sua funcionalidade. Ou seja, são os servidores de maior prioridade:

- Asterisk, para fornecer um serviço de PABX interno aos funcionários;
- Apache, para exibir as páginas da sua Intranet.

Internamente a empresa possui uma rede de 10Mbit/s e precisa garantir que o serviço de VoIP tenha uma Qualidade de Serviço necessária para que os funcionários possam se comunicar com qualidade. Além disso, deve garantir uma prioridade para que o tráfego do Apache tenha uma hierarquia inferior ao Asterisk mas inferior aos demais.

Em números, vamos admitir que o tráfego restante seja mínimo e não exceda 1,5Mbit/s. Assim, vamos garantir 3,5Mbit/s para o tráfego do Apache e o restante (5Mbit/s) para o Asterisk.

## Regras para o controle do tráfego

Primeiramente, devemos remover qualquer qdisc associado à interface que será utilizada. No nosso caso, eth0. O `2> /dev/null > /dev/null` serve para enviar a saída do comando (se existir) para o dispositivo `/dev/null`, que descarta tudo que lhe é enviado:

```
tc qdisc del dev eth0 root 2> /dev/null > /dev/null
```

Agora devemos associar uma qdisc HTB à interface. Também vamos associar a ela o manipulador 1:. Segundo [7], ele será referenciado por comandos para ajudar a criar a estrutura hierárquica. E o default 30 indica que todo tráfego que não estiver associado a uma classe específica será feito pela classe 1:50. Segundo [9], os manipuladores são escritos no formato `x:y`, onde `x` é um inteiro que identifica o qdisc e `y` identifica a classe pertencente a ele. Para manipular um qdisc qualquer, o valor da classe deve ser zero ou vazio - o valor 1: é semelhante a 1:0.

```
tc qdisc add dev eth0 root handle 1: htb default 30
```

O comando abaixo define a banda da classe pai da interface. Ou seja, a banda total disponível, 10Mbit/s.

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 10mbit
```

Agora vamos definir três classes: 1:10, 1:20 e 1:30. A primeira especifica a banda mínima de 5Mbit/s e banda total de 10Mbit/s. A segunda garante uma banda mínima de 3.5Mbit/s e máxima de 10Mbit/s. Por fim, a terceira garante 1.5Mbit/s de banda mínima e 10Mbit/s de banda máxima.

```
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 5mbit ceil 10mbit  
tc class add dev eth0 parent 1:1 classid 1:20 htb rate 3.5mbit ceil 10mbit
```

```
tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1.5mbit ceil 10mbit
```

Explicando: a banda mínima é o que uma aplicação terá de garantia quando estiver utilizando a interface controlada pelo qdisc definido. Nenhuma classe pode interferir na banda garantida de outra classe caso esta esteja em uso. Porém, classes filhas poderão atingir o limite definido em ceil se, e somente se, nenhuma outra classe estiver utilizando o restante da banda.

Agora temos que criar manipuladores sfq 10: a 30: sob as classes 1:10 a 1:30. O qdisc sfq (Stochastic Fairness Queuing) é um algoritmo proposto por John Nagle em 1987 que, segundo [7], objetiva fornecer uma distribuição justa entre os diversos tráfegos de uma mesma classe, fazendo com que as várias conexões dividam a banda de forma equilibrada entre si. Este qdisc evita que uma conexão de uma classe roube recursos de outras na mesma classe. Já o parâmetro perturb 10 define o tempo em segundos onde será verificada a solicitação de uso de banda por conexão. O manipulador 30: receberá tratamento padrão (FIFO) através do qdisc pfifo.

```
tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10  
tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10  
tc qdisc add dev eth0 parent 1:30 handle 30: pfifo
```

Por fim, vamos definir filtros para que o tráfego utilize as classes acima definidas. O parâmetro protocol define o protocolo que a regra se aplica e prio 1 define a prioridade do tráfego e varia de 0 a 15, sendo 0 o de maior prioridade e 15 o de menor. O filtro u32 é aplicado para identificar endereços ip e/ou portas tcp/udp [7].

```
tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \  
  match ip dport 5060 0xffff flowid 1:10  
tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \  
  match ip dport 80 0xffff flowid 1:20
```

O primeiro filtro define que o tráfego na porta de destino 5060 receberá prioridade 1 e será enquadrada na classe 10 da qdisc 1. O segundo define que o tráfego na porta de destino 80 receberá prioridade 1 e será enquadrada na classe 20 da qdisc 1. Os outros tipos de tráfego serão enquadrados na classe 30 da qdisc 1.

Assim o tráfego será filtrado conforme as especificações da porta.

Note que quando definimos os filtros apareceu o valor hexadecimal 0xffff. Isto significa que estamos aplicando uma máscara para definir um intervalo de portas.

```
5060 (dec) 0001001111000100  
0xffff (hex) 1111111111111111 (máscara)  
----- AND (E lógico)  
0001001111000100 (5060)
```

Assim, dport 5060 0xffff deverá cobrir apenas a porta 5060.

Abrangendo o conceito acima, podemos definir um filtro que abrange um intervalo de portas. Como exemplo, vamos definir um filtro-exemplo para o intervalo 8192-16383. Aplicando a máscara hexadecimal 0xe000 à porta 10000 obtemos o seguinte (os valores binários foram separados com espaços para melhorar a visualização):

```
10000 (dec) 001 0011100010000
0xe000 (hex) 111 0000000000000
```

Observe que o valor 0xe000 convertido para binário contém uma seqüência de zeros. Estes zeros que fazem parte da máscara, ao ser realizada a operação AND, definirão o início (valor obtido após a operação AND) e o fim (valor obtido através da inversão dos bits – e somente estes – que sofreram um AND com os “0” da máscara) do intervalo.

```
10000 (dec) 001 0011100010000
0xe000 (hex) 111 0000000000000
                ----- AND
                001 0000000000000 (8192) – após a operação binária
                001 1111111111111 (16383) – apenas os bits separados são invertidos
```

As portas que definem o início e o fim do intervalo são obtidas através da parte variável dos valores binários, uma vez que  $10000000000000_2$  é igual a  $8192_{10}$  e  $1111111111111_2$  é igual a  $16383_{10}$ . Assim, a linha a seguir define um filtro para o intervalo de portas 8192-16383 associando o seu tráfego à uma classe qualquer.

```
tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
match ip dport 10000 0xe000 flowid 1:666
```

## Exemplo de um script rc para Slackware

Crie um arquivo chamado rc.htb no diretório /etc/rc.d:

```
# cd /etc/rc.d
# touch rc.htb
```

Copie e cole o código fonte a seguir no arquivo (você pode utilizar o editor de sua preferência para efetuar este procedimento):

```
#!/bin/bash

INTERFACE=eth0
BANDA_LIMITE=10mbit

function iniciar()
{
    tc qdisc add dev $INTERFACE root handle 1: htb default 30

    tc class add dev $INTERFACE parent 1: classid 1:1 htb \
rate $BANDA_LIMITE
```

```

tc class add dev $INTERFACE parent 1:1 classid 1:10 htb \
    rate 5mbit ceil $BANDA_LIMITE
tc class add dev $INTERFACE parent 1:1 classid 1:20 htb \
    rate 3.5mbit ceil $BANDA_LIMITE
tc class add dev $INTERFACE parent 1:1 classid 1:30 htb \
    rate 1.5mbit ceil $BANDA_LIMITE

tc qdisc add dev $INTERFACE parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev $INTERFACE parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev $INTERFACE parent 1:30 handle 30: pfifo

tc filter add dev $INTERFACE parent 1:0 protocol ip prio 1 u32 \
    match ip dport 5060 0xffff flowid 1:10
tc filter add dev $INTERFACE parent 1:0 protocol ip prio 1 u32 \
    match ip dport 80 0xffff flowid 1:20
}

function parar()
{
    tc qdisc del dev $INTERFACE root 2> /dev/null > /dev/null
}

function reiniciar()
{
    parar
    iniciar
}

case "$1" in
    'start')
        iniciar
        ;;
    'restart')
        reiniciar
        ;;
    'stop')
        parar
        ;;
    *)
        echo "usage: start|restart|stop"
esac

```

Em seguida, dê permissão de execução ao arquivo:

```
# chmod +x rc.htb
```

O script será executado na inicialização do sistema. Para iniciá-lo, utilize o comando:

```
# ./rc.htb start
```

Você pode, ainda, utilizar os parâmetros restart (para reiniciar) e stop (para parar o script).

## Referências

- [1] MARTINS, Joberto. Qualidade de Serviço em Redes IP. 2003. (URL: <http://www.jsmnet.com>).
- [2] SANTOS, Ana Paula Silva dos. Qualidade de Serviço na Internet. NewsGeneration, v. 3 nº 6, 12 de novembro de 1999. (URL: <http://www.rnp.br/newsgen/9911/qos.shtml>. Acessado em 08 de março de 2006).
- [3] Quality of Service. Wikipedia, The Free Encyclopedia, 13 de março de 2006. (URL: [http://en.wikipedia.org/w/index.php?title=Quality\\_of\\_service&oldid=43632285](http://en.wikipedia.org/w/index.php?title=Quality_of_service&oldid=43632285). Acessado em 14 de março de 2006).
- [4] DiffServ. Wikipedia, The Free Encyclopedia, 25 de fevereiro de 2006. (URL: <http://pt.wikipedia.org/w/index.php?title=DiffServ&oldid=1529273>. Acessado em 14 de março de 2006).
- [5] KUROSE, James F.; ROSS, Keith W.. Redes de computadores e a internet: uma abordagem top-down. 3. ed. São Paulo: Pearson Addison Wesley, 2005. 634 p.
- [6] OYAMA, Cybelle Suemi Oda; LUCENA, Sidney Cunha de. Considerações acerca do estabelecimento de QoS no RNP2. NewsGeneration, v. 6, nº 3, 21 de maio de 2002. (URL: [http://www.rnp.br/newsgen/0205/qos\\_rnp.html](http://www.rnp.br/newsgen/0205/qos_rnp.html). Acessado em 21 de março de 2006).
- [7] LESSA, Carlos Virgílio Beltrão. Controle de Banda com HTB. Artigos br-linux.org, 13 de janeiro de 2004. (URL: <http://br-linux.org/tutoriais/001648.html>. Acessado em 31 de março de 2006).
- [8] BROWN, Martin A.. Traffic Control HOWTO. The Linux Documentation Project, novembro de 2003. (URL: <http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/>. Acessado em 05 de abril de 2006).
- [9] DEVERA, Martin.. User guide. HTB manual, 05 de maio de 2002. (URL: <http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm>. Acessado em: 06 de abril de 2006).

## Licença

Este trabalho está licenciado sob uma Licença Creative Commons Atribuição-Use Não-Comercial-Compartilhamento pela mesma Licença 2.5 Brasil. Para ver uma cópia desta licença, visite <http://creativecommons.org/licenses/by-nc-sa/2.5/br/> ou envie uma carta para Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.