UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SOFTWARE DE CONTROLE DE BANDA PARA *GATEWAYS* LINUX

JAISON DALLABONA

BLUMENAU
2010

2010/2-14

JAISON DALLABONA

SOFTWARE DE CONTROLE DE BANDA PARA *GATEWAYS* LINUX

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Francisco Adell Péricas – Orientador

BLUMENAU
2010

2010/2-14

SOFTWARE DE CONTROLE DE BANDA PARA *GATEWAYS* LINUX

Por

JAISON DALLABONA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Périca, Mestre – Orientador, FURB
Membro:	Prof. Paulo Fernando da Silva, Mestre – FURB
Membro:	Prof. Sérgio Stringari, Mestre – FURB

Blumenau, 30 de novembro de 2010.

Dedico este trabalho a minha família, amigos e todos que me ajudaram ao longo deste desafio.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que esteve sempre ao meu lado.

Aos meus amigos, que contribuíram de alguma forma para conclusão deste trabalho.

Ao meu orientador, Francisco Adell Péricas pelo apoio durante a jornada da execução deste trabalho.

Se pudermos atravessar a barreira da dor, poderemos ser campeões, se não esqueça. É isto que falta a grande maioria, garra para dizer que vamos chegar até o fim passe o que passar.

Arnold Schwarzenegger

RESUMO

Este trabalho apresenta o desenvolvimento de um *software* para controle de *Quality of Services* (QoS) de redes de computadores, partindo do princípio que não exista nenhuma técnica de QoS na rede. O propósito deste *software* é melhorar a qualidade do tráfego da rede fazendo com que não haja necessidade de contratação de *links* maiores onde muitas vezes o problema não é resolvido. O propósito deste *software*, também é fazer com que o administrador da rede consiga ter controle de largura de banda, classificação do tráfego da rede, podendo priorizar o que for mais importante dentro da rede. O controle é feito tanto para *downloads* como *uploads*. Os protocolos utilizados no controle serão *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP) e *Internet Control Message Protocol* (ICMP).

Palavras-chave: Qualidade de Serviço. Controle de tráfego. Priorização de pacotes.

ABSTRACT

This paper presents the development of *software* to control the QoS of network computers, assuming that there is no technique of network QoS. The purpose of this *software* is to improve the quality of the network traffic so that there is no need to hire more *links* where often the problem is not resolved. The purpose of this *software* is to make the network administrator can have control, classification of network traffic can prioritize what is most important in the control network.The *downloads* as is done for *uploads*. The protocols will be used to control TCP, UDP and ICMP

Key-words: Quality of Service. Traffic control. Packet prioritization

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CBQ – *Class Based Queueing*

DS – *Differentiated Services*

GW – *gateway*

HTB – *Hierarchical Token Bucket*

HTML – *Hyper Text Markup Language*

ICMP – *Internet Control Message Protocol*

KBPS – *Kilobits per second*

MP3 – *Moving Pictures Expert Group Layer 3*

P2P – *Peer to Peer*

PHP – *Hypertext Preprocessor*

QoS – *Quality of Services*

RED – *Random Early Detection*

RF – requisitos funcionais

RNF – requisitos não funcionais

TBF – *Token Bucket Filter*

TC – *Traffic Control*

TCP – *Transmission Control Protocol*

ToS – *Type of Services*

UDP – *User Datagram Protocol*

SUMÁRIO

INTRODUÇÃO

Com o avanço das redes de comunicação, o número de usuários tem se multiplicado a cada dia, o mesmo ocorrendo com aplicativos. Junto a este crescimento vem a necessidade de transmissões mais rápidas e confiáveis, que garantam a satisfação destes serviços, que são cada vez mais exigentes (STACHLEWSKI, 2008).

As redes de computadores, tanto as públicas quanto as privadas, estão crescendo muito e cada vez se integrando mais, convergindo para uma enorme infra-estrutura global. Em consequência a este crescimento, a possibilidade de gerenciar sistematicamente uma grande quantidade de sistemas de hardware e de *software*, componentes constituintes destas redes, tem tido cada vez mais importância (PÉRICAS, 2003, p. 121).

Conforme Stato (2009), as redes corporativas trouxeram a necessidade de se fazer um controle de tráfego dos dados. Novos serviços e tecnologias demandam um controle mais robusto e maior qualidade de banda dependendo do serviço requerido. Estes serviços trafegam muitas vezes necessitando ter um tratamento diferenciado em relação a outros, para que funcionem perfeitamente. A solução proposta para este problema é a utilização de QoS. QoS é uma técnica que está diretamente relacionada ao tráfego de dados, em outras palavras, controla o tráfego de uma rede definindo prioridades e limites de forma a melhorar o uso de determinados serviços, bem como utilizá-los de maneira mais eficiente. QoS permite garantir um número determinado de *bytes*, reserva de largura de banda, prioridade, dentre outros serviços que trafegam pela rede. Funciona como uma reserva de *bytes*, de forma que quando requerido tenha disponibilidade suficiente e torne-se funcional aos serviços solicitados pelo usuário.

Conforme Jucá (2005), as aplicações em tempo real requerem garantias rigorosas para que cheguem a seu destino. Aplicações que não sejam em tempo real poderão ter um tempo maior de resposta, um atraso nos pacotes enviados, sem percepção significativa ao usuário. Estas configurações envolvem QoS para garantir que a aplicação em tempo real seja prioritária diante de outros serviços.

O controle de QoS pode ser feito em um *gateway* Linux que é um sistema de código aberto multi-usuário. Se o destino do pacote não tiver origem a rede interna, é o *gateway* quem faz o encaminhamento dos pacotes para estes alcançarem seus destinos. Qualquer pacote que requisitar algum endereço fora da rede interna, obrigatoriamente passará pelo

gateway, caso contrário o pacote não chegará ao destino. Portanto o responsável pelas rotas de saída da rede é o *gateway*. O *software* desenvolvido neste trabalho permitirá todo e qualquer pacote que passar pelo *gateway* será tratado para que possa aplicar regras de QoS, determinando qual será sua prioridade na rede e sua respectiva largura de banda. A priorização dos serviços disponíveis poderá ser configurada pelo administrador da rede, através de uma interface web, conforme demanda exigida.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um *software* que permite definir um controle de banda, QoS, via web para um *gateway* Linux, permitindo priorizar serviços essenciais dentro de uma rede. O *software* receberá os dados que o administrador da rede informou através da interface web, onde a mesma chamará um *script* que fará a aplicação das regras propriamente ditas no sistema operacional.

Os objetivos específicos do trabalho são:

a) monitorar regras de QoS cadastradas remotamente via web;
b) aplicar regras de QoS remotamente via web;
c) controlar a banda consumida dentre serviços solicitados na rede;
d) priorizar serviços em todo ambiente da rede que tenha como saída o *gateway* Linux;
e) minimizar a largura de banda para serviços menos importantes da rede;
f) verificar *hosts online* na rede;
g) verificar a quantidade de *bytes* consumidos em cada regra de QoS aplicada.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos. No primeiro capítulo tem-se a introdução, a justificativa do trabalho, o objetivo geral e objetivos específicos. No segundo capítulo é exposta a fundamentação teórica, onde são abordados temas relevantes como

protocolos de rede, algoritmos de QoS, ferramentas de rede como Iptables, Iproute, *Traffic Control* (TC), infra-estrutura de rede antes e após aplicação de regras de QoS. Além também de trabalhos correlatos. No terceiro capítulo é abordado o desenvolvimento do trabalho. Para um melhor entendimento esta seção foi divida em requisitos principais, especificação, implementação, técnicas, ferramentas utilizadas e resultados. O quarto capítulo expõem a conclusão final e os resultados obtidos com e sem o uso das técnicas apresentadas neste trabalho.

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados alguns assuntos relevantes sobre o tema deste trabalho. Será apresentado um estudo sobre conceito de QoS, infra-estrutura da rede para aplicação e funcionamento do QoS em *gateways* Linux. Ao final serão expostas as técnicas adotadas para efetuar QoS no ambiente em que o administrador de rede desejar.

CONTROLE DE BANDA DE UMA REDE

Com o crescimento de aplicações exigindo cada vez mais recursos de rede tornou-se necessário, além de técnicas de redução de congestionamento, o fornecimento por parte das redes de parâmetros de desempenho com uma determina qualidade que permitirá assim uma comunicação mais confiável, ou ainda a inviabilidade de sua utilização.

Este tipo de controle se faz necessário porque o *TCP/IP*[1] por si só, não fornece um controle direto sobre a taxa de transmissão. A princípio, cada *host*[2] transmite pacotes o mais rápido possível de acordo com a capacidade dos dispositivos e o meio de comunicação disponível. Este comportamento pode ser problemático quando outros *script* também utilizam e compartilham o mesmo meio físico, competindo entre si a largura de banda disponível. Desta forma, um único usuário pode comprometer o desempenho dos demais, caso esteja ocupando uma grande parcela da capacidade do *link* durante a comunicação (MOSCHETO, 2006).

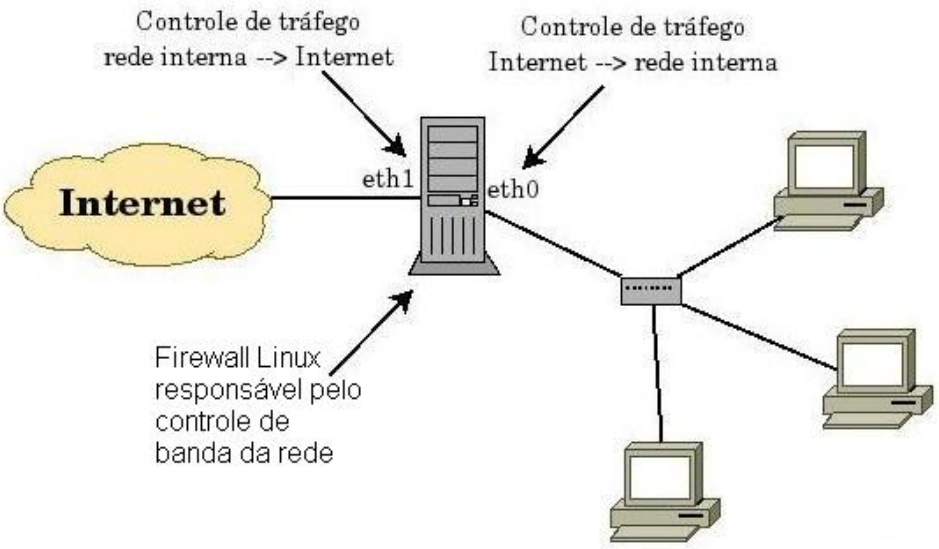
Conforme Jucá (2005), através de um roteador *firewall*[3] Linux , o tráfego de *download* e *upload* dos computadores da rede local podem ser controlados definindo-se um *qdisc* por dispositivo de rede. A Figura 1 mostra a topologia de rede e controle feito pelo *firewall gateway* Linux .

---

[1] TCP/IP é um conjunto de protocolos de comunicação entre computadores em rede. Seu nome vem de dois protocolos: o TCP *Transmission Control Protocol* - Protocolo de Controle de Transmissão e o IP *Internet Protocol* - Protocolo de Interconexão.

[2] *Host* são computadores ou equipamentos integrados a uma rede.

[3] *Firewall* é um dispositivo de segurança de rede, permitindo ou negando as transmissões de uma rede a outra.



Fonte: adaptado de Stato (2009).

Figura 1 – Esquema de rede com *gateway* Linux sendo aplicado em uma rede

O QoS busca suprir as expectativas de atender os usuário em termos de resposta e qualidade, muitas vezes prejudicada, por não existirem filtros de controle. A qualidade do tráfego de rede depende do tipo de aplicação que se deseja usar. Existem aplicações que podem sofrer atraso sem a percepção do usuário e outras que um mínimo de atraso é visível e poderá torná-la insustentável dentro do ambiente (JUCÁ, 2005).

QoS está diretamente relacionado ao tráfego, em outras palavras, ao controlar o tráfego da rede definindo prioridades e limites de forma a melhorar o uso de determinados serviços, bem como utilizá-los de maneira mais eficiente. Com o QoS é possível organizar a rede para que determinados serviços tenham precedência dentre os demais e desta forma, combater o mau uso da banda para fins que não sejam os da empresa. Quando implanta-se um sistema de QoS, tem-se em mente a diminuição no atraso e na perda de pacotes entre outros. Esses itens são requisitos para uma análise da necessidade de implantação do QoS (STATO, 2009, p. 187).

A lógica é equivalente a de dois rios que deságuam em um mesmo ponto, disputando a passagem por um mesmo canal. O que acontece se o volume de água de um dos rios for muito maior que o outro? O rio de maior volume (largura de banda) trafegará pelo canal em velocidade maior e o rio de menor volume terá dificuldade, cada vez maior, para trafegar. Ou seja, a distribuição de água pelo canal se tornará injusta. Dependendo da diferença de volume entre os rios, o rio de menor volume poderá deixar de trafegar pelo canal dominado pelo rio de maior volume. Esta situação pode ser observada com freqüência maior em redes

onde o acesso de *Internet* é compartilhado sem a utilização de filtros, pois um único usuário pode facilmente comprometer o acesso dos demais, caso ocupe boa parte da largura de banda ao realizar *downloads* de músicas mp3[4], vídeos e fotos, por exemplo. Um provedor de acesso também não deve bloquear o uso de ferramentas *P2P*[5] (e alguns clientes farão esta solicitação), mas precisará evitar este tipo de situação (JUCÁ, 2005, p.234).

Muitos pensam que a Qualidade do Serviço não é necessária e que é suficiente utilizar uma largura de banda maior para se obter um adequado QoS em todas as aplicações. Sustentam que a implementação do QoS é muito complicado e o aumento da largura de banda é mais simples. Embora isto seja parcialmente certo, pois há uma certa dificuldade de entendimento de algumas técnicas de QoS existentes hoje. Primeiro será necessário identificar os problemas a serem resolvidos para depois avaliar se eles poderão ser superados aumentando a largura de banda, que em muitos casos não resultará na solução do problema e sim fornecimento de mais munição para os serviços que por ventura estejam sendo usados indevidamente na rede. Se a largura de banda de todas as conexões de rede fossem infinitas, elas nunca congestionariam e não seria necessário aplicar técnicas de QoS. De fato, partes de certas redes possuem largura de banda bem razoável e foram cuidadosamente desenvolvidas para tornar mínimos os congestionamentos. Um exemplo claro é o VOIP onde em uma ligação não passará o consumo de 250kbps/s em geral. Ao se ter um *link* com 1 Mbps/s poderia-se fazer 4 ligações simultâneas sem nenhuma interferências ou perda nas chamadas. Mas neste caso há que se garantir que haja esta largura de banda. Porém, as conexões de grande largura de banda não estão disponíveis em toda a rede, desde a fonte até o destino do tráfico. As variações em larguras de banda podem chegar a ser potenciais pontos de congestionamento, gerando assim uma Qualidade do Serviço imprevisível. Os fornecedores, para atenderem as explosivas necessidades da Internet, têm incorporado larguras de banda a suas redes IP de forma indiscriminada. Algumas delas oferecem conexões de baixa latência em redes nacionais ou continentais. O tráfego deve ser tratado de forma adequada para atingir um determinado nível de QoS. Se o ponto ao qual se incorpora a rede de acesso se converte num ponto de congestionamento, a Qualidade do Serviço de ponta a ponta pode chegar a ser deficiente, embora o rendimento da rede geral seja excelente. Em casos de congestionamentos, lentidão de determinados serviços na rede o primeiro passo a ser feito é a verificação de quais serviços

---

[4] Mp3 é um formato utilizado para músicas, uma abreviação de *MPEG1 Layer-3* e foi criado pelo Instituto Fraunhofer, que detém a patente e cobra para o seu uso desde 1998.

[5] P2P ou *Peer-to-Peer*, é uma arquitetura de sistemas distribuídos caracterizada pela descentralização das funções na rede, onde cada nodo realiza tanto funções de servidor quanto de cliente. Exemplos de *softwares* p2p emule, kazza, shareaza, entre outro.

estão fazendo um consumo indevido da largura de banda e classificar estes serviços menos importantes com uma largura de banda menor e priorizando os serviços mais importantes, fornecendo maior largura de banda e priorização dos pacotes (FERREO, 2002).

REDES SEM CONTROLE DE BANDA

Todo tráfego que for requisitado em uma determinada rede sem tratamento de QoS, será respondido o mais rápido possível conforme largura de banda na rede. Imagine-se uma rede que possua um *link* de 2Mb/s, e 50 computadores, exemplos atuais como prédios, pequenas empresas, *lan houses*. Estas redes poderão utilizar qualquer tipo de programa de acesso à *Internet*.

Em uma rede liberal, programas como o Emule ou Kazza, que são *softwares* de compartilhamento que utilizam paradigma P2P para transferência de dados, poderão causar um enorme congestionamento. Um único Emule instalado em um computador da rede poderá abrir dezenas de conexões simultâneas. Com isso, caso algum outro usuário, em outro computador, deseje visitar algum *site*, por exemplo, haverá uma máquina estabelecendo uma conexão, lutando contra outro computador com 60 conexões como no exemplo. Haverá, em outras palavras, 1 contra 60. Com isso haverá uma grande desvantagem para um dos dois. Isso ocorre porque, da forma como o Kernel do Linux trabalha, os pacotes são enfileirados em um *buffer*[6] e saem na mesma ordem em que chegam. Aumentar o *link*, ao contrário do que se pensa, não resolveria o caso, pois só se está dando mais poder de fogo ao usuário do Emule e se continuaria sendo o 60º da fila (MOTTA, 2007).

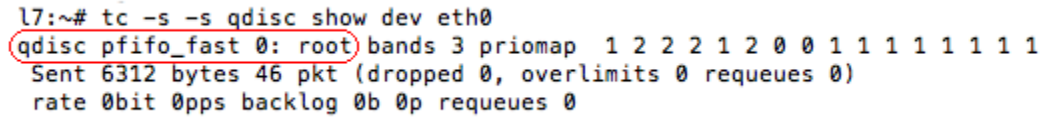
Bloquear o Emule torna e força o usuário de tal programa a achar uma forma de burlar o controle.

É esse tipo de problema que acontece em muitas redes, principalmente no ramo empresarial que necessita de uma demanda de banda enorme e muitas vezes usuários utilizando *softwares* que consigam ter êxito na conexão, poderá causar transtornos na rede inteira.

O algoritmo de enfileiramento padrão adotado pelo Kernel do Linux é conhecido como *pfifo_fast*. O próprio nome quase revela a forma como o algoritmo gerencia a fila, o primeiro pacote que chega é o primeiro a sair (JUCÁ, 2005).

---

[6] *Buffer* é uma região de memória temporária utilizada para escrita e leitura de dados.



Fonte: adaptado de Jucá (2005).

Figura 2 – Demonstração da utilização do algoritmo *pfifo_fast* no Linux

Com tudo isso se não houver a utilização de algoritmos e ferramentas que farão e auxiliarão a criação de regras de QoS esta é impossibilitada de acontecer pois o Kernel do Linux utiliza o algoritmo de enfileiramento *pfifo_fast.*

UTILIZAÇÃO DO ALGORITMO HTB PARA REGRAS DE QOS

Segundo Stato (2009, p. 247), o algoritmo HTB veio com o intuito de substituir o *Class Based Queueing* (CBQ), por ser mais compreensível e rápido. Ele foi criado por Martin Devera e incluído no Kernel a partir do 2.4.20 onde implementa uma fila com suporte a várias classes para controle de tráfego. O HTB cria *links* virtuais onde existe a possibilidade de filtrar quanto a prioridade dos pacotes que ali trafegarão, ou seja, serviços que necessitem de uma taxa com um mínimo de largura possível serão tratadas com este mínimo que o administrador configurou fazendo uso das ferramentas e do algoritmo HTB.

O algoritmo HTB garante que o total de serviço provido para cada classe é no mínimo o total requerido por esta e conforme o que foi designado para cada classe. Quando uma classe requisita menos que o total a ela reservado, a largura de banda remanescente é distribuída para outras classes que requisitem serviços, caso isto seja uma configuração em que o administrador opte fazer (DEVERA, 2002).

Conforme Jucá (2005, p. 257), todos os recursos analisados no CBQ estão presentes no HTB, tanto a disposição hierárquica em forma de árvore, quanto a garantia ou o compartilhamento de banda. A principal diferença é precisão, pois não depende de cálculos de tempo de inatividade (não é dependente das características da interface de rede). Este qdisc é suportado nativamente pelo Kernel. Ao contrário do CBQ, o número de parâmetros necessários para adotar a disciplina de enfileiramento HTB é menor. As principais características, parâmetros e descrição da utilização do algoritmo HTB são:

a) *Default*: os pacotes que não forem classificados serão submetidos a um nó

configurado no campo padrão (*default*). Para definir o nó padrão, utiliza-se o parâmetro *default*. Todo o tráfego que não tiver classificado em algum momento obrigatoriamente será encaminhado à classe *default*;

b) *Priority*: as classes com maior prioridade apresentam preferência de atendimento maior e são consultadas antes;

c) *Rate*: taxa de transmissão mínima que a classe poderá atingir. Mesmo que haja muito tráfego, obrigatoriamente a classe que estiver atribuída com um valor *rate* terá garantido no mínimo o valor *rate* atribuído;

d) *Ceil*: taxa de transmissão máxima que a classe poderá atingir (quando outras classes puderem emprestar banda). Para tomar emprestado de outras classes uma fatia de banda, é necessário à configuração *ceil* um valor superior a *rate*;

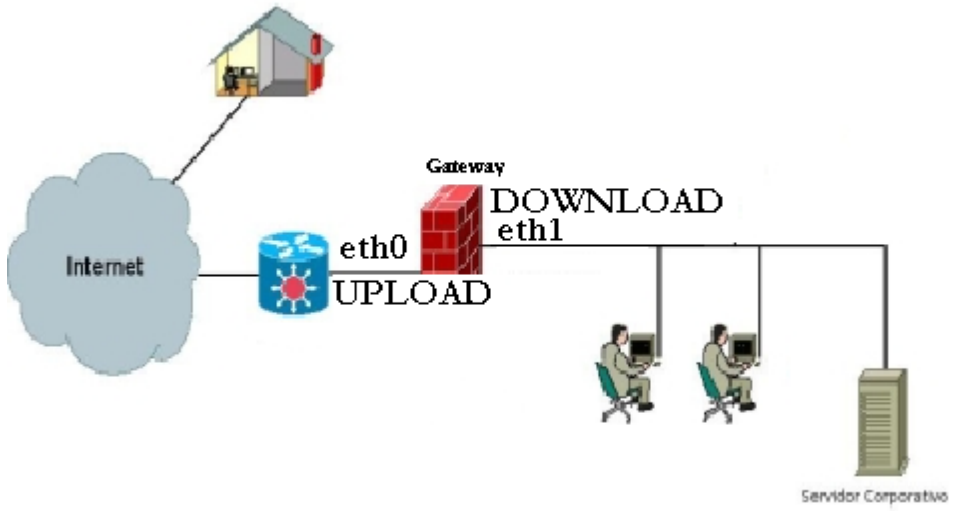
e) *Burst*: define quantos *bytes* que poderão ser transmitidos em rajadas, durante o consumo da taxa atribuída a *ceil* (ao exceder a *rate*). O *burst* de uma classe qualquer deve ser superior ao valor atribuído aos seus respectivos filhos. Do contrário, é possível que as requisições das classes filhas sejam superiores ao suportado pela classe em questão (pai);

f) *Cburst*: determina quantos *bytes* poderão ser enviados na capacidade máxima do dispositivo durante uma rajada (excedendo *ceil*). O valor ideal deve corresponder ao tamanho do pacote. Assim como o *burst*, o valor deste parâmetro deve ser superior ao atribuído aos respectivos filhos.

Conforme Lessa (2004), o Kernel do Linux oferece diversos recursos que permitem o total controle no envio e recebimento de pacotes. Este conjunto de recursos é denominado qdisc algo como regras de enfileiramento. Dentre estas regras existe uma denominada HTB. O HTB fornece meios para controlar o tráfego de entrada e saída de um *link*. Através dele pode-se simular vários *links* lentos em um único *link* físico e estabelecer quais tipos de tráfegos vão transitar em cada um destes *links* virtuais. O que precisa ser feito é definir a divisão em *links* e decidir como os pacotes trafegarão neles. Para a utilização do HTB, é necessário um Kernel com versão a partir da 2.4.20 em diante. A ferramenta TC é utilizada para manipular qdiscs HTB. A ferramenta TC faz parte do pacote IProute2 que já acompanham as distribuições Linux. Ela é utilizada para instruir o Kernel a tratar o tráfego de rede. É importante dizer que o HTB funciona com qualquer distribuição Linux que tenham as ferramentas instaladas para que exista QoS. Pode-se utilizar a distribuição em que se tenha maior afinidade, basta suprir os requisitos já citados.

Para que se entenda como e onde é feito o controle de *download* e *upload* no *gateway*

Linux a figura 3 mostra esta explanação.



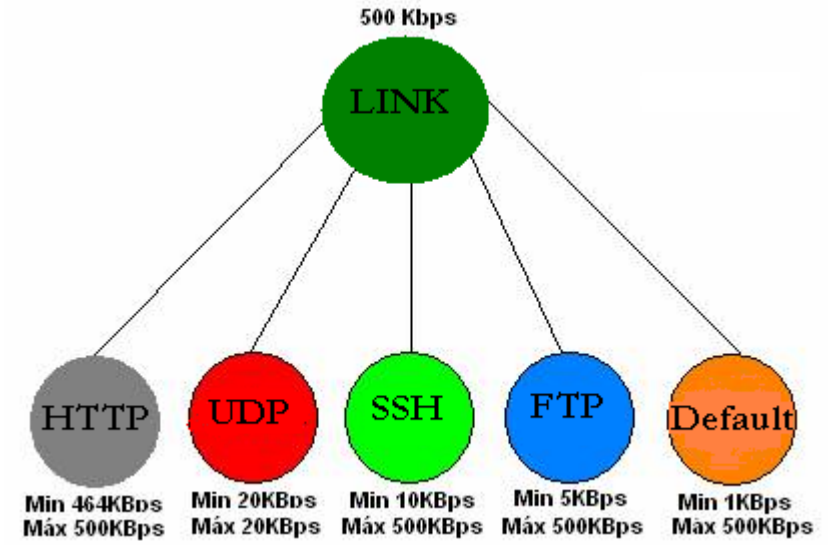
Fonte: adaptado de Stato (2009).

Figura 3 – Esquema de rede com *gateway* Linux sendo aplicado em uma rede

Conforme Jucá (2005, p. 235), o controle de *download* é feito *limit*ando a taxa de transmissão entre o *gateway* Linux e os *scripts* da rede local. Como a interface de rede que permite conectividade com a rede local é a *eth1*, conforme topologia de rede da figura 3, o *limit*e será imposto a esta interface e aplicado quando for necessário integrar pacotes aos *scripts* da rede local. Assim, é "garantido" que os pacotes encaminhados à rede local não trafeguem a uma taxa superior a estabelecida. Para controlar o tráfego de *upload* adotar-se-á um raciocínio equivalente ao tratado para *downloads*. A *limit*ação deverá ser imposta a interface que permite conectividade com a internet, ou seja, a interface de rede eth0 conforme topologia de rede da figura 3. O controle será aplicado quando for necessário entregar pacotes originados pela rede local e destinados à internet, no caso do *upload*. Logo, será preciso configurar um qdisc para a interface *eth0* e *limit*ar o tráfego de saída gerado por esta interface.

O método para confecção deste trabalho é o método chamado *Classful* qdisc, que poderá emprestar *link* de outros *links* criados nas regras de QoS caso os mesmos não estejam sendo utilizados ou estejam ociosos.

Segundo Stato (2009, p. 248), o exemplo seguinte mostra o caso em que a largura de banda esteja ociosa, que poderá ser emprestada para qualquer um dos serviços mostrados na figura 4. Este método de empréstimo entre serviços chama-se *Classfull* qdisc. Dentro da rede tem-se parte da banda determinado para serviços como: http, udp, ssh, ftp, voip entre outros.

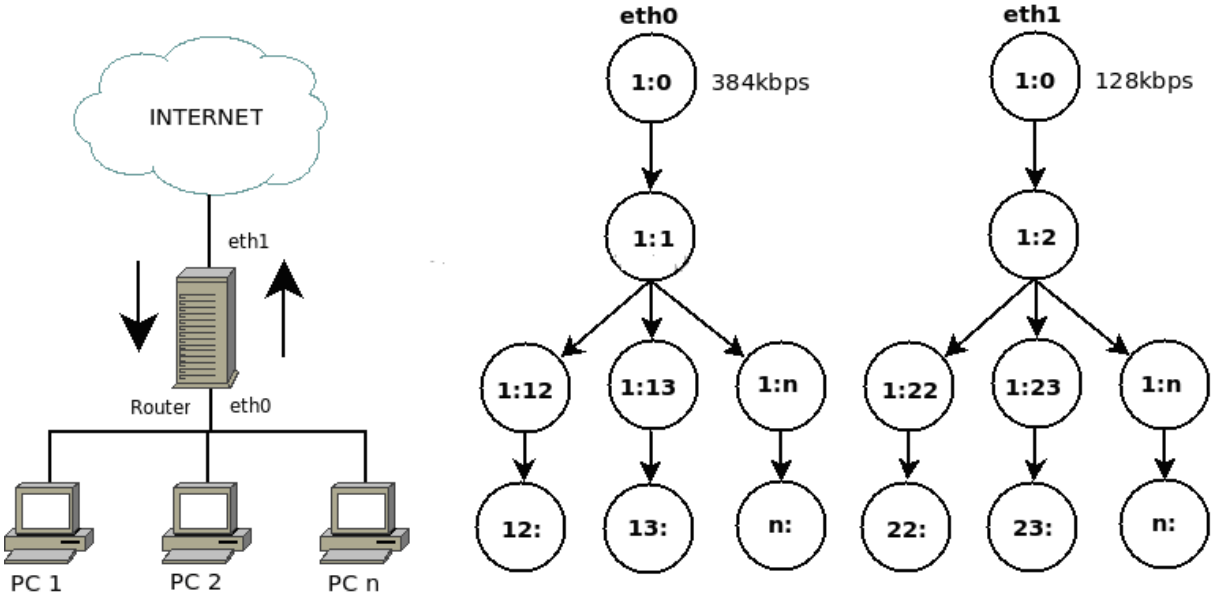


Fonte: Adaptado Stato (2009)

Figura 4 – Disponibilização de banda por serviços

O *link* terá 500Kbps ligado à internet através da interface eth1 do *gateway* Linux. Dentro do *link* proposto será garantida uma determinada largura da banda para cada serviço que será implantado através de filtros ligados diretamente a cada classe. No modelo da figura 4, cria-se uma *qdisc* principal que seria o *link* de 500Kbps, em seguida os *links* filhos com suas determinas larguras de banda conforme figura 4. Atenção para a classe *default*: esta é a classe em que não receberá tratamento, ou seja, todo *download* feito que não for, http, udp, ssh, ftp, será redimensionado para classe *default* que terá no mínimo 1Kbps garantido conforme figura 4 e no máximo 500Kbps caso nenhum outro serviço esteja sendo utilizado.

Segundo Jucá (2005, p. 242), a grande vantagem deste qdisc *Classfull* é a disposição de classes em forma de árvore (subdividas hierarquicamente). Este tipo de solução possibilita a implementação de diferentes larguras de banda, limitadas por classe ou nó filho, e garante uma largura de banda mínima não compartilhada ou compartilhada. Para explicar melhor, imagine-se como seria a subdivisão, em forma de árvore, dos serviços de internet solicitados por *script* de uma rede local. O primeiro nó da árvore representaria a largura de banda total que será dividida. Para aplicar uma divisão justa melhor utilização do *link*, fará-se-ia a divisão da largura de banda em três partes conforme figura 5.



Fonte: Motta (2007)

Figura 5 – Mostra esquema de prioridade do algoritmo HTB

A Identificação de cada nó é feita com a combinação de dois números, conhecidos como "*major*" e "*minor*" (*major:minor*). Os nós que representam um qdisc são identificados apenas pelo número "*major*" ou "*major:0*". Pode-se observar que "1:", "12:", "13:" , "22:" e "23:" são nós qdisc, os demais identificam as classes (1:1, 1:2, 1:12, 1:13, 1:22 , 1:23 e 1:n). O parentesco é definido administrativamente e pode ser visualizado através da ligação existente entre os nós (não necessariamente por nós de mesmo número "major"). Acompanhando a ligação estabelecida entre os nós, pode-se concluir que a classe "22:" é filho da classe "1:22" que é filho de "1:2".

Conforme Jucá (2005, p. 243), o primeiro nó da árvore corresponde ao "*root* qdisc" (raiz), que a partir de uma solicitação do Kernel, remove ou envia pacotes à fila (controlando assim a taxa de transmissão). A remoção da fila é feita a partir de consulta às classes filho. Os demais nós não possuem comunicação direta com o Kernel. A forma como a fila será manipulada depende do algoritmo adotado, como HTB, por exemplo. Através de filtros específicos é determinado à qual classe um pacote deve ser submetido. Ou seja, classificar o pacote. O termo classificar é empregado nos filtros que identificam à qual posição da árvore o pacote será submetido. O *root* qdisc realiza consultas constantes às classes filhas para obter o pacote que deve ser removido da fila. Portanto, para remover um pacote que foi submetido à classe 1:23 é necessário que o *root* qdisc consulte os nós filhos percorrendo a árvore hierarquicamente.

Em termos de configuração, a identificação do nó qdisc é feita pelo parâmetro "*handle*" e as classes por "*classid*" (ambos baseando-se no número "major:minor"). A ligação entre os nós é feita segundo figura 5, estabelecendo o parentesco entre os nós. Este parentesco é definido pelo parâmetro "*parent*", identificando o nó pai através do respectivo número "major:minor".

O qdisc principal do HTB controla a divisão de banda. Quando uma classe solicita banda, é disponibilizado um valor de banda mínimo que é especificado em *rate*. Caso a calasse não esteja utilizando sua banda, o qdisc-pai distribui pelas outras classes conforme solicitado. Até o máximo especificado em *ceil*.

Caso a configuração do HTB seja apenas manter um valor de banda que não seja ultrapassado por ninguém, basta colocar o valor de *ceil* e *rate* iguais, de forma que a classe no máximo chegará ao valor de *ceil*.

UTILIZAÇÃO DO IPTABLES EM CONJUNTO AO HTB

Segundo Motta (2007), o Iptables é um *firewall*, um programa com o objetivo de proteger máquinas contra acessos indesejados, tráfego indesejado, proteger serviços que estejam rodando na máquina e bloquear a passagem de determinas coisas que não se deseja receber (como conexões vindas da *Internet* para a rede local, evitando acesso aos dados corporativos de uma empresa ou a dados pessoais). A partir do Kernel do Linux 2.4, foi introduzido o *firewall* Iptables (também chamado de *netfilter*) que substitui o Ipchains dos Kernels da série 2.2 para baixo. Este novo *firewall* tem como vantagem ser muito estável, confiável, e permitir muita flexibilidade na programação de regras pelo administrador do sistema, mais opções disponíveis ao administrador para controle de tráfego e controle independente do tráfego da rede devido à nova organização das etapas de roteamento de pacotes.

O Iptables é um *firewall* em nível de pacotes e funciona baseado no endereço/porta de origem/destino do pacote e prioridade. Ele funciona através da comparação de regras para saber se um pacote tem ou não permissão para passar. Em *firewalls* mais restritivos, o pacote é bloqueado e registrado para que o administrador do sistema tenha conhecimento sobre o que está acontecendo no sistema.

Ele também pode ser usado para modificar e monitorar o tráfego da rede, fazer *Network Address Translation* (NAT), uma técnica que consiste em reescrever os endereços IP de origem de um pacote que passam por um *router* ou *firewall* de maneira que um computador de uma rede interna tenha acesso ao exterior, redirecionamento de pacotes, marcação de pacotes, modificar a prioridade de pacotes que chegam/saem do sistema, contagem de *bytes*, dividir tráfego entre máquinas, criar proteções e muito mais. O tráfego vindo de máquinas desconhecidas da rede pode também ser bloqueado/registrado através do uso de regras de acesso. As possibilidades oferecidas pelos recursos de filtragem Iptables como todas as ferramentas Linux maduras, dependem da imaginação, pois ele garante uma grande flexibilidade na manipulação das regras de acesso ao sistema, precisando apenas conhecer quais interfaces o sistema possui, o que deseja bloquear, o que tem acesso garantido, quais serviços devem estar acessíveis para cada rede, e iniciar a construção do *firewall*.

O Iptables ainda tem a vantagem de ser modularizável. Funções podem ser adicionadas ao *firewall* ampliando as possibilidades oferecidas.

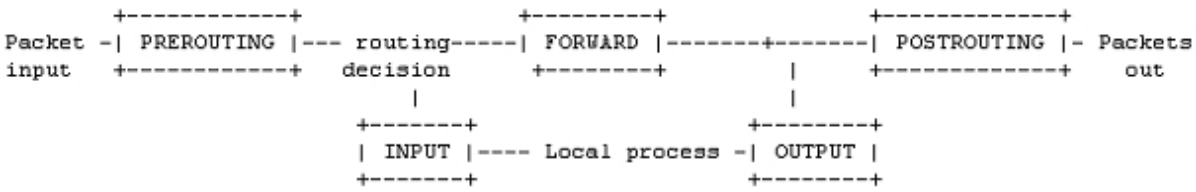
Um *firewall* não funciona de forma automática, é necessário pelo menos conhecimentos básicos de rede TCP, roteamento e portas para criar as regras que farão à segurança do sistema. A segurança do sistema depende do controle das regras que serão criadas, portanto as falhas humanas são garantia de mais de 95% de sucesso nas invasões.

Enfim o Iptables é um *firewall* que agradará tanto a pessoas que desejam uma segurança básica em seu sistema, quando administradores de grandes redes que querem ter um controle minucioso sobre o tráfego que passa entre as interfaces de rede (controlando tudo o que pode passar de uma rede a outra), controlar o uso de tráfego, monitoração, controle de banda através de marcação dos pacotes e muito mais.

Neste ponto viu-se o que é a ferramenta Iptables e qual sua devida importância dentro de uma rede. Com esta ferramenta, pode-se fazer a marcação dos pacotes, sabendo em qual classe, porta, determinado serviço está passando, operando pela rede, consequentemente pode-se classificar tal serviço para uma determinada classe e priorizá-lo conforme a necessidade, a demanda requerida. É através desta ferramenta que será feita toda marcação dos pacotes, ou seja, tudo que precisar ser classificado será marcado através da ferramenta Iptables em conjunto com outras ferramentas que no final fará o que se propôs: o controle de QoS na rede através do *gateway* Linux .

Segundo Carvalho (2006), classificando os tráfegos de interesse, precisa-se ter o entendimento sobre a máquina de estados do Iptables de forma apenas básica. Deve-se seguir

o modelo abaixo da figura 6.



Fonte: Carvalho (2006)

Figura 6 – Esquema da máquina Iptables

Iptables é utilizado para criar, manter e inspecionar as tabelas de filtro de pacotes ip no Kernel do Linux. Para que se possa manipular isso de forma eficiente, o Iptables criou uma serie de tabelas e cada uma contém *chains* (cadeias, rotinas) pré-definidas ou criadas pelo usuário que são executadas à medida que os pacotes chegam ao sistema operacional:

a) *Mangle* é muito utilizada para a manipulação de pacotes. Geralmente se usa para mudar (*mangle*) algum valor como *Type of Services* (ToS), as marcações FW, TTL entre outros. É aqui que se faz a marcação dos pacotes para um tratamento diferenciado de QoS no *proxy*/*gateway*. É altamente recomendado não realizar nenhum filtro nesta tabela, já que existe uma com essa finalidade;

b) *NAT* deve ser utilizada para as necessidades de tradução de endereços de rede. É nesta tabela que se colocam as regras para compartilhar uma conexão de internet com uma LAN, existe a possibilidade de alguém localizado na internet acesse um recurso interno da rede, como um servidor web e coisas desse gênero. O *NAT* possui três *chains* disponíveis que são *Prerouting, Postrouting e Output;*

c) *Filter* é uma tabela que deve ser usada para filtrar pacotes, ou seja, permitir ou restringir o acesso. É aqui que se verificar a direção de um pacote, o que ele contém e tomar uma ação com ele, seja aceitar ou rejeitar. O *filter* possui três *chain*s disponíveis que são I*nput, Forward e Output;*

d) *Prerouting* é a primeira a ser processada pelo Iptables em qualquer tabela. O *prerouting* Ela tem esse nome, pois é utilizada antes mesmo do Kernel do Linux tomar alguma decisão de roteamento baseado em seu endereço de origem/destino, marcas FW por exemplo na marcação de pacotes;

e) *INPUT* é utilizada somente para pacotes que possuem o endereço de destino o servidor local. Pacotes com outros destinos que não seja o servidor, não serão afetados por essa *chain*;

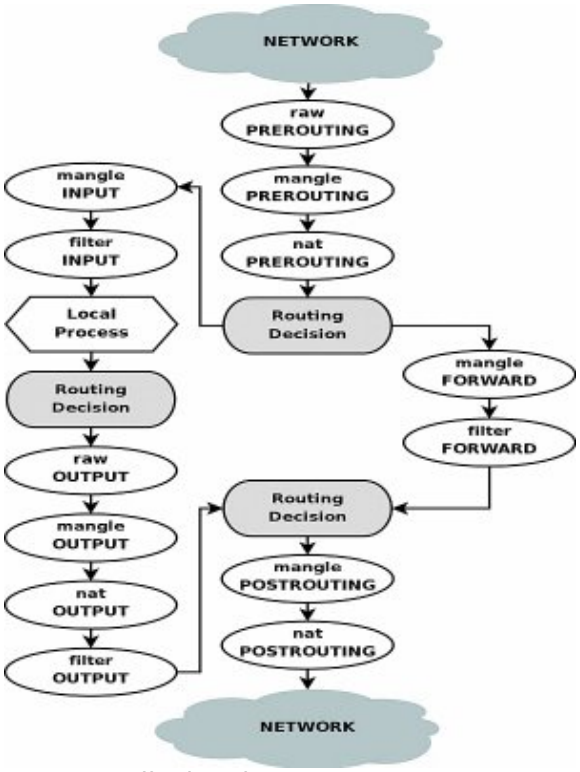
f) *FORWARD* pacotes que passam nesta *chain*, possuem endereço de destino que

não seja o computador/servidor. São pacotes destinados, por exemplo a uma LAN, para a *internet* ou alguma outra rede que o Linux tenha conectado e que este seja responsável por fazer roteamento. Aqui, pode-se identificar esses pacotes;

g) *OUTPUT* possui a função de inspecionar os pacotes gerados pelo computador/ servidor local. Se proibir que um computador/servidor local faça *ping* por exemplo aqui é o local, tabela *filter*;

h) *POSTROUTING* de forma análoga a *chain prerouting*, possui função exatamente oposta a ela. Os pacotes que passam por aqui já tiveram sua decisão de roteamento tomada. Aqui é o local onde se pode aplicar o *Masquerade, Snat, Dnat*.

Com o fluxograma da figura 7 abaixo, fica mais claro a explanação deste assunto:



Fonte: Carvalho (2006)

Figura 7 – Esquema da máquina Iptables

Em geral, cada uma destas *chains* é processada antes da *chain* correspondente na tabela *filter* e NAT para definir opções especiais para o tráfego (por exemplo, o *chain* PREROUTING da tabela *Mangle* processado antes do PREROUTING da tabela NAT). O *chain* OUTPUT da tabela *Mangle* corresponde ao OUTPUT da tabela NAT.

Opções como o ToS é especificado nesta tabela para classificar e aumentar consideravelmente à velocidade de tráfego considerados em tempo real. Mesmo após o tráfego ser estabelecido, as *chain*s da tabela *Mangle* continuam ativos para garantir que as

opções especiais relacionadas com a conexão continuem fazendo efeito.

Todo tráfego quando alcançar o *gateway* Linux, independentemente da interface utilizada, é tratado pela *chain* PREROUTING. De maneira análoga, todo o tráfego de saída é tratado pela *chain* POSTROUTING. As *chain*s de INPUT e OUTPUT são utilizadas apenas para tratamento de pacotes que possuam origem e destino o próprio *gateway*, e não o tráfego que sofrerá apenas roteamento e encaminhamento. Forçando valor de DSCP para protocolos críticos, como por exemplo *Internet Control Message Protocol* (ICMP).

```
iptables -t mangle -A PREROUTING -p icmp -j DSCP --set-dscp 46
iptables -t mangle -A PREROUTING -p icmp -j MARK --set-mark 0x1
iptables -t mangle -A PREROUTING -p icmp -j RETURN
```

Fonte: Motta (2007)

Figura 8 – Regras com marcação para protocolo ICMP

Pode-se observar a utilização da *chain* PREROUTING, como descrito anteriormente, onde ocorre a classificação do tráfego entrante por quaisquer interfaces, ou seja, em quaisquer direções (Rede Interna --> Internet ou Internet ---> Rede Interna).

Segundo Stato (2009), pode-se fazer o uso de diversos modelos de regras e classificadores com o Iptables. Pode-se utilizar a *tables Mangle* para marcar pacotes. É possível marcar os pacotes utilizando a *chain Classify* ou simplesmente MARK. *Chain* Classifiy permite que se configure um valor de prioridade, então será utilizado esta classificação de pacote dentro de uma classe do HTB. Exemplo de classificação Figura 9.

```
iptables -t mangle -A FORWARD -o eth0 -p udp -s --sport 53 -j CLASSIFY --set-class 1:11
```

Fonte: Motta (2007)

Figura 9 – Regras com marcação para serviço *Domain Name System* (DNS)

Neste caso está sendo classificado a porta 53, serviço de DNS, classificando-o na classe 1:11.

Outro exemplo seria a utilização do filtro MARK. Neste modelo trabalha-se com filtros que serão classificados e associados às regras e não diretamente como no exemplo da figura 8 que classifica a regra diretamente com o marcador *set-class*.

Na figura 10 tem-se o exemplo com o filtro MARK que é o número entre *handle* e FW.

```
tc filter add dev eth1 parent 1:0 protocol ip handle 110 fw classid 1:11
```

Fonte: Adaptado Stato (2009)

Figura 10 – Regras com marcação para protocolo POP3

A chave filtro está em *handle* e FW *classid* y:yy, pois o *handle* vai ser a marca que

ele procurará no pacote, enquanto FW será a classe para qual o pacote será encaminhado. No primeiro item tem-se *handle* 111, ou seja serão verificados pacotes que contenham marca 111 e depois FW classid 1:>11, tais pacotes serão encaminhados para a classe 1:11. Basta criar a regra de Iptables conforme figura 11.

```
iptables -t mangle -A FORWARD -o eth1 -s --sport 110 -j MARK --set-mark 111
```

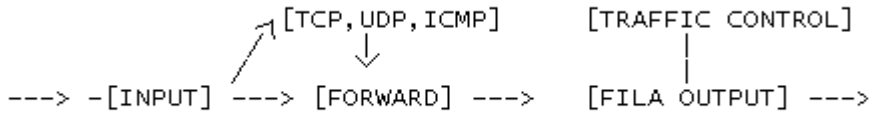
Fonte: Adaptado Stato (2009)

Figura 11 – Regras com marcação para protocolo POP3 utilizando *set-mark*

Nesta regra o Iptables está filtrando todos os pacotes que tem origem na rede local pelo serviço de POP3, que é o recebimento de *emails* através da porta 110. Já no *filter* do TC, os pacotes marcados com 111 serão encaminhados para 1:11. E assim para cada marca feita pelo Iptables, será encaminhada para classe correta.

A FERRAMENTA TRAFFIC CONTROL

O TC trabalha com uma fila chamada *queuing* que respeita a forma em que os pacotes são recebidos. Através do tratamento dado, pode ser alterada a velocidade de transmissão e o pacote terá maior prioridade; então esse pacote será o primeiro a sair da fila. Na figura 12 vê-se uma idéia de onde o TC atua:



Fonte: Adaptado Stato (2009)

Figura 12 – Atuação do TC

Segundo Stato (2009), os pacotes entram pela interface de entrada e são enviados ao *input* (como se fosse uma fila de pacotes de entrada). Em seguida eles são encaminhados para a fila local (TCP,UDP) ou encaminhados (*Forwarding*), e depois são tratados pelo TC. Mas como o Linux tratará isto? O TC possui quatro componentes principais: qdiscs*, Classes, filter e Policers.*

Os qdiscs são as filas ou algoritmos de enfileiramento. Eles são responsáveis pelo enfileiramento e saída de pacotes.

O Linux suporta vários tipos de filas de disciplina. Essas filas são divididas em duas categorias *Classless e Classfull*.

As *Classless* são responsáveis por classificar os pacotes, é a forma como vai-se controlar o tráfego, dividir as partes que serão usadas por determinados pacotes. A *Classless* é associada ao qdisc, pois para a manipulação dos pacotes, é preciso ter um qdisc e escolher que tipo de qdisc que se usará , e então se for o caso, criar classes para trabalhar com o qdisc.

Pode-se escolher trabalhar com uma qdisc específico como HTB já explicado em capítulos anteriores, e em seguida criar várias classes que tratarão pacotes com marcas, ou campos que estejam marcados com determinados números, como foi o caso da marcação dos pacotes com o auxilio da ferramenta Iptables, descrito no capítulo anterior.

Isto é válido se for escolhido um qdisc que possa alterar as classes, pois em algumas delas não é possível que as classes sejam definidas, o que é justamente a diferença entre *Classless e Classfull.*

As qdisc do tipo *Classless* não podem conter classes definidas por usuário, já as *Classfull* podem conter subclasses criadas ou definidas por usuário, podendo dar maior flexibilidade para tratamento de banda (STATO, 2009).

Este trabalho foi desenvolvido utilizando o modelo *Classfull*, justamente pelo fato de ter maior flexibilidade de configuração nas classes utilizadas. Exemplo de classes são FiFo, SFQ, TBF, DS_MARK. Um exemplo do *Classfull* é o HTB, inclusive este foi o algoritmo utilizado para o desenvolvimento destra trabalho.

Outro componente do TC é o *filter*. Ele é responsável por classificar o pacote e distribuí-lo dentro de suas classes atuando diretamente com um classificador.Um classificador é utilizado dentro do *filter* para identificar pacotes e fluxos e, posteriormente, os separando em suas respectivas classes. Existem vários classificadores que poderão ser usados para verificação de endereços de origem, destino, portas origem e destino, marcação de alguns campos entre outras coisas. Um dos mais conhecidos é o U32 que é muito utilizado devido a sua flexibilidade para verificar os campos dos pacotes. Existem outros como *FW, parent, prio, handle, TCindex, protocol e rsvp*.

Para manipular as qdisc, classes e filtros, usa-se o comando TC seguido dos itens que se quer manipular.

Então através do comando TC, pode-se adicionar, alterar, excluir e visualizar. Por exemplo, para adicionar um qdisc trabalhando diretamente com o HTB usa-se o comando da figura 13.

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

Fonte: Adaptado Stato (2009)

Figura 13 – exemplo de utilização de um qdisc-*root*

Neste exemplo, está sendo adicionado um qdisc (ou uma disciplina de enfileiramento) no dispositivo eth0. O *root* significa que qdisc é a raiz, ou seja a primeira. Imagine um processo que gere novos processos, e esses novos processos geram mais processos. Então o primeiro processo, é o processo-pai, e posteriormente terá processos-filhos. Assim será feito com a qdisc em relação às classes é justamente isto. Criou-se um processo inicial, posteriormente cria-se a classe-filha.

O *handle* nada mais é do que um identificador, um nome qualquer. A única obrigatoriedade aqui é que termine em zero quando for a raiz da qdisc. Por fim informa-se qual será o algoritmo que será utilizado com a qdisc, que neste caso é o HTB.

Consequentemente pode-se criar as classes-filha. O comando é parecido conforme figura 14.

```
tc class add dev eth0 parent 1:0 classeid 1:10 htb rate 256kbps
tc class add dev eth0 parent 1:0 classeid 1:20 htb rate 128kbps
```

Fonte: Adaptado Stato (2009)

Figura 14 – exemplo de qdisc classes-filhas

Criou-se duas classes-filha do primeiro qdisc o 1:0, referenciada pelo parâmetro *parent*, as classes são a 1:1 e 1:10, usando o algoritmo HTB e banda garantida de 128kbps e 256kbps.

Pode-se criar qdisc de classes, gerando uma qdisc-filha, conforme figura 15.

```
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
```

Fonte: Adaptado Stato (2009)

Figura 15 – exemplo de qdisc classes-filhas

Na figura 15 tem-se uma particularidade que é o uso do algoritmo SFQ (no próximo capítulo será dada uma visão geral sobre outros algoritmos para QoS e o SFQ será um deles), que na realidade é a especificação para uso do algoritmo SQF com uma opção chamada *perturb*.

Para verificar quais regras estão vigentes, pode-se utilizar o comando TC para o mesmo, conforme figura 16.

```
debianl :~# tc qdisc show dev eth0
qdisc htb 1: root r2q 10 default 0 direct_packets_stat 490
qdisc sfq 10: parent 1:10 limit 127p quantum 1514b perturb 10sec
qdisc sfq 20: parent 1:20 limit 127p quantum 1514b perturb 10sec
```

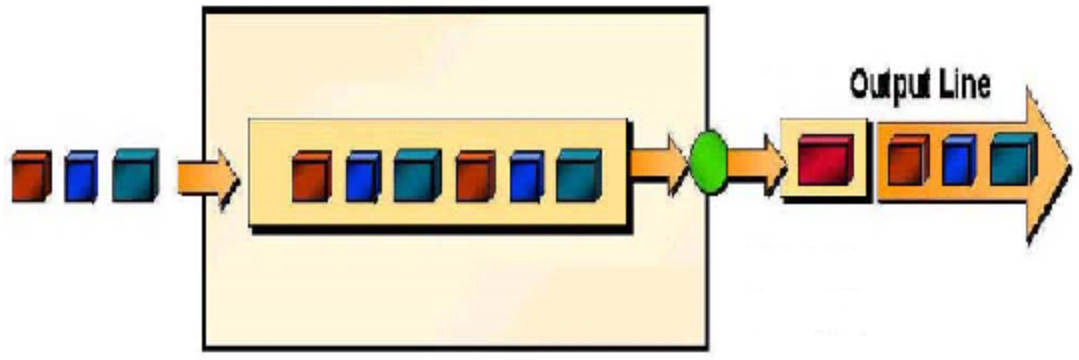
Fonte: Adaptado Stato (2009)

Figura 16 – visualizando regras vigentes com comando TC

VISÃO GERAL DE OUTROS ALGORITMOS PARA UTILIZAÇÃO DE QOS

Aqui será fornecido uma visão geral de outros algoritmos que poderiam ser utilizados para efetuar um QoS. Este trabalho foi desenvolvido apenas utilizando o algoritmo HTB já explicado nos capítulos anteriores. Aqui será dado uma visão geral de outros algoritmos que poderiam ser utilizados também para fornecer um QoS diferente ou em conjunto com o algoritmo HTB.

Segundo Stato (2009), conhecido como o esquema FiFo é o qdisc padrão utilizado no Linux, mas com alguns itens adicionados, como prioridade e também usado na maioria dos roteadores. O processo é bem simples: os pacotes que entram primeiro saem primeiro. O algoritmo FIFO não garante um tempo de resposta rápido, pois é extremamente sensível a ordem de chegada de cada processo e dos antecessores (se existirem) e se processos que tendem a demorar mais tempo chegarem primeiro o tempo médio de espera acabam sendo aumentado. A figura 17 mostra os diferentes tipos de tráfegos entrando e saindo na mesma ordem, ou seja, não há tratamento de entrada e saída, que é justamente isto que o algoritmo FiFo faz como descrito anteriormente.



Fonte: Adaptado Stato (2009)

Figura 17 – Diferentes tipos de tráfego

O algoritmo PRIO, segundo Stato (2009, p. 203), é uma classe que atua com

algoritmos que podem trabalhar com prioridade para serviços diferenciados. Neste tipo de algoritmo, pacotes são classificados primeiro pelo sistema, em seguida colocados em filas diferentes conforme sua prioridade. Quando é criada uma fila do tipo PRIO, automaticamente são criadas três filas !:1, 1:2 e 1:3. Com essa estrutura criada, pode-se criar filtros para serem aplicados diretamente nessas filas, ou se for o caso, criar qdisc-filha para atuarem com um algoritmo diferente.

Quando um pacote entra na fila, ele é enviado a um subqdisc ou um qdisc-filho. O encaminhamento poderá ser baseado de algumas formas. Em um filtro propriamente criado pelo administrador, que informa para qual fila será enviado determinado pacote, e através do PRIOMAP caso não seja especificado nenhum filtro o PRIO verificará a TC_PRIO prioridade e decidirá para qual fila que o pacotes será encaminhado.

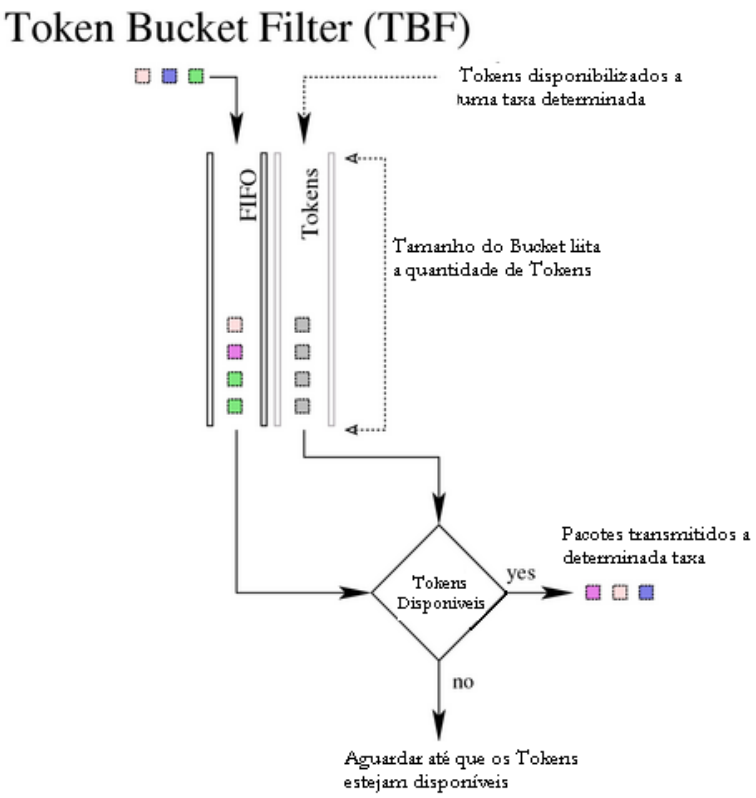
O campo ToS já citado neste trabalho, é quem fará a priorização. A figura 18 mostra os *bits* de definição de ToS:

Binário	Decimal	Tipo
1000	8	Minimize delay (md)
0100	4	Maximize throughput (mt)
0010	2	Maximize reliability(mr)
0001	1	Minimize monetary cost (mmc)
0000	0	Normal Service

Fonte: Adaptado Stato (2009)

Figura 18 – Campos de TOS

Segundo Stato (2009, p. 211), o algoritmo *Token Bucket Filter* (TBF) consegue controlar a taxa máxima de processamento de pacotes e seu funcionamento se dá através dos *Buckes e Tokens.* O *Bucket* seria como um buffer onde é colocado o *Token*. Como exemplo dá-se o *Bucket* com um balde que conforme os pacotes vão entrando na fila, ele também vai recebendo *Tokens*. O *Token* é um tipo de pacote que enche o balde, no caso o *Bucket*. Quando um *Bucket* enche-se de *Tokens*, os pacotes que estão na fila real são enviados para o seu destino a uma taxa de velocidade, ou até rajadas que podem ser definidas pelo administrador. Para facilitar o entendimento a figura 19 mostra o esquema dos *Bucket* e *Token*s.



Fonte: Adaptado Stato (2009)

Figura 19 – Funcionamento através dos *buckets e tokens*

A medida que os pacotes vão entrando na fila, os *Tokens* vão enchendo o *Bucket*. Tão logo encha o *Bucket* (que esta pré-configurado com uma velocidade), serão enviados os pacotes a determinada velocidade. Deve-se verificar o tamanho do buffer que é a quantidade de *Tokens* que se pode armazenar e que está relacionada diretamente com a quantidade de pacotes em *bytes*. Parâmetros que podem e deve ser adicionados a criação de uma qdisc do TBF:

a) *Limit ou latency: limit*e é o numero de *bytes* que podem ficar na fila aguardando os *Tokens* para se tornarem disponíveis. Também pode ser especificado através da latency, onde é informado o tempo máximo que um pacote poderá aguardar por um *Tokens*;

b) *Burst/buffer/maxbusrt:* tamanho do *bucket* em *bytes*. Esta é a quantidade máxima de *bytes* que poderá ter *Tokens* para envio imediato;

c) *MPU Minimum Packet Unit,* determina o tamanho de um pacote para uso de um *Token*, normalmente 64 *bytes*;

d) *Rate* taxa de transmissão média*;*

e) *Peakrate* se existirem *Tokens* disponíveis para pacotes que estão chegando, os pacotes podem ser enviados imediatamente. Algumas vezes pode não ser isso o desejável. O *peak rate* é usado para especificar o quão rápido o *Bucket* está autorizado a esvaziar-se. Recomendado para plataforma Intel é 100mbit/s;

f) *Mtu/minburst* especifica o tamanho do Peakrate do Bucket, o tamanho da MTU.

A utilização do algoritmo TBF, comum qdisc-filho para tratamento de um qdisc do tipo PRIO, onde será usado algum item relacionado ao próprio algoritmo. Figura 20 relata um exemplo da utilização do algoritmo TBF:

```
tc qdisc add dev eth0 root tbf rate 0.5mbit burst 5kb latency
70ms peakrate 1mbit minburst 1540
```

Fonte: Adaptado Stato (2009)

Figura 20 – Funcionamento através da utilização do algoritmo TBF

Primeiro é informado ao qdisc qual o algoritmo que foi usado no caso o TBF. É selecionada uma taxa máxima de transmissão de 0.5Mbps em *buffer* de 5k a uma taxa máxima que o *Bucket* segurará antes de se esvaziar de 10 Mbit. O tempo máximo que um pacote fica na fila é de 70 ms, e por último, o tamanho de MTA que é de 1540.

Para verificar a regra feita acima utiliza-se o comando conforme figura 21.

```
debian:~# tc qdisc ls dev eth0
qdisc tbf 8001: root rate 500000bit burst 5Kb peakrate
1000Kbit minburst 1539b lat 70.0ms
```

Fonte: Adaptado Stato (2009)

Figura 21 – Algoritmo TBF em funcionamento

Assim verifica-se as regras que estão na interface em que foram aplicadas as regras. Caso haja necessidade de excluir alguma regra, poderá ser feito o comando conforme figura 22.

```
tc qdisc del dev eth0 root tbf rate 0.5mbit burst 5kb latency 70ms peakrate
1mbit minburst 1540
```

Fonte: Adaptado Stato (2009)

Figura 22 – Exemplo de exclusão de uma regra utilizando do algoritmo TBF

Após excluído, pode-se ver as regras *default* que já foi mencionado neste trabalho em que o Kernel do Linux utiliza por default o algoritmo *pfifo_fast* vê-se após excluir a regra da figura 22 o que o comando da figura 23 retorna.

```
debian:~# tc qdisc ls dev eth0
qdisc pfifo_fast 0: root bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

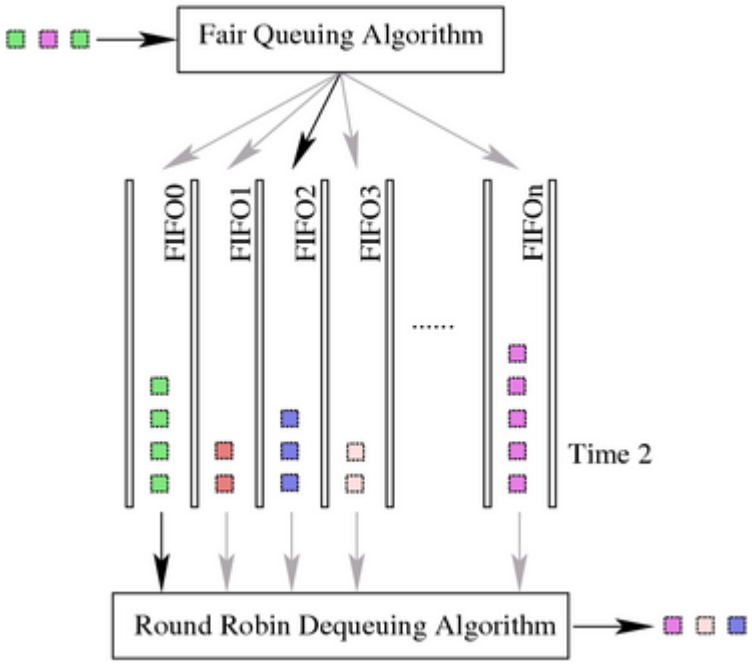
Fonte: Adaptado Stato (2009)

Figura 23 – Após exclusão Kernel do Linux volta utilizar algoritmo *default pfido_fast*

Segundo Stato (2009), o algoritmo *Stochastic Fair Queuing* (SFQ) é baseado no *Fair Queuing* proposto por John Nagle em 1987. Ele foi projetado para assegurar que cada fluxo tenha acesso à rede de modo justo, impedindo que certos tráfegos não consumissem mais do que sua quota de largura de banda. Em contra-partida, ele não possui distinção de pacotes e é um dos mais simples em relação a outros algoritmos da família *Fair Queuining*.

No FQ os pacotes chegam e são classificados em fluxos pelo sistema. Após isso, eles são organizados por um classificador e enviados a outro algoritmo conhecido com Round Robin.

Round Robin é um algoritmo largamente usado. A idéia principal desse algoritmo é ter uma pequena unidade de tempo denominada *timeslice* ou *quantum*, e ter todos os processos serem armazenados em filas.Um escalonador percorre a fila alocando CPU para cada processo durante um *quantum*, se o processo não terminar após um *quantum* ocorre uma preempção, e o processo é inserido no fim da fila, mas se o processo terminar antes de um *quantum*, a CPU é liberada para execução de novos processos. Os pacotes que chegam às filas são classificados segundo três itens: endereço de origem, porta de origem e endereço de destino. Após a classificação são enviados para a fila dedicada a esse fluxo, e então as filas são tratadas um pacote por vez por um algoritmo Round Robin, que mantém a justiça no uso da banda, já que tratará pacote a pacote em várias filas, conforme pode ser visto na Figura 24.



Fonte: Adaptado Stato (2009)

Figura 24 – Algoritmo SFQ

Como visto, os pacotes chegam e em seguida são tratados pelo algoritmo FQ, que são colocados em filas (FIFO1, FIFO2, FIFOn...) para posteriormente serem tratados novamente pelo Round Robin (algoritmo que se encarrega de enviar os pacotes de forma justa).

A classificação dos fluxos é feita através do endereço de origem, porta de origem e endereço de destino (conhecidos com tuplas em determinada fila).

Mas já dentro da fila, cada fluxo de pacote é associado a um identificador que permite que ele seja reclassficado dentro da fila em relação aos outros fluxos, permitindo uma melhor otimização da banda, ou seja, aqueles pacotes gigantes de *download* ou P2P não vão poder usar todo *limit*e de banda disponível, o que deixava outras aplicações sem recursos para uso de banda. Alguns parâmetros usados pelo SFQ:

a) *Perturb* reconfigura o *hash* após o número de segundos especificado. Se não estiver ativo, o *hash* nunca será reconfigurado, o que não é recomendado. Um valor comumente recomendado é 10 segundos;

b) *Quantum* é a quantidade em *bytes* removidos da fila para que a próxima fila entre no algoritmo. É a quantidade de *bytes* que é removida por vez. O valor padrão é 1 MTU.

Um exemplo da utilização do algoritmo SFQ é apresentado na Figura 25.

```
tc qdisc add dev eth0 root sfq perturb 10
```

Fonte: Adaptado Stato (2009)

Figura 25 – Comando de adição de regra utilizando o algoritmo SFQ

Em seguida o comando para visualizar a regra na figura 26.

```
tc -s -d qdisc show dev eth0
qdisc sfq 8003: root limit 127p quantum 1514b flows 127/1024 perturb 10sec
Sent 2258 bytes 13 pkt (dropped 0, overlimits 0 requeues 0)
rate 0bit 0pps backlog 0b 0p requeues 0
```

Fonte: Adaptado Stato (2009)

Figura 26 – Saída do comando da figura 25

Observa-se que o número 8003 é um número atribuído automaticamente ao *handle*, o *limit* 127p é o *limit* de 127 pacotes que podem esperar nesta fila, 127/1024 *hashbucket*s disponíveis para trabalhar com pacotes, e serão ativados 127 por vez e *perturb* 10 são os *handle*s que serão reconfigurados a cada 10 segundos.

Segundo Stato (2009), o algoritmo *Random Early Detection* (RED) tem função de controle de fila. A maioria dos algoritmos trabalha com uma fila. O processo é conhecido como FIFO e *DropTail.* O primeiro que entra é o primeiro que sai, e se a fila estiver cheia, os pacotes são descartados assim que acabam de chegar ao final da fila. Em alguns algoritmos, o primeiro da fila é descartado.

O RED foi criado com o objetivo de prevenir alguns fenômenos como o *lock-out* e o *starvation*, ambos relacionados aos recursos da banda que são tomados por pequenos fluxos fazendo com que a rede fique ocupada para os demais. O RED responde mais rapidamente a estes fluxos que ocasionam esse problema, e não espera que ocorra um congestionamento. Antes de ocorrer, o algoritmo já trata o descarte de pacotes de forma equilibrada entre todas as filas e fluxos. Através de seu algoritmo é implementado um controle de tamanho das filas, e elas são sempre comparadas com um mínimo/máximo passado através de comando. Quando a fila permanece na média de acordo com um mínimo/máximo, nenhum pacote é descartado, mas caso o tamanho seja superado, ocorre o descarte de pacotes que chegam baseados na probabilidade máxima indicada, fundamentado em uma função que considera o tamanho da fila.

Os parâmetros do RED são:

a) *Min* tamanho médio da fila, onde se inicia a probabilidade de descarte. O valor

deve ser calculado em torno de um terço do máximo, Max/3=min;

b) *Max* tamanho médio da fila com probabilidade máxima de descarte;

c) *Probability* probabilidade de descarte. Flutua entre 0.0 e 1.0. Os valores recomendados estão entre 0.01 e 0.02 que são 1 e 2% respectivamente;

d) *Limit* tamanho máximo da fila em *bytes*. Deve ser superior ao tamanho Max e burst e é recomendado 8 x Max;

e) *Avpkt* tamanho médio de pacotes em *bytes*. O valor recomendado é 1.000;

f) *Burst* usado para determinar a velocidade que o tamanho médio da fila é influenciado pelo tamanho real. Recomenda-se 2x mim +m /3 * avpkt;

g) *Bandwidth* tamanho médio da fila. Quando o *link* está ocioso, ou seja, quando não há tráfego, normalmente é configurado para o tamanho da banda na interface. O seu uso é opcional;

h) *Enc* pode ser usado para marcar ou recusar pacotes, opcionalmente para *script* com taxa de uso excessiva, uma alternativa ao descarte de pacotes. Sem essa marca, os *scripts* só serão notificados através de *drop*s, e com o uso de ENC, os pacotes serão marcados para descarte, mas não serão descartados em um primeiro momento, e os *script* serão avisados do uso excessivo da taxa de banda.

O descarte ocorrerá somente se a fila atingir o limite máximo. O comando da figura 27 mostra o seguinte cenário com descarte iniciando a partir dos 12.000 *bytes*, e a probabilidade máxima de descarte é de 2%. A probabilidade máxima atingida aos 32.000 *bytes*, velocidade medida de 20 k*bytes* e descarte acima de 256.000 *bytes*. No caso de ECN, em vez de descartados, os pacotes são marcados.

```
tc qdisc add dev eth0 root red limit 2560000 min 12000 max 32000 avpkt
1000 burst 20 probability 0.02 bandwidth 512
```

Fonte: Adaptado Stato (2009)

Figura 27 – Regra com uso do algoritmo RED

Pode-se visualizar o resultado com o comando TC conforme a Figura 28.

```
debian:~# tc -s -d qdisc show dev eth0
qdisc red 8004: root limit 2500Kb min 12000b max 32000b ewma 4 Plog 20 Scell_log 24
Sent 1546 bytes 9 pkt (dropped 0, overlimits 0 requeues 0)
rate 0bit 0pps backlog 0b 0p requeues 0
marked 0 early 0 pdrop 0 other 0
```

Fonte: Adaptado Stato (2009)

Figura 28 – Resultado do comando da figura 27

Por fim o algoritmo CBQ é muito semelhante ao HTB, exceto ao modo utilizado em que o HTB incorpora o formato *Classefull* já citado neste trabalho em capítulos anteriores.

Segundo Stato (2009) o *Class Based Queuing* (CBQ) é uma *qdisc* do tipo *Classless* que implementa uma hierarquia de classes de tráfego, ele pode tanto priorizar como dividir banda. Esse algoritmo é muito usado através de um *script* chamado *shaperd*, que facilita muito o seu uso. O algoritmo CBQ classifica os pacotes em uma árvore de hierarquia de classes, onde cada folha (*leaves*) pode ser considerada uma fila desta árvore e pode trabalhar com algoritmos diferentes em seu interior, ou tratar os pacotes encaminhados para determinadas filas por meio das marcas efetuadas anteriormente através do Iptables.

Pode-se ainda subdividir uma das filas para melhor tratamento da divisão de banda. Esse processo é feito através da criação de classes adicionais dentro da fila qdisc.

Os parâmetros de qdisc utilizados pelo CBQ são:

a) *Parent* informa qual qdisc é instanciada, com o que ela está relacionada;

b) *Handle* atribui um *handle* para a qdisc (um nome);.

c) *Avpkt* é o tamanho médio de um pacote medido em *bytes*;

d) *Bandwidth* é a largura de banda usado para determinar o *idle*. É necessário informar à largura de banda da interface física ou da qdisc *root*;

e) *Ceil* é o tempo que um pacote leva para ser transmitido por um dispositivo, relacionando diretamente com uma taxa.

Os parâmetros de Classe são:

a) *Avpkt* é o tamanho médio de um pacote medido em *bytes*;

b) *Weight* auxilia o balanceamento de envio de pacotes através do processo *Weighted* Round Robin. As classes configuradas com uma banda de valor alto tendem a enviar um tráfego maior. Através desta opção, limita-se o uso da banda, forçando essa classe a enviar em taxas pré-fixadas. Geralmente o valor é *rate*/10;

c) *Allot* especifica quantos *bytes* uma qdisc poderá enviar da fila em cada fase do processo;

d) *Prio* é a prioridade de envio (quanto menor o número, maior a prioridade)*;*

e) *Rate* é a taxa máxima da classe e de todos seus filhos, informação obrigatória;

f) *Bandwidth* é usado para determinar o *maxidle*, que é calculado usando os valores de *maxburst* ou *minburst*;

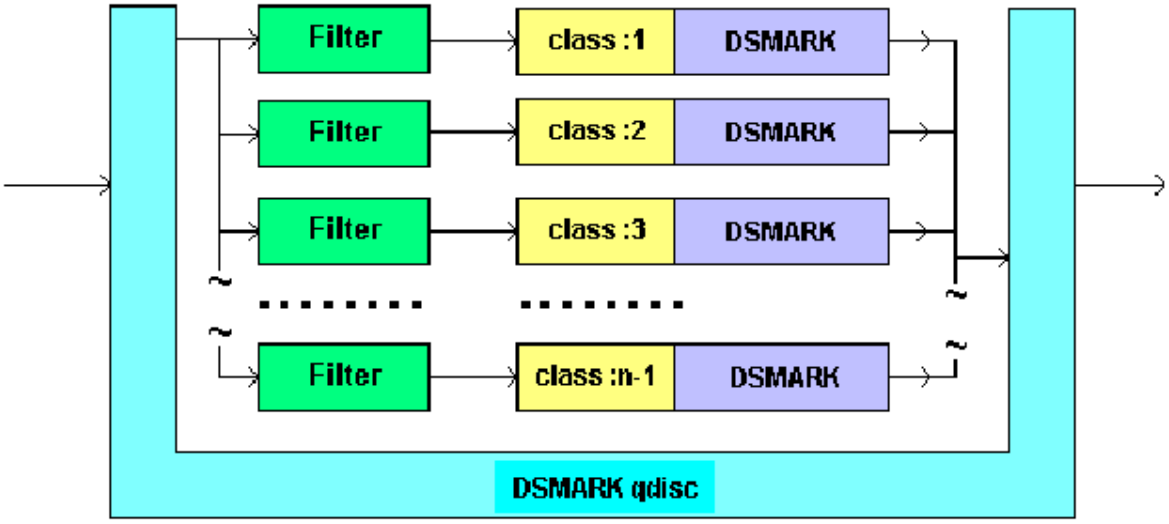
g) *Maxburst* é usado para calcular o *maxidle* (valor máximo de rajada para envio). Nos parâmetros *avgidle*, limita-se o valor de rajadas, de forma que quando ultrapassado, os pacotes começam a ser descartados. Através do uso de *maxburst*, será permitido o envio de rajadas até o valor definido em *maxburst*, mesmo que o

avgidle seja menor até o valor definido em *maxbursts*;

h) *Bounded* especifica que esta classe não pedirá emprestada a banda para outras classes;

i) *Isolated* especifica que esta classe não emprestará banda a outras classes.

Segundo Stato (2009), o algoritmo DSMARK qdisc, é um algoritmo de QoS com capacidade para trabalhar com *Differentiated Services* (DiffServ ou DS). O DS é uma arquitetura ou tipo de QoS que se baseia no valor do campo DS contido no pacote. O DSMARK, diferente dos outros algoritmos que foram apresentados, não é capaz de gerenciar fila e nem priorizar serviços, entre outras funções; Seu objetivo principal é marcar os pacotes no campo DS. A Figura 29 mostra o funcionamentodo DSMARK.



Fonte: Adaptado Stato (2009)

Figura 29 – Funcionamento do DSMARK

Suas classes são numeradas de 1, 2, 3, 4,...,n-1, onde n é um parâmetro que define o tamanho da tabela interna desejada para implementar a qdisc.

Após a separação de tráfego através dos filtros, os pacotes são encaminhados para suas classes, e então o DSMARK estará atuando e marcando ou remarcando os pacotes. Em uma qdisc comum, como o HTB, neste momento estariam sendo aplicadas as especificações em relação a prioridade desejada e configurada anteriormente.

Os pacotes são marcados com um valor inteiro que é definido em cada classe quando configura-se a fila na saída, antes de irem para interface.

Através do uso do parâmetro DSCP do Iptables, podem-se marcar todos os pacotes e

deixar que o roteador de borda tome conta do restante, mas antes precisa-se saber para qual classe que serão enviados os pacotes marcados e quais pacotes marcar.

No Iptables não se pode tomar decisões, como no caso do DSMARK. A opção DSMARK, com o uso de *mask* e *value* através de *bits* de marcação, poderá manter ou alterar tanto a classe como a precedência de *drop* identificando assim qual pacote irá seguir o caminho de qual classe.

Com o Iptables pode-se setar diretamente um pacote já com a classe e precedência final desejada. Seu uso é bem mais simples através do alvo DSCP. Um exemplo de regra segue a figura 30.

```
iptables -t mangle -A PREROUTING -s 192.168.0.0/24 -j DSCP --set-dscp 0x30
```

Fonte: Adaptado Stato (2009)

Figura 30 – Regra para setar pacotes de uma rede com a classe 0x30

Ainda tem-se a possibilidade de capturar o tráfego que venha com marcação, utilizando o Iptables para isto. Mas não tem sentido em capturar o tráfego entrando, a não ser que seja um provedor e estivesse tentando garantir a qualidade de serviço para os clientes.

Segundo Stato (2009), o algoritmo *Policing* verifica os pacotes, se eles podem ou não entrar no domínio do DiffServ. É muito usado na entrada dos roteadores de borda, evitando o consumo de banda já dentro do domínio.

O tráfego é classificado e o que for violado poderá ser descartado, reclassficando ou permitindo. A forma de criar as regras de policiamento não será tão diferente do que foi visto até o momento. Os pacotes são classificados e colocados na suas classes predefinidas. É preciso configurar as regras de policiamento para que sejam aplicados os valores correspondentes em suas classes. Pacotes que excederem essas regras serão descartadas, relcassificados ou permitidos. O policiamento poderá tomar decisões quando determinados pacotes excederem a banda permitida. Por exemplo, se for configurado uma banda para policiar uma taxa de 4M*bits* e o tráfego correspondente a taxa for maior que 5M*bits* pode-se parar todo tráfego de 5M*bits*, ou apenas fazer com que o policiamento deixe passar 4M*bits* e policiar o 1M*bit* excedente, deixando o restante ser enviado para a classe configurada.

Se a banda exceder a taxa configurada, pode-se então *drop*ar, reclassficiar ou mandar verificar se outro filtro poderá corresponder a esse tráfego, tomando novas atitudes. Para se ter um resultado satisfatório, pode-se policiar o tráfego usando as três formas: ações tomadas com itens de permissão, reclassificação ou simplesmente *drop*.

TRABALHOS CORRELATOS

Tratando-se em controle de tráfego de rede existem vários *softwares* no mercado. Muitos destes são pagos, principalmente aqueles voltados para sistema operacional Windows. Como o foco deste trabalho é um controle sobre *gateway* Linux, há como base um *software* chamado WebHTB.

Conforme Delicostea (2009), WebHTB é um conjunto de *softwares* que ajudam a simplificar o difícil processo de alocação de largura de banda, o tráfego, tanto para *upload* e *download*, permitindo gerar e verificar os arquivos de configuração, visão geral do tráfego em tempo real para cada cliente em separado.

Tendo em vista que WebHTB é um *software* livre, voltado para sistemas operacionais *Linux*, o WebHTB utiliza um conjunto de ferramentas que auxiliam na aplicação do QoS na rede. Algumas das ferramentas utilizadas é o Iptables, o algoritmo HTB e que serão utilizadas neste presente trabalho. Abaixo na Figura 31 há um exemplo da operacionalidade do WebHTB.

Fonte: Delicostea (2009)

Figura 31 – Ajustando a largura de banda e *host* que será aplicado o filtro

Este *software* foi desenvolvido em conjunto com as ferramentas já citadas, uma interface desenvolvida em *Hypertext Preprocessor* (PHP), *Java Script, Ajax* e *MySQL*, proporcionando uma interface amigável, facilitadora para administração do controle de banda na rede.

Em relação ao controle de banda, existe a ferramenta HTB-tools. Segundo Stato

(2009, p. 263), conforme informado no site do HTB-tools, é uma ferramenta para controle e administração de banda, um *software* com vários recursos para auxiliar e simplificar o processo de alocação de banda, tráfego, *download, upload*, gerando e checando configurações para cada cliente isoladamente.

A Figura 32 mostra um exemplo de arquivo de configuração da ferramenta HTB-tools.

```
class Wireless {
        bandwidth 480;
        limit 512;
        burst 2;
        priority 1;

        client cliente_1 {
            bandwidth 192;
            limit 256;
            burst 2;
            priority 1;
            src { 192.168.1.2/24; };
            dst { 192.168.1.2/24; };
        };

        client cliente_2 {
            bandwidth 192;
            limit 256;
            burst 2;
            priority 1;
            src { 192.168.2.2/24; };
            dst { 192.168.2.2/24; };
        };

};

class default { bandwidth 8; };
```

Fonte: adaptada Stato (2009)

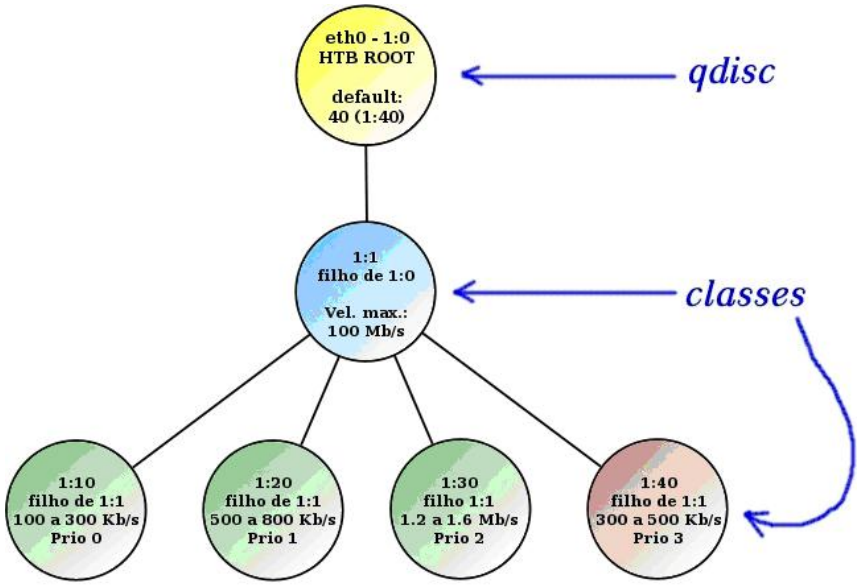
Figura 32 – Exemplo de configuração do HTB-tools

Como exemplo de que o algoritmo *Hierarchical Token Bucket* (HTB) é muito menos compreensível que a ferramenta HTB-tools, nas figura 33 e 34 vê-se um exemplo de arquivo de configuração da utilizando o algoritmo HTB e um esquema de como funciona após implantado.

Segundo Stato (2009, p. 245), o HTB foi criado por Martin Devera e incluído no *kernel* a partir do 2.4.20 e implementa uma fila com suporte a várias classes para controle de tráfego, onde são usados parâmetros como:

a) *rate*: largura de banda disponível para uso nessa classe;

b) *ceil*: largura máxima de banda que uma classe poderá consumir.



Fonte: Motta (2009)

Figura 33 – Mostra esquema de prioridade de uma ferramenta HTB

```
#!/bin/bash

tc qdisc del dev eth1 root

tc qdisc add dev eth1 root handle 1:0 htb default 40
tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100mbit

tc class add dev eth1 parent 1:1 classid 1:10 htb rate 100kbit ceil 300kbit prio 0  # DNS / ACK / SYN / FIN
tc class add dev eth1 parent 1:1 classid 1:20 htb rate 500kbit ceil 500kbit prio 1  # E-mail
tc class add dev eth1 parent 1:1 classid 1:30 htb rate 300kbit ceil 500kbit prio 2  # FTP
tc class add dev eth1 parent 1:1 classid 1:40 htb rate 300kbit ceil 500kbit prio 3  # Geral
```

Fonte: adaptada de Stato (2009)

Figura 34 – Mostra um trecho de configuração que utiliza o algoritmo HTB

O *software* desenvolvido em relação ao HTB-tools terá hierarquia de tráfego, ou seja, quando existir largura de banda sobrando para um determinado serviço o mesmo poderá receber emprestado de sua classe principal, quando serviços com maior prioridade não forem requisitados. Será determinado para cada serviço um valor mínimo e máximo. O máximo poderá atingir o valor que sua classe principal tiver. Um serviço com menos prioridade dentro do cenário, poderá consumir a largura de banda total predominada, mas caso algum serviço com maior prioridade seja requisitado, o menos prioritário consequentemente terá sua taxa de transferência diminuída, mas terá um mínimo de transferência garantida pelo valor já estabelecido. O algoritmo HTB e as ferramentas de auxílio são responsáveis por este controle de ajustes das taxas. Quem tiver maior prioridade sempre terá maior largura disponível e

velocidade de resposta dos pacotes. Assim que um serviço com prioridade maior parar de requisitar largura de banda os serviços que estão sendo utilizados tiverem folga no *link* irão aumentar suas taxas até que seja concluído ou que não seja requisitado algum outro serviço com maior prioridade.

Para configuração do HTB-tools é necessário editar um arquivo de configuração dentro de um diretório no Linux. Este arquivo é igual à Figura 3 e é o responsável pela configuração. O software proposto terá uma interface Web, acessível dentro da rede de qualquer computador que tiver um navegador instalado. Ao contrário do HTB-tools que somente poderá ser configurado onde estiver instalado ou através de acesso remoto.

DESENVOLVIMENTO

O presente capítulo apresenta a especificação, a implantação e os testes do *software* referentes às ações de qualidade de serviço. Na primeira seção, é feito o levantamento das informações. Na segunda seção são descritos os requisitos do *software*. A terceira seção mostra a especificação do *software* através do diagrama de caso de uso. A quarta seção descreve a implementação, as técnicas e ferramentas utilizadas e a operacionalidade da implementação através de um caso de uso. Na quinta e última seção estão descritos os resultados obtidos.

3.1 LEVANTAMENTO DAS INFORMAÇÕES

Faz-se um levantamento sobre o uso de recursos de uma rede através de estatísticas dos usuários, do registro do consumo de banda sempre alto em determinados horários, muitas vezes, consumo indevido não tendo utilidade alguma dentro da rede corporativa, inclusive tráfego indesejado como *worms*, vírus e todas as pragas virtuais possíveis que fazem comunicação com a rede para a internet acabam entupindo a rede deixando o tráfego importante de fora ou concorrendo com programas P2P, *bittorrent* entre outros. Com tudo isso, chegou-se a fazer bloqueios, mas determinados serviços não podem ser bloqueados, mas pode-se fornecer baixa prioridade. O QoS neste caso fará com que serviços não priorizados sejam menos importantes fazendo com que o tráfego importante seja prioritário com a largura de banda que é necessário para seu funcionamento.

3.2 REQUISITOS PRINCIPAIS DO SISTEMA

Nesta seção serão apresentados os principais requisitos funcionais (RF), requisitos não funcionais (RNF), e sua rastreabilidade com seus respectivos casos de uso. No quadro 1 são apresentados os requisitos funcionais da ferramenta.

Requisitos Funcionais	Caso de Uso
RF01: O *software* deve permitir cadastrar parâmetros da rede	UC01
RF02: O *software* deve permitir configurar largura de banda por serviço de rede	UC02
RF03: O *software* deve permitir configurar prioridade de serviços da rede	UC03
RF04: O *software* deve permitir visualizar prioridades aplicadas para rede	UC04
RF05: O *software* deve permitir visualizar equipamentos ativos na rede	UC05
RF06: O *software* deve permitir visualizar o consumo do link por serviço	UC06
RF07: O *software* deve permitir visualizar a largura de banda reservada por equipamento de rede	UC07

Quadro 1: Requisitos funcionais

O Quadro 2 lista os requisitos não funcionais previstos para o *software*.

Requisitos Não Funcionais
RNF01: O *software* deverá utilizar sistema operacional Linux como *gateway* padrão da rede
RNF02: O *software* deverá utilizar versão do Apache 2.0
RNF03: O *software* deverá utilizar o algoritmo HTB presente a partir do Kernel 2.4.22
RNF04: O *software* deverá utilizar a ferramenta Iptables
RNF05: O *software* deverá ser implementado em PHP e *shell script*

Quadro 2: Requisitos não funcionais

3.3 ESPECIFICAÇÃO

Esta seção apresenta o diagrama que será necessário para o entendimento da ferramenta de que fará o QoS em *gateway* Linux. Para a especificação do *software* utilizou-se a notação UML, sendo o diagrama gerado através da ferramenta *Enterprise Architect.*

3.3.1 Diagrama de Caso de Uso e Diagrama de Atividades

Esta sub-seção apresenta o diagrama de caso de uso do sistema e o diagrama de

atividades, sendo que o detalhamento dos principais casos de uso estão descritos no Apêndice A. Na figura 35, tem-se o diagrama de caso de uso dos cadastros do *software*.

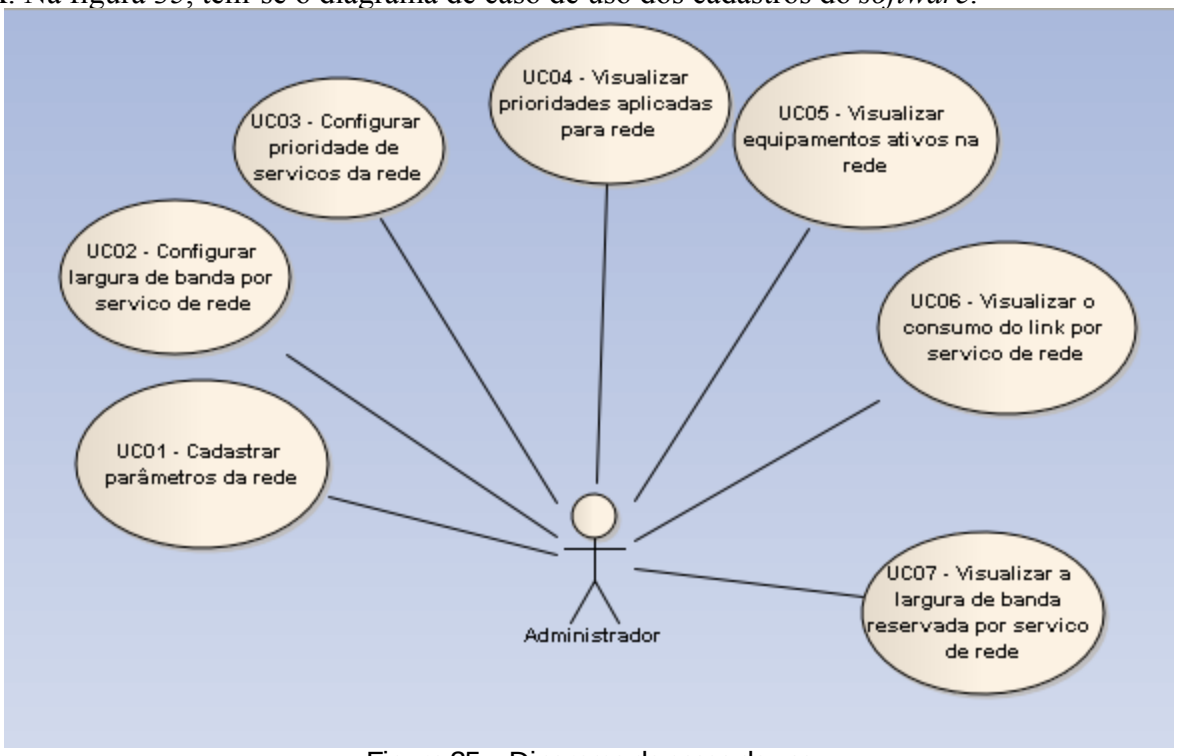


Figura 35 – Diagrama de casos de uso

Na figura 36 tem-se o diagrama de atividades que mostra as principais transações do *software.*

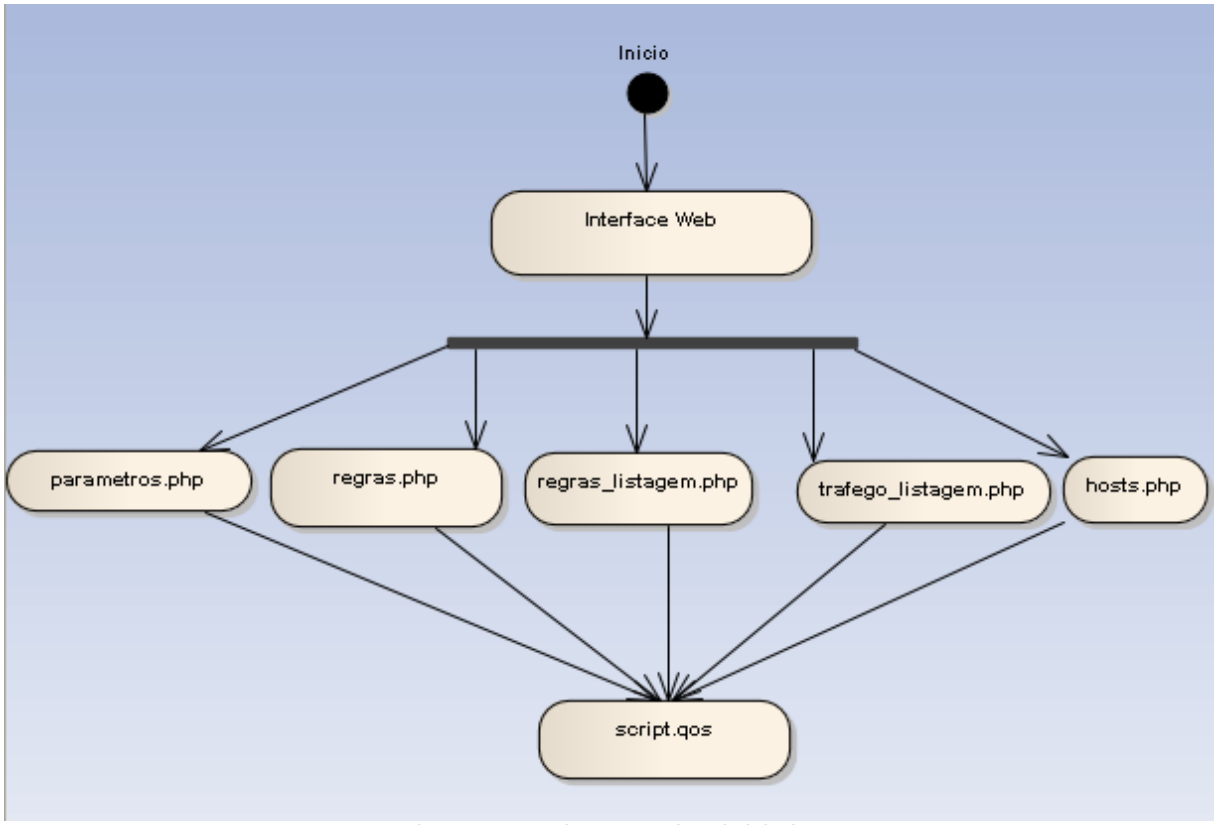


Figura 36 – Diagrama de atividades

Através da interface web de acordo com a seleção dos menus é apresentada uma tela para que o usuário preencha as informações necessárias e o *script* correspondente é chamado para validar as opções, configurar o ambiente e então o *script.qos* é chamado para executar as regras. Os *scripts* são: parametros.php que serve para capturar os dados inseridos conforme Figura 39 que mostra os campos a serem preenchidos na tela parâmetros. O *script* regras.php é responsável por gerar as informações que as regras de QoS utilizarão através do *script.qos* que é o responsável pela execução das regras. O *script* regras_listagem.php é o responsável pelo relatório das regras criadas. Este *script* listas as regras que foram cadastradas através da tela Cadastro de Regas. O *script* trafego_listagem.php é responsável por mostrar quantos *bytes* trafegaram pelo *gateway* conforme as regras criadas. Por fim, o script hosts.php é quem mostra quais *hosts* estão conectados ao *gateway*. Todos os *scripts* comunicam-se com o script.qos que após receber os dados vindo dos *script* acima, este é executado pelo sistema operacional onde chamará as ferramentas e *scripts* utilizados para que o QoS seja aplicado. Tudo que for feito através da interface web é o script.qos quem executará, capturará as informações e executará no *shell* do Linux assim aplicando as regras de QoS.

3.4 IMPLEMENTAÇÃO

Esta seção demonstra os detalhes da implementação do projeto, bem como as técnicas e ferramentas utilizadas.

3.4.1 Técnicas e Ferramentas utilizadas

Para implementação do software, utilizada a ferramenta Adobe Dreamweaver CS3, pois é uma ferramenta que auxilia no desenvolvimento de websites permitindo a criação de arquivos *Hyper Text Markup Language* (HTML), na qual foram estruturadas as sessões da ferramenta, que receberão as informações a partir das páginas e serão convertidas em *shell script* para assim aplicar as regras, onde o Kernel do Linux interpretará e aplicará. Para validação de algumas informações foi utilizado Java Script.

Para o desenvolvimento foi utilizado a ferramenta TC que acompanha o pacote

IProute2, o algoritmo HTB, a ferramenta Iptables, phpMyAdmin 2.13 e o interpretador de páginas PHP 5.

3.4.2 Operacionalidade da implementação

Nesta sub-seção é apresentada a seqüência de telas com as operações para cada tipo de serviço requisitado, para conseguir utilizar corretamente a ferramenta de controle de banda. Também serão apresentados trechos dos códigos fonte de algumas rotinas mais importantes da ferramenta.

3.4.2.1 Logar no Sistema

Na tela apresentada na figura 37, o administrador informar o usuário que será o usuário *root* do sistema operacional para que possa acessar o sistema.

Figura 37 – *Login* do *software*

Informando o *login* e senha, o administrador clica em *login*. A ferramentra irá verificará se a senha está correta. Caso a senha não esteja correta retornará a tela para reescrever a senha. Quando houver a validação do *login*, a ferramenta apresentará a tela inicial de parâmetros com as opções para o cadastro.

3.4.2.1 Barra de Menus

A barra de menu indica todas as opções por onde o administrador pode navegar. Na Figura 38 é demonstrada a barra de menu acessível ao administrador. A barra é estática, ou seja, é vista em todas as páginas possibilitando ao administrador o acesso imediato aos menus Parâmetros, Cadastro de Regras, Listagem de Regras, *hosts online*, consumo de banda e sair.

Figura 38 – Menu da *software*

3.4.2.2 Tela de Parâmetros

Após o *login* no *software*, será direcionada à tela de parâmetros que indica todos os campos necessários para posteriormente fazer-se o cadastro das regras. Nesta tela todos os campos são obrigatórios. Os campos descrito na figura 39 tratarão das regras padrão, ou seja, a largura de banda destinada para as regras que não serão classificadas como o link máximo de *download* e *upload*, as interfaces de rede e o caminho para o *script* de *firewall* do sistema.

Depois de preenchidos os dados clicando em confirmar, estes serão salvos e a ferramenta não perguntará nada para confirmar e continuará na tela parâmetros. Se clicar em limpar todas os campos serão limpos, mas o *software* perguntará se deseja realmente limpar os campos conforme a figura 41. Após confirmar é retornado à tela parâmetro com todos os campos em branco. Em regras padrão, existe um ícone a direita em que mostra uma ajuda caso o administrador não saiba para que serve estes campos. Na figura 40 esta tela de ajuda é mostrada.

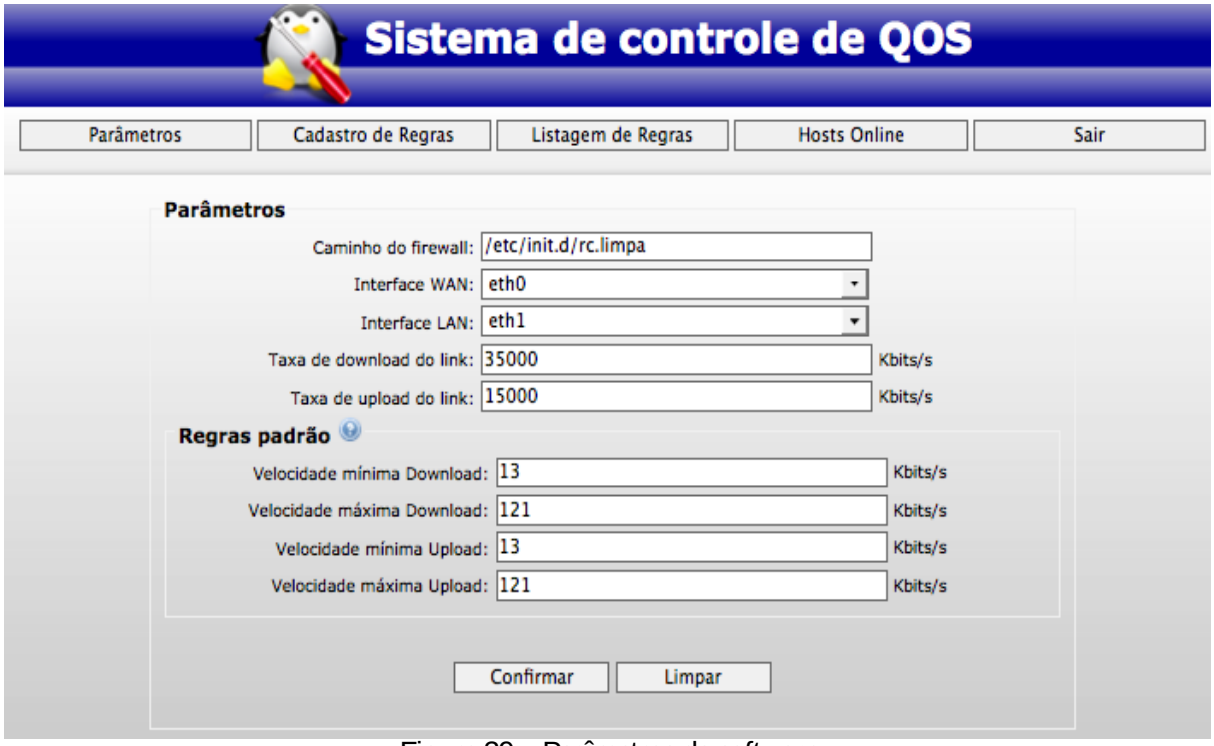


Figura 39 – Parâmetros da *software*

Caso o administrador clicar no link ao lado de Regras padrão a tela da figura 40 será apresentada.

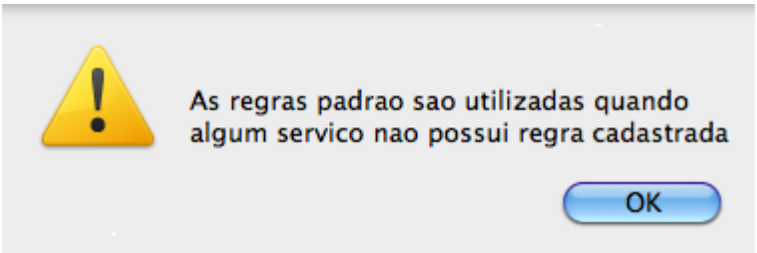


Figura 40 – Tele de ajuda para regras padrão

Caso o administrador clicar em limpar a mensagem para confirmar, conforme a tela da figura 41.

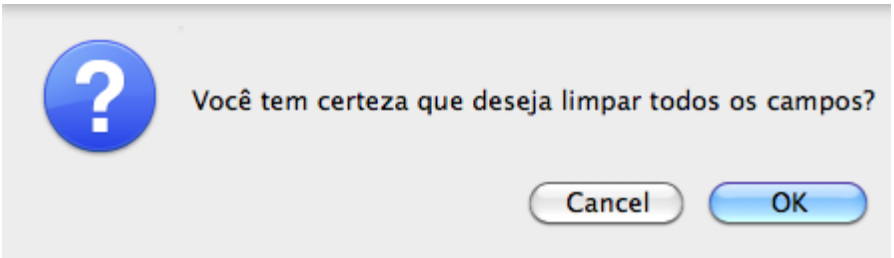


Figura 41 – Parâmetros da *software*

3.4.2.3 Cadastro de Regras

No menu Cadastro de Regras é feita a aplicação das regras que o administrador deseja efetuar e priorizar dentro da rede. Nesta tela conforme figura 42 todos os campos são obrigatórios. Para efetuar o cadastro é só clicar em Confirmar. Após isto o *software* retorna para a mesma tela de parâmetros com os campos atuais, onde o administrador pode excluir a regra feita, clicando no item Excluir conforme figura 39. Ou clicando em Novo para efetuar um novo cadastro. Na figura 42 é mostrado quando o administrador clica sobre o menu Cadastro de Regras pela primeira vez e existe apenas o botão Confirmar.

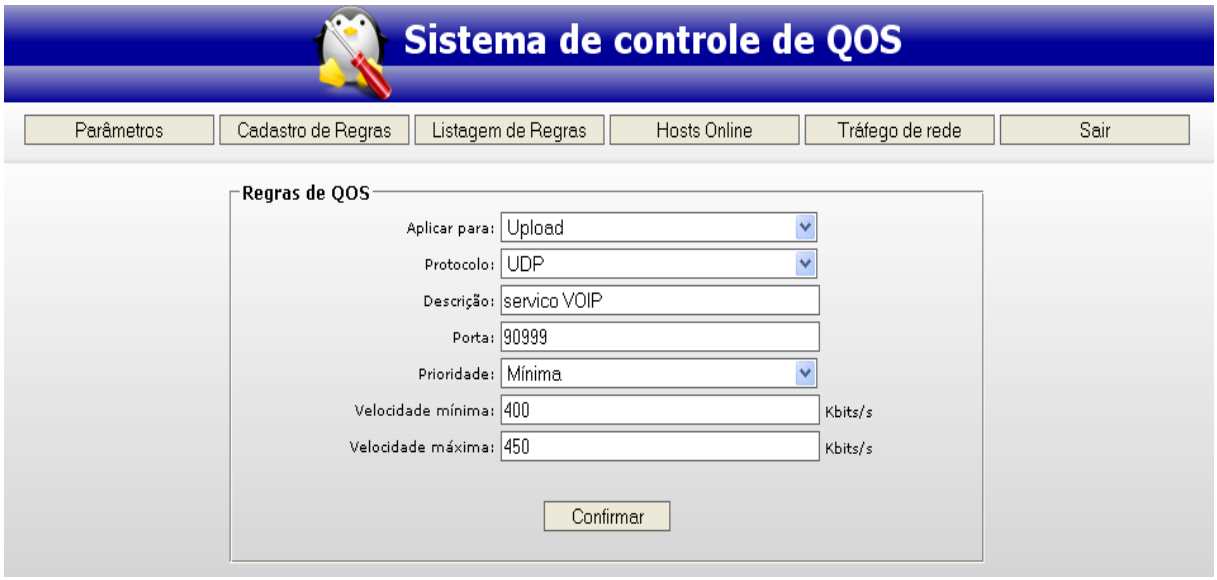


Figura 42 – Tela de cadastro de regras do *software*

Agora depois de incluída uma regra aparecerá mais dois botões como descrito, conforme figura 43.

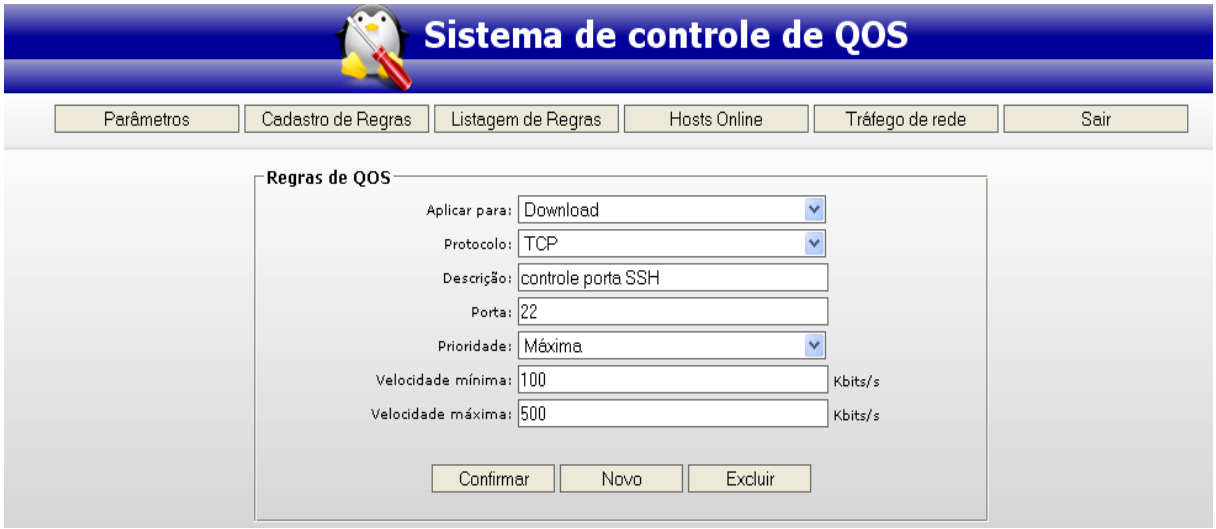


Figura 43 – Tela de recadastro do *software*

3.4.2.4 Tela de Listagem de Regras

Nesta tela (figura 44) é apresentado um relatório das regras aplicadas. Nesta tela poderão ser visualizar as regras aplicadas com suas respectivas características: descrição da regra, protocolo utilizado, porta, prioridade, velocidade máximo e mínima e aplicação se é para *download* e *upload*. A identificação é mostrada através de uma seta para cima ou para baixo assim diferenciando *download* e *upload*. Com o mouse também passando por cima das setas mostrará se é *download* ou *upload*. Se clicar sobre a regra ela irá cair na tela de cadastro de regra onde poderá ser feita a exclusão da mesma ou edição.A primeira coluna que aparece é uma coluna onde se pode selecionar todas as regras ou ir selecionando-as para poder excluí-las se for necessário. Caso selecionado alguma regra existe o botão Excluir Selecionados para tal ação. A figura 44 mostra a tela de listagem de regras.

Figura 44 – Tela de listagem de regras do *software*

O arquivo principal que fará a aplicação das regras é o rc.qos. Este é um *shell script*, e é nele que a interface web captura as informações em que o administrador informa e grava para que o sistema operacional possa entender e fazer com que o QoS aconteça. Na figura 45, há um trecho do *script* mostrando a marcação de pacotes de algumas regras já inseridas como exemplo pelo administrador.

```
script.qos.txt     [----]  0 L:[149+20 169/255] *(3774/8913b)= #  35 0x23
#classe root down
tc class add dev $INIF parent 1:0 classid 1:1 htb rate $TOTALRATEDOWN burst $BURSTMAX cburst $CBURSTMAX


tc class add dev $INIF parent 1:1 classid 1:5   htb rate $TOTALRATEDOWN ceil $TOTALRATEDOWN quantum $QUANTUMMAX
#subclasses_down

#/subclasses_down.
tc class add dev $INIF parent 1:1 classid 1:200 htb rate $MINRATED200   ceil $MAXRATED200   quantum $QUANTUM


tc filter add dev $INIF parent 1:0 protocol ip prio 5   handle $MARKPRIOD5   fw classid 1:5
#filtros_de_pacotes_down

#/filtros_de_pacotes_down.
tc filter add dev $INIF parent 1:0 protocol ip prio 200 handle $MARKPRIOD200 fw classid 1:200

tc qdisc add dev $INIF parent 1:5   sfq perturb 16 quantum $QUANTUMMAX
#fila_de_disciplina_down

#/fila_de_disciplina_down
tc qdisc add dev $INIF parent 1:200 sfq perturb 16 quantum $QUANTUM
```

Figura 45 – Trecho de código do PHP

Na figura 46 tem-se um trecho de código da classe principal desenvolvida em PHP. Esta, é a responsável por varrer o arquivo rc.qos mencionado na figura 30, capturar os dados que o administrador inseriu na interface web e ajusta-lo no *script* rc.qos para que o sistema operacional faça a execução das regras corretamente.

```
parametros.php     [----]  0 L:[  1+ 0   1/167] *(0   /7912b)= <  60 0x3C
<?php^
require_once "includes/validaLogin.inc.php";^
require_once "classes/parametro.class.php";^
require_once "classes/geral.class.php";^
^
// instacia as classes ^
$objParametro = new Parametro();^
$objGeral = new Geral();^
// Pega as interfaces de rede^
$strInterfaces = $objParametro -> pegaInterfaces();^
// se foi submitado o formulario salva os dados no arquivo do script ( ele verifica se todos os campos foram enviados para fazer o salvamento)^
if($_POST["caminho_firewall"] && $_POST["interface_wan"] && $_POST["interface_lan"] && $_POST["taxa_download"] && $_POST["taxa_upload"])^
{^
    $objGeral -> varreScript("#firewall", "sh", "sh ".$_POST["caminho_firewall"]);^
    $objGeral -> varreScript("#interfaces", "EXIF=", 'EXIF="'.$_POST["interface_wan"].'"');^
    $objGeral -> varreScript("#interfaces", "INIF=", 'INIF="'.$_POST["interface_lan"].'"');^
    $objGeral -> varreScript("#ratesDown", "TOTALRATEDOWN=", 'TOTALRATEDOWN="'.$_POST["taxa_download"].'kbit"', true);^
    $objGeral -> varreScript("#ratesUP", "TOTALRATEUP=", 'TOTALRATEUP="'.$_POST["taxa_upload"].'kbit"', true);^
    $objGeral -> varreScript("#minRateD", "MINRATED200=", 'MINRATED200="'.$_POST["velocidade_minimaD"].'kbit"', true);^
    $objGeral -> varreScript("#maxRateD", "MAXRATED200=", 'MAXRATED200="'.$_POST["velocidade_maximaD"].'kbit"', true);^
    $objGeral -> varreScript("#minRateU", "MINRATEU400=", 'MINRATEU400="'.$_POST["velocidade_minimaU"].'kbit"', true);^
    $objGeral -> varreScript("#maxRateU", "MAXRATEU400=", 'MAXRATEU400="'.$_POST["velocidade_maximaU"].'kbit"', true);^
^
```

Figura 46 – Trecho de código do PHP

3.4.2.5 Tela de *Script Online*

Nesta tela é apresentada a quantidade e o número IP de cada computador que estiver

conectado ao *gateway* Linux (figura 47).

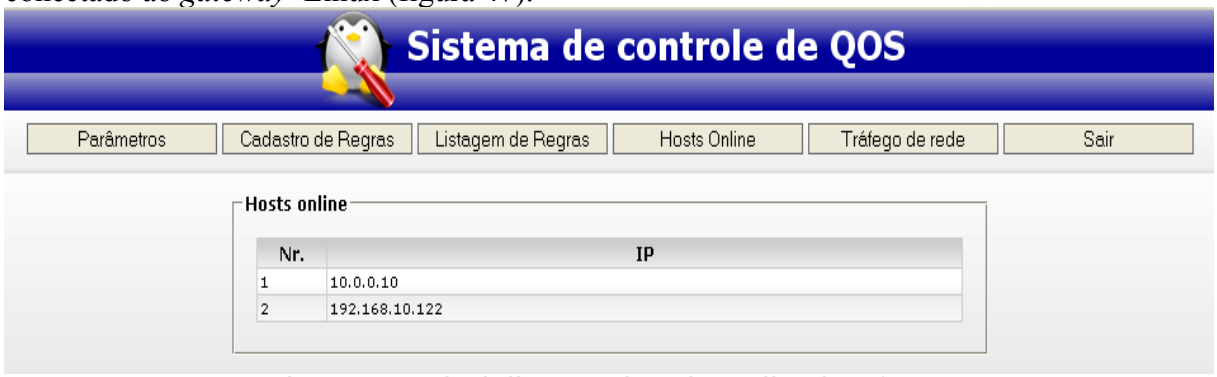


Figura 47 – Tela de listagem de *script online* do *software*

Na figura 48 é apresentado o trecho do código onde é feita a captura dos IPs que passam pelo *gateway* Linux . É utilizado o comando arp para encontrar os ips conectados no *gateway* .

```
classes/ho~.class.php   [----] 10 L:[  9+ 0   9/103] *(136 /2480b)= .  10 0x0A
        function __construct()
        {
                $this -> objConfig = new Config;
                $this -> objGeral = new Geral;
        }


        public function retornaArp()
        {
                $arp = shell_exec("/usr/bin/sudo /usr/sbin/arp");
                /*$arp = "Address                  HWtype  HWaddress
                        10.1.1.1                 ether   00:1e:58:1a:27:6a   C
                        192.168.0.67             ether   00:13:3b:03:03:17   C
                        192.168.0.6                      (incompleto)
                        10.1.1.66                ether   00:4f:62:04:26:08   C
                $arp = stripcslashes(nl2br(strip_tags(($arp))));
                $arp =  trim(preg_replace('!\s+!', ' ', $arp));
```

Figura 48 – Tela do código que captura os ips conectados ao *gateway* Linux

3.4.2.6 Tela Consumo de Banda

A tela Trafego de rede lista a quantidade de *bytes* consumidos por serviço, as regras cadastradas. Nesta tela é mostrado o total de *bytes* por serviço, conforme figura 49.

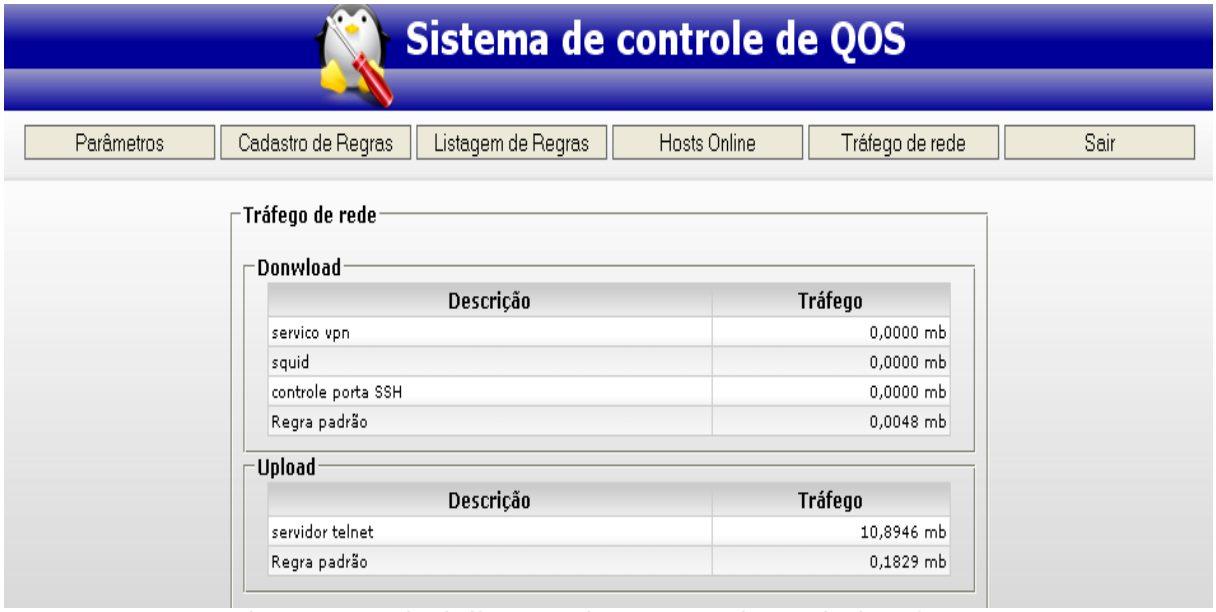


Figura 49 – Tela de listagem de Consumo de Banda do *software*

A ferramenta responsável pela medição do consumo de banda é o TC. Com isso dá-se a possibilidade de saber a quantidade de *bytes* que passa em cada regra definida pelo administrador. Conforme figura 50 vê-se o comando TC com seus parâmetros, assim pode-se capturar a quantidade de *bytes* que passam pela regra. Neste exemplo a regra é default 400 e passaram 2312 *bytes*.

```
l7:/var/www/qos/qos-05-11-2010/qos# tc -s -s qdisc show dev eth0
qdisc htb 1: root r2q 1 default 400 direct_packets_stat 0
 Sent 2312 bytes 16 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
qdisc sfq 8003: parent 1:25 limit 127p quantum 12187b perturb 16sec
```

Figura 50 – Tela de listagem de Consumo de Banda da ferramenta

3.4.2.7 Tela Sair

O menu Sair após o administrador efetuar *login*, poderá fazer o *logout*, que é a saída do sistema através do menu Sair. Após clicar em Sair o sistema volta para tela inicial de *login*, conforme figura 37.

3.5 RESULTADOS E DISCUSSÃO

O desenvolvimento deste trabalho permitiu automatizar e agilizar os processos que envolvem atividades dentro de uma regra com o controle de banda muitas vezes negligenciado por administradores.

Aumentar o *link*, ao contrário do que se pensa, não resolveria o problema de lentidão na rede, pois só se está dando mais poder de fogo ao usuário que utiliza tráfego indevido (MOTTA, 2007).

A ferramenta fará com que este problema citado pelo Motta (2007) seja sanado e de fácil entendimento para o administrador. QoS existe pouco material difundido em português e de difícil entendimento.

Segundo Stato (2009), QoS é um assunto tão importante e nenhum material publicado, ou ferramentas mais claras, concretas, com facilidade ampla para um melhor entendimento. Não se precisa ter um conhecimento profundo no Linux para poder entender o uso do *firewall* e das demais ferramentas abordadas, precisa-se de apenas um conhecimento básico sobre o assunto.

Com este *software* a facilidade de se efetuar um controle de banda na rede é muito mais fácil do que programar os *scripts* propriamente dito. A interface web é quem faz este elo entre as informações que o administrador passará e as conexões com *scripts* e ferramentas para efetuar o controle. A principal vantagem é que o próprio administrador não precisa estar na rede em que deseja efetuar o controle, como é um sistema web, pode-se fazer o controle, acesso de qualquer lugar que tenha comunicação com a internet. O administrador apenas precisa saber o que ele quer fazer, com isto conseguirá efetuar o controle através da interface web da software, para que todo tráfego da rede seja controlado conforme prioridade e largura de banda requerida por serviço.

CONCLUSÕES

O *software* desenvolvido para este trabalho atendeu a todos os requisitos que foram estabelecidos na proposta inicial, apesar de ter havido algumas dificuldades na linguagem de programação.

A interface web proporcionou aos usuários da rede um acesso fácil e rápido. A forma simples de fazer o controle facilita muito ao administrador .

Com os estudos e implementações que foram feitos neste trabalho, conclui-se que implementar um controle de banda numa rede não é uma tarefa simples, pois exige o conhecimento de várias técnicas, algoritmos e ferramentas específicas do Linux para que se aplique o QoS com sucesso.

Conforme Stato (2009), QoS é uma técnica que engloba muitas ferramentas e conhecimentos específicos para que se consiga obter sucesso neste cenário.

> Resolvi escrever o livro, pois um assunto tão importante como controle de banda e nenhum material nacional publicado. Estas técnicas são voltadas para administradores de redes, analista e amantes do Linux. (STATO, 2009, p. 353).

O intuito deste trabalho é facilitar este controle, pois através da interface web, é transparente para o administrador, assim conseguirá liberar banda para os serviços de uma rede. A maior facilidade, é que se pode fazer o controle de qualquer lugar que tenha conexão com a internet, acesso remoto. As regras são dinâmicas, não há necessidade de alterar código ou se entender o *script* que faz a interação com as ferramentas e algoritmos do sistema operacional. A única resposta que o administrador terá são os serviços requeridos se enquadrando dentro das regras que foram acrescentadas na interface web.

Com isto pode-se ter um controle maior e eficaz do tráfego real de que uma rede necessita, inclusive deixando o tráfego inesperado, que muitas vezes não se pode simplesmente bloqueá-lo, determina-se uma prioridade e largura de banda baixa, apenas passando como padrão sua porta de origem. Assim o administrador poderá ter mais folga do *link* em serviços que são realmente importantes e que são fundamentais para o bom desempenho da rede

4.1 EXTENSÕES

Há outras funcionalidades que estão implementadas no *software* como:

a) quantidade de *bytes* consumidos por serviço;
b) *script online*

Todas estas funcionalidades estão sendo tratadas através de ferramentas que o próprio sistema já traz por *default*. No caso da medição do consumido em *bytes* por serviço é feito a utilização da ferramenta TC. Para saber quais *hosts* estão *online* faz-se a utilização da ferramenta arp listando quais *hosts* estão conectados ao *gateway* .

Assim, o *software* faz com que haja um controle de quem está conectado ao *gateway*, qual das regras possui mais consumo de banda, contudo o administrador poderá constatar se precisará ter mais largura de banda em uma determina regra pelo seu alto consumo em relação a outras por exemplo.

REFERÊNCIAS BIBLIOGRÁFICAS

CARVALHO, G. **Qualidade de Serviços para *Gateway* s Linux (QoS).** Rio de Janeiro, 2006. Disponível em <http://www.vivaoLinux .com.br/artigo/Qualidade-de-Servicos-para-Gateway s-Linux -(QoS) >. Acesso em: 01 mar. 2010.

DELICOSTEA, D. **Webhtb**. Romênia, 2009. Disponível em < http://webhtb.nethd.ro/>. Acesso em: 02 fev. 2010.

DEVERA, M. **HTB Home.** República Checa, 2002. Disponível em <http://luxik.cdi.cz/ ~devik/qos/htb/>. Acesso em: 27 nov. 2010.

FERREO, C. **A importância da qualidade do serviço.** São Paulo, 2002. Disponível em <http://www.anixtersoluciones.com/latam/br/informao_geral/12693/ a_importancia_da_qualidade_do_servico_qos___parte_i_po.htm>. Acesso em: 15 out. 2010.

JUCÁ, L. H. **Técnicas avançadas de conectividade e firewall em GNU/Linux .** Rio de Janeiro: BRASPORT, 2005.

LESSA, C. V. B. **Controle de banda com HTB.** São Paulo, 2004. Disponível em < http://br-linux.org/tutoriais/001648.html>. Acesso em: 27 nov. 2010.

MOTTA, F. J. E. **Controle de tráfego com TC, HTB e Iptables.** São Paulo, 2007. Disponível em <http://eriberto.pro.br/wiki/index.php? title=Controle_de_tr%C3%A1fego_com_TC%2C_HTB_e_Iptables>. Acesso em: 01 abr. 2010.

MOSCHETO, R. **Controle de banda com HTB-Tools.** Rio de Janeiro, 2006. Disponível em <http://www.vivaoLinux .com.br/artigo/Controle-de-banda-com-HTBTools/>. Acesso em: 10 mar. 2010.

PÉRICAS, F. A. **Rede de computadores:** conceitos e arquitetura internet. Blumenau: EDIFURB, 2003.

STATO, A. F. **Linux controle de redes:** firewall Iptables, balanceamento de link, qualidade de serviços e roteamento dinâmico**.** Florianópolis: VISUAL BOOKS, 2009.

STACHLEWSKI, S. R. **QoS Quality of Service.** Natal, 2008. Disponível em: <http:// www.dimap.ufrn.br/~glaucia/RAV/QoS-Artigo.pdf**>.** Acesso em: 01 abr. 2010.

APÊNDICE A – Detalhamento dos casos de uso

No Quadro 3 apresenta-se o caso de uso "Cadastrar parâmetros da rede".

Nome do Caso de Uso	Cadastrar parâmetros da rede
Descrição	Administrador acessa a aplicação web informa caminho do *firewall*, *interfaces* de rede, taxa máxima de *download e upload* para *link* total, taxa máxima de *download* e *upload* para regras que não terão tratamento.
Ator	Administrador.
Pré-condição	Administrador deve saber quais parâmetros são necessários.
Fluxo principal	1. O *software* requisita os dados para ao administrador; 2. *Software* valida os dados que o administrador digitou.
Cenário – Visualização	*Software* mostra os parâmetros cadastrados pelo Administrador.
Cenário – Edição	1. *Software* mostra os parâmetros cadastrados; 2. Administrador seleciona um parâmetro para edição; 3. Administrador altera o parâmetro e seleciona opção confirmar; 4. *Software* mostra os parâmetros cadastrados com o registro alterado.
Cenário – Inclusão	1. *Software* mostra parâmetros cadastrados; *2.* Administrador inclui um novo parâmetro; 3. *Software* mostra os parâmetros cadastrados.
Cenário – Exclusão	1. *Software* mostra os parâmetros cadastrados; 2. Administrador seleciona um parâmetro para exclusão; 3. *Software* exclui o parâmetro e mostra os registros restantes.
Pós-condição	O controle de banda começa a funcionar com os parâmetros fornecidos pelo administrador.

Quadro 3 – Descrição do caso de uso UC01

No Quadro 4 apresenta-se o caso de uso "Configurar largura de banda por serviço de rede".

Nome do Caso de Uso	Configurar largura de banda por serviço de rede.
Descrição	Administrador acessa a aplicação web, consulta o endereço ip já preenchido passando os parâmetros que deseja aplicar.
Ator	Administrador.
Pré-condição	Administrador deve saber quais parâmetros deseja aplicar para o endereço desejado.
Fluxo principal	1. O *software* requisita os dados para o administrador; 2. *Software* valida os dados que o administrador digitou.
Cenário – Visualização	*Software* mostra o endereço ip com as regras aplicadas.
Pós-condição	Administrador visualizou o endereço ip com regras aplicadas.

Quadro 4 – Descrição do caso de uso UC02

No Quadro 5 apresenta-se o caso de uso "Configurar prioridade de serviços da rede".

Nome do Caso de Uso	Configurar prioridade de serviços da rede.
Descrição	Administrador acessa a aplicação web, consulta o endereço ip já preenchido passando os parâmetros que deseja aplicar.
Ator	Administrador.
Pré-condição	Administrador deve saber quais parâmetros deseja aplicar para o endereço desejado.
Fluxo principal	1. O *software* requisita os dados para o usuário; 2. *Software* valida os dados que o usuário digitou.
Cenário – Visualização	*Software* mostra o endereço ip com as regras aplicadas.
Pós-condição	Administrador visualizou o endereço ip com regras aplicadas.

Quadro 5 – Descrição do caso de uso UC03

No Quadro 6 apresenta-se o caso de uso "Visualizar prioridades aplicadas para rede".

Nome do Caso de Uso	Visualizar prioridades aplicadas para rede.
Descrição	Administrador visualiza regras estabelecidas para a rede.
Ator	Administrador.
Pré-condição	Administrador deve selecionar listrar regra da rede, para visualizar as regras específicas para a rede.
Pós-condição	Administrador visualizou as regras aplicadas para a rede.

Quadro 6 – Descrição do caso de uso UC04

No Quadro 7 apresenta-se o caso de uso "Visualizar *hostsonline* na rede".

Nome do Caso de Uso	Visualizar *hostsonline* na rede.
Descrição	Administrador visualiza os *hostsonline* que estão se comunicando com o *gateway* Linux no momento da requisição feita pelo administrador.
Ator	Administrador.
Pré-condição	Administrador deve selecionar listrar ips ativos para visualizar os ips que estão comunicando-se com o *gateway* Linux .
Pós-condição	Administrador visualizou as regras aplicadas para a rede.

Quadro 7 – Descrição do caso de uso UC05

No Quadro 8 apresenta-se o caso de uso "Visualizar consumo do *link* por serviço de rede".

Nome do Caso de Uso	Visualizar consumo do *link* por serviço de rede
Descrição	Administrador visualiza o consumo da largura de banda da rede no momento em que requisita-lo.
Ator	Administrador.

Pré-condição	Administrador deve selecionar listar mostrar consumo atual para visualizar quanto de banda está sendo consumida.
Pós-condição	Administrador visualizou o consumo da rede.

Quadro 8 – Descrição do caso de uso UC06

No Quadro 9 apresenta-se o caso de uso "Visualizar a largura de banda reservada por serviço de rede ".

Nome do Caso de Uso	Visualizar a largura de banda reservada por serviço de rede
Descrição	Administrador visualiza a largura de banda reservada por serviço de rede.
Ator	Administrador.
Pré-condição	Administrador deve selecionar listagem de regras para visualizar qual a largura de banda foi cadastrada.
Pós-condição	Administrador visualizou a largura de banda por serviço de rede.

Quadro 9 – Descrição do caso de uso UC07