UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CENTRO DE CIÊNCIAS DA NATUREZA - CCN
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA - DIE
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IMPLANTAÇÃO DE CONTROLE DE BANDA PARA OTIMIZAÇÃO DO USO DE REDE DE COMPUTADORES ATRAVÉS DE QoS

Nathan Franklin Saraiva de Sousa

TERESINA - PI
FEVEIRO/2005

UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CENTRO DE CIÊNCIAS DA NATUREZA - CCN
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA - DIE
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IMPLANTAÇÃO DE CONTROLE DE BANDA PARA OTIMIZAÇÃO DO USO DE REDE DE COMPUTADORES ATRAVÉS DE QoS

Nathan Franklin Saraiva de Sousa

Trabalho apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Federal do Piauí - UFPI, para demonstrar as atividades do estágio Supervisionado realizado na FAPEPI.

TERESINA -PI
FEVEREIRO/2005

DE SOUSA, Nathan Franklin Saraiva.
“Implantação de Controle de Banda para otimização do uso de rede de computadores através de QoS”/ Nathan Franklin Saraiva de Sousa:UFPI - FAPEPI , Ciência da Computação, 2005.
p.69

Trabalho de Conclusão de Curso, Curso de Bacharelado em Ciência da Computação, Teresina – PI, 2005.
Orientador: Profº Msc. Magno Alves dos Santos
Orientador de Campo: Carlos Giovanni Nunes de Carvalho

QoS
DiffServ
IntServ
Controle de Banda
HTB
.

Estagiário: Nathan Franklin Saraiva de Sousa.

Matrícula: 01N1023-6.

Ano: 2004 **Período:** 2º .

Curso: Bacharelado em Ciência da Computação.

Início do estágio: 01/06/2004.

Término do estágio: 25/02/2005.

Docente orientador: Magno Alves dos Santos.

Orientador de Campo: Carlos Giovanni Nunes de Carvalho.

Coordenador de estágio: Flávio Ferry.

Telefone para contato: (xx86) 223 - 3810.

E-mail: nathan_ufpi@yahoo.com.br

Dedicatória

Dedico este trabalho aos meus pais, Luiz Gonzaga de Sousa e Rosa Saraiva de Sousa.

Agradecimentos

- primeiramente a Deus por ter me dado saúde e paz para enfrentar os obstáculos encontrados;
- aos meus pais, que me incentivaram e foram essenciais e fundamentais não só no curso mas em toda a minha vida;
- aos meus irmãos, irmãs, tias e minha prima Nasaré que me deram apoio e acreditaram em mim em todos os momentos;
- aos amigos conquistados no decorrer do curso, como Richardson Lima, Rômulo Araújo, Rodrigo Veras e David Pereira, e aos amigos de infância pela força e companheirismo encontrados neles;
- um agradecimento especial ao colega Alan Kardec e a FAPEPI pela ajuda que pra mim foi decisiva na elaboração desse trabalho;
- ao meu professor orientador Magno Alves, ao meu orientador de campo Carlos Giovanni e a todos os professores do DIE pelo apoio teórico e pela dedicação à atividade docente.

Resumo

A Internet nos últimos anos teve um crescimento estrondoso sendo uma realidade para milhares de pessoas em todo mundo. Juntamente com esse crescimento foram surgindo novas aplicações e tipos de tráfegos como video e voz que exigem cada vez mais recursos da rede. No entanto a internet não conseguiu evoluir diante desse crescimento e dessas novas aplicações, sendo que o serviço proporcionado hoje pela internet não é adequado para esse novo cenário. É necessário então inserir serviços adicionais de forma a proporcionar uma melhor adaptação da internet. Com isso surgiu a idéia de aplicação de Qualidade de Serviço(QoS) e Controle de banda para otimização do uso de redes de computadores(não somente a internet). Pensando nisso, esse trabalho procura fazer um estudo sobre Qualidade de Serviço, abordando aspectos gerais e descrevendo as principais abordagens para QoS existentes hoje como o DiffServ e Intserv. Além disso, procurou-se aplicar controle de banda em redes de computadores, tendo o Linux como SO adotado, para ser aplicado a rede do PoP-PI e ser estendido para algumas instituições públicas.

Palavras-chaves: Internet, QoS, IntServ, DiffServ, Controle de Banda, HTB, Linux.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

Atualmente a internet(e as redes IP modernas) assumiu proporções nunca imagináveis sendo uma realidade para milhares de pessoas ao redor do mundo, integrando diversos tipos de tráfegos tais como voz, vídeo, imagens e dados. Porém em seu projeto inicial essa evolução não era prevista.

Em decorrência disso, possui vários problemas de infra-estrutura, tais como a falta de controle de erros, atrasos na transmissão por congestionamento das filas dos equipamentos e a falta de priorização de serviços e classes, tornando com isso maior necessidade de formas de processamento, tratamento e encaminhamento de informações ainda mais sofisticado.

Pode-se verificar que o serviço proporcionado hoje em dia pela Internet não é adequado para atender à demanda de aplicações avançadas[9], como videoconferências e telemedicina. Isso também atinge a qualquer rede de computadores(não especificamente a Internet) existentes hoje.

Nesse cenário, surgiu a idéia de uma solução utilizando qualidade de serviço (QoS) e controle de banda para otimização do uso de rede de computadores para tráfego de voz, dados, serviços multimídia e aplicações avançadas. Esforços em pesquisas tem sido despendido no sentido de implantar QoS nas redes de computadores e na própria Internet capaz de prover serviços com tráfego diferenciado.

Este tutorial tem como finalidade apresentar um estudo sobre Qualidade de Serviço(QoS) e implantação de controle de banda em redes de computadores, considerando as classes de serviços, perfil de tráfego e o planejamento, para ser empregado no PoP-PI, localizado na Fundação de Amparo à Pesquisa do Estado do Piauí(FAPEPI) e estendido para algumas instituições públicas como a Universidade Federal do Piauí(UFPI) e a Universidade Estadual do Piauí(UESPI). Com isso, busca-se resolver os problemas existentes nas redes de computadores dessas instituições com relação ao congestionamento e, conseqüentemente o atraso na transmissão.

O projeto foi dividido em duas partes: uma fazendo um estudo geral sobre Qualidade de Serviço e apresentando as diversas formas de abordagem de QoS tais como o DiffServ e InterServ, e outra voltada para a parte prática com a aplicação de controle de banda e propondo uma solução a ser empregada nas instituições acima citadas.

2. Qualidade de Serviço

Nesse capítulo procurou-se fazer uma visão abrangente e didática sobre Qualidade de serviço, falando o que é QoS, os parâmetros de QoS e por que QoS é tão importante.

2.1 O que é QoS

O termo Qualidade de Serviço ou QoS(Quality of Service) já foi mencionado anteriormente nesse trabalho, mas o que significa QoS. O conceito de QoS é um dos tópicos mais confusos e difíceis de se definir em redes de computadores atualmente[7]. Por isso, serão mostradas algumas definições.

- IS0[1] - De acordo com a ISO, QoS é o efeito coletivo do desempenho de um serviço, o qual determina o grau de satisfação de um usuário desse serviço [8].
- Redes de Computadores - Para redes, QoS é o desempenho de uma rede relativa às necessidades das aplicações, além de ser o conjunto de tecnologias que possibilita às redes oferecer garantias de desempenho[19].
- Sistemas Multimídias - Nesse caso, qualidade de serviço é definida como representação do conjunto de características qualitativas e quantitativas de um sistema multimídia distribuído, necessário para alcançar a funcionalidade de uma aplicação [24].

Com a união de todos esses conceitos podemos elaborar um conceito mais abstrato de Qualidade de Serviço que é algum método ou serviço que proporcione uma diferenciação de tráfego aplicando-se alguns parâmetros de qualidade de forma a oferecer garantias de desempenho à rede. Mas quando falamos em QoS surge a idéia também de Classe de Serviço(CoS).

[1] International Standard Organization, responsável por vários padrões internacionais inclusive o relacionado com a internet.

O termo CoS e QoS são praticamente sinônimos. QoS é algo mais amplo e abrangente envolvendo garantias de acordo com alguns parâmetros e CoS significa que serviços(ou tráfego) podem ser colocados em classes e ter um tratamento diferenciado. Proporcionar um tratamento diferenciado a uma determinada classe significa oferecer garantias de desempenho e portanto é como se o conceito de QoS englobasse a idéia de CoS, ou seja, quando falamos de CoS de uma forma de outra estamos falamos de QoS.

2.4 Parâmetros de QoS

Diferentes aplicações têm diferentes requerimentos. Aplicações geram tráfico em diferentes taxas e geralmente esperam que a rede seja capaz de transmitir o tráfego na taxa gerado por elas. Certas aplicações podem tolerar algum tipo de atraso na transmissão enquanto outras não, como é o caso das aplicações multimídia. Estes requerimentos podem ser representados com esses seguintes parâmetros de QoS[15]:

- Largura de banda - taxa de transmissão dos dados de uma aplicação;
- Atraso - tempo necessário para um pacote percorrer a rede, desde o emissor até o receptor;
- Jitter - variação do atraso entre pacotes consecutivos
- Perda(Loss) - porcentagem de perdas de pacotes.

Se os recursos da rede fossem infinitos, todas as aplicações poderiam transmitir com largura de banda requerida, com zero de atraso, zero de jitter e zero de perdas. Entretanto, os recursos de rede não são infinitos e em certas partes da rede haverá a falta de recursos para atender as requisições das aplicações. QoS atua diretamente controlando os recursos da rede para o tráfego das aplicações, administrando todas as requisições.

2.3 Aumento da Largura de Banda versus QoS

Existe um debate sobre a necessidade de QoS nas redes atuais, visto que a velocidade do meio físico, com o surgimento de novas tecnologias, está ficando geometricamente maior e atenderia todos usuários sem problemas[16]. Além disso,

uma questão fundamental sobre a implantação de QoS é decidir se o custo final de sua aplicação é menor do que o custo de aumentar a largura de banda em locais onde há congestionamento.

Mas é inegável que somente o aumento da banda não é suficiente para garantir a qualidade do serviço à aplicação, pois em se tratando de redes compartilhadas por múltiplos usuários e muitas vezes a longas distâncias, como é o caso da Internet, pode haver congestionamento, provocando atrasos inadmissíveis em certas aplicações sensíveis, como voz e videoconferência [16].

Porém muitos preferem aumentar a quantidade de banda disponível, usando assim a "força bruta", por ser mais abundante e barata do que gerenciá-la através QoS, por ser mais complexo e caro.

No entanto, isso só vem a ser uma solução momentânea pois não importa quanta largura de banda houver sempre surgirão novas aplicações que irão consumi-la. Um exemplo atual disso é o caso do FTP (File Transfer Protocol) que procura utilizar toda a banda disponível consumindo assim boa parte do link. Com base nisso, a Internet2[2] elegeu QoS com uma das suas principais prioridades, juntamente com a questão da segurança e do comércio eletrônico.

É válido lembrar que a introdução de QoS está cercada de falsas expectativas, geralmente comum com o surgimento de tecnologias que propõem ser inovadoras e que podem causar futuramente grandes mudanças nos serviços oferecidos pelas redes de computadores. Como exemplos dessas expectativas podemos citar:

1. QoS não compensa imperfeições na rede decorrentes de projetos mal feitos ou de situações extremas de congestionamento;
2. Não aumenta a largura de banda fazendo aparecer mais banda onde não existe;
3. QoS é injusta pois aumenta recursos para algumas classes de tráfego e diminui os recursos para outras;

[2] Nova internet que foi projetada para resolver os problemas da internet atual, adotando novas tecnologias como Ipv6 e QoS.

4. Só funciona para determinados tipos de tráfego;
5. Além disso, QoS só é necessário em redes onde ocorrem congestionamento, caso contrário a sua utilização é irrelevante.

2.4 A Internet Atual

Atualmente, a Internet está baseada na arquitetura TCP/IP que é um conjunto de padrões que especifica detalhes de comunicação, interconexão e roteamento. Essa arquitetura foi adotada devido sua interoperabilidade, sua simplicidade em relação ao modelo OSI e rapidez que os padrões são criados.

O termo TCP/IP vem dos dois principais protocolos que é o Transmission Control Protocol (TCP) e o Internet Protocol (IP). A arquitetura TCP/IP é formada por quatro camadas: **aplicação** que contém os protocolos de alto nível tais como o HTTP (protocolo de transferência de hipertexto) e o FTP (protocolo de transferência de arquivos), **transporte** que permiti uma conexão fim a fim e onde estão definidos o TCP e o UDP(User Datagram Protocol), **inter-redes** que une os protocolos da camada superior com as tecnologias de transmissão, e **interface de rede** onde estão localizados as tecnologias de transmissão utilizado pelo protocolo IP para transmitir pacotes, tais como Ethernet, ATM, Frame Relay.

A Internet assumiu dimensões gigantescas que não estava programada no seu projeto inicial e, portanto apresenta problemas a serem resolvidos.

Um exemplo disso é que o protocolo IP versão quatro (Ipv4) apresenta algumas limitações, fruto de suas simplicidade original, que limitam a implementação de QoS nas redes baseadas neste protocolo. Entre elas podemos citar, segundo [6]:

- É um protocolo sem conexão, não havendo mecanismos de controle de admissão e sem nenhuma garantia de serviço;
- A rede só fornece serviço do tipo “melhor esforço”, onde o tráfego é processado de forma rápida quando possível sem haver nenhum tipo de classificação, priorização e reserva de recursos;

- É adotado o esquema de roteamento implícito, onde cada pacote é inspecionado e analisado individualmente em cada nó de roteamento, acarretando uma sobrecarga na transmissão dos mesmos.

Essas fraquezas do protocolo IP fazem com que ocorram congestionamentos nos roteadores causando atrasos, perdas de pacotes e fazendo com que aplicações de tempo real sejam impossíveis de ser transmitidos através da Internet de forma satisfatória. O modelo do "melhor esforço" não consegue oferecer sempre um serviço de boa qualidade.

Para oferecer garantias de serviço, os serviços IP precisam ser suplementados. Para isso, é necessário adicionar algum tipo de inteligência à rede, para que ela possa diferenciar tráfego e possibilitar níveis diferentes de serviços para usuários e aplicações distintos [9]. IP e largura de banda são necessários mas não suficientes. Redes IP necessitam de mecanismos de gerenciamento ativo da largura de banda, ou em outras palavras, necessitam de QoS.

2.5 IPv6 e QoS

O IPv6 (Internet Protocol Version 6) , também conhecido como IPng (IP Next Generation) é uma evolução do IPv4 e surgiu em resposta as deficiências verificados no IPv4, tais como: número de endereços IP's, segurança e encaminhamento. Dentre as principais melhorias trazidas pela nova versão podemos destacar:

- Extensão da capacidade de endereçamento e roteamento (Endereço de 128bits, ao contrário do IPv4 com apenas 32 bits);
- Suporte a mecanismos de segurança, incluindo encriptação e autenticação;
- Mobilidade;
- Suporte a multicast e anycast;
- Seleção de rota;
- Suporte para tráfego com Qualidade de Serviço (QoS) garantida, próprio para aplicações multimídia e em tempo-real.

Essa nova versão tem embutido no cabeçalho o campo *Flow Label* e campos de prioridades que permitem que diferentes tipos de tráfego e de aplicações recebam tratamento distinto na rede. Esse tratamento inclui as definições de largura de banda constante (essencial na transmissão de vídeo), reserva de largura de banda (espaço dedicado a certo tipo de aplicações), baixa latência e perda mínima de pacotes de informação em trânsito.

Com isso, o protocolo IPv6 pode dar suporte a implantação de QoS em redes de computadores.

3. Abordagens para QoS

Essa seção apresenta as principais abordagens para implantação de QoS que estão em discussão atualmente. A comunidade acadêmica e comercial apresenta um grande interesse em relação ao tema. A IETF (Internet Engineering Task Force) procura regulamentar essas abordagens e tenta criar um padrão para provisão de QoS.

3.1 Serviços Integrados (IntServ)

A IETF criou o grupo de trabalho Integrated Services (IntServ), com o objetivo de viabilizar uma rede de serviços integrados de forma a se adaptar à nova era de comunicações multimídia que está se estabelecendo, ou seja, o fluxo de áudio, vídeo, video on demand e dados clássicos através da mesma infra-estrutura de rede.

O termo serviços integrados é empregado para designar um modelo de serviços para a Internet que inclui o serviço de melhor esforço, serviços de tempo real e serviços de compartilhamento controlado de enlace[9]. Além disso, o modelo de serviços integrados é caracterizado pela reserva de recursos.

Os níveis de qualidade de serviço especificado pelo IntServ são o *Controlled-Load Service*(Serviço de Carga Controlada) e o *Guaranteed Service*(Serviço Garantido).Eles serão apresentados a seguir, em seguida, definiremos o protocolo de reserva de recursos utilizado pelo IntServ que é o RSVP, por padrão de fato.

3.1.1 Serviço Garantido

O Serviço Garantido oferece um nível assegurado de largura de banda, um limite rígido de atraso fim a fim e uma proteção contra a perda de pacotes nas filas para os pacotes que estiverem obedecendo ao perfil de tráfego contratado [9]. É destinado a aplicações em tempo real que não toleram atrasos. Este serviço fornece limites exatos (provados matematicamente) para o atraso máximo dos datagramas na rede.

Os roteadores no caminho do fluxo devem suportar o serviço garantido e isso assegura uma largura de banda para um determinado tráfego. Qualquer pacote que se enquadre nesse tráfego terá um atraso limitado.

Os elementos intermediários descrevem o tráfego através de um condicionador de tráfego tipo balde de fichas(*token bucket)*, que computa o máximo atraso que um pacote pode sofrer nas suas filas. Somando todos os atrasos devidos a todos elementos de determinado caminho, é possível obter o máximo atraso do pacote por aquele caminho[16].

Para assegurar as especificações o serviço garantido necessita de um controle de admissão na qual pode ser utilizado qualquer protocolo de reserva de recursos, mas somente o protocolo RSVP foi especificado.

3.1.2 Serviço de Carga Controlada

O Serviço de Carga Controlada não garante um serviço quantitativo tão rígido como o serviço garantido. As aplicações têm seus fluxos tratados semelhantes ao serviço de melhor esforço em um rede levemente carregada (sem congestionamento) e têm como garantias:

- Poucos descartes de pacotes nas filas, e com isso um número muito alto de pacotes transmitidos chegarão ao destino.
- Boa parte dos pacotes terão atrasos próximos ao atraso mínimo, fazendo com que os atrasos máximos e mínimos sejam bem próximo.

Para assegurar essas garantias, a aplicação, que está requisitando o serviço, necessita fornecer aos equipamentos de rede (roteadores) uma estimativa de seu tráfego, através do parâmetro Tspec(Traffic Specification). Os pacotes que ficarem dentro das especificações terão todos os recursos necessários para garantir uma qualidade na transmissão e aqueles pacotes que ficarão fora poderão ser descartados ou sofrer atrasos.

Segundo [9] o serviço de carga controlada dá suporte a várias aplicações desenvolvidas para a internet atual, que têm seus desempenhos prejudicados com o congestionamento das redes. Algumas dessas são aplicações de tempo real e que funcionam bem em redes de carga leve, mas que tem seu desempenho

minimizado frente a uma rede carregada (congestionada). Então, o serviço de carga controlada, que simula uma rede sem congestionamento, seria útil para esses tipos de funções.

3.1.3 Protocolo RSVP (Protocolo de Reversa de Recursos)

O RSVP (Resource Reservation Protocol) é um protocolo de controle que roda sobre IP e que permite reserva de recursos entre origem e destino na camada 3 do modelo OSI, ou seja, protocolo que faz reserva de recursos fim a fim. O RSVP é utilizado por sistemas finais para requisitar à rede parâmetros específicos de QoS para as aplicações. Também é usado pelos roteadores para repassar as requisições de QoS para todos os outros roteadores(nós) intermediários e para estabelecer e manter informações de estado que possibilitam oferecer o serviço requerido. As requisições RSVP, sendo aceitas por todos os roteadores intermediários, terão como resultado a reserva de recursos feita em todos os roteadores ao longo do caminho.

Segundo [9], as características mais importantes desse protocolo são:

- O RSVP faz reservas para aplicações tanto de unidifusão como para multidifusão;
- RSVP é simplex, ou seja, faz reservas somente para fluxos unidirecionais. Para obter reservas duplex, deve-se solicitar duas reservas simplex distintas nos dois sentidos;
- O receptor é responsável por iniciar e manter as reservas para os fluxos;
- O estado das reservas no RSVP é “leve” (*soft-state*), ou seja, tem um tempo máximo (geralmente 30s) de validade depois do qual ele expira. O receptor constantemente “refresca” o estado das reservas. Isso permite que ele se adapte automaticamente a alterações no roteamento;
- O RSVP não é um protocolo de roteamento. Ele usa as rotas escolhidas por qualquer protocolo de roteamento;

- O RSVP transporta e mantém informações sobre o controle de tráfego e controle de políticas tais como: identificação dos recursos requeridos pelos usuários e as aplicações correspondentes;
- Todos os nós no meio do caminho de dados devem suportar RSVP;
- O RSVP suporta tanto o IPv4 quanto o IPv6.

As duas principais mensagens do protocolo RSVP são PATH, que é originado no transmissor e RESV, que é originado no receptor. Mensagens PATH contêm a especificação do fluxo (formato de dados, endereço destino, porta destino) e características de tráfego (***Tspec e ADSPEC***) que são informadas ao receptor para conferir se os nós de rede podem fazer a reserva requerida. Além disso, à medida que a mensagem PATH segue pela rede armazenam-se informações de roteamento reverso em todos os nós por onde passa, para que a mensagem RESV possa percorrer o mesmo caminho. Deve-se frisar que a mensagem PATH não faz a reserva de recursos, ela é feita pelo receptor através da mensagem RESV, e que o caminho da mensagem RESV é exatamente o mesmo da mensagem PATH.

Após chegar uma mensagem PATH, o receptor envia uma mensagem RESV de volta em direção ao transmissor solicitando aos elementos de rede uma reserva de recursos de acordo com os parâmetros especificados em PATH. Cada roteador (nó), entre o receptor e transmissor, pode aceitar ou rejeitar a reserva solicitada, de acordo com a quantidade de recursos disponíveis e a política de controle de admissão adotada. Se a requisição é rejeitada por algum nó, é então enviada uma mensagem de erro para o receptor e a sinalização é finalizada. Caso contrário, a reserva dos recursos é efetuada para o fluxo e o tráfego garantindo. A partir daí, os dados começam a ser transmitido pelo caminho especificado pelo RSVP.

Além disso, durante a transmissão, tanto o transmissor quanto o receptor devem continuar enviando essas mensagens, com o propósito de manter o estado da sessão, pois como já foi dito, o RSVP é baseado em “soft-state” e precisa de tempos em tempos dar um "refresh" para evitar que a sessão seja desfeita e para adaptar-se as mudanças na rede.

A figura 3.1 mostra todo o processo de sinalização realizada pelo protocolo RSVP.

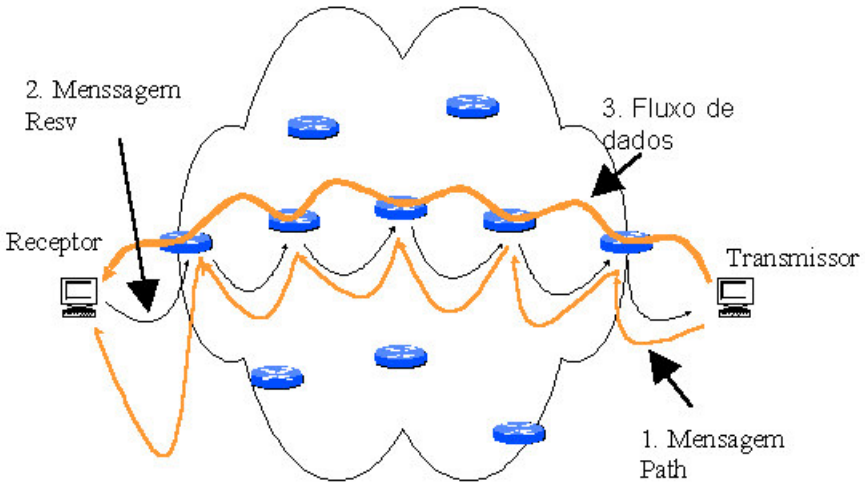

Figura 3.1: Sinalização RSVP

3.1.4 Problemas com IntServ

O modelo IntServ/RSVP é questionado atualmente pois possui problemas de escalabilidade, ou seja, não é escalável para ser utilizado em uma rede de grande porte, como a Internet. Segundo [9] os principais problemas são:

- Todos os roteadores têm que implementar RSVP, classificação, controle de admissão e escalonamento de pacotes.
- A quantidade de informação de estado aumenta proporcionalmente ao número de fluxos que um roteador tem que tratar exigindo um maior espaço de armazenamento (mais memória) e sobrecarregando os roteadores.
- Para cada fluxo deve haver sinalização a cada nó (sistema final ou roteador), aumentando o overhead[3] da rede.
- Com a troca de rota nos roteadores, o tráfego naquela rota usa o serviço do "melhor esforço" até um novo *refresh.*

[3] Custo adicional pela introdução de mais informações na rede, diminuindo o desempenho desta.

3.2 Serviços Diferenciados(DiffServ)

A partir dos problemas encontrados na implantação do IntServ, a IETF propôs uma arquitetura de serviços diferenciados chamados DiffServ(Differentiated Services) voltada para oferecer QoS na Internet.

O DiffServ não utiliza sinalização em cada nó e não faz reserva de recursos para cada fluxo de dados tornando-se, assim, escalável. O objetivo da arquitetura é fazer a agregação de fluxos em classes, chamados de Behavior Aggregate(BA), a partir da marcação dos pacotes de acordo com os tipos de serviços desejados.

Para isso, é utilizado o campo DS(Differrentiated Services) do cabeçalho IP, que é o antigo campo ToS(Type of Service) do cabeçalho IPv4 ou o campo “classe de tráfego”(Traffic Class) no caso do IPv6. Com a configuração do campo DS é possível determinar o comportamento do pacote na rede e o tipo de serviço recebido, ou seja, sua prioridade diante dos outros pacotes da rede[16].

O campo DS é formado por 8 bits e dividido em DSCP(DiffServ Code Point) e CU(Currently Undefined), como mostra a figura 3.2. O DSCP é formado por 6 bits e é responsável por especificar o tipo de comportamento, chamado de *Per-Hop Behavior*(PHB), que o pacote vai ser submetido em cada nó(roteador) da rede. O campo CU é formado por 2 bits(os bits 6 e 7) e não é padronizado pelo DiffServ, portanto deve ser ignorado.

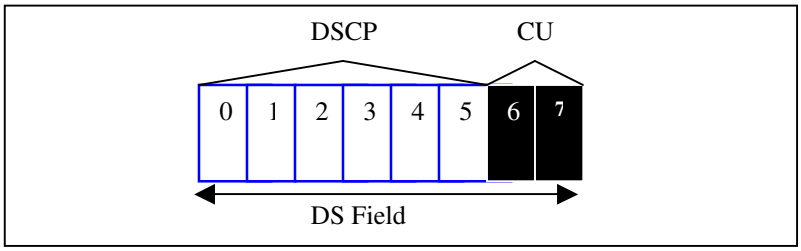

Figura 3.2: Campo DS

Além disso, o campo DSCP é capaz de identificar até 64 diferentes tipos de códigos e recomenda-se que sejam alocados em três diferentes conjuntos, conforme a tabela a seguir:

Conjunto	Códigos	Descrição
1 (32 códigos)	xxxxx0	A serem definidos por padronização do IEEE
2 (16 códigos)	xxxx11	Para uso experimental ou local (EXP/LU)
3 (16 códigos)	xxxx01	Para uso experimental ou local (EXP/LU), mas se o conjunto 1 exaurir, servirá para mais padrões

Tabela 3.1: Códigos do DSCP

3.2.1 Arquitetura DiffServ

Uma rede que oferece serviço DiffServ é chamada de Domínio DS e os roteadores habilitados são chamados de nós DS. A arquitetura DiffServ é baseada em um modelo simples, onde a classificação e mapeamento dos pacotes são feitos somente no nó de entrada do domínio DS, aos nós intermediários cabem ler o DSCP e mapear para uma classe(PHB). Os nós de borda mapeiam o valor contido no DSCP para um PHB que deve atendido em cada nó intermediário entre a origem e destino.

Com essa arquitetura, todo o processamento e complexidade ficam nos equipamentos de borda e para os nós intermediários ficam os mecanismos mais simples, temos assim um aumento da escalabilidade.

Domínios DS negociam entre si contratos de serviços (SLA – Service Level Agreements) que visam o oferecimento de garantias de QoS para as aplicações. Todos os pacotes que são enviados de um domínio para outro são policiados nos nós DS para verificar sua conformidade com os contratos. Todos os pacotes que pertencem a um mesmo BA em um Domínio DS são tratados em todos os roteadores pelo mesmo PHB[9]. Aqueles pacotes que não pertencem a um BA são tratados com o serviço padrão, o qual é equivalente ao serviço de melhor esforço.

A arquitetura lógica DiffServ é formado pela concatenação de vários Domínios DS na qual os SLA's são negociados em cada um dos nós DS entre os domínios existentes. Para estabelecer uma SLA na qual são estabelecidos parâmetros de vazão, perdas e atrasos, os usuário não negocia direto com o domínio final(a não ser que sejam adjacentes), ele negocia com o mais próximo Domínio DS até chegar o domínio final. Essa arquitetura somente provê a diferenciação de serviços em única direção do fluxo, tornado os nós de borda, ora

nós de ingresso ora de regresso sendo portanto assimétrico. A figura 4 resume todo essa arquitetura lógica:

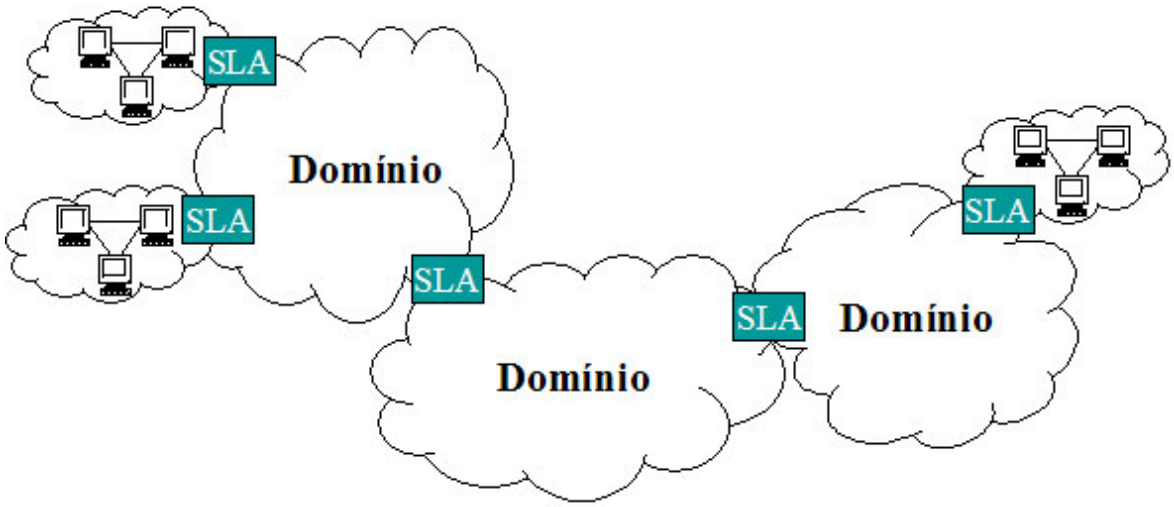

Figura 3.3: Arquitetura DiffServ

Na arquitetura DiffServ temos o condicionador de tráfego que é responsável pela medição, modelagem, policiamento e marcação do tráfego entrante no domínio de acordo com as regras especificadas na SLA[17]. A figura 5 mostra todos os estágios do condicionador de tráfego.

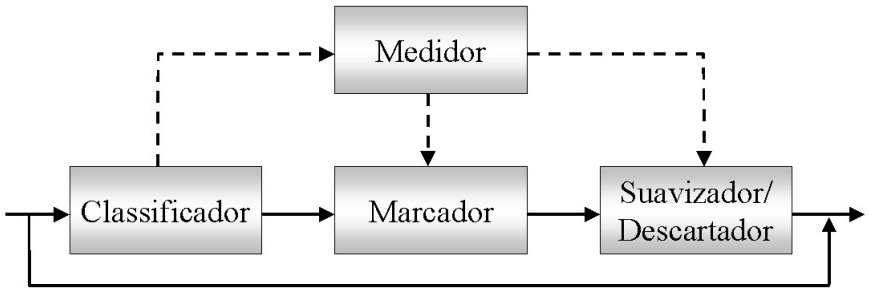

Figura 3.4: Condicionador de Tráfego

O Classificador seleciona os pacotes dentro de um fluxo através do conteúdo do campo DSCP do cabeçalho IP. Existem dois tipos de classificadores: BA(Behavior Aggregate) que é baseado somente no DSCP e o MF(Mult-Field) que é baseado em vários parâmetros do pacote: endereço de origem e destino, número da porta e no próprio DSCP. O Medidor de tráfego mede o fluxo para verificar se os pacotes selecionados pelo classificador estão de acordo com o perfil de tráfego contratado (SLA). Para isso pode ser utilizado balde de fichas. Depois da medição, o pacote pode sofrer alguns tipos de ações: suavização, descarte, marcação ou

contabilização para cobrança. O suavizador introduz um atraso no tráfego que esteja fora da PHB, para colocá-lo dentro do perfil. O descartador descarta os pacotes que estão fora do perfil com o objetivo de colocar o fluxo dentro do perfil. Já o marcador é responsável por marcar determinado código de bits no campo DSCP, para que os roteadores intermediários possam fazer a diferenciação de tráfego. Outra função é o rebaixamento dos pacotes fora do perfil.

É válido dizer que o condicionador de tráfego em um domínio DS é necessário somente nos roteadores de borda, onde o fluxo de dados é menor, e cabendo aos roteadores intermediários somente o encaminhamento dos pacotes através do PHB.

Atualmente existem dois tipos de PHB's que estão sendo padronizados pelo IETF para a implementação de Serviços Diferenciados: o Encaminhamento Expresso (Expedited Forwarding – EF) e o Encaminhamento Assegurado (Assured Forwarding – AF). Eles serão analisados a seguir. Além desses dois tipos, temos também o PHP BE(Best Effort) que é o padrão da internet e pelo qual todos os pacotes são tratados da mesma maneira com a única preocupação de alcançarem seu destino.

3.2.2 Encaminhamento Expresso (EF)

O PHB EF oferece garantia de uma taxa mínima de transmissão pré-configurada com a proposta de Serviços Diferenciados. Assim, ele pode ser utilizado para a obtenção de um serviço fim-a-fim com baixa perda, baixo retardo, baixo variação de retardo(jitter) e banda passante assegurada através de um domínio DS[12], como se os dois pontos(origem e destino) estivessem conectados por uma linha dedicada.

Segundo [16] perdas de pacotes, atrasos e jitter ocorrem devido a existência de filas nos roteadores e ao trânsito na rede. Portanto, para oferecer um serviço EF para algum tráfego agregado significa manter as filas nulas ou muito baixas. Para isso, a taxa de saída dos roteadores DiffServ devem ser maior que a taxa de entrada e eles devem ser configurados de tal forma que o tráfego agregado de entrada tenha uma taxa de saída mínima, através de uma política de filas, a fim de não eliminar os fluxos de melhor esforço.

O EF também é conhecido pelos usuários como serviço premium e é ideal para aplicações que necessitam de rapidez, transmissão constante e pouco ou nenhum erro, tais como aplicações multimídias e de tempo real.

3.2.3 Encaminhamento Assegurado (AF)

O PHB AF pode ser utilizado por um domínio DS provedor para oferecer níveis diferentes de garantias de encaminhamento de pacotes recebidos de um domínio DS cliente[9]. São definidas quatro classes AF com três níveis de precedência de descarte, gerando doze códigos diferentes de DSCP para implementação.

Os pacotes serão marcados de acordo com a requisição do cliente para cada uma das quatros classes e para cada classe serão reservados recursos, como largura de banda e *buffers.* Em caso de haver algum congestionamento na rede, o nível de precedência presente no campo DSCP determina quais pacotes serão descartados primeiros.

Com isso, o nível de garantia de encaminhamento de um pacote depende da quantidade de recursos reservados pela classe a qual ele pertence, a carga atual da classe AF e o nível de descarte do pacote.

O Encaminhamento Assegurado é voltado para clientes que precisam de segurança para seus provedores de serviços no momento que haja um congestionamento e aplicações que exigem maior confiabilidade do que a oferecida pelo serviço de “melhor esforço”[3].

Em suma, o PHB AF pode ser usado para implementar um serviço mais ou menos semelhante ao serviço de carga controlada do IntServ , porém ele é mais flexível[9].

3.2.4 Análise Crítica do DiffServ

Como foi visto o Serviço Diferenciado possui algumas vantagens tais como:

- É uma forma simples de diferenciar classes de serviços, sendo base para uma tarifação diferenciada;

- A gerência de classes de tráfego se aplica aos fluxos agregados sem utilizar um protocolo de sinalização;
- Resolução dos problemas de escalabilidade.

Muitas são as vantagens encontrados com o DiffServ, porém também são encontrados alguns problemas, como:

- Possui número limitado de classes;
- Complexidade crescente das técnicas de configuração e do dimensionamento do núcleo da rede;
- Baixa granulosidade, ou seja, não possui um bom mecanismo de distinção frente à diversidade de fluxos presentes[12];
- Esquemas de prioridade relativa garantem que uma aplicação gerando tráfego de determinada prioridade terá melhor desempenho que outra gerando tráfego de menor prioridade, podendo fazer até com que ambas as aplicações tenha um desempenho muito diferente das reais necessidades [12].

3.3 IntServ x DifServ

Os modelos IntServ e DiffServ são os que estão em maior discussão atualmente, por isso mostraremos a seguir um quadro comparativo entre eles.

	Integrated Services (IntServ)	**Differentiated Services (DiffServ)**
Fluxo	Fluxo individual	Agregação de fluxos
Estado nos roteadores	Por fluxo	Por agregação
Classificação do tráfego	Vários campos do cabeçalho IP	O campo DS(6 bits) do cabeçalho IP
Tipo de serviço	Garantias estatísticas ou determinísticas	Garantias absolutas ou relativas
Protocolo de sinalização	RSVP	Não requerido
Escalabilidade	Limitado pelo número de fluxos	Limitado pelo número de classes de serviços
Descrição da rede	Baseado nas características	Baseado no uso da classe

	do fluxo e os requerimentos de Qos	
Gerenciamento da rede	Similar as redes de comutação de circuitos	Similar as redes IP' s existentes

Tabela 3.2: Quadro comparativo entre IntServ e DiffServ

4. Implantação de Controle de Banda

Depois dessa parte teórica sobre Qualidade de Serviço, nesse capítulo estaremos iniciando a parte principal desse tutorial que mostrará a aplicação do controle de banda em redes de computadores, e ao final propondo uma solução de uso desta para otimização de redes de computadores.

4.1 Desenvolvimento e Dificuldades encontradas

Para a proposta de aplicação de QoS e controle de banda em redes de computadores, inicialmente procurou-se obter um maior conhecimento sobre o assunto através da aquisição de material sobre QoS e controle de banda. Por ser um assunto relativamente novo no âmbito comercial, mas ser algo já discutido no campo acadêmico, foi encontrado boa parte do material de consulta na Internet e em trabalhos científicos.

Depois desse primeiro contato, procurou-se iniciar a parte de configuração de roteadores que pudessem implementar QoS(ou controle de banda) e ser estendido a rede de computadores do PoP-PI.

Com relação aos roteadores, tentou-se configurar um roteador IBM 2210 serie 1s8. Depois de analisar o manual dele e pesquisar na internet verificou-se que ele implementava QoS. Partiu-se então para a sua configuração, mas depois de alguns testes e pesquisando mais sobre o modelo descobriu-se que a serie 1s8 não tinha suporte a QoS ou qualquer controle de banda.

Foi deixado de lado a parte de roteadores e buscou-se a configuração de estações de forma a proporcionar QoS aos sistemas finais. Inicialmente, tentou-se configurar estações Windows 2000 e XP, pois de acordo com a Microsoft eles têm suporte a QoS. No entanto, essa proposta teve que ser abandonada devido a alguns motivos, tais como:

- Documentação que a Microsoft dispõem é incompleta e muito vaga e não conseguiu sanar dúvidas sobre a configuração de estações;

- Tecnologia ainda muito imatura e não muito testada na prática;
- Principalmente, porque necessita de roteadores dedicados que suportam o mesmo protocolo usado pelo windows.

A partir de todas essas dificuldades e estudando mais afundo, concluiu-se que para aplicar QoS é necessário trabalhar com roteadores intermediários e isso é praticamente impossível no atual estágio da internet. Procurou-se, então trabalhar somente com o controle da rede interna (intranet) e para isso foram buscadas ferramentas e alternativas que pudessem proporcionar isso.

Nesse ponto em diante, começaremos a voltar as atenções para o controle de banda e qualidade de serviço que pudessem ser aplicados em uma rede local. Encontrou-se no Linux, um sistema operacional bem maduro em relação a QoS e controle de banda, algumas ferramentas e pacotes já bem testados e conhecidos como o traffic controller(tc), CBQ, HTB, entre outros.

Controle de banda significa proporcionar algum tipo de gerenciamento (controle) sobre o tráfego de rede de forma a proporcionar garantias (largura de banda) para determinado tipo de tráfego e com isso melhorar o desempenho da rede. A partir do momento que estamos proporcionando garantias, estamos proporcionando algum nível de qualidade de serviço, por isso quando falamos em controle de banda de uma forma de outra estamos falando de QoS. Além disso, largura de banda é um dos parâmetros de QoS.

Desse ponto em diante, o trabalhou voltou-se a aplicar controle de banda em redes locais tendo o Linux como SO adotado. Nessas próximas seções estaremos detalhando todo o processo de aplicação de controle de banda usando a ferramenta HTB(Hierarchical Token Bucket) do linux mostrando e analisando todos testes que foram feitos durante o projeto. Escolheu-se essa ferramenta por ser mais compreensível e intuitivo do que as já existentes e por ter encontrado uma boa documentação.

4.2 Ferramenta HTB

Nos sistemas operacionais Linux quando o kernel tem de enviar para a rede vários pacotes através de um dispositivo de rede, ele precisa decidir quais enviará primeiro, quais retardará e quais descartará. Utilizando diversos algoritmos o

escalonador de pacotes(scheduling) do kernel tentam ordenar ou re-ordenar os pacotes para serem enviados. O método padrão para o escalonador de pacotes é a FIFO (First In, First Out) - o primeiro que chega é o primeiro que sai, mas existem outros métodos como o SFQ (Stochastic Fair Queuing), que procura dar a cada fluxo(flow) uma chance de transmitir seus pacotes, ou o RED(Random Early Detection), que procura descartar pacotes de forma randômica ao se atingir uma determinada condição de forma a evitar que o link fique congestionado.

O kernel do linux oferece diversos recursos que permitem enviar pacotes de forma que for mais recomendado mantendo total controle no envio de pacotes. Este conjunto de recursos é denominado Queueing Discipline (qdisc), ou seja, disciplina de enfileiramento. Dentre estas regras existe uma denominada Hierarchical Token Bucket - HTB[4].

O Hierarchical Token Bucket (HTB) ou Balde de Fichas Hierárquico é uma ferramenta Linux que faz controle de banda do link de saída e propõe-se ser um substituto mais compreensível, intuitivo e rápido para o CBQ qdisc . Tanto o CBQ quanto o HTB ajudam você a controlar o uso da largura de banda no tráfego de saída em um link. Ambos permitem você usar um link físico para simular vários links lentos e enviar diferentes tipos de tráfego em diferentes links virtuais. Em ambos casos, tem-se de especificar como dividir o link físico em links simulados e decidir, através das regras pré-estabelecidas, como cada pacote será enviado e em que link virtual[10]. É importante frisar que o HTB funciona em qualquer distribuição Linux.

O HTB já vem instalado em kernels a partir da versão 2.4.20 e para trabalhar com kernels mais antigos é necessário aplicar o patch e recompilá-los. Baixe os fontes do kernel e execute **patch -p1 -i htb3_2.X.X.diff**(depende da versão do htb) para aplicar o patch. Então execute **make menuconfig;make bzImage** . Não

esqueça de habilitar QOS e HTB. Para obter a ferramenta “tc” atualizada, faça o download do arquivo em http://luxik.cdi.cz/~devik/qos/htb/v3/htb3.6-020525.tgz, descompacte-o com o comando "tar zxvf htb3.6-020525.tgz" e copie o arquivo "tc" para diretório /sbin.

O HTB utiliza o conceito de balde de fichas combinado com um sistema de classes e filtros, que permite a configuração de um controle do tráfego bastante preciso e complexo.

Para facilitar o estudo do HTB é necessário definir alguns conceitos.

Class	As classes são uma forma de se dividir o tráfego para um tratamento diferenciado.
Filter	Os filtros exercem a função de determinar em que classe um pacote será enfileirado dentro de um qdisc classful[4].
Classificador	Utilizado para identificar certos padrões e/ou características dos pacotes e fluxos permitindo a separação. Exs: U32(permite a identificação de qualquer parte de um pacote IP) e FW.

Tabela 4.1: Conceito de class, filter e classificador.

Como o HTB é um qdisc com suporte a classes, ele pode ser utilizado como um qdisc (um escalonador/shaper) ou como uma classe. Quando utilizado como classe, o htb possui apenas um parâmetro (opcional) "default" que define em qual subclasse um pacote deverá ser enfileirado caso não seja classificado por nenhum filtro[22].

Mas já como um escalonador o HTB suporta basicamente os seguintes parâmetros:

Parâmetro	O que significa
Rate	Velocidade de transmissão garantida
Burst	Tamanho máximo de bytes para rajadas
Ceil	Velocidade máxima que a classe pode assumir

[4] Qdisc que pode ter subclasses definidas pelo usuário

Tabela 4.2: Parâmetros do HTB

Para tornar o entendimento do HTB mais dinâmico, será mostrado o script do ambiente de testes 1, descrevendo como está divido as classes e explicando comando por comando.

No teste temos uma classe qdisc htb 1:0 e uma classe raiz 1:1 com suas filhas. A estrutura se assemelha com uma árvore.

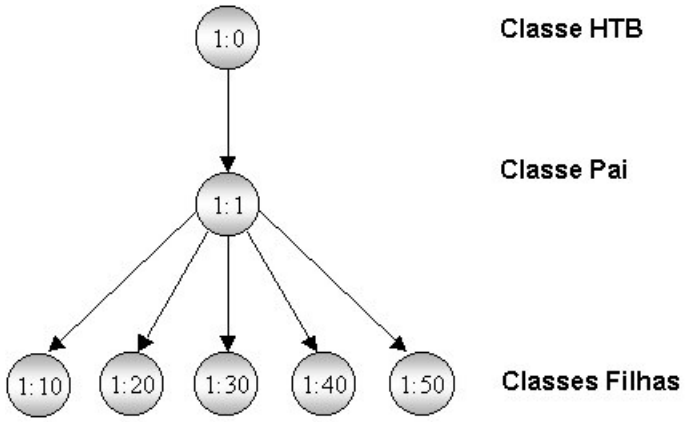

Figura 4.1: Hierarquia das classes

Veremos agora o script:

```
tc qdisc add dev eth0 root handle 1: htb default 50
```

Vincula a regra de enfileiramento HTB a interface eth0 da placa de rede e associa a ela um handle (manipulador) 1: construindo a estrutura hierárquica. O parâmetro "default 50" define que qualquer tráfego que não tiver uma classe associada será tratado pela classe 1:50.

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 1024kbit
```

Cria uma classe raiz 1:1 logo abaixo da classe htb e com taxa de transferência de 1024Kbit/s.

```
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 512kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:20 htb rate 256kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:30 htb rate 128kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:40 htb rate 64kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:50 htb rate 32kbit ceil 1024kbit
```

Nos 5 comandos acima são criadas as classe filhas da classe raiz 1:1 identificando-as com 1:10, 1:20, 1:30, 1:40 e 1:50 e com uma taxa de transferência (rate) de 512Kbit, 256Kbit, 128Kbit, 64Kbit e 32Kbit, respectivamente, e podendo atingir 1024Kbit através do parâmetro "ceil". Com essa formação, uma classe filha(irmã) pode pedir emprestado largura de banda excedente de outra classe filha.

```
tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev eth0 parent 1:30 handle 30: sfq perturb 10
tc qdisc add dev eth0 parent 1:40 handle 40: sfq perturb 10
tc qdisc add dev eth0 parent 1:50 handle 50: sfq perturb 10
```

Com esses comandos, cria-se manipuladores sfq 10:, 20:, 30:, 40: e 50: com o objetivo de fazer uma distribuição justa entre os diversos tráfegos de uma mesma classe, fazendo com que cada conexão seja distribuída igualmente na classe. O parâmetro "perturb 10" é o tempo em segundo onde será verificado a solicitação de uso de banda por conexão.

```
U32="tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 "
U32=$U32"match ip src 192.168.0.0/24 match ip dst 192.168.1.0/24"
```

Criou-se uma variável "U32" para diminuir as linhas de comando a seguir. O classificador "u32" é utilizado para identificar endereços ip' s ou para identificar portas tcp/udp dentro do pacote ip. O parâmetro "prio" identifica qual a prioridade que os pacotes terão, variando de 0 a 15, e quanto menor o "prio" maior será a prioridade do pacote.

```
$U32 match ip sport 22 0xffff flowid 1:10
$U32 match ip sport 8800 0xffff flowid 1:10
$U32 match ip sport 8080 0xffff flowid 1:20
$U32 match ip sport 8008 0xffff flowid 1:30
$U32 match ip sport 8000 0xffff flowid 1:40
```

Os comandos acima definem como o tráfego será direcionado para cada classe com base nos endereços e portas de origem e endereços de destino através dos parâmetros src, sport e dst respectivamente. Com "flowid" é estabelecida a associação destes com suas classes. Todo tráfego não incluído em um destes filtros será tratado através da classe 50. O número hexadecimal após a porta pode ser utilizado para identificar uma faixa de portas consecutivas de 32bits(é recomendável usar 0xffff).

A partir dessa breve explanação da ferramenta HTB podemos verificar seu grande poder sobre o controle de banda. Apesar de existirem equipamentos dedicados que fazem esse controle, o uso do HTB pode ser uma alternativa viável, prática e de baixo custo. Isso que foi mostrado foi uma pequena demonstração do potencial do HTB, configurações mais complexas podem ser feitas para atender diversas situações. Mais informações sobre a ferramenta podem ser encontradas nas referências bibliográficas.

4.3 Ambientes de Teste

Nessa seção serão apresentados os ambientes de testes montados e utilizados durante o projeto, que fazem controle de banda com Qualidade de Serviço(QoS) em redes de computadores. Esses ambientes foram montados para estudo e verificação da influência do controle de banda nas transmissões e para determinar qual configuração é mais eficiente.

Foram desenvolvidos dois ambientes de testes. Em ambos os cenários foram utilizados a ferramenta Linux Hierarchical Token Bucket - HTB (seção 4.2) para fazer o controle de banda do link de saída. No primeiro ambiente foram feitos alguns testes para entender o funcionamento básico do pacote HTB e estudar os seus resultados e erros. Já no segundo, foi desenvolvido um ambiente para interligar uma intranet com controle de banda à Internet.

Veremos a seguir os dois ambientes de testes.

4.3.1 Ambiente de Teste 1

Nesse primeiro ambiente foi projetado um cenário bem simplificado para fazer um estudo sobre o HTB entendendo o seu funcionamento e verificando e

analisando seus resultados. Foram utilizados dois computadores com o sistema operacional Linux - Red Hat 9.0 - com versão do kernel 2.4.20. Foi escolhida essa distribuição porque já traz módulo HTB packet scheduler ativado e a ferramenta "tc"[5] (Traffic Control), ambos necessários para a implementação do ambiente de teste.

Foi configurada uma intranet (Anexo I) entre esses dois computadores, na qual um dos computadores é o cliente, e o outro será o servidor que fará o controle de banda do link de saída para o cliente. Teremos o servidor - *Zeus* - interligado com o cliente - *Odin* - através de um cabo par trançado crossover RJ-45 com link de 10 Mbit/s.

Foi imaginado um cenário onde é necessário definir prioridades para algumas aplicações que exigem mais importância. No teste, as aplicações de maior grau de prioridade são as que atendem nas portas tcp 8800 e 22, logo depois as que atendem na porta 8080, depois as que atendem na porta 8008 e por fim as que atendem na porta 8000. Todos os demais fluxos são tratados com menor grau de importância e igualmente.

O link entre o servidor e o cliente foi simulado como um link de 1024kbit/s. Então, o HTB foi configurado de forma a garantir largura de banda de 512Kbit/s nas portas 22 e 8800, 256Kbit/s na porta 8080, 128Kbit/s na porta 8008, 64Kbit/s na porta 8000 e 32Kbit/s para o restante do tráfego.

Porta	8800/22	8080	8008	8000	Restante
Velocidade(Kbit/s)	512	256	128	64	32

Tabela 4.3: Relação entre as portas e as taxas de transmissão

Para completar o ambiente habilitou-se o Apache[6], na mesma máquina onde o HTB está configurado, ou seja, no *Zeus*, para funcionar com servidor web e para aceitar conexões nas portas referenciadas na tabela 4.3. Para isso, basta acrescentar quatro linhas no arquivo httpd.conf (geralmente encontrado em /etc/httpd/conf - Red Hat 9.0) com os seguintes comandos: Listen 8800, Listen 8080, Listen 8008 e Listen 8000(tudo isso logo após o comando Listen 80 que já

[5] Ferramenta que faz parte do pacote iproute2 e é utilizada para instruir ao kernel como tratar tráfego de rede.

existe no arquivo). Após isso, reiniciar o apache com o comando /etc/init.d/httpd restart ou /etc/init.d/httpd stop e /etc/init.d/httpd start.

Foram colocados também no diretório onde ficam as páginas do apache(no Red Hat 9.0 é em /var/www/html) arquivos com tamanhos de 512 kbytes, 1024 kbytes, 2048 kbytes, 4096 kbytes e 8192 kbytes, utilizou-se para isso arquivos do OpenOffice com os nomes 512.doc, 1024.doc, 4096.doc e 8192.doc, respectivamente. A seguir temos a estrutura desse ambiente de teste.

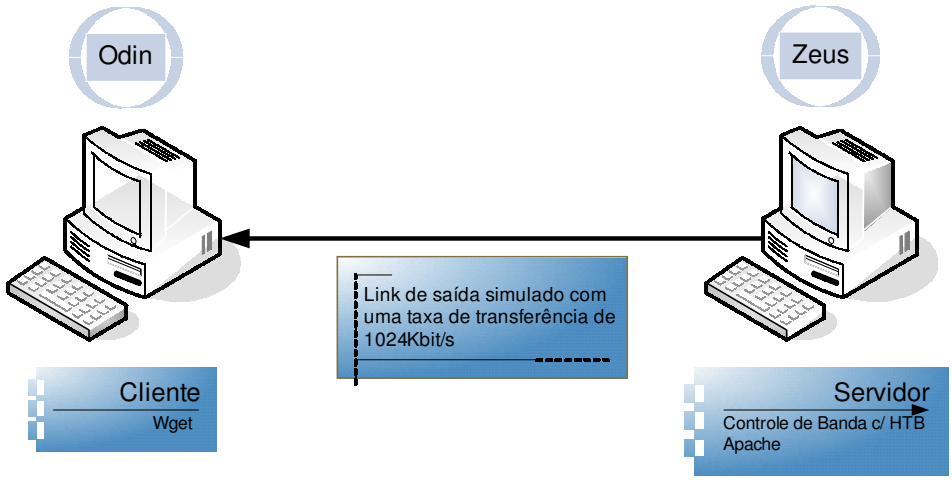

Figura 4.2: Estrutura do ambiente de teste 1

Agora para finalizar o ambiente de teste falta o principal, o script HTB. Para facilitar os testes colocou-se o script dentro de /etc/rc.d/rc.local no *Zeus* para toda vez que reiniciar o computador esse script ser ativado. Mas pode ser inserido comando por comando no terminal root.

A seguir temos todo o script original HTB que fará o controle de banda.

```
tc qdisc add dev eth0 root handle 1: htb default 50
tc class add dev eth0 parent 1: classid 1:1 htb rate 1024kbit
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 512kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:20 htb rate 256kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:30 htb rate 128kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:40 htb rate 64kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:50 htb rate 32kbit ceil 1024kbit
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
```

[6] Servidor web mais utilizado em todo mundo

```
tc qdisc add dev eth0 parent 1:30 handle 30: sfq perturb 10
tc qdisc add dev eth0 parent 1:40 handle 40: sfq perturb 10
tc qdisc add dev eth0 parent 1:50 handle 50: sfq perturb 10
U32="tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 "
U32=$U32"match ip src 192.168.0.0/24 match ip dst 192.168.1.0/24"
$U32 match ip sport 22 0xffff flowid 1:10
$U32 match ip sport 8800 0xffff flowid 1:10
$U32 match ip sport 8080 0xffff flowid 1:20
$U32 match ip sport 8008 0xffff flowid 1:30
$U32 match ip sport 8000 0xffff flowid 1:40
```

4.3.1.1 Resultados

Para verificar os resultados, no *Odin* (cliente) utilizou-se o wget que é um programa que faz download de um servidor web, define a porta que o servidor está ouvindo e mostra a taxa de transferência final dos dados. Escolheu-se esse programa por ser simples e por atender todas as exigências. Mas é preciso lembrar que os valores das taxas de transferências são apenas aproximações dos reais valores, não podendo fazer uma análise mais minuciosa.

Foram feitos vários testes para verificar o funcionamento do HTB fazendo pequenas alterações no script original. As larguras de banda especificadas no script HTB são em Kbit/s enquanto que os resultados do wget são em Kbytes/s. Para verificar se os resultados estão iguais basta dividido Kbit/s por 8, teremos então a velocidade em Kbytes/s.

Para um primeiro teste e para verificar a distribuição eqüitativa dentro de uma mesma classe, baixou-se duas vezes o arquivo 1024.doc simultaneamente pela porta 8800. Veja o resultado:

```
[root@cliente root]# wget http://192.168.1.3:8800/1024.doc; rm -f 1024.doc
--17:51:23--  http://192.168.1.3:8800/1024.doc          => `1024.doc'
Conectando-se a 192.168.1.3:8800... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,080,832 [application/msword]

100%[==========================>] 1,080,832     63.68K/s    ETA 00:00

17:51:39 (63.68 KB/s) - `1024.doc' recebido [1080832/1080832]
```

```
[root@cliente root]# wget http://192.168.1.3:8800/1024.doc; rm -f 1024.doc
--17:51:23--  http://192.168.1.3:8800/1024.doc          => ` 1024.doc.1'
Conectando-se a 192.168.1.3:8800... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,080,832 [application/msword]

100%[=========================>] 1,080,832     63.53K/s    ETA 00:00

17:51:40 (63.53 KB/s) - ` 1024.doc.1'   recebido [1080832/1080832]
```

As taxas de ambos os downloads são praticamente iguais e idênticas ao especificado no script(64KB/s = 512Kbit/s). Isso significa que o controle de banda do HTB está funcionado corretamente no link de saída.

Os testes a seguir consistem em fazer downloads simultâneos nas portas 80, 8000, 8008, 8080 e 8800 com os arquivos 512.doc, 1024.doc, 2048.doc, 4096.doc e 8192.doc, respectivamente, e que estão localizado no *Zeus*(servidor). Veja os resultados:

Teste 1 - Utilizou-se o script original especificado no projeto.

Resultados Obtidos:

```
[root@cliente root]# wget http://192.168.1.3:80/512.doc; rm -f 512.doc
--18:12:36--  http://192.168.1.3/512.doc          => ` 512.doc'
Conectando-se a 192.168.1.3:80... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 524,288 [application/msword]

100%[=========================>] 524,288       4.02K/s    ETA 00:00

18:14:43 (4.02 KB/s) - ` 512.doc'   recebido [524288/524288]

[root@cliente root]# wget http://192.168.1.3:8000/1024.doc; rm -f 1024.doc
--18:12:36--  http://192.168.1.3:8000/1024.doc          => ` 1024.doc'
Conectando-se a 192.168.1.3:8000... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,080,832 [application/msword]

100%[=========================>] 1,080,832     8.61K/s    ETA 00:00

18:14:39 (8.61 KB/s) - ` 1024.doc'   recebido [1080832/1080832]

[root@cliente root]# wget http://192.168.1.3:8008/2048.doc; rm -f 2048.doc
--18:12:37--  http://192.168.1.3:8008/2048.doc          => ` 2048.doc'
Conectando-se a 192.168.1.3:8008... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
```

```
Tamanho: 2,147,840 [application/msword]

100%[=========================>] 2,147,840     15.96K/s    ETA 00:00

18:14:49 (15.96 KB/s) - ` 2048.doc'   recebido [2147840/2147840]

[root@cliente root]# wget http://192.168.1.3:8080/4096.doc; rm -f 4096.doc
--18:12:38--  http://192.168.1.3:8080/4096.doc          => ` 4096.doc'
Conectando-se a 192.168.1.3:8080... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 4,194,304 [application/msword]

100%[=========================>] 4,194,304     31.39K/s    ETA 00:00

18:14:48 (31.39 KB/s) - ` 4096.doc'   recebido [4194304/4194304]

[root@cliente root]# wget http://192.168.1.3:8800/8192.doc; rm -f 8192.doc
--18:12:38--  http://192.168.1.3:8800/8192.doc          => ` 8192.doc'
Conectando-se a 192.168.1.3:8800... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 8,379,904 [application/msword]

100%[=========================>] 8,379,904     63.25K/s    ETA 00:00

18:14:48 (63.25 KB/s) - ` 8192.doc'   recebido [8379904/8379904]
```

Observando os resultados desse primeiro teste, pode-se perceber que as taxas de transferência(em vermelho) foram bastante aproximados as que foram especificado no script. Por exemplo, na transferência do arquivo 2048.doc pela porta 8008 na qual se enquadra na classe :30 e cuja a taxa de transferência é de 128Kbit/s(16KB/s), tem-se no teste a taxa de transferência de 15,96KB/s. Se você for atencioso, percebe-se que nas duas primeiras transferências, as taxas ficaram acima da taxa rate especificado no script. Isso ocorreu porque a classe pai 1:1 emprestou largura de banda. Na soma de todas as taxas rate(Largura de banda garantida) das classes filhas tem-se um total de 992Kbit/s, portanto a classe pai de 1024Kbit/s tem uma sobra de (1024 – 992) 32Kbits/s=4KB/s que pode ser distribuída entre as classes filhas.

Teste 2 - Nesse segundo teste foi alterado o parâmetro "sfq perturb 10" que faz uma distribuição justa entre os diversos tráfegos de uma mesma classe fazendo com que cada conexão seja atendida igualmente na classe, por "pfifo limit

5" que é o padrão do linux que faz com que em cada classe o primeiro pacote que chega é o primeiro que sai(FIFO- First In First Out) e "limit 5" é a quantidade de bytes que podem aguardar na fila, ou seja, 5 bytes. Resultados obtidos:

```
[root@cliente root]# wget http://192.168.1.3:80/512.doc; rm -f 512.doc
--18:44:40--  http://192.168.1.3/512.doc          => ` 512.doc'
Conectando-se a 192.168.1.3:80... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 524,288 [application/msword]

100%[==========================>] 524,288        5.79K/s    ETA 00:00

18:46:08 (5.79 KB/s) - ` 512.doc'    recebido [524288/524288]

[root@cliente root]# wget http://192.168.1.3:8000/1024.doc; rm -f 1024.doc
--18:44:40--  http://192.168.1.3:8000/1024.doc          => ` 1024.doc'
Conectando-se a 192.168.1.3:8000... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,080,832 [application/msword]

100%[==========================>] 1,080,832      8.41K/s    ETA 00:00

18:46:46 (8.41 KB/s) - ` 1024.doc'    recebido [1080832/1080832]

[root@cliente root]# wget http://192.168.1.3:8008/2048.doc; rm -f 2048.doc
--18:44:41--  http://192.168.1.3:8008/2048.doc          => ` 2048.doc'
Conectando-se a 192.168.1.3:8008... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 2,147,840 [application/msword]

100%[==========================>] 2,147,840     16.26K/s    ETA 00:00

18:46:50 (16.26 KB/s) - ` 2048.doc'    recebido [2147840/2147840]

[root@cliente root]# wget http://192.168.1.3:8080/4096.doc; rm -f 4096.doc
--18:44:41--  http://192.168.1.3:8080/4096.doc          => ` 4096.doc'
Conectando-se a 192.168.1.3:8080... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 4,194,304 [application/msword]

100%[==========================>] 4,194,304     32.00K/s    ETA 00:00

18:46:49 (32.00 KB/s) - ` 4096.doc'    recebido [4194304/4194304]

[root@cliente root]# wget http://192.168.1.3:8800/8192.doc; rm -f 8192.doc
--18:44:42--  http://192.168.1.3:8800/8192.doc          => ` 8192.doc'
Conectando-se a 192.168.1.3:8800... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
```

Tamanho: 8,379,904 [application/msword]

100%[=========================>] 8,379,904 64.93K/s ETA 00:00

18:46:48 (64.93 KB/s) - ` 8192.doc' recebido [8379904/8379904]

Pelos resultados pode-se perceber que somente a classe 40: (transferência na porta 8000 do arquivo 1024.doc) não teve sua taxa de transferência maior que do teste anterior, mas no geral houve uma melhora no desempenho das taxas de transmissão de todas as classes e a largura de banda excedente da classe pai(1:) foi mais bem distribuída entre as classes filhas.

Teste 3 - Teste feito com as mesmas alterações anteriores mais a introdução do parâmetro "burst 16k" na classe pai. O novo comando fica assim:

tc class add dev eth0 parent 1: classid 1:1 htb rate 1024kbit burst 16k

Com isso pretende-se ter taxas de transmissão com rajadas de 16k(burst = rajada). Como estou fazendo download de um servidor Apache e o tráfego www é em rajada, procurou-se com a introdução desse novo parâmetro melhorar o tempo de resposta, pois durante a inatividade o busrt ficará "carregando". Os resultados obtidos são:

[root@cliente root]# wget http://192.168.1.3:80/512.doc; rm -f 512.doc
--18:54:55-- http://192.168.1.3/512.doc => ` 512.doc'
Conectando-se a 192.168.1.3:80... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 524,288 [application/msword]

100%[=========================>] 524,288 5.59K/s ETA 00:00

18:56:27 (5.59 KB/s) - `512.doc' recebido [524288/524288]

[root@cliente root]# wget http://192.168.1.3:8000/1024.doc; rm -f 1024.doc
--18:54:55-- http://192.168.1.3:8000/1024.doc => ` 1024.doc'
Conectando-se a 192.168.1.3:8000... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,080,832 [application/msword]

100%[=========================>] 1,080,832 8.49K/s ETA 00:00

18:56:59 (8.49 KB/s) - ` 1024.doc' recebido [1080832/1080832]

```
[root@cliente root]# wget http://192.168.1.3:8008/2048.doc; rm -f 2048.doc
--18:54:56--  http://192.168.1.3:8008/2048.doc          => ` 2048.doc'
Conectando-se a 192.168.1.3:8008... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 2,147,840 [application/msword]

100%[=========================>] 2,147,840     16.28K/s    ETA 00:00

18:57:05 (16.28 KB/s) - ` 2048.doc'   recebido [2147840/2147840]

[root@cliente root]# wget http://192.168.1.3:8080/4096.doc; rm -f 4096.doc
--18:54:57--  http://192.168.1.3:8080/4096.doc          => ` 4096.doc'
Conectando-se a 192.168.1.3:8080... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 4,194,304 [application/msword]

100%[=========================>] 4,194,304     32.09K/s    ETA 00:00

18:57:05 (32.09 KB/s) - ` 4096.doc'   recebido [4194304/4194304]

[root@cliente root]# wget http://192.168.1.3:8800/8192.doc; rm -f 8192.doc
--18:54:58--  http://192.168.1.3:8800/8192.doc          => ` 8192.doc'
Conectando-se a 192.168.1.3:8800... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 8,379,904 [application/msword]

100%[=========================>] 8,379,904     65.00K/s    ETA 00:00

18:57:04 (65.00 KB/s) - ` 8192.doc'   recebido [8379904/8379904]
```

Houve uma relativa melhora no desempenho das transmissões com o acréscimo desse novo parâmetro(burst 16k) com uma boa distribuição da largura de banda excedente da classe pai e, em geral, aumento das taxas de transmissões, com exceção da classe :50 (classe default).

Teste 4 - Depois dos testes anteriores faremos um último teste que aproveitará as mesmas alterações anteriores mais a alteração do burst da classe pai de 16k pra 36k. Procurou-se com isso um melhor aproveitamento da largura de banda da classe pai que é de 4KB/s. Resultados obtidos:

```
[root@cliente root]# wget http://192.168.1.3:80/512.doc; rm -f 512.doc
--18:20:11--  http://192.168.1.3/512.doc          => ` 512.doc'
Conectando-se a 192.168.1.3:80... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
```

Tamanho: 524,288 [application/msword]

100%[==========>] 524,288 5.98K/s ETA 00:00

18:21:21 (5.98 KB/s) - ` 512.doc' recebido [524288/524288]

[root@cliente root]# wget http://192.168.1.3:8000/1024.doc; rm -f 1024.doc
--18:20:12-- http://192.168.1.3:8000/1024.doc => ` 1024.doc'
Conectando-se a 192.168.1.3:8000... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,080,832 [application/msword]

100%[==========>] 1,080,832 8.61K/s ETA 00:00

18:22:14 (8.39 KB/s) - ` 1024.doc' recebido [1080832/1080832]

[root@cliente root]# wget http://192.168.1.3:8008/2048.doc; rm -f 2048.doc
--18:20:12-- http://192.168.1.3:8008/2048.doc => ` 2048.doc'
Conectando-se a 192.168.1.3:8008... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 2,147,840 [application/msword]

100%[==========>] 2,147,840 16.31K/s ETA 00:00

18:22:21 (16.27 KB/s) - ` 2048.doc' recebido [2147840/2147840]

[root@cliente root]# wget http://192.168.1.3:8080/4096.doc; rm -f 4096.doc
--18:20:14-- http://192.168.1.3:8080/4096.doc => ` 4096.doc'
Conectando-se a 192.168.1.3:8080... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 4,194,304 [application/msword]

100%[==========] 4,194,304 32.32K/s ETA 00:00

18:22:21 (32.05 KB/s) - ` 4096.doc' recebido [4194304/4194304]

[root@cliente root]# wget http://192.168.1.3:8800/8192.doc; rm -f 8192.doc
--18:20:15-- http://192.168.1.3:8800/8192.doc => ` 8192.doc'
Conectando-se a 192.168.1.3:8800... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 8,379,904 [application/msword]

100%[==========>] 8,379,904 65.77K/s ETA 00:00

18:22:19 (65.07 KB/s) - ` 8192.doc' recebido [8379904/8379904]

Com esse último teste, verificou-se um melhor aproveitamento da largura excedente da classe pai. Vejamos o quadro comparativo entre os testes feitos mostrando as taxas (KB/s) de todas as classes nos testes e quanto foi utilizado da largura de banda excedente da classe pai.

	Testes			
	Teste 1	**Teste 2**	**teste3**	**Teste4**
classe 50	4,02	5,79	5,59	5,98
classe 40	8,61	8,41	8,49	8,39
classe 30	15,96	16,26	16,28	16,27
classe 20	31,39	32	32,09	32,05
classe 10	63,25	64,93	65	65,07
Uso excedente	-0,77	3,39	3,45	3,76

Tabela 4.4: Quadro comparativo dos testes

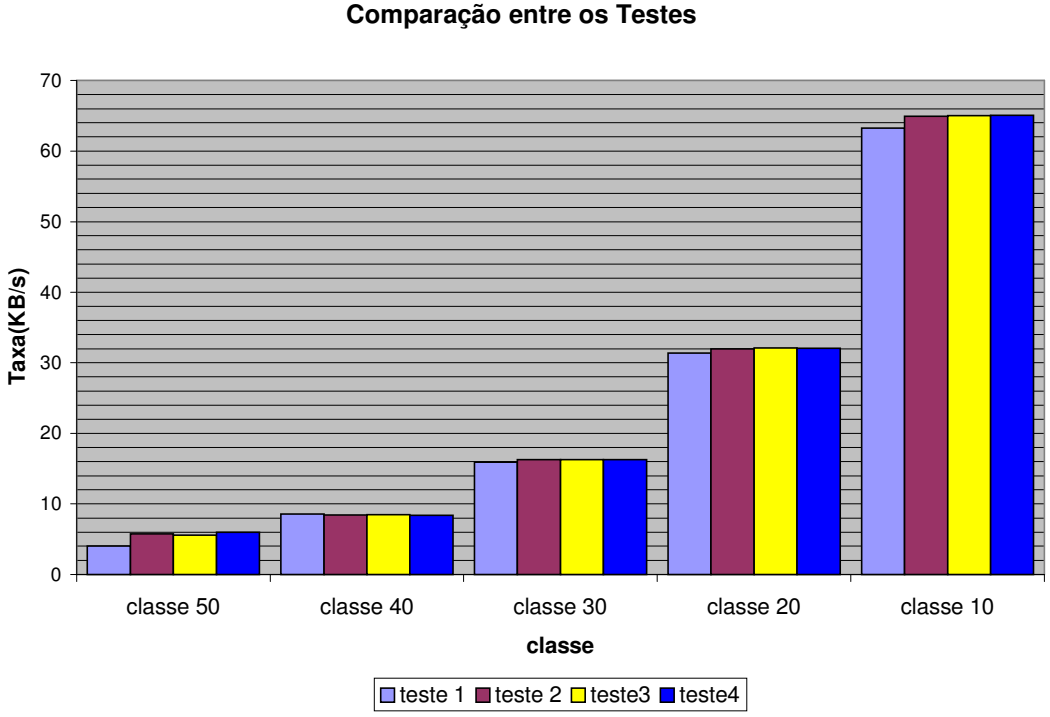

Figura 4.3: Gráfico dos testes

A partir da análise do quadro e do gráfico, pode-se perceber que no teste 1 obteve-se taxas próximos aos especificados no script, no entanto a banda excedente da classe pai não foi usado por todas as classes. No teste 2, conseguiu-se aumentar as taxas de transferências com a utilização mais distribuída da banda

excedente entre as classes filhas. Com o teste 3, a largura de banda em excesso foi mais bem utilizada do que no teste anterior e todas as classes utilizaram desse excedente.

Porém, apesar da pequena diferença das taxas dos três últimos testes, podemos concluir que no último teste (teste 4) obteve-se um melhor desempenho em relação aos outros pois houve um aproveitamento maior da largura de banda que sobrou da classe pai e essa banda foi mais bem distribuída entre as classes. Chegou a utilizar 3,76KB/s de 4KB/s excedentes. Por isso, será utilizada essa configuração no próximo ambiente de teste.

4.3.2 Ambiente de Teste 2

Depois dos resultados obtidos no primeiro ambiente de teste e das conclusões obtidas, para esse segundo ambiente procurou-se montar um intranet ligada a internet, onde o fluxo de dados vindo da internet sofria um controle de banda antes de entrar na intranet. Buscou-se dessa forma verificar o uso de controle de banda com HTB diante do fluxo da internet e a partir daí procurar uma solução para ser aplicado a redes de computadores do PoP-PI, estendendo essa solução à UFPI e UESPI.

Para isso era necessário configurar um gateway que ficasse entre a internet e a intranet, e onde seria feito todo o controle de banda utilizando o HTB.

De início, procurou-se transformar esse gateway em um roteador através do uso do NAT e do pacote Linux shorewall, para todas as máquinas da intranet ter números ip' s inválidos. No entanto todo esse processo de roteamento com o shorewall era um gargalo nos fluxos de dados. Um exemplo disso é que fora da intranet da FAPEPI consegui-se fazer downloads a 200KB/s, enquanto que quando usava o roteador os downloads do mesmo arquivo no mesmo site caiam para 40KB/s, ou seja, uma redução de 80% da velocidade. Tornando assim um ambiente impróprio para realização dos testes.

Para solucionar esse problema em vez de utilizar um roteador como gateway, ele foi configurado para ser apenas uma ponte(bridge). Para isso, foi utilizado o mesmo sistema operacional do primeiro ambiente de teste que é o linux

Red Hat 9.0 com kernel 2.4.20, por já está trabalhado com ele e por atender todas as exigências. Sendo uma ponte, os pacotes que passem por ele só atingiriam a camada 2 (camada de enlace) ao contrário do roteador. Assim pode-se ter um ambiente propicio para meus testes.

4.3.2.1 Arquitetura do ambiente

Após essa primeira etapa, montou-se todo o ambiente de testes utilizando os seguintes equipamentos e softwares:

- Computadores IBM NetVista, com 196 MB de memória RAM, processador Intel Pentium III de 800 MHz, HD 20 GB , placa ethernet VIA compatible Fast Ethernet Adaptor;
- Um Hub IBM 8224, com 24 portas;
- Roteador CISCO SYTEMS 3700 series;
- Sistema operacional Linux Red Hat 9.0, versão do kernel 2.4.20;
- HTB 3.6;
- Bridge-Utils 1.0.4.

Como já foi dito configurou-se um dos computadores para ser uma bridge (ver anexo II) e funcionar como um gateway entre a minha intranet e a internet. A intranet projetada é formado por dois computadores ligados através de um hub, e este é ligado à bridge. Esta por sua vez é ligada ao roteador Cisco que é ligado a internet através do backbone da RNP(Rede Nacional de Ensino e Pesquisa). A Figura 4.4 representa a estrutura da rede.

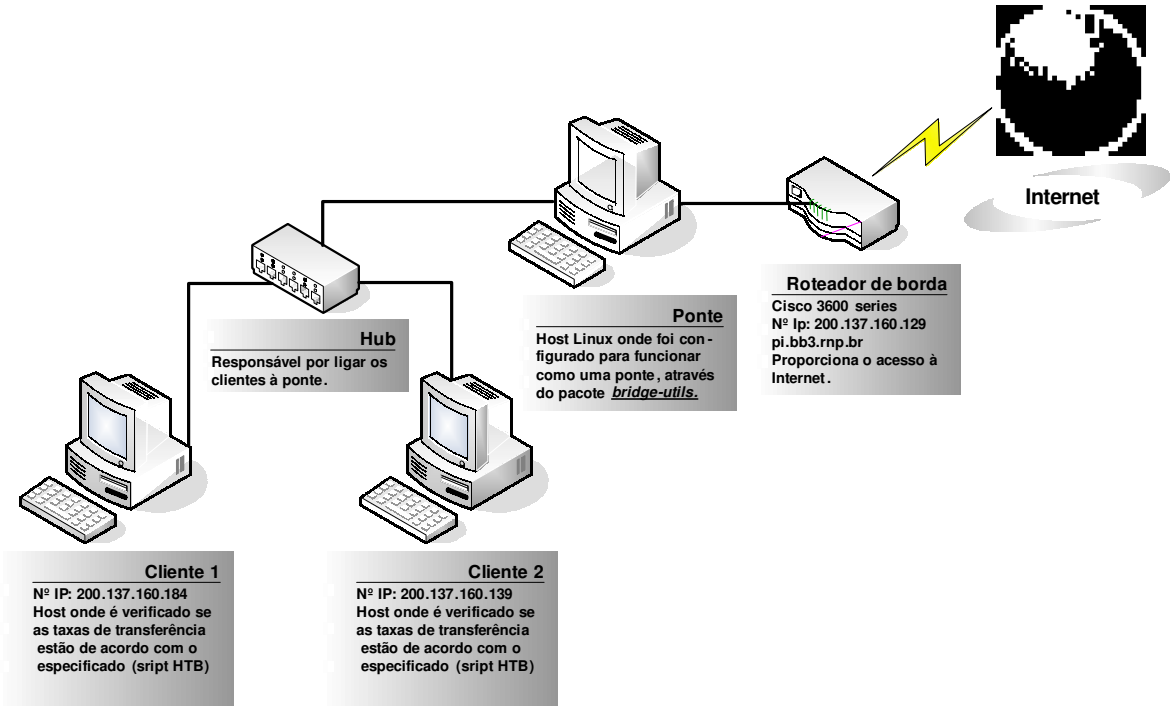

Figura 4.4: Arquitetura do ambiente de teste 2

Na configuração da ponte(Bridge) foram instaladas duas placas de rede: uma com interface para a internet(eth0) e outra com a interface para a intranet(eth1). Então tudo que vinha da internet chegava na eth0, passava para a eth1 que transmitia para as estações (cliente 1 e cliente 2) através do hub. Para o tráfego da intranet para a internet o processo era o inverso.

Como já foi mencionado no primeiro ambiente de testes, o HTB já vem incluso na versão 2.4.20 do kernel, então não foi preciso instala-lo. Como o HTB faz o controle do fluxo de saída então para implantar o controle de banda na intranet é necessário aplicar o HTB na placa eth1, que é a placa que recebe o fluxo da eth0 vindo da internet.

Para verificar o comportamento diante do tráfego da internet defini-se alguns tipos de tráfegos para o fazermos o controle de banda priorizando, através de mais largura de banda garantida, primeiramente o tráfego HTTP (www), depois o SMTP(e-mail), logo em seguida o FTP(transferência de arquivos), depois o Telnet, e por último todo o restante do tráfego.

Foi criada uma classe HTB chamada de 1:0 e uma classe default 50 para os tráfegos que não pertencem a nenhuma classe. A partir da classe 1:0 foram criada

uma classe pai 1:1 com 512Kbit/s e depois 4 classes filhas a :10, :20, :30, :40 para serem alocados os tipos de tráfegos. Veja a tabela 4.4 a seguir.

	HTTP (80)	SMTP (25)	FTP (20/21)	Telnet (23)	Restante
Classe	1:10	1:20	1:30	1:40	1:50
Taxa de transmissão	256Kbit/s	128Kbit/s	64Kbit/s	32Kbit/s	16Kbit/s

Tabela 4.4: Velocidades definidas para cada tipo de tráfego

Na tabela 4.4 vemos que para o tráfego http cuja a porta de origem é a porta 80, foi reservado uma largura de banda de 256, para o tráfego smtp cuja porta é a 25, foi reservado uma banda de 128Kbit/s, para o tráfego ftp cujas portas são a 20 e 21, temos uma reserva de 64Kbit/s, e para todos os outros tipos de tráfegos foi reservado uma largura de banda de 16Kbit/s. A hierarquia de classes se assemelha a do ambiente 1(veja figura 4.1).

Para fazermos a diferenciação de tráfego utilizei o mesmo classificador do primeiro ambiente que é o u32 que analisa os cabeçalhos IP' s e clasifica os pacotes pelo número da porta de origem e pelo número Ip de destino. E para esse ambiente, tudo que era para a rede 200.137.160.0/24 era alocado em uma das classes filhas(1:10,1:20,1:30,1:40,1:50) dependendo da porta de origem. Para ver o script htb completo veja anexo III.

4.3.2.2 Testes

Para realizar os testes simulou-se todos os tipos de tráfegos da tabela 4.4. Para isso, foram colocados arquivos de tamanhos variados no servidor web da FAPEPI, que é um servidor linux apache. Depois foi feito com que o servidor escutasse nas porta 210, 230, 250 e 8008. A partir daí, e sabendo que o servidor web fica fora da rede interna da FAPEPI, foram feitos downloads nessas portas simulando assim tráfegos FTP (Porta 210), Telnet (Porta 230), SMTP (Porta 250) e um outro tipo de trafego qualquer na internet (Porta 8008). Além disso, fiz download na porta default do apache (Porta 80) simulando tráfego HTTP. Não foram utilizados as reais portas das aplicações (Porta 21, 23 e 25) devido o apache

bloquear as portas que já são definidas, para efeito de teste foi colocado um zero ao final de cada porta(Porta 210, 230, 250).

Dessa forma o tráfego entrante na intranet vinha com a porta em que o servidor estava escutando, por exemplo, quando fazia downloads na porta 250 todos pacotes tinham como porta de origem a porta tcp 250, com isso pode-se aplicar as regras de enfileiramento do HTB simulando, assim um tráfego SMTP. Para todos as aplicações foram realizados os mesmos testes.

4.3.2.3 Resultados

Para fazer os downloads utilizei o mesmo programa do 1º ambiente, o wget. Abaixo temos alguns resultados obtidos (nos dois primeiros minutos).

Atente-se que as velocidades estão em Kbytes/s enquanto que no script estão em Kbit/s e que os resultados foram colhidos nos 2 primeiros minutos.

Mostraremos primeiramente o resultado da simulação de tráfego FTP utilizado a porta 210 e fazendo download de um arquivo de 1,8Mb.

Tempo(s)	Velocidade(KB/s)
0	9,04
15	7,86
30	7,8
45	7,78
60	7,71
75	7,7
90	7,7
105	7,7
20	7,7

Tabela 4.5: Simulação FTP

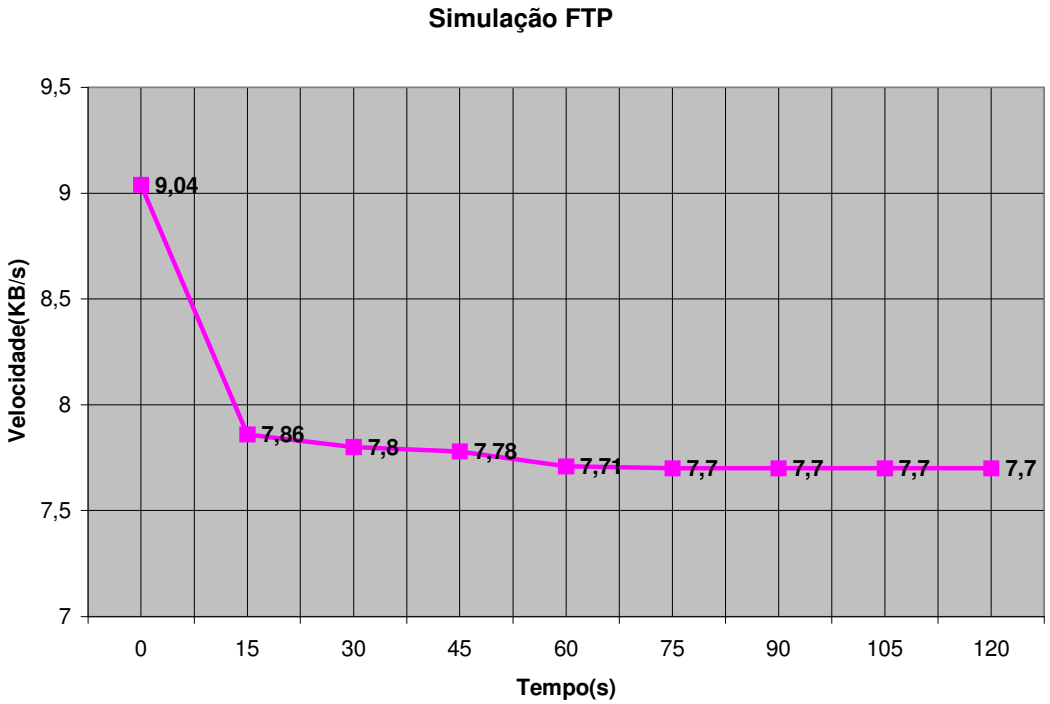

Figura 4.5: Gráfico da Velocidade X Tempo (FTP)

Agora mostraremos o resultado da simulação de tráfego Telnet(porta 230) fazendo download de um arquivo de 1,8Mb.

Tempo(s)	Velocidade(KB/s)
0	5
15	3,9
30	3,82
45	3,83
60	3,83
75	3,83
90	3,83
105	3,83
120	3,83

Tabela 4.6: Simulação Telnet

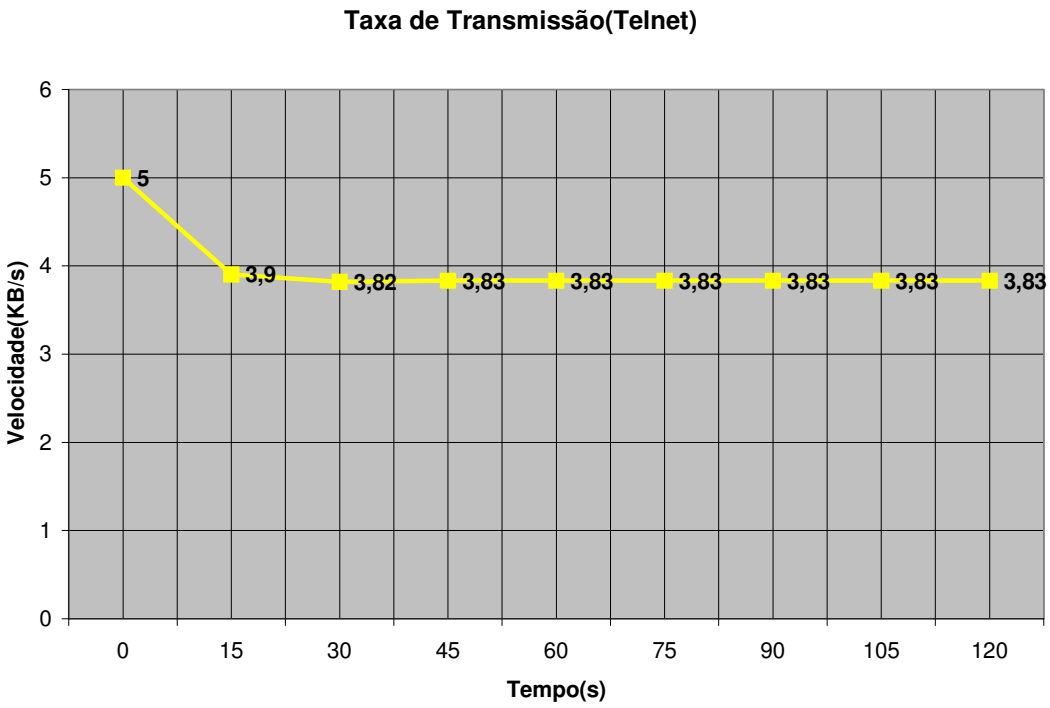

Figura 4.6: Gráfico da Velocidade X Tempo (Telnet)

Para a simulação do tráfego SMTP(porta 250) com um arquivo também de 1,8MB temos como resultado:

Tempo(s)	Velocidade(KB/s)
0	16,52
15	15,6
30	15,5
45	15,42
60	15,4
75	15,4
90	15,4
105	15,4
120	15,39

Tabela 4.7: Simulação SMTP

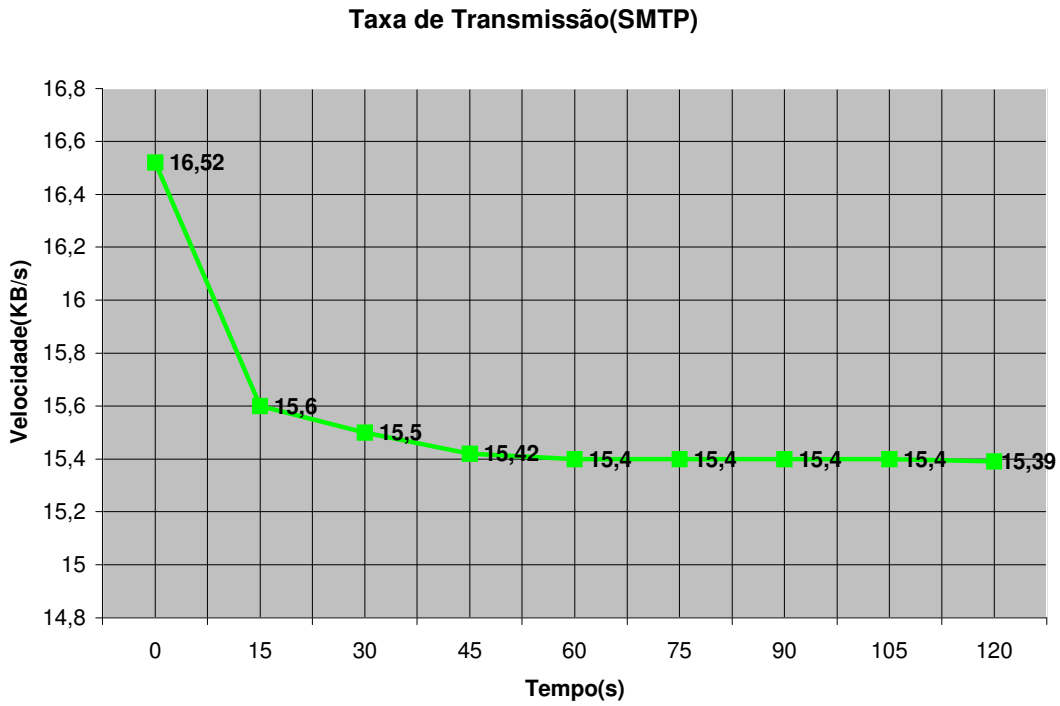

Figura 4.7: Gráfico da Velocidade X Tempo (SMTP)

Por último temos o resultado da simulação de tráfego HTTP na porta 80, fazendo download de um arquivo de 4,5 MB.

Tempo(s)	Velocidade(KB/s)
0	32,67
15	31,9
30	30,8
45	30,79
60	30,75
75	30,72
90	30,71
105	30,69
120	30,69

Tabela 4.8: Simulação HTTP

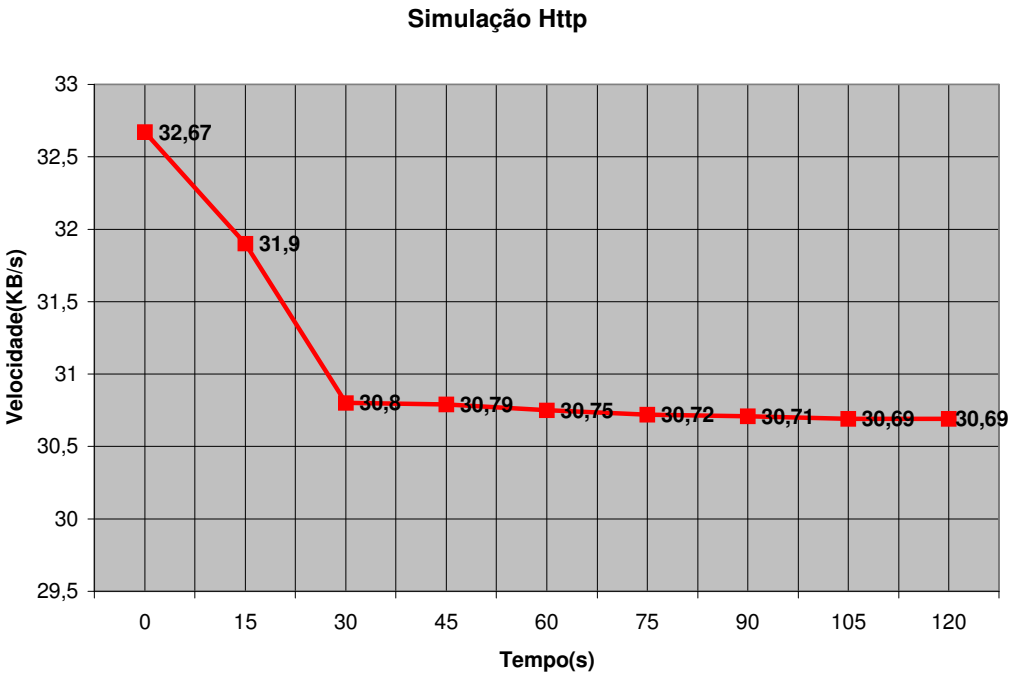

Figura 4.8: Gráfico da Velocidade X Tempo (HTTP)

A figura 4.9 mostra todos os resultados obtidos em um único gráfico para fins de observação das velocidades de downloads de cada tipo de tráfego.

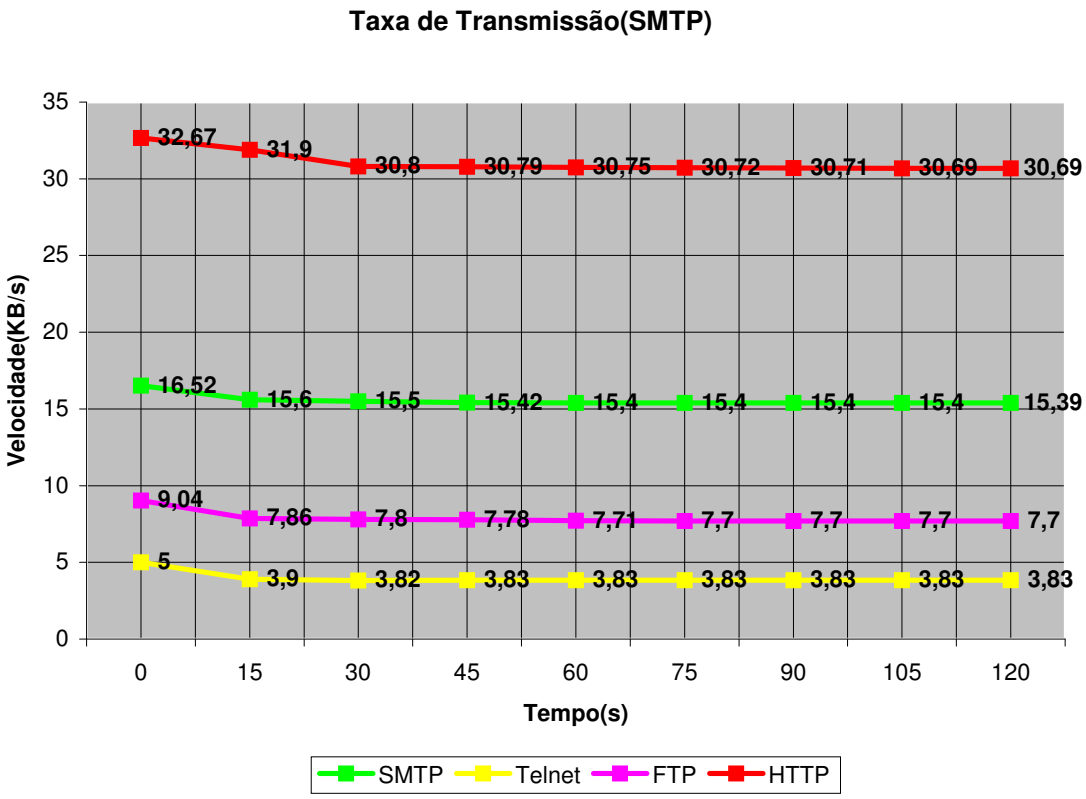

Figura 4.9: Resultados Obtidos com os 4 tipos tráfego

4.3.2.4 Análise de Resultados

Como vemos nos gráficos, nos 15 primeiros segundos houve uma queda brusca na velocidade de download e no restante do tempo a velocidade se tornou constante, isso é devido o fato do HTB está se adequado ao tipo de tráfego, realizando um certo tipo de "calibramento" no pacotes para adequa-los para a classe correspondente.

Todos as taxas de downloads ficaram próximas às especificadas no script HTB (ver anexo III). A perda de velocidade deve-se a diversos fatores como: capacidade de processamento do servidor e do cliente, perdas e erros nos pacotes

resultando em retransmissões, qualidade dos equipamentos intermediários, dentre outros.

Para ver o resultado final do *wget* desses testes, veja anexo IV. Além a ferramenta "tc" oferece recursos estatísticos da distribuição dos pacotes para cada classe, para mais informação veja anexo V.

Com esse ambiente de teste verificou-se que o controle de banda feito pelo HTB foi bem satisfatório diante do tráfego da internet fornecendo requisitos para possibilitar a aplicação desse tipo de controle em um ambiente real.

5. Solução

Depois de todos os estudos feitos durante a elaboração desse trabalho e dos testes realizados com a ferramenta HTB uma das soluções encontradas para a aplicação de controle de banda para otimização da rede de computadores do PoP-PI seria:

- Aquisição de um computador com boa capacidade de processamento(Pentium IV ou similar) e com duas placas de redes de excelente qualidade, necessários para o computador não se tornar um "gargalo" na rede;
- Configurar esse computador para ser uma ponte(bridge), porque assim pode-se espetar ela em qualquer lugar sem alterar a topologia da rede ;
- Essa ponte(bridge) teria como SO o linux (qualquer distribuição, preferível o Red Hat por ser a utilizado em todos os testes) com kernel a partir da versão 2.4.20, pois já traz o suporte ao HTB incluído;
- A ponte funcionaria como um gateway entre o firewall e rede interna;
- Na placa de rede com interface à rede interna aplicar-se-ia o controle de banda com o HTB, privilegiando os hosts e os tráfegos mais importantes fornecendo mais largura de banda em detrimento do tráfego menos importante. Então todo tráfego vindo da internet para rede interna seria tratado pelo HTB;

Uma outra solução seria a aplicação do HTB na placa de rede com interface para a rede interna do próprio firewall, sendo este Linux com kernel a partir da versão 2.4.20. Isso traria como vantagem a redução dos custos, pois não seria necessário a aquisição de um computador com duas placas de rede, entretanto causaria um maior processamento no firewall podendo acarretar um menor desempenho da rede.

Com isso, construímos um ambiente em que teríamos um controle sobre o tráfego interno, evitando congestionamentos, e consequentemente atrasos, otimizando assim o uso da rede.

6. Conclusão

É inegável que o serviço oferecido hoje pela intenet não seja adequado para tráfegos como voz e video. É nesse ponto que a introdução de QoS deixa de ser apenas um assunto de pesquisa e passa a ser uma exigência do mercado, na qual possibilitaria aos usuários a utilização de aplicações multimídia de boa qualidade e aos provedores de serviços fazer uma cobrança diferenciada de acordo com o tipo de tráfego e qualidade oferecida.

Em detrimento disso alguns fabricantes, como a Cisco, estão correndo a passos largos fabricando equipamentos com suporte a QoS para ser empregados na internet. Porém não existe nenhuma solução definitiva para a internet atual.

Como não existe ainda uma solução geral para QoS na internet, para amenizar esse problema pode-se melhorar o desempenho de redes corporativas adotando níveis de QoS. Uma das maneiras de resolver isso é através do controle de banda, na qual pode-se definir hierarquias do uso da rede, priorizando hosts, redes ou tipos de tráfegos com mais largura de banda.

É nesse sentido que este trabalho apresentou um estudo sobre QoS e uma aplicação de controle de banda em redes corporativas (rede do PoP -PI, UFPI e UESPI) de forma a proporcionar um melhor rendimento à rede. Para fazer esse controle utilizou-se o Linux por ser uma alternativa viável e de grande flexibilidade.

Espera-se ao final desse trabalho que o leitor tenha uma visão do problema existente hoje nas redes em relação a falta de QoS e da possível solução através do Linux e da ferramenta HTB.

7. Trabalhos Futuros

Como sugestão para trabalhos futuros poder-se-ia explorar mais a ferramenta HTB, pois ela é muito poderosa e não foi explorada ao máximo. Na classificação e filtração dos pacotes sugere-se trabalhar não só com a camada de enlace (camada 2) através do número da porta de origem e endereço de destino, mas também trabalhar com a camada de aplicação(camada 7) através do pacote L7_filter do Linux , que classifica os pacotes de acordo com a aplicação.

Por último, pode-se sugerir como outra linha de pesquisa fazer aplicação de QoS em cima de uma rede IPv6, utilizado todos os recursos que o IPv6 oferece para diferenciação e priorização de pacotes.

8. Bibliografia

[1] A Case for Relative Diffrentiated Services and the Proportional Differentiation Model, University of Winsconsin-Madison, 1999.

[2] ALBERTI, Antônio M., MENDES Leonardo S., Desenvolvimento de Modelo de Simulação para a Análise de Qualidade de Serviço em redes ATM, Unicamp, 2003.

[3] ALVES, Nilton, DOMINGUEZ, Kelly S. P., Modelos de Qualidade de Serviço - Aplicações em IP, CBPF, 2000.

[4] Controle de Banda com HTB, Carlos Virgílio Beltrão Lessa, disponível em: http://br-linux.org/tutoriais/001648.html

[5] DIAS, Roberto Alexandre, RIBAS, Júlio César da Costa , Uso controlado e eficiente de recursos de redes IP usando tecnologia MPLS, Cefet-SC, UFSC e W2B.

[6] DIAS, Roberto A., Serviços Diferenciados Baseado na Tecnologia MPLS em redes Heterogêneas, Cefet - SC.

[7] FERGUSON, P., HUSTON, G., "Quality of Service: Delivering QoS on the Internet and in Corporate Networks", John Wiley & Sons, 1998.

[8] SO/IEC DIS 13236,"Information Technology - Quality of Service – Framework", ISO/OSI/ODP, Julho 1995.

[9] KAMIENSKI, Carlos Alberto, SADOK, Djamel, Qualidade de Serviço na Internet, UFPE.

[10] HTB Linux queuing discipline manual - user guide, Martin Devera, disponível em: http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm

[11] MONTEIRO, José Augusto S. *et. al ,* GT - QoS: Documento de Avaliação dos Pilotos, RNP, 2003.

[12] OLIVEIRA, Renato D. V., FARINES, Jean M., Medições e Testes para Aplicações Envolvendo Mídias Contínuas em Redes IP com Serviços Diferenciados, UFSC.

[13] Projeto UCER: Uso controlado e eficiente de recursos de rede IP usando tecnologia MPLS, Roberto Alexandre Dias e Júlio César da Costa Ribas, disponível em: http://www.ucer.inf.ufsc.br.

[14] QoS for Real Time Applications over Next Generation Data Networks. Disponível em: http://www.engr.udayton.edu/faculty/matiquzz/ Pres/ QoS.pdf.

[15] Quality of Service Technical White Paper, Microsoft Corporation, 1999.

[16] ROESLER, Valter, QoS em Redes de Computadores, UNISINOS, UFRGS, 2001.

[17] SOUSA, Rayner G., FAINA, Luís Fernando, Reconfiguração Dinâmica das Classes DiffServ no Suporte a QoS face à Dinâmica da Rede, FIMES, UFU.

[18] TANENBAUM, Andrew S., Redes de Computadores, Tradução da 3ª edição, Rio de Janeiro,1997.

[19] TEITELMAN, B., HANSS, T., "QoS Requirements for Internet2", Internet2 QoS Work Group Draft, Abril 1998.

[20] TEIXEIRA, Márcio A., BARBAR, Jamil S., Gerenciamento de QoS para as Aplicações Multimídia no Sistema Final, UFU.

[21] URL: http://re.a.la/man/tc/

[22] URL: http://www.pop-pr.rnp.br/tiki-index.php? page=Roteadores+ Controle +de+Trafego

[23] URL: http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=1136

[24] VOGEL, L. A. *et al*., "Distributed Multimedia and QoS: A Survey", IEEE Multimedia, Verão1995.

Anexo I - Montando uma pequena intranet com Red Hat 9.0

Para montarmos uma pequena intranet formada por dois computadores linux interligados por um cabo crossover é necessário realizarmos algumas configurações.

Primeiramente configuraremos um computador que funcionará como servidor.

Entre em **Iniciar Aplicações** - > **Configurações do Sistema** -> **Rede**. Abrirá uma janela com as configurações de rede. Na aba **Dispositivos** clique no dispositivo eth0 e em seguida clique em **Editar**. Aparecerá uma nova janela. Desmarque a opção de obter ip automaticamente (caso essa opção esteja marcado) depois marque em definir endereços Ip' s estaticamente. Coloque o endereço Ip da máquina(por exemplo 192.168.1.3) e a máscara da sub-rede(no exemplo, 255.255.255.0). Verifique na aba **Dispositivo de Hardware** se o hardware selecionado é o eth0. Clique em **Ok** e volte para a janela anterior. Agora desative e ative novamente a eth0 e, para maior segurança, reinicie o computador.

Pronto seu servidor já está configurado. Vamos agora configurar o outro computador que funcionará como cliente.

No cliente segue o mesmo caminho para chegar na janela de rede (**Iniciar Aplicações** - > **Configurações do Sistema** -> **Rede**). Clique no dispositivo eth0 e depois em **Editar**. Da mesma forma que no servidor, você tem que desmarcar a opção obter endereços Ip' s automaticamente e marcar a opção para fixar o número ip, colocando o novo endereço (por exemplo, 192.168.1.2) e a máscara da sub-rede (255.255.255.0). Cuidado para não colocar servidor e cliente em redes diferentes. Verifique também na aba **Dispositivo de Hardware** se o hardware selecionado é o eth0. Agora clique em **OK** e volte para a janela anterior. Na aba **Servidores** coloque um novo servidor, clicando em **Novo**, e na nova janela que aparecerá digite nos campos o endereço do servidor(no exemplo, 192.168.1.3), o nome da máquina(servidor.localdomain) e os nomes alternativos(usei o default, localdomain). Como foi feito no servidor, desative e ative a placa. Pronto, verifique

se o cabo está bem conecto e já está pronta sua intranet. Para testar você pode dar um ping de uma máquina para outra com o comando (por exemplo):

"ping 192.168.1.3" , se não aparecer nenhum erro e se mostrar os tempos de respostas, isso indicará que sua intranet foi configurada correta e que já pode ser utilizada. Foi essa configuração que foi utilizada no primeiro ambiente de teste.

Anexo II – Script para montar uma bridge (ponte)

A seguir segue o script que foi utilizado para configurar uma ponte utilizando como sistema operacional o Red Hat 9.0. Para montarmos uma bridge é necessário ter duas placas de rede instalados(eth0, eth1) e ter instalado o pacote Bridge-Utils que pode ser obtido através do site Http://rpm.pbone.net/index.php3/stat/4/idpl/1607563/com/bridge-utils-1.0.4-1mdk.i586.rpm.html .

Em cada comando tem-se ao lado um comentário explicando-o. Para se tornar uma configuração persistente esse script foi colocado dentro de /etc/rc.d/rc.local.

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don'  t
# want to do the full Sys V style init stuff.
#
#
#############################################################################
############### Script para a configuração da ponte  #################
#
ifconfig eth0 down            # Desativa a interface eth0
ifconfig eth1 down            # Desativa a interface eth1
ifconfig eth0 0.0.0.0         # Coloca o endereço 0.0.0.0 a interface eth0
ifconfig eth1 0.0.0.0         # Coloca o endereço 0.0.0.0 a interface eth1
brctl addbr ponte             # Cria a interface de bridge chamada de “ponte”
brctl addif ponte eth0        # Adiciona a interface eth0 a bridge
brctl addif ponte eth1        # Adiciona a interface eth0 a bridge
ifconfig ponte up 0.0.0.0     # Ativa a ponte sem ip(0.0.0.0)
#
#############################################################################
```

Anexo III – Sript HTB para o 2° ambiente de teste

Temos a seguir o script completo HTB que foi utilizado no segundo ambiente de teste e que propunha fazer o controle de banda dos tráfegos http, smtp, ftp, telnet e ssh. Para o script se tornar persistente foi colocado dentro de /etc/rc.d/rc.local.

```
#!/bin/sh
touch /var/lock/subsys/local
#
#########################################################################
#########################  Script HTB ###################################
#
tc qdisc add dev eth1 root handle 1: htb default 50
tc class add dev eth1 parent 1: classid 1:1 htb rate 512kbit burst 36k
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 256kbit
tc class add dev eth1 parent 1:1 classid 1:20 htb rate 128kbit
tc class add dev eth1 parent 1:1 classid 1:30 htb rate 64kbit
tc class add dev eth1 parent 1:1 classid 1:40 htb rate 32kbit
tc class add dev eth1 parent 1:1 classid 1:50 htb rate 16kbit
tc qdisc add dev eth1 parent 1:10 handle 10: pfifo limit 5
tc qdisc add dev eth1 parent 1:20 handle 20: pfifo limit 5
tc qdisc add dev eth1 parent 1:30 handle 30: pfifo limit 5
tc qdisc add dev eth1 parent 1:40 handle 40: pfifo limit 5
tc qdisc add dev eth1 parent 1:50 handle 50: pfifo limit 5
U32="tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 "
U32=$U32"match ip dst 200.137.160.184"
$U32 match ip sport 80 0xffff flowid 1:10 #porta do www(htpp)
$U32 match ip sport 25 0xffff flowid 1:20 #porta do smtp(e-mail)
$U32 match ip sport 20 0xffff flowid 1:30 #porta1 do ftp
$U32 match ip sport 21 0xffff flowid 1:30 #porta2 do ftp
$U32 match ip sport 22 0xffff flowid 1:40 #porta do ssh
$U32 match ip sport 23 0xffff flowid 1:40 #porta do telnet

#########################################################################
```

Anexo IV - Resultados do wget para o ambiente de teste 2

Para os testes realizados no ambiente de teste 2, mostaremos a seguir os resultados obtidos com o uso do wget, que faz downloads e mostra as taxas de transferência.

```
[root@localhost root]# wget http://200.137.160.188:250/testeHtb/testeSmtp.zip
--18:48:02--  http://200.137.160.188:250/testeHtb/testeSmtp.zip
        => ` testeSmtp.zip.6'
Conectando-se a 200.137.160.188:250... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,900,384 [application/zip]

100%[====================================>] 1,900,384     15.35K/s    ETA 00:00

18:50:03 (15.35 KB/s) - ` testeSmtp.zip.6'   recebido [1900384/1900384]

[root@localhost root]# wget http://200.137.160.188:210/testeHtb/testeFtp.zip
--19:11:17--  http://200.137.160.188:210/testeHtb/testeFtp.zip
        => ` testeFtp.zip.1'
Conectando-se a 200.137.160.188:210... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,900,384 [application/zip]

100%[====================================>] 1,900,384      7.66K/s    ETA 00:00

19:15:19 (7.66 KB/s) - ` testeFtp.zip.1'   recebido [1900384/1900384]

[root@localhost root]# wget http://200.137.160.188:230/testeHtb/testeTelnet.zip
--19:24:24--  http://200.137.160.188:230/testeHtb/testeTelnet.zip
        => ` testeTelnet.zip'
Conectando-se a 200.137.160.188:230... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 1,900,384 [application/zip]

100%[====================================>] 1,900,384      3.82K/s    ETA 00:00

19:32:29 (3.82 KB/s) - ` testeTelnet.zip'   recebido [1900384/1900384]

[root@localhost root]# wget http://200.137.160.188:80/testeHtb/testeHttp.zip
--19:49:51--  http://200.137.160.188/testeHtb/testeHttp.zip
        => ` testeHttp.zip.3'
Conectando-se a 200.137.160.188:80... connected.
HTTP requisição enviada, aguardando resposta... 200 OK
Tamanho: 4,765,124 [application/zip]

100%[====================================>] 4,765,124     30.66K/s    ETA 00:00

19:52:23 (30.66 KB/s) - ` testeHttp.zip.3'   recebido [4765124/4765124]
```

Anexo V – Estatísticas do TC

A ferramenta **tc** permite você obter estatísticas sobre regras de enfileiramento no Linux. O fragmento abaixo foi obtido durante a simulação no segundo ambiente de teste.

```
[root@fw root]# tc -s -d qdisc
qdisc pfifo 50: dev eth1 limit 5p
Sent 1077599 bytes 8502 pkts (dropped 613, overlimits 0)

qdisc pfifo 40: dev eth1 limit 5p
Sent 1992377 bytes 1323 pkts (dropped 41, overlimits 0)

qdisc pfifo 30: dev eth1 limit 5p
Sent 2104289 bytes 1395 pkts (dropped 54, overlimits 0)

qdisc pfifo 20: dev eth1 limit 5p
Sent 1987571 bytes 1316 pkts (dropped 41, overlimits 0)

qdisc pfifo 10: dev eth1 limit 5p
Sent 42035264 bytes 28317 pkts (dropped 708, overlimits 0)

qdisc htb 1: dev eth1 r2q 10 default 50 direct_packets_stat 0 ver 3.7
Sent 49197100 bytes 40853 pkts (dropped 1457, overlimits 55367)
```

As cinco primeiras regras são filhos de HTB. O parâmetro *Sent* indica quantos bytes foram enviados pela classe, o *pkts* mostra quantos pacotes foram enviados, o *dropped* indica quantos pacotes foram rejeitados, o *overlimits* diz quantas vezes a regra atrasou um pacote. Já o *direct_packet_stat* diz quantos pacotes foram enviados por intermédio de fila direta.