

UNIVERSIDADE FEDERAL DE SANTA CATARINA

**Experimentos com
Ferramentas de Software para Controle de Qualidade
de Serviço no FreeBSD**

LEONARDO LUIZ RIBEIRO DE TOLEDO

**Florianópolis - SC
2004**



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Experimentos com Ferramentas de Software para Controle de Qualidade de Serviço no FreeBSD

Relatório do projeto apresentado como requisito para conclusão do bacharelado em Ciências da Computação da Universidade Federal de Santa Catarina.

AUTOR: LEONARDO LUIZ RIBEIRO DE TOLEDO
ORIENTADOR: DR. MARIO ANTONIO RIBEIRO DANTAS
BANCA EXAMINADORA: DR. ROBERTO WILLRICH
DR. ROSVELTER COELHO DA COSTA

Florianópolis, fevereiro de 2004

RESUMO

A Internet, desde seu início, utiliza-se de um mecanismo de filas do tipo FIFO (First In First Out) também chamado de modelo do melhor esforço. Este modelo já não atende as demandas dos novos aplicativos. É preciso oferecer garantias da qualidade do serviço na rede afim de viabilizar essas novas aplicações. Para tal, surgiram arquiteturas de Qualidade de Serviço (QoS) que, para atenderem as demandas com êxito, precisam estar presentes em todos os pontos envolvidos na transmissão dos dados.

Em uma transmissão típica via Internet é raro encontrar pontos que disponibilizam essas tecnologias. Algumas organizações adotam medidas como a reserva de recursos para determinadas aplicações ou priorização de certos tipos de tráfego. Nesse trabalho faz-se uma breve pesquisa das ferramentas de software disponíveis em um sistema operacional gratuito (FreeBSD) que implementem essas exigências. Após, são realizados alguns experimentos que visam o aprendizado da instalação e configuração de algumas das ferramentas. Ao final discute-se os resultados encontrados.

Dedicatória

**Para minha mãe, Yara;
Meu pai, Pedro;
Meu irmão, Pedro Eduardo;
Minha namorada, Gabriela; e
Meus amigos.**

Agradecimentos

A minha família e namorada pelo apoio e incentivo dado em todas as horas.

Ao professor Mário Dantas, pelo apoio e orientação.

Aos professores e colegas da UFSC que de alguma forma contribuíram com minha formação acadêmica e pessoal.

A Deus.

ÍNDICE

LISTA DE ABREVIATURAS	8
LISTA DE FIGURAS	9
1.1 Considerações Iniciais	10
1.2 Objetivos do Trabalho	11
1.2.1 Objetivo geral	11
1.2.2 Objetivos específicos	11
1.3 Motivação	11
2 QUALIDADE DE SERVIÇO (QoS)	13
2.1 Parâmetros de QoS	14
2.2 Melhor esforço (Best Effort)	15
2.3 Mecanismos IETF para QoS em Redes IP	15
2.3.1 Serviços Integrados (IntServ)	15
2.3.2 Serviços Diferenciados (DiffServ)	16
2.3.3 Multiprotocol Label Switching (MPLS)	17
2.4 Qualidade de Serviço em Redes Corporativas	17
2.5 Mecanismos provedores de QoS	18
2.5.1 Algoritmos de tratamento de filas	18
2.5.2 Mecanismos de Controle de Congestionamento	20
2.5.3 Políticas de Tráfego, Adequação e Controle de Admissão	22
3 O SISTEMA OPERACIONAL FREEBSD	25
3.1 Breve História	25
3.2 Características	26
4 PROJETO E RESULTADOS EXPERIMENTAIS	28
4.1 Ferramentas	28
4.1.1 Ferramentas auxiliares	29
4.2 Ferramentas Escolhidas	29
4.2.1 DUMMYNET	29
4.2.2 ALTQ (Alternate Queueing)	30
4.3 Experimentos	31
4.3.1 Ambiente	31
4.3.2 Experimento com DUMMYNET	31
4.3.3 Experimentos com AltQ	36
5 CONCLUSÕES E TRABALHOS FUTUROS	40
5.1 Conclusões	40
5.2 Trabalhos Futuros	41
ANEXOS	45

LISTA DE ABREVIATURAS

AS - Assured Forwarding
ATM – Asynchronous Transfer Mode
BE - Best-Effort
BSD – Berkeley Software Distribution
CBR - Constant Bit Rate
CDNR- Diff-Serv Traffic Conditioner
CoS – Class of Service (Classe de Serviço)
DiffServ – Differentiated Services
FIFO - First In First Out
IEEE – Institute of Electrical and Electronic Engineers
IETF – Internet Engineering Task Force
IPv4 – Internet Protocol version 4 (Protocolo Internet versão 4)
PHB – Per-Hop Behavior
PQ - Priority Queueing
QoS – Quality of Service (Qualidade de Serviço)
RSVP – Resource Reservation Protocol (Protocolo de Reserva de Recursos)
RTT - Round Trip Time
SLA – Service Level Agreement (Acordo por Nível de Serviço)
TCP – Transmission Control Protocol
TCP/IP – Transmission Control Protocol/ Internet Protocol
TOS – Type of Service
UDP – User Datagram Protocol
WAN – Wide Area Network
WFQ – Weighted Fair Queueing

LISTA DE FIGURAS

Figura 1 - Topologia do ambiente de experimentos	36
Figura 2 – Tempos de transmissão para cada situação	39
Figura 3 – Comparação dos tempos de transmissão com “rede livre” versus “Rede Congestionada”.....	39
Figura 4 – Comparação de taxas de transmissão ideais com as taxas obtidas	40
Figura 5 - Comparação de tempos de transmissão utilizando AltQ	44

1 INTRODUÇÃO

1.1 Considerações Iniciais

A cada dia cresce o número de serviços oferecidos via Internet. Dentre estes serviços pode-se citar: sites de conteúdo, comércio eletrônico, e-mails, troca de arquivos, streams de áudio e vídeo, entre outros. Com todas essas novas funcionalidades, a Internet está deixando de ser apenas uma ferramenta de auxílio para a produtividade da empresa e está se tornando parte do negócio.

Para transmitir todas essas informações, a Internet utiliza-se, desde seu início, do modelo do melhor esforço (Best Effort). Este modelo é baseado em filas do tipo FIFO (First In First Out) e não estabelece nenhum critério de tipo ou prioridade.

Até poucos anos, o modelo do melhor esforço conseguia atender as necessidades dos usuários da rede, mas com o surgimento de aplicações de tempo real, videoconferência e telefonia IP, este modelo se tornou obsoleto. Este tipo de aplicação necessita de uma série de requisitos para que possa funcionar corretamente. Em uma aplicação de transferência de arquivo não pode haver perda de pacotes pois isto comprometeria a integridade do arquivo, já se tratando uma conversação por VoIP, pequenas perdas são aceitas, já um atraso muito grande, inviabiliza a aplicação [17]. Tendo isto em mente pode-se afirmar que aplicações diferentes requerem tratamentos diferentes.

Com a necessidade de suprir estas deficiências, novos protocolos e padrões estão sendo adotados. Eles são chamados de Arquiteturas de Qualidade de Serviço [22]. Dentre essas arquiteturas destacam-se, por sua grande aceitação, a de Serviços Integrados (IntServ) e a de Serviços Diferenciados (DiffServ). Ambas arquiteturas contam com grupos de trabalho no IETF (Internet Engineering Task Force).

Esta tecnologia deve estar presente nos roteadores, pois são estes, os responsáveis pelo encaminhamento dos pacotes no nível inter-rede. O grande problema é que fora de uma organização o acesso aos roteadores é restrito às companhias de telecomunicação.

Existe a possibilidade de contratação de conexões especiais em que todas as características dos níveis dos serviços providos, responsabilidades entre cliente e provedor ficam firmadas em um Contrato de Nível de Serviço (SLA). Com esse tipo de contrato tem-se uma garantia da qualidade do serviço. Isso é viável para, por exemplo, conexões entre a sede de uma empresa com sua filial. Mas ao se tratar da Internet torna-se inviável devido a enorme quantidade de nós presentes. Além disso, esse tipo de contrato pode não estar disponível em todas as regiões e quando está, geralmente, é um serviço de custo elevado.

Uma arquitetura muito comum em empresas que acessam a Internet é o uso de um computador de borda (C1) que faça tradução de endereços, conhecida também como NAT. C1 fica presente na rede interna de uma organização e também na Internet. Os computadores da rede interna mandam as requisições para C1 que por sua vez repassa-as para Internet. As

respostas chegam a C1 e são encaminhadas para quem as requisitou. Nessa arquitetura todo o tráfego de e para a Internet passa por C1.

Como implementar as arquiteturas de QoS definidas pela IETF é impraticável sem o controle dos pontos intermediários, pode-se adotar o uso de ferramentas de software para classificar, restringir e/ou priorizar o tráfego de certas informações no computador de borda. Pode se dizer, então, que estas são ferramentas para controle de qualidade do serviço, QoS.

Paralelo a isto está o sistema operacional FreeBSD, desenvolvido pela Universidade da Califórnia, Berkeley, e que vem se popularizando cada vez mais. Considerado uma referência na implementação dos protocolos da família TCP/IP, o sistema é têm uma implementação robusta, focada para a área de redes de computadores. Não exige muito poder de processamento e é compatível com os SO's da família Linux. Essas características o torna uma alternativa muito interessante para prover soluções da área de redes.

A proposta deste projeto é um estudo das ferramentas de software disponíveis e que vêm sendo utilizadas no sistema operacional FreeBSD com fins de prover qualidade de serviço.

1.2 Objetivos do Trabalho

1.2.1 Objetivo geral

Esse projeto tem como objetivo aprofundar conceitos no que diz respeito a qualidade de serviço (QoS) em redes de computadores em seu âmbito geral e também a exploração prática desse tema através de experimentos em um sistema operacional gratuito utilizando ferramentas de software disponíveis.

1.2.2 Objetivos específicos

Realizar um estudo aprofundado dos recursos do sistema operacional FreeBSD no que diz respeito a parte de conectividade utilizando o protocolo TCP/IP e aplicação de parâmetros de QoS em suas configurações.

Verificar a utilização de ferramentas softwares disponíveis para o sistema operacional FreeBSD que permitam implementar mecanismos de controle de tráfego para prover QoS em redes IPv4 em computadores de borda.

1.3 Motivação

Qualidade de serviço não é um assunto novo e já vem sendo amplamente discutido por profissionais da área de redes de computadores. Muitas dessas discussões, porém, se desenvolvem somente na teoria e pouco se faz na prática.

A maior motivação desse trabalho é extrapolar a teoria e observar aplicações práticas do assunto enquanto, paralelamente, aprende-se a trabalhar em uma plataforma robusta e de grande aceitação no âmbito das soluções e serviços de conectividade e redes de computadores.

O alto custo de links de comunicação por parte das empresas de telecomunicações também estimulam o desenvolvimento do assunto pois com os conhecimentos adquiridos pode-se melhorar a qualidade de serviços ao invés da contratação de novos serviços.

2 QUALIDADE DE SERVIÇO (QoS)

Já tem algum tempo que o modelo do melhor esforço não atende mais as necessidades de todas as aplicações que se utilizam do protocolo IP. É preciso fornecer mecanismos à rede para que esta possa atender as novas necessidades das aplicações, oferecendo uma boa qualidade nos serviços oferecidos.

Qualidade de serviço é algo difícil de definir. Em geral, assume significados diferentes para pessoas distintas. A pouco tempo, QoS tinha conotações voltadas para algumas tecnologias de rede, principalmente a ATM, porém o termo já está sendo mais amplamente utilizado. Hoje já se fala em QoS em sistemas de arquivo, gerência de memória, além de outros.

Em um sistema multimídia distribuído a qualidade de serviço pode ser definida como “a especificação qualitativa e quantitativa dos requisitos de uma aplicação que um sistema multimídia deveria satisfazer a fim de obter a qualidade desejada” [12]. Em redes de computadores, QoS é utilizado para definir tanto o desempenho de uma rede relativa às necessidades das aplicações, quanto ao conjunto de tecnologias que possibilitam à rede oferecer garantias de desempenho.

Uma discussão surge quando se trata QoS em redes de alto desempenho. De um lado, estão os que acreditam que com as novas tecnologias (fibras óticas e WDM) a taxa de transmissão será abundante e barata, então, todas as aplicações se resolverão com melhor esforço. Do outro lado, diz-se que uma alta taxa de transmissão não elimina a necessidade de QoS. Não importa o quão alta é a taxa de transmissão, surgirão sempre novas aplicações consumi-la e assim, mecanismos para prover QoS serão sempre necessários [10].

Novas aplicações têm a tendência de requererem grandes taxas de transmissão o que provocará um aumento de carga nos backbones Internet provocando um certo congestionamento. De fato, existe uma relação entre as aplicações e a rede. As aplicações estimulam cada vez mais melhorias na rede, e a rede, a medida que surgem melhorias, estimulam o surgimento de novas aplicações. Esta relação permite supor que no futuro as aplicações exigiram redes com alto desempenho e com qualidade.

São inúmeras as definições de qualidade de serviço encontradas na literatura, dentre elas pode-se citar:

“Qualidade de Serviço é definida como a capacidade da rede de prover serviço de encaminhamento de dados de forma consistente e previsível” [17]

De maneira semelhante Ferguson [6] definiu QoS como: “Capacidade da rede de fornecer tratamento especial a certos tipos de tráfego previsivelmente”.

“Habilidade de um elemento da rede, seja uma aplicação, host, roteador ou outro dispositivo, ter algum nível de garantia que seu tráfego e exigências de serviços podem ser satisfeitas.” [8]

“QoS é um termo que freqüentemente possui duplo significado: refere-se ao desempenho de uma rede em relação as necessidades de uma aplicação e ao conjunto de tecnologias que permitem a rede satisfazer estas garantias de desempenho” [20]

“Termo empregado para definir os parâmetros específicos necessários para uma determinada aplicação do usuário. Estes parâmetros de serviço podem ser definidos em termos de largura de banda, latência e jitter, visando que a aplicação possa obter uma melhor qualidade ao longo da rede” [5]

Neste trabalho, serão analisadas ferramentas que de alguma forma contribuam com a diferenciação do tratamento do tráfego para permitir melhor utilização da rede e garantir a funcionalidade de aplicações mais críticas. Em outras palavras, ferramentas que se opõem a idéia do melhor esforço serão consideradas ferramentas de QoS.

2.1 Parâmetros de QoS

Ao proporcionar garantias de transmissão para certos tipos de serviço, pode se dizer que se está provendo QoS. Deve-se, então, definir alguns parâmetros que serão utilizados como requisitos para uma aplicação que deseja uma transmissão com QoS [7]. Dentre esses parâmetros destacam-se:

- Atraso fim-a-fim: É o tempo despendido entre o envio de um pacote do emissor, através da rede, até o receptor. Em um roteador, o atraso é o tempo entre a recepção de um pacote e sua transmissão, também referido como atraso de propagação [13]. Um atraso grande prejudica o bom funcionamento dos protocolos de transporte, como o TCP. Algumas aplicações (multimídia por exemplo) exigem o cumprimento de níveis máximos de retardo para funcionar adequadamente.
- Variação do atraso (jitter): É a variação no atraso fim-a-fim em redes de pacotes. Variações muito acentuadas do retardo prejudicam a qualidade do serviço oferecido a algumas aplicações. Em aplicações multimídia esse efeito pode ser percebido observando distorção ou fragmentação da mídia na recepção
- Largura de banda: É uma medida de capacidade de transmissão de dados ao longo do tempo, ou seja, a máxima taxa de transmissão de dados que pode ser sustentada entre dois pontos finais. Além dos limites físicos (tecnologia utilizada) a largura de banda é limitada também pela quantidade de fluxos que compartilham a utilização de determinados componentes da rede.

- **Confiabilidade:** Como uma propriedade dos sistemas de transmissão, pode ser vista como a taxa de erros do meio físico. Um dos principais componentes para expressar a confiabilidade é o roteamento, que pode atrasar, alterar a ordem ou mesmo descartar pacotes.

Um serviço com qualidade pode ser visto como aquele que provê baixo atraso e variação do atraso, grande quantidade de banda e muita confiabilidade. Quando se refere a QoS na Internet, no entanto, a questão diz respeito à diferenciação em uma ou mais dessas métricas de qualidade para uma determinada categoria de tráfego. Na prática, os provedores e operadoras de telecomunicação baseiam a qualidade dos serviços oferecidos no preço que o cliente pode pagar, quanto maior a qualidade exigida pelo cliente, mais ele terá que pagar.

2.2 Melhor esforço (Best Effort)

A Internet atual utiliza um modelo de serviço de melhor esforço [11], que significa que todos os usuários e aplicações têm o mesmo tratamento nos roteadores no caminho entre origem e destino dos pacotes. Em situações de congestionamento, roteadores guardam pacotes em filas na ordem estrita de chegada (FIFO). Quando a capacidade da fila se esgota, os pacotes são simplesmente descartados. Esse modelo apresenta uma grande simplicidade e robustez, que foi o motivo do sucesso da Internet. No entanto, não permite o desenvolvimento de aplicações avançadas e a diferenciação de serviços entre usuários.

2.3 Mecanismos IETF para QoS em Redes IP

A IETF tem dado grande atenção à introdução de QoS na Internet e propôs alguns modelos de serviços e mecanismos para atenderem essa demanda. Entre eles, com mais destaque estão: Arquitetura de Serviços Integrados, Arquitetura de Serviços Diferenciados e MPLS apresentados a seguir. Uma característica comum deste tipo de arquitetura é o fato delas proverem QoS fim-a-fim¹. Isto significa que é preciso que todos os roteadores entre origem e destino implementem a arquitetura.

2.3.1 Serviços Integrados (IntServ)

A Arquitetura de Serviços Integrados (IntServ) [2] sugere uma extensão para a atual arquitetura da Internet de modo a oferecer novos serviços. O foco principal de IntServ são as aplicações de

¹ QoS fim-a-fim é a habilidade da rede em prover o serviço requerido por um tráfego específico de uma extremidade a outra da rede (CISCO, 1999)

tempo real, que necessitam de garantias rígidas de QoS para funcionar corretamente. Para isso, foram propostas duas novas classes de serviços, além do serviço de melhor esforço existente:

- Serviço garantido (Guaranteed Service): [16] fornece limites rígidos aos quais os pacotes estarão condicionados nos roteadores, garantindo assim, um limite do atraso e da taxa de transmissão.
- Serviço de Carga Controlada (Controlled Load Service): [21] fornece um fluxo de dados com QoS muito próxima que o mesmo fluxo receberia de uma rede não sobrecarregada. Usa, porém, um controle de admissão para assegurar que este serviço será recebido mesmo que a rede esteja congestionada.

Esse modelo propõe que os recursos (principalmente dos roteadores) sejam disponibilizados de forma a atender às necessidades das aplicações. Para tal deve-se fazer uma reserva de recursos além de um controle de admissão. A reserva de recursos pode ser implementada de forma estática ou dinâmica. O uso de um protocolo de sinalização deve ser feito. O RSVP é um exemplo desse protocolo. Esses procedimentos geram tráfego extra na rede e aumentam proporcionalmente de acordo com o número de fluxos utilizados pelas aplicações. Este fato inviabiliza o uso desta arquitetura em ambientes com grande quantidade de aplicações, ou seja, é uma arquitetura pouco escalonável.

2.3.2 Serviços Diferenciados (DiffServ)

A arquitetura Serviços Diferenciados (DiffServ) [1] se destaca pela sua escalonabilidade que pretende ser obtida através da agregação de fluxos e separação das funções dos roteadores de borda e de núcleo nas grandes redes de backbone. As redes que implementam serviços diferenciados são chamadas Domínios DS.

Domínios DS negociam entre si contratos que visam o provimento de garantias mínimas de QoS para as aplicações dos usuários. Todos os pacotes que fluem de um domínio para outro são fiscalizados (policiados) nos roteadores de borda para verificar sua conformidade com os contratos.

No centro da rede, os roteadores simplesmente encaminham os pacotes para os seus destinos, oferecendo algumas garantias de QoS a determinados pacotes. Ou seja, pacotes distintos podem ter tratamentos distintos nos roteadores, para sua aderência a seus requisitos de QoS. Esse tratamento específico de encaminhamento é chamado de PHB (Per-Hop Behavior).

A combinação do PHB no centro da rede com as regras de policiamento na borda, permitem a criação de vários serviços em uma rede de Serviços Diferenciados. Existem dois padrões de PHB no IETF:

- Encaminhamento Expresso (EF);
- Encaminhamento Assegurado (AF)

O PHB AF pode ser utilizado por serviços que necessitam de garantias não muito rígidas, para obter diferenciação (preferência) aos seus pacotes fluindo na rede.

Por outro lado, o PHB EF define garantias mais rígidas de QoS para aplicações muito sensíveis a variações de características temporais da rede. Apesar de ser escalonável, DiffServ não oferece a garantia rígida de recursos para todos os fluxos, como o IntServ. As reservas de recursos são feitas para agregações, ou seja, grandes conjuntos de fluxos. Um fluxo individual pode não atingir as suas necessidades em termos dos parâmetros de QoS, como largura de banda e atraso. Esse tipo de QoS algumas vezes é chamado de Classes de Serviço (CoS). Neste caso, garantias são obtidas somente através de uma correta provisão dos recursos da rede, algo que não é fácil de ser alcançado [10].

2.3.3 Multiprotocol Label Switching (MPLS)

Quando um roteador recebe um pacote ele faz uma busca na sua tabela de roteamento e então, baseado no endereço IP do pacote, ele decide para onde enviá-lo. Essa busca pode levar bastante tempo, dependendo do tamanho da tabela de cada roteador. MPLS rompe com esse paradigma, usando um rótulo de tamanho fixo a partir do qual o roteador decide por onde enviar os pacotes.

MPLS [15] é na realidade a padronização de várias implementações existentes no mercado da técnica de encaminhamento de pacotes baseado em rótulos (label switching). Essa forma de encaminhamento proporciona algumas vantagens em relação a maneira tradicional, como:

1. melhor desempenho no encaminhamento de pacotes;
2. criação de caminhos (chamados de LSPs -Label Switched Paths) entre roteadores,
3. possibilidade de associar requisitos de QoS baseados no rótulo carregado pelos pacotes.

2.4 Qualidade de Serviço em Redes Corporativas

Habilitar QoS requer a cooperação de todos os níveis da rede, desde o topo até a base, bem como de todos os elementos da rede de fim-a-fim. Qualquer garantia de QoS será no máximo tão boa quanto o elo mais fraco na “cadeia” entre origem e destino [18].

Em redes corporativas, geralmente, tem-se a contratação de um link dedicado com alguma companhia de telecomunicação e/ou provedor de Internet. São vários o tipos de contrato, seus valores variam de acordo com a tecnologia utilizada e taxa de transmissão disponibilizada. O contrato estabelece uma taxa máxima de transmissão para envio (upload) e uma taxa máxima para recepção (download), números IP válidos além de outros parâmetros. A esse tipo de contrato dá-se o nome de SLA (Service Level Agreement).

Com esse tipo de acesso, não se tem garantia de QoS utilizando arquiteturas fim-a-fim. Para fora da corporação a Internet é vista como uma grande nuvem onde persiste o modelo do melhor esforço. Sem garantias é possível controlar QoS de duas maneiras [13]:

- **Reserva de Recursos:** os recursos da rede são divididos de acordo com os requisitos da aplicação, e sujeitos à política de administração de largura de banda.
- **Priorização:** o tráfego da rede é classificado e os recursos de rede são divididos de acordo com critérios de políticas de administração da largura de banda. Para habilitar QoS os mecanismos de classificação dão tratamento preferencial a aplicações identificadas como tendo requisitos mais exigentes.

Essas duas maneiras podem ser implementadas juntas ou separadas acomodando diferentes exigências operacionais em diferentes contextos de redes.

2.5 Mecanismos provedores de QoS

Os mecanismos usados para prover QoS atuam de duas maneiras: preventiva e reativa. Na forma preventiva são implementados controle de admissão, adequação, políticas de tráfego e disciplina de filas. Na forma reativa, os mecanismos executam ações pré-programadas ao momento em que se detecta qualquer situação que provoque desvanecimento do nível de qualidade dos serviços, tais como os mecanismos de controle de fluxo do TCP e o gerenciamento ativo de filas.

2.5.1 Algoritmos de tratamento de filas

A política de tratamento de filas é um ponto fundamental para se obter êxito na busca de QoS. A eficiência no processamento do tráfego na rede deve estar adequada para situações de grande tráfego, buscando sempre evitar congestionamentos. Um congestionamento é quando a capacidade de saída de pacotes é menor do que a taxa de chegada, ou seja, um nó da rede não consegue processar seu tráfego de forma que vão acumulando pacotes a serem processados. Esses pacotes ficam em fila esperando seu processamento.

Uma fila é um espaço na memória reservada para armazenar pacotes a serem processados. Como é um espaço limitado, se a fila encher, alguns pacotes serão perdidos. Além disso, o tempo que um pacote fica na fila significa atraso na transmissão. Existem uma série de algoritmos para tratamento das filas, a seguir, é apresentado um resumo do funcionamento de alguns dos principais algoritmos [13]:

2.5.1.1 FIFO

O FIFO (Fist In First Out) é um dos algoritmos mais simples e antigos. Como o nome já diz: primeiro que entra, primeiro que sai. Seu funcionamento se resume no armazenamento de pacotes em caso de rede congestionada e o envio deles quando a rede não mais estiver congestionada, sendo que esse envio obedece a ordem de chegada. Quando a fila encher (acabar a memória especificada), os pacotes que chegam são descartados.

A grande maioria dos equipamentos encaminhadores de pacotes adotam esse algoritmo

como padrão. O algoritmo FIFO atende de forma eficiente casos com tráfego de rajadas com picos de pequena duração, já que, nesses casos, a fila não chega a ficar muito grande deixando o retardo médio sem prejuízos para a transmissão. Existem, porém, algumas deficiências [13]:

- ele não toma nenhuma decisão sobre prioridade do pacote;
- a ordem de chegada determina a taxa de transmissão que será obtida, a prioridade e a alocação de buffers; e
- não prove proteção contra aplicações ou fontes de tráfego com comportamento prejudicial [8].

Se acaso acontecer de uma grande quantidade de tráfego por um período prolongado, fatalmente a qualidade do serviço ficará comprometida prejudicando as aplicações.

2.5.1.2 PQ (Priority Queueing)

Tido como uma variação do FIFO, o PQ ou PRR (Priority Round Robin), leva em conta o critério de prioridade das classes de serviço, ou seja, as filas são armazenadas em classes. Por exemplo, um pacote de uma determinada classe de serviço só pode ser tratado caso não haja nenhum pacote em uma fila de uma classe com maior prioridade. Dessa forma, aplicações mais prioritárias terão seu tráfego atendido com maior prioridade sendo processado antes de outros de menor prioridade.

Isso equivale a ter várias filas com prioridades diferentes e as filas de menor prioridade só serão processadas caso as de maior prioridade estejam totalmente ociosas. O uso do PRR é bastante útil quando é preciso garantir que um tráfego de missão crítica atravessasse várias redes e tenha tratamento privilegiado em cada nó [23].

Embora bastante flexível em relação aos protocolos, este mecanismo de enfileiramento apresenta vulnerabilidades [8]. Se o volume de tráfego de maior prioridade for alto, o tráfego normal esperando para entrar na fila pode ser descartado por insuficiência de espaço de armazenamento.

2.5.1.3 Enfileiramento baseado em classes (CBQ)

O CBQ (Class-Based Queuing) ou CQ (Custom Queuing) foi projetado para admitir que aplicações com especificações de taxa de transmissão mínimas ou exigências de latência controlada, compartilhem a rede [24]. É uma variação do PQ, porém, com mais flexibilidade. No CBQ, pode-se definir qual o nível de preferência de cada fila além da quantidade de tráfego a ser escoado em cada passagem na rota do serviço [8].

É um mecanismo baseado em classes que se comporta como um alocador de porções da capacidade de transferência para certos tipos de tráfego. O fato é que pode-se prover maiores recursos para as classes de maior prioridade e menores para as de menor prioridade. Dessa forma há um tratamento mais justo do que o apresentado no mecanismo PQ, pois é melhor uma simples redução dos recursos do que sua total ausência.

CBQ vêm sendo considerada uma boa alternativa para implementação de mecanismos baseados em classes de serviço (CoS), além de um excelente método para gerenciamento de recursos de filas [13]. Um grande problema do CBQ é quanto a sua escalabilidade devido a grande capacidade de processamento exigida na reordenação de pacotes e gerenciamento das filas, principalmente em comunicações de alta velocidade.

2.5.1.4 WFQ

O mecanismo de enfileiramento WFQ (Weighted Fair Queuing) é um algoritmo que procura prover a entrega de pacotes de forma previsível, além de assegurar que os fluxos não sofram por falta total de buffers [8]. Este mecanismo dá aos fluxos de baixo volume de tráfego tratamento preferencial, permitindo assim, que os fluxos com alto volume de dados utilizem o restante da capacidade de enfileiramento. Ele pode ser usado para prover tempo de resposta consistente a usuários pesados e leves de modo semelhante, sem ter que adicionar largura de banda em excesso [13]

Ao mesmo tempo em que escalona o tráfego interativo para a frente da fila visando a redução do tempo de resposta, ele compartilha a banda restante entre os fluxos com alta taxa de consumo [8].

O mecanismo WFQ possui algumas das características dos mecanismos de enfileiramento por prioridade (PQ) e enfileiramento baseado em classes (CBQ) e, pelos mesmos motivos, apresenta problemas de escalabilidade.

É um mecanismo de enfileiramento que se adequa bastante a arquitetura IntServ que opera por fluxos, como à DiffServ que opera por classes [23]. A cada fluxo ou classe é atribuído um peso que será utilizado para determinar a fração de tempo ou recurso que será alocada ao fluxo ou classe correspondente. Contudo, não são fornecidas formas de avaliar ou ajustar estes parâmetros para alterar o comportamento, pois o método de preferir alguns fluxos sobre outros é estaticamente definido na implementação específica de cada fabricante [8].

2.5.2 Mecanismos de Controle de Congestionamento

Nas redes atuais pode se dizer que é mais provável que as perdas de pacotes ou os atrasos ocorram por congestionamento do que por algum defeito no hardware. O fato é tanto as perdas como atrasos prejudicam as aplicações chegando até ao ponto de torná-las inviáveis. Com fins de amenizar as conseqüências causadas pelos picos de tráfego, que congestionam a rede, surgiram alguns mecanismos:

2.5.2.1 Controle de Congestionamento do TCP

O protocolo de transporte TCP possui um interessante mecanismo de controle de congestionamento. Em sua implementação são utilizadas várias técnicas como o algoritmo de janelas deslizantes que é um mecanismo muito eficiente de controle de fluxo, importante especialmente na Internet onde existem equipamentos com as mais diversas capacidades de transmissão.

Existe também o mecanismo de reação a congestionamentos [25] que detecta o congestionamento usando as perdas de pacotes como referência e responde imediatamente reduzindo de forma drástica a taxa de transmissão de dados. O TCP trabalha o tempo todo tentando ajustar a sua taxa de transmissão para um valor ótimo, onde maximize a taxa e minimize as perdas.

Essas técnicas e algoritmos deixaram o TCP muito mais estável e eficiente, melhorando seu desempenho em relação às versões anteriores que não implementavam esses algoritmos.

2.5.2.2 Gerenciamento Ativo de Filas (RED)

Em pequenas redes os algoritmos de controle de congestionamento do TCP são eficientes pois a quantidade de fluxos ativos simultaneamente é pequena, já em redes grandes pode ocorrer o problema de sincronização global. Numa situação de congestionamento com muitas sessões TCP ativas simultaneamente, as perdas de pacotes podem fazer com que os fluxos TCP executem o procedimento padrão de diminuir sua taxa de transferência. Em seguida, as sessões TCP começam a incrementar sua taxa de transmissão exponencialmente e provocando novamente o congestionamento dando origem a um ciclo vicioso. Este é um dos maiores problemas que não são resolvidos pelos algoritmos de controle de congestionamento do TCP.

Os picos de tráfego ou os tráfegos em rajada de grande volume e curta duração são outro problema. Esta situação provoca congestionamentos com pequena duração de tempo. Isto se resolveria com um simples aumento do tamanho dos buffers nos roteadores, porém geraria o problema da latência induzida e o jitter, causados pelo tamanho excessivo das filas.

Para que tratar esses dois problemas e não gerar mais problemas para a rede, pode-se realizar uma gestão apurada na rede, monitorando seu tráfego. Assim os congestionamentos seriam detectados e seria possível distinguir entre curta e longa a duração permitindo escolher a melhor maneira de reagir. Isto é o chamado gerenciamento ativo de filas. O mecanismo mais conhecido e aceito que implementa esse gerenciamento é o RED (Random Early Detection).

O RED é um mecanismo de contenção de congestionamento designado para redes de pacote comutado que tem como base controlar o tamanho médio da fila e indicar nós finais quando eles devem parar de enviar pacotes temporariamente. O RED leva uma enorme vantagem sobre o mecanismo de controle de congestionamento do TCP por descartar pacotes de forma aleatória antes do congestionamento, ou seja, o RED pede para origem do pacote para diminuir

sua taxa de transmissão visando evitar o congestionamento. Como isso é feito aleatoriamente, o problema da sincronização global é eliminado.

O RED funcionaria como um meio de causar o abandono do tráfego, o que não interrompe os fluxos e sim adapta a sua taxa de transmissão à taxa que a rede possa suportar sem congestionamentos. O RED faz justamente o oposto do tail-drop. Ele monitora o tamanho médio da fila e usa um critério randômico para escolher pacotes e notificar os transmissores da respectiva conexão de que um congestionamento está impedindo a transmissão daquele fluxo. O algoritmo de descarte de pacotes do RED utiliza o tamanho médio da fila e dois gatilhos para o cálculo da probabilidade de descarte.

O RED só funciona adequadamente com protocolos de transporte que reajam à perdas de pacotes com a diminuição do tamanho da janela de transmissão (no caso o TCP), em casos de uso dos protocolos AppleTalk ou Novell Netware que respondem a pacotes descartados através de retransmissão dos mesmos, o RED não terá nenhum efeito positivo, podendo inclusive prejudicar o desempenho da rede [23].

O RED usa a técnica do gerenciamento ativo de filas pra descartar pacotes e sinalizar congestionamento para a fonte de tráfego TCP. Isso trás o inconveniente de interromper o tráfego e tentar a retransmissão o que contribui com o congestionamento.

2.5.2.3 ECN

ECN (Explicit Congestion Notification) é uma outra solução que sinaliza o congestionamento para a fonte do tráfego TCP utilizando marcas no cabeçalho dos pacotes IP sem que haja descarte de pacotes. Com sua utilização a necessidade do roteador descartar um pacote selecionado randomicamente é eliminada.

Um algoritmo, baseado no RED, faz uma sinalização para o gerador do tráfego TCP que a fila está próxima de sua capacidade máxima e pede que o mesmo se adapte à situação, ocorrendo um descarte de pacotes. Já no ECN isso não ocorre pois faz-se uma marcação nos flags do cabeçalho IP CE (*Congestion Experienced*) e ECT (ECN-Capable Transport). Assim o transmissor recebe esse pacotes marcados e intende que deve reduzir seu tráfego e evitar assim, o descarte de pacotes.

2.5.3 Políticas de Tráfego, Adequação e Controle de Admissão

Existem os casos onde o congestionamento se dá por períodos mais longos, e logo, alguns pacotes impreterivelmente serão descartados. Através de ações preventivas pode-se chegar a uma solução que possibilite um amplo controle do tráfego que será admitido para o interior da rede. Dentre essas ações destacam-se as Políticas de Tráfego, Adequação e Controle de Admissão.

2.5.3.1 Política de Tráfego

Política de dados é o nome dado aos critérios que estabelecem como serão utilizados os recursos da rede, além, das prioridades para tipos diferentes de tráfego. Sem esse tipo de abordagem não é possível implementar QoS [23].

Essas políticas devem estar bem claras na implementação de QoS, pois são nelas que estão definidas as chamadas classes de perfis de tráfego que são descrições do serviços, prioridades e recursos que cada aplicação ou tipo de tráfego terá disponível. Essa negociação geralmente é feita entre um provedor de serviços com o usuário através dos contratos de SLA (Service Level Agreement).

2.5.3.2 Adequação de Tráfego e o Token Bucket

Adequar um tráfego é na verdade garantir que seu tratamento esteja de acordo com o que foi descrito em sua política. Para isso, pode-se utilizar o modelo do token ou ficha que é um modelo conceitual de execução de políticas de tráfegos, adequação e também, controle de admissão de tráfego. Esse modelo na prática faz com que o throughput do tráfego seja constante fazendo com que a rede admitia somente um tráfego que ela é possa suportar.

Leaky bucket ou balde furado é uma implementação bem simples deste modelo. Esse nome se dá devida a analogia de seu funcionamento com um balde d'água furado. O tráfego de pacotes seria o fluxo de água e o buffer o balde.

Também existe o Token Bucket ou balde de símbolos que difere do leaky bucket, porque permite pequenas variações no tráfego quando ocorre congestionamento, já o leaky bucket força um fluxo de saída constante em qualquer situação.

Estes algoritmos adequam o tráfego admitindo-o de modo que se obtenha m throughput médio de longo prazo correspondente à taxa de geração de tokens e uma variação correspondente à profundidade do balde.

2.5.3.3 Controle de Admissão

Controlar e regular o volume de tráfego admitido na rede é imprescindível para que se possa fornecer um serviço adequado aos usuários e suas aplicações. Para isso deve-se implementar mecanismos que controlem a admissão de novos tráfegos. Esse mecanismo irá determinar se existe ou não recursos disponíveis para atender às requisições. O fato é que um novo fluxo só será admitido se a rede dispor dos recursos necessários para atendê-lo sem causar danos aos demais fluxos.

A estratégia de controle de admissão deve estar de acordo com as políticas estabelecidas de forma garantir o cumprimento das mesmas. São necessárias que todas as informações sobre

os recursos disponíveis estejam centralizadas ou que haja um mecanismo de troca de informações sobre esses recursos. Isso irá permitir uma análise da rede e para se verificar se a admissão do tráfego será possível. Todos os parâmetros devem ser levados em conta.

3 O SISTEMA OPERACIONAL FREEBSD

Toda aplicação necessita de uma base sólida para ser executada com correção. Os serviços de redes na maioria das vezes devem estar disponíveis a todo momento. Para isso o sistema operacional precisa oferecer um ambiente robusto e flexível garantindo maior segurança, disponibilidade e confiabilidade do sistema. Este capítulo apresenta, de maneira breve, características do sistema operacional FreeBSD, um S.O. que fornece, além do que já foi mencionado, uma solução gratuita, com pouca exigência de hardware e boa performance.

3.1 Breve História ²

A Universidade da Califórnia em Berkeley (UCB) recebeu 1974 a versão 4 do UNIX. Ao receber este pacote de código fonte, Berkeley iniciou o desenvolvimento de uma série de aperfeiçoamentos e assim, estabeleceu um primeiro estágio para o desenvolvimento de sua própria versão do UNIX. Em 1975 Ken Thompson, um pesquisador da AT&T, inicia uma licença sabática em Berkeley.

Em 1978 a Berkeley Software Distribution (BSD) lança a série 2.xBSD para PDP-11. Neste ano saiu também a série 3BSD com incluía suporte a memória virtual. Até esse momento o UNIX só havia sido executado em plataformas DEC e a DEC, por sua vez, recusava-se a suportar o UNIX dando preferência para seu próprio sistema operacional.

Em 1979 saiu a V7 e o UNIX foi portado para o novo VAX. Esta foi a primeira versão vendida comercialmente e a partir deste momento a AT&T começou a proteger seus interesses comerciais. Neste mesmo ano a Microsoft adquire uma licença da AT&T e lança uma versão do UNIX, chamava-se XENIX e rodava em microcomputadores de arquitetura da Intel.

Em 1980 o BSD UNIX sai do Bell Labs através de um contrato com a DARPA (Defense Advanced Research Projects Agency), fundando o Computer Systems Research Group (CSRG). Este grupo, não somente desenvolveria o BSD UNIX, mas também seria responsável pela primeira implementação da família de protocolos TCP/IP em um sistema operacional.

Em 1982 a HP anuncia suporte ao UNIX (HP/UX) em suas estações da linha 9000. A DEC lança o ULTRIX. A IBM lança o CPIX. Em 1983 é lançado o System V (SV) da AT&T e o 4.2BSD, esta última pode ser considerada uma das mais importantes versões do Unix por facilitava muito a conexão do Sistema a redes locais além de ter integrado os softwares que implementam TCP/IP e sockets.

Em 1985 a AT&T criou o System V Interface Definition (SVID), em uma tentativa de padronização. Neste mesmo ano o padrão POSIX foi estabelecido. Em 1986 a IBM lança o AIX. Em 1991 a AT&T estabelece o UNIX System Laboratories (USL), em cooperação com a Novell, Amdahl, Fujitsu, Sun, Motorola, ICL, Olivetti, NEC, OKI Electric, III of Taiwan e a Toshiba. A Novell e o USL iniciam o desenvolvimento e a distribuição do UnixWARE. A Sun cria a SunSoft e

anuncia o Solaris. A Sun Microsystem também lançou sua primeira versão Unix a partir do BSD. Neste ano Linus Torvalds inicia o desenvolvimento do Linux.

Em 1992 a AT&T vende sua participação na Sun. Neste mesmo ano o 4.4BSD é lançado para várias plataformas: HP 9000/300, Sparc, 386, DEC e outros. Também neste ano Bill Jolitz lançou o 386BSD, primeiro derivado BSD para a plataforma i386.

Em 1993 o USL é adquirido pela Novell. Em outubro a marca Unix é cedida pela Novell para a X/Open. O CSRG de Berkeley é dissolvido e tanto a AT&T quanto Berkeley estão fora do desenvolvimento do Unix. É criada a BSDi (Berkeley Software Design, Inc.) que é processada pela USL. O projeto FreeBSD então é estabelecido a partir do “Unofficial 386BSD PatchKit”.

Em 1994 é a Novell que processa a BSDi. A SunSoft, AT&T GIS, Novell e a Fujitsu pagam um milhão de dólares para juntar-se a OSF. A Sun compra uma licença do Unix da Novell por U\$82 milhões.

Em 1996 o FreeBSD já era bastante popular em inúmeros ISPs (Internet Service Providers – Provedores de Internet) em todo o mundo. Neste ano o FreeBSD torna-se a plataforma da Yahoo!.

Em 1999 a série 3.x inicia uma seqüência de aperfeiçoamentos ao sistema, alcançando uma surpreendente estabilidade. Em 2000 a série 4.X é distribuída, oferecendo inúmeros aperfeiçoamentos e correções. Atualmente a versão em desenvolvimento é a 5.x sendo a 5.1 prevista para o final de 2003.

É importante lembrar que o Bell Labs desenvolveu outros sistemas operacionais nas décadas de 1980 e 1990. Merecem destaque os projetos Plan9 e Inferno. Os pesquisadores Ken Thompson e Dennis Ritchie participam destes e outros projetos também.

3.2 Características

- Código aberto: acesso completo ao seu código fonte
- Multitarefa: capacidade de realizar várias tarefas ao mesmo tempo com menos recursos de hardware
- Multi-usuário: suporta vários usuários compartilhando recursos e aplicativos simultaneamente.
- Capacidade de Rede TCP/IP: foi o sistema operacional pioneiro na implementação desse protocolo. É capaz de atuar como um servidor de serviços de Internet, estabelecer rotinas de roteamento e firewall.
- Memória Protegida: diferentes aplicações ou usuários não interferem entre si. Ou seja, mesmo que uma aplicação trave, isso não atrapalhará os outros processos ou usuários que estão utilizando o sistema.
- Módulos de compatibilidade Binária Linux, AT&T SysVR4 e SCO Unix: com a adição desses módulos o sistema será compatível com os demais sistemas relacionados.

² Essa sessão foi retirada e adaptada de [14] e [9]

- Pacotes de Aplicativos: existem mais de 7000 aplicativos disponíveis para download na Internet, além dos que já vêm embutidos nos arquivos imagem do sistema.
- Multi processamento Simétrico: implementa suporta a hardware com vários processadores.
- Ferramentas de desenvolvimento: suporte às linguagens C, C++, Perl, Java, Php entre outras.
- Livre distribuição: software e documentação com distribuição livre, devendo apenas obedecer o BSD Copyright conforme Anexo I.

Outra característica importante do FreeBSD é a forma que são tratadas suas versões. A versão que está sendo desenvolvida recebe o nome de CURRENT. Em intervalos de tempo (3 ou 4 vezes ao ano) é lançada uma versão em que as pendências mais críticas são arrumadas e então é disponibilizada para download com o nome RELEASE. Após vários testes realizados sobre a versão RELEASE, os erros são reportados e corrigidos. Quando não se encontra mais erros em uma versão, é então disponibilizada uma nova versão chamada STABLE. Como a versão STABLE já foi testada de forma intensiva, têm-se uma garantia de que o sistema é estável e seguro.

4 PROJETO E RESULTADOS EXPERIMENTAIS

Esse projeto tem como objetivo aprofundar conceitos no que diz respeito a qualidade de serviço (QoS) em redes de computadores em seu âmbito geral e também a exploração prática desse tema no sistema operacional FreeBSD.

Espera-se realizar um estudo aprofundado dos recursos do sistema operacional FreeBSD no que diz respeito a parte de conectividade utilizando o protocolo TCP/IP e aplicação de parâmetros de QoS em suas configurações.

Através de uma pesquisa feita no site do S.O. FreeBSD e na Internet pretende-se verificar a utilização das ferramentas que fornecem suporte à qualidade de serviço.

4.1 Ferramentas

Na primeira etapa fez-se uma breve pesquisa em ferramentas de software de QoS compatíveis com o FreeBSD. Dessa pesquisa foi realizada uma pré escolha das ferramentas. Como critério para a escolha optou-se por ferramentas gratuitas, e relacionadas com o sistema operacional escolhido. Na etapa seguinte, procura-se avaliar as ferramentas verificando a instalação, configuração e características. Dentre as ferramentas pré escolhidas pode-se citar:

ALTQ: Alternate Queueing for BSD UNIX

Fornecer disciplinas de fila além de outros componentes relacionados QoS requeridos compartilhar recursos e a qualidade do serviço.

PF: Packet Filter

Firewall padrão de sistemas OpenBSD que está sendo portado para o FreeBSD e que oferece possibilidade de controle de largura de banda.

IPFW e DUMMYNET

Firewall mais utilizado no FreeBSD e que oferece possibilidade da criação de canais com características pré-definidas.

Zebra

É um software que implementa roteamento baseado em redes IPv4. Possui suporte para arquiteturas de provimento de QoS.

EclipseBSD

Sistema Operacional baseado em FreeBSD que busca implementar QoS tanto nível de rede como também de sistema.

4.1.1 Ferramentas auxiliares

Outras ferramentas não ligadas ao escopo do trabalho foram utilizadas, dentre elas, destacam-se:

MGEN

A ferramenta MGEN é composta pelos módulos mgen para a geração de tráfego, drec para a recepção na estação remota e o mcalc para geração da informações a partir do arquivo de saída gerado pelo drec. Esta ferramenta permite a geração controlada de tráfego UDP, sendo possível a geração de um ou mais fluxos unicast ou multicast com a taxa de bits desejada. A ferramenta permite também controlar o tempo do experimento e a variação do tamanho do pacote UDP utilizado. [13]

NETPERF

O Netperf é uma ferramenta de benchmark para medir desempenho de rede de computadores. Ela foi desenvolvida pela Divisão de Informações de Redes da Hewlett Packard Company. A ferramenta é composta de dois programas básicos: netperf e netserv. Baseado no modelo cliente- servidor, onde o tráfego é gerado com o programa netperf que é receitado pelo programa netserver.

4.2 Ferramentas Escolhidas

Existem muitas ferramentas disponíveis, ou sendo desenvolvidas, que atendam às necessidades de se prover QoS Seguindo as idéias básicas dos modelos IntServ e DiffServ, escolhe-se respectivamente os softwares Dummynet e AltQ. O Dummynet implementa pipes (dutos) com características que permitem reservar recursos. O AltQ implementa uma série de algoritmos de gerenciamento de filas que traz a possibilidade priorizar determinados tipos de tráfego. Nas próximas sessões procura-se apresentar de forma mais completa estes dois softwares.

4.2.1 DUMMYNET

O Dummynet foi proposto por Luigi Rizzo como uma alternativa aos simuladores de redes existentes até então. A justificativa dada [26], era que, parâmetros operacionais como largura de

banda, atrasos e tamanho de filas eram fáceis de se controlar nos simuladores, porém, eles só ofereciam essas possibilidades no modelo simulado não havendo possibilidade de simular em um sistema real.

O Dummynet implementa algumas modificações na pilha do protocolo o que permite experimentos em um sistema real. Ele trabalha através de interceptação das comunicações na camada em teste e simulação de efeitos de filas finitas, limitação de banda e atrasos de comunicação. Como é implementado na pilha real do sistema, permite o uso de geradores de tráfego além do uso de tráfego real.

Ele trabalha em conjunto com o firewall padrão da maioria das versões 4.x do FreeBSD, o IPFW. Para utilizá-lo basta adicionar regras ao firewall com a seguinte sintaxe:

```
# ipfw add pipe [N] tráfego
# ipfw pipe [N] config bw [B] delay [D] queue [Q] plr [P]
```

Onde:

- N é o número do pipe criado
- B é a banda definida para o pipe
- D é o atraso que terá os pacotes
- Q é a fila que será posto os pacotes
- P é o número de pacotes que serão descartados
- tráfego é a regra que define qual protocolo/porta faz parte do pipe

É importante lembrar que os tipos de tráfego que não estiverem definidos em algum pipe serão tratados por melhor esforço. Todos os passos para a instalação do Dummynet são mostrados no Anexo IV.

4.2.2 ALTQ (Alternate Queueing)

Como foi visto na seção 2.5, o controle das filas é um elemento essencial para o controle de tráfego, acontece que a maioria dos sistemas utilizam somente o algoritmo FIFO. O ALTQ, é um framework que permite o uso de uma série de outras disciplinas de controle de fila.

Tendo como sistema de origem o OPENBSD, o ALTQ já foi implementado como uma simples extensão do kernel do FreeBSD, além de algumas correções nos drivers de alguns dispositivos. Vários algoritmos de gerência de fila são implementados tais como CBQ, RED, WFQ, HFSC, PRIQ, JoBS, RIO e Blue.

4.3 Experimentos

Afim de verificar o funcionamento das ferramentas e também observar o comportamento em diferentes situações realiza-se alguns experimentos.

4.3.1 Ambiente

Para a realização dos experimentos, foram utilizados três computadores:

M1 – Pentium III 450 MHz com 96 Mbytes de memória RAM, placa de rede VIA VT6105 Rhine III 10/100BaseTX, SO FreeBSD v 4.9 STABLE

M2 – Pentium I 133MHz com 64Mbytes de memória RAM, placa de 3Com 3c905-TX Fast Etherlink, SO FreeBSD v 4.9 STABLE

M3 – Pentium III 600MHz com 128Mbytes de memória RAM, placa de rede SIS 900 Fast Ethernet, SO MS Windows 98 SE.

A Figura 1 apresenta a topologia do ambiente

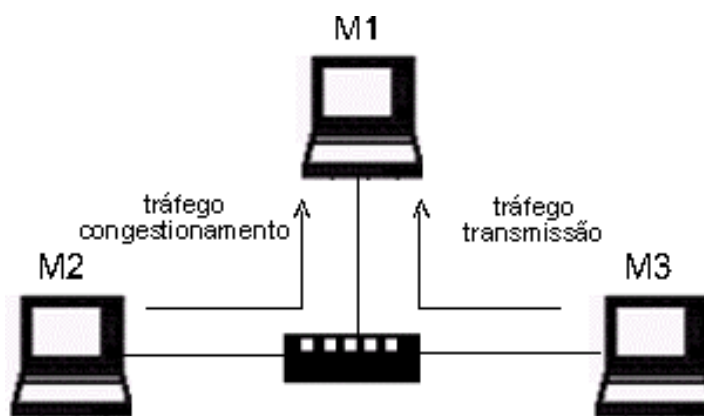


Figura 1 – Topologia do ambiente de experimentos

As ferramentas serão observadas em M1. M2 servirá para, em algumas circunstâncias, gerar um tráfego afim de ocupar ou congestionar a rede. M3 será usado para realizar as transmissões via SFTP. Os três computadores estão conectados, via par-trançado, a um hub padrão ethernet, o que forçara as placas de rede a trabalharem a até 10Mbps. Configurou-se as placas fornecendo a cada uma um endereço IP e através do utilitário ping, verificou-se as conexões. Foram instaladas em M1 e M2 as ferramentas MGEN e NetPerf, que geram tráfego entre as duas máquinas.

4.3.2 Experimento com DUMMYNET

O experimento consiste em realizar a transmissão de um arquivo de 10 Mbytes de M3 até M1 utilizando o software SSH Secure Shell observando diversas situações. Cada transmissão foi

repetida 7 vezes para cada situação, o maior e o menor tempo são descartados. O tempo de transmissão então é calculado com a média aritmética simples dos 5 tempos restantes. Com esse procedimento tenta-se excluir quaisquer anormalidades que possam ocorrer em determinadas transmissões ficando com um valor médio mais próximo possível do real.

Situações:

a) Rede livre e sem pipe

Tempo médio de transmissão: **48:3** segundos

Média da taxa de transmissão: **1696** Kbps

b) Rede congestionada e sem pipe

Para congestionar a rede, foi utilizado o software MGEN que, gerou um tráfego que saturou o canal, conforme Anexo II. Vários valores para o tamanho do pacote e para o número de pacotes por segundo foram testados até que o canal ficasse saturado porém sem perdas. Valores acima de [800 1024] causavam logo um estouro da capacidade da pilha, e o sistema retornava a seguinte mensagem:

“UdpSocket::SendTo(): sendto() error:: No buffer space available”

Tempo médio de transmissão: **1:09:9** minutos

Média da taxa de transmissão: **1168** Kbps

c) Rede livre e com pipe de 800Kbps

Um pipe foi criado e logo após, foi configurado para uma taxa de transferência de 100Kbytes por segundo.

```
M1# ipfw add 10 pipe 1 tcp from any to any
```

```
M1# ipfw pipe1 config bw 800Kbps
```

Tempo médio de transmissão: **2:49.5** minutos

Média da taxa de transmissão: **480** Kbps

d) Rede livre e com pipe de 2400Kbps

Tempo médio de transmissão: **49.4** segundos

Média da taxa de transmissão: **1656** Kbps

Para alterar as configurações do pipe basta configurar com os novos parâmetros:

```
M1# ipfw pipe1 config bw 2400Kbytes
```


e) Rede livre e com pipe de 4000Kbps
Tempo médio de transmissão: **39:8** segundos
Média da taxa de transmissão: **2056** Kbps

f) Rede livre e com pipe de 8000Kbps
Tempo médio de transmissão: **28:3** segundos
Média da taxa de transmissão: **2888** Kbps

g) Rede congestionada e com pipe de 800Kbps
Tempo médio de transmissão: **2:14:2** minutos
Média da taxa de transmissão: **608** Kbps

h) Rede congestionada e com pipe de 2400Kbps
Tempo médio de transmissão: **1:08:4** minutos
Média da taxa de transmissão: **1192** Kbps

i) Rede congestionada e com pipe de 4000Kbps
Tempo médio de transmissão: **1:04:7** minutos
Média da taxa de transmissão: **1264** Kbps

j) Rede congestionada e com pipe de 8000Kbps
Tempo médio de transmissão: **40:1** segundos
Média da taxa de transmissão: **2040** Kbps

Resultados

A figura 2 demonstra graficamente o comportamento geral dos tempos de transmissão.

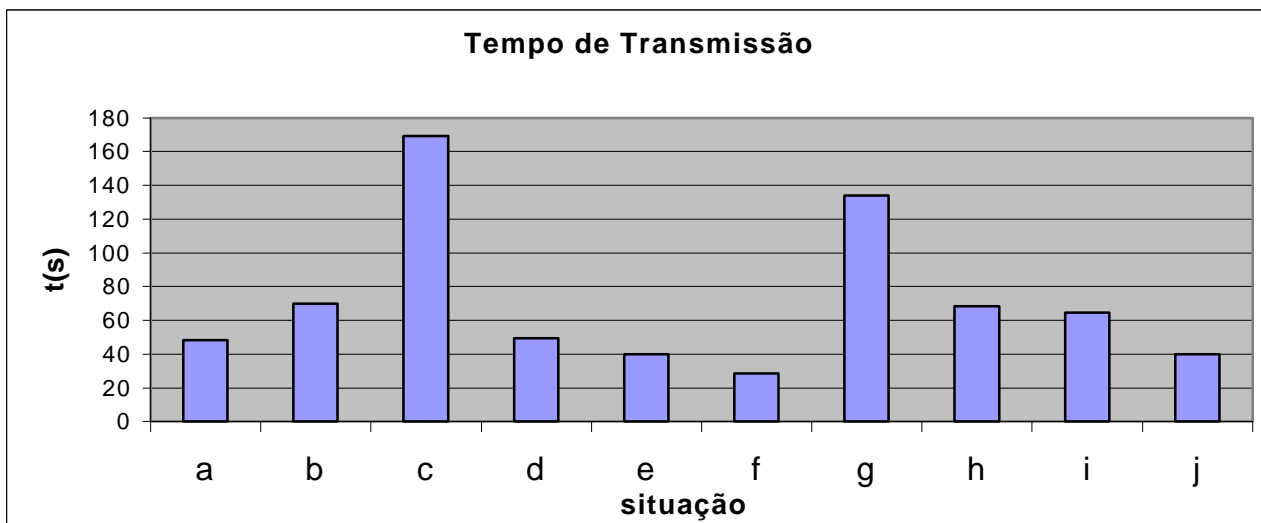


Figura 2 – Tempos de transmissão para cada situação

A figura 3 trás uma comparação entre tempos em que a rede não tinha tráfego e os tempos que estava congestionada para os mesmo valores atribuídos aos pipes.

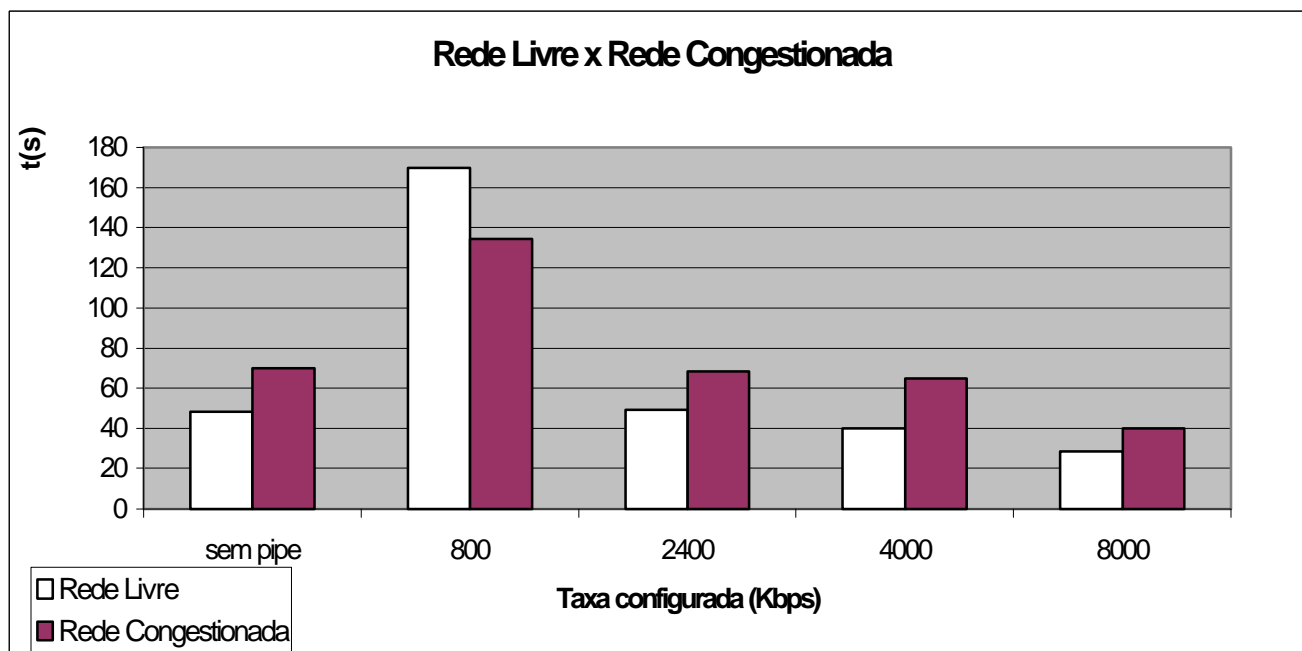


Figura 3 – Comparação dos tempos de transmissão com “rede livre” versus “Rede Congestionada”

Se considerarmos que cada bit transmitido seja um bit de dado (payload) pode-se afirmar que as taxas de transferência ideais são iguais às taxas nominais especificadas ou configuradas. Ou seja, uma rede padrão ethernet teria uma taxa uma taxa de transmissão 10Mbits por segundo. Claro que além de ignorar as informações de cabeçalho dos pacotes, não está se levando em conta outra série de fatores que afastam essa taxa da realidade (perdas, colisões, handshake do protocolo, etc). Utiliza-se esses valores apenas para observar o aumento que seria esperado com as diferenças entre as taxas configuradas e as taxas obtidas. A figura 4 mostra essa comparação.

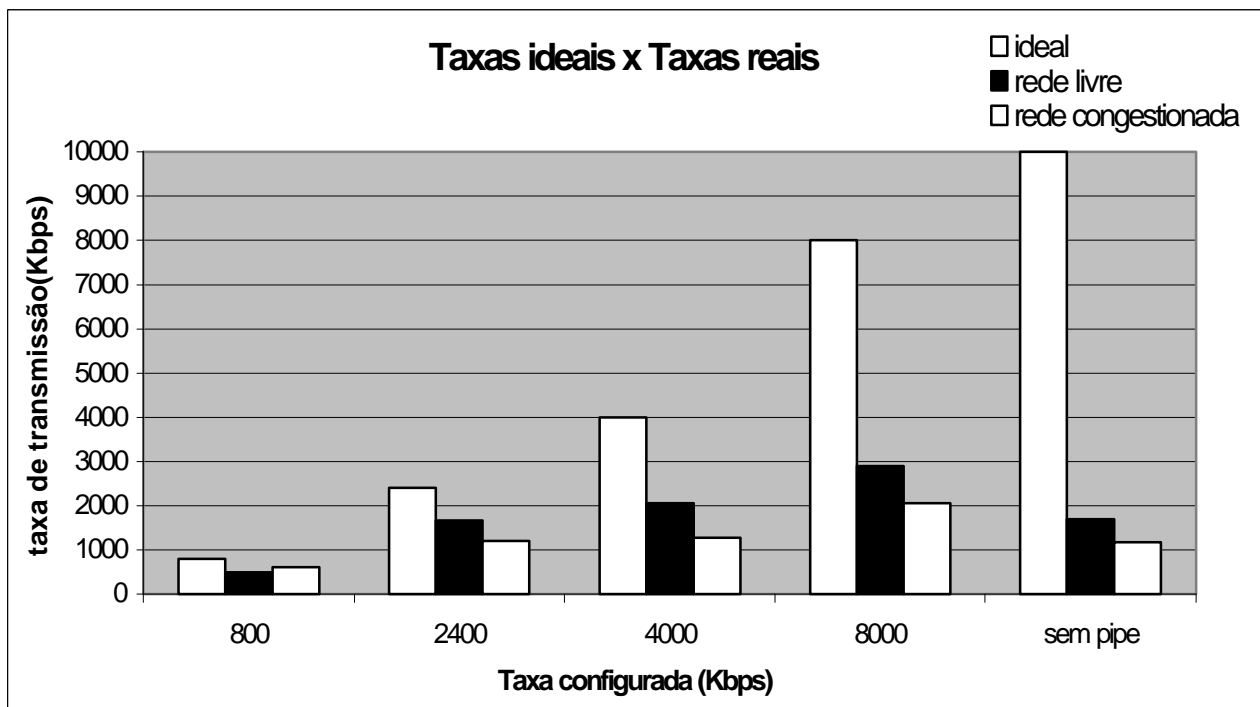


Figura 4 – Comparação de taxas de transmissão ideais com as taxas obtidas

O que se pode observar nessa comparação é o comportamento irregular da taxa real quanto à taxa ideal. Por exemplo, na transmissão com rede livre com o pipe configurado a 4000Kbps a taxa média encontrada foi 2056Kbps. Supõem-se então que ao dobrar o valor do pipe para 8000Kbps, a taxa real aumente proporcionalmente, ou seja, também dobre. Isso não acontece, a taxa da transmissão foi de 2888Kbps, um aumento de 40%, onde teoricamente, se esperava 100%.

Apesar da reserva de recursos ser a mesma, observou-se uma variação nas taxas de transferência em caso de congestionamento na rede. Acredita-se que este fato está relacionado com o protocolo de acesso ao meio, pois durante os experimentos com a rede congestionada observou-se um alto índice de colisões indicados pelo hub.

4.3.3 Experimentos com AltQ

Para utilizar o AltQ deve-se configurar o arquivo `/etc/altq.conf`. Nesse arquivo pode-se configurar para uma mesma rede uma infinidade de opções diferentes. O AltQ permite o uso de diversos algoritmos para tratamento de fila (BLUE, CBQ, FIFO, HFSC, PRIQ, RED, RIO, WFQ, JoBS e CDNR). Além disso, permite atribuir largura de banda, definir prioridades para determinados tráfegos, tamanho do pacotes, atraso máximo permitido entre outros.

Esta fora do escopo deste trabalho analisar detalhadamente a ferramenta e explorar todas essas característica do software.

Para verificar o funcionamento da ferramenta, faz-se transmissões de um arquivo de 10Mbytes em diversas situações da máquina M3 para a máquina M1. Instala-se a ferramenta AltQ em M1 e cria-se um arquivo `/etc/altq.conf`.

Arquivo `altq.conf`³

```

01 # Configura a interface para um link de 10Mbits
02 interface vr0 bandwidth 10M cbq
03
04 # Meta classes
05 #
06 class cbq vr0 root NULL pbandwidth 100
07 class cbq vr0 ctl_class root pbandwidth 10 control
08
09 # Classe que a transferência de encaixa
10 # (neste caso configurada para 5% do tráfego,
11 # o que seria no caso de 500Kbits )
12 class cbq vr0 def_class root borrow pbandwidth 5 default
13
14 # Define a classe traf da meta classe root com banda de 85%
15 # Nesta classe passaria o restante do tráfego
16 class cbq vr0 traf root pbandwidth 85
17     filter vr0 traf 0 0 M2 netmask 255.255.255.0 0 6
18     filter vr0 traf M2 netmask 255.255.255.0 0 0 6
19 # (Para o experimento foi configurada para todo tráfego de M2,
20 # onde será gerado o tráfego pelo Netperf)

```

Todos caracteres escritos a direita do caracter # são ignorados e considerados como comentários. Na linha 02 é configurada a capacidade de transmissão do link para a interface vr0. Na linha 06 é criada uma metaclassa (ou classe raiz) onde outras classes derivarão dela. Na linha

³ Os números que aparecem à esquerda do arquivo são apenas para numerar as linhas. Não aparecem no arquivo original.

12 é criada uma classe padrão que será utilizada por todos os pacotes que não estiverem definidos em nenhuma outra classe. É essa classe que será utilizada para transmissão do arquivo.

Nas linhas 16 até 18 define-se uma classe que será utilizada para congestionamento da rede utilizando, desta vez, o software NetPerf, conforme Anexo III. Para esse experimento escolheu-se o algoritmo CBQ, que pode ser observado em quase todas as linhas. Para facilitar a atribuição de largura de banda pode-se utilizar a palavra chave pbandwidth que ao invés de fornecer o valor em bps, fornece-se em porcentagens.

As situações observadas:

- a) 0,5% da banda ou 50kbps com rede livre
Tempo médio de transmissão: **47,6** segundos
Média da taxa de transmissão: **1720** Kbps

- b) 0,5% da banda ou 50kbps com tráfego Netperf
Tempo médio de transmissão: **1:26:3** minutos
Média da taxa de transmissão: **944** Kbps

- c) 1% da banda ou 100kbps com rede livre
Tempo médio de transmissão: **47,7** segundos
Média da taxa de transmissão: **1712** Kbps

- d) 1% da banda ou 100kbps com tráfego Netperf
Tempo médio de transmissão: **1:18:3** minutos
Média da taxa de transmissão: **1040** Kbps

- e) 3% da banda ou 300kbps com rede livre
Tempo médio de transmissão: **48** segundos
Média da taxa de transmissão: **1704** Kbps

- f) 3% da banda ou 300kbps com tráfego Netperf
Tempo médio de transmissão: **1:12:6** minutos
Média da taxa de transmissão: **1128** Kbps

- g) 5% da banda ou 500kbps com rede livre
Tempo médio de transmissão: **47.3** segundos
Média da taxa de transmissão: **1728** Kbps

h) 5% da banda ou 500kbps com tráfego Netperf

Tempo médio de transmissão: **1:09:3** minutos

Média da taxa de transmissão: **1176** Kbps

i) 10% da banda ou 1Mbps com rede livre

Tempo médio de transmissão: **54:1** segundos

Média da taxa de transmissão: **1512** Kbps

j) 10% da banda ou 1Mbps com tráfego Netperf

Tempo médio de transmissão: **1:27:3** minutos

Média da taxa de transmissão: **936** Kbps

Resultados

Observando a figura 5, gráfico comparativo dos tempos de transmissão com e sem o tráfego gerado pelo NetPerf, observa-se que o tempo de transmissão, quando não se tem tráfego NetPerf, permanece praticamente constante. Isto se deve ao fato de que a reserva de uma classe pode ser usada por outra se esta não estiver sendo utilizada, ou seja, os recursos não ficam estagnados à espera de utilização.

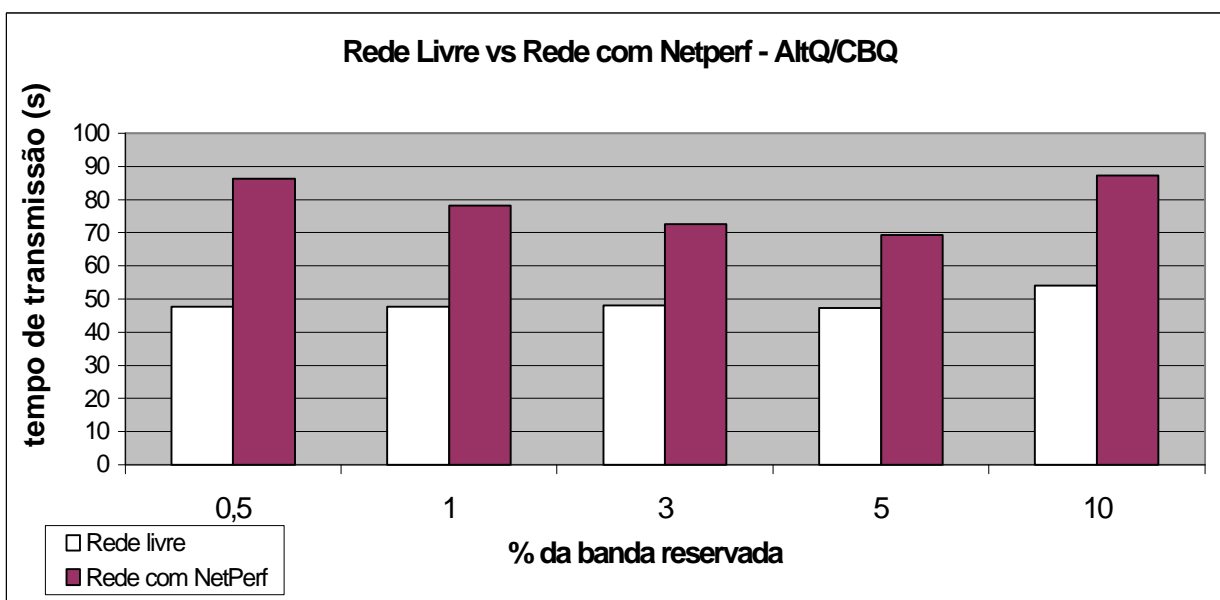


Figura 5- Comparação de tempos de transmissão utilizando AltQ

Pode-se também, perceber um comportamento inesperado quando o valor da reserva atingiu 10%, a tendência de diminuir o tempo da transmissão quando em uma situação de

reserva maior não foi observada. Repetindo várias vezes e não alterando resultados percebeu-se que o led indicador de colisão do hub permanecia praticamente aceso durante toda transmissão.

5 CONCLUSÕES E TRABALHOS FUTUROS

5.1 Conclusões

As ferramentas escolhidas foram observadas em características diferentes. No Dummynet observou-se sua semelhança ao paradigma do IntServ (reserva de recursos, de forma bem simplificada). A possibilidade de fazer reservas de banda garantindo que serviços mais importantes não sejam prejudicados por um congestionamento na rede é um ponto forte. Seu grande problema é o fato de ser estático, ou seja, os valores ficam fixos aos configurados, não importando a situação que a rede se encontre. A reserva de recursos pode causar congestionamento mesmo com banda disponível, como em uma situação em que um tipo de tráfego com grande reservas não estar sendo utilizado e outro tipo, com reservas menores, estar sendo muito requisitado. Nesse caso recomenda-se evitar pequenas reservas deixando o melhor esforço tratar o tráfego.

Dentre as vantagens observadas do Dummynet pode se ressaltar sua facilidade de instalação e configuração. O software foi desenvolvido para o FreeBSD e nas versões mais atuais basta somente adicionar o módulo e recompilar o kernel do sistema. Além do uso de pipes para priorizar certos tipos de tráfego, pode se utiliza-los para deteriorar outros tipos. Isso se torna útil quando a intenção é desestimular o uso de alguma aplicação, o que é muito interessante para empresas que têm problemas com funcionários acessando vídeos, rádios ou fazendo uso de software que compartilham arquivos (Kaaza, E-Donkey, Morpheus, etc).

O AltQ demonstrou uma característica muito interessante quando se fala em reserva de recursos, o fato de um recurso reservado para uma classe poder ser utilizado por uma outra classe quando a primeira não estiver precisando, nos traz uma idéia de prioridade e uso eficaz da rede. Isso também evita desperdícios.

Pela documentação observada da ferramenta AltQ, percebe-se esta é uma ferramenta de grande flexibilidade e certamente indispensável para trabalhos com maior complexidade no que se diz respeito ao provimento de QoS. O ponto fraco em relação ao Dummynet é quanto a instalação que, de certa forma, é mais trabalhosa e é necessária de mais atenção para o perfeito funcionamento. Isto já foi observado e versões mais recentes do FreeBSD deverão trazer maiores facilidade.

O principal problema encontrado foi quanto ao ambiente de testes. Ao longo dos experimentos percebeu-se que a topologia era totalmente inadequada, pois os problemas gerados por conflitos (ou colisões) prejudicaram muito os resultados. O Anexo VI trás uma sugestão de uma topologia que apresentaria melhor resultados. Com esse fato torna-se mais visível a necessidade de mais laboratórios e equipamentos a serem disponibilizados aos alunos de graduação.

5.2 Trabalhos Futuros

Como trabalhos futuros sugere-se:

1. Repetir os experimentos em um ambiente mais adequado.
2. Um estudo mais detalhado de cada ferramenta observando seus vários outros recursos.
3. Uma análise das outras ferramentas disponíveis e que não foram escolhidas.
4. Utilizar meios mais precisos de medição e geração de tráfego além de isolar o problema quanto a fatores externos.
5. Testar, não só largura de banda e priorização, com outros parâmetros de qualidade de serviço como tamanho dos buffers das filas de entrada e saída, algoritmos de tratamento de fila, controle de admissão, atrasos e variação do atraso (jitter), dentre outros.
6. Fazer um estudo estatístico, em ambientes reais, em grupos com características de tráfego semelhantes afim de encontrar padrões nas definições dos parâmetros de QoS para cada grupo. Esses padrões auxiliariam na configuração de um novo ambiente que se encaixe em um determinado grupo.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] Black, D et al **An Architecture for Differentiated Services**, Request for Comments 2475. Disponível em:<<http://www.ietf.com/rfc/rfc2475.txt>> 1998.
- [2] Braden, R. ; Clark, D. & Shenker, S. **Integrated Services in the Internet Architecture: an Overview**. Request for Comments 1633. Disponível em: <<ftp://www.ietf.org/rfc/rfc1633.txt>> 1994.
- [3] Cisco System Inc. **Quality of Service Overview**. Disponível em:<http://www.cisco.com/univercd/cc/td/doc/product/software/ios102/12cgcr/qos_c/qcintro.htm>
- [4] Comer, D. E. **Interligação em rede com TCP/IP – Princípios protocolos e Arquitetura**. Volume 1. Editora Campus. Rio de Janeiro, 1998
- [5] Dantas, M. A. R. **Tecnologias de Redes de Comunicação e Computadores**. Axcel Books. Rio de Janeiro, 2002
- [6] Ferguson, P. & Huston, G **Quality of Service**. Wiley Computer Publishing,1998.
- [7] Ferguson, P. & Huston, G., **Quality of Service in the Internet: Fact, Fiction, or Compromise?**, INET'98, julho 1998.
- [8] Ferguson, P. & Huston, G **Quality of Service: Delivering QoS on the Internet and Corporate Networks**. Wiley Computer Publishing,1999.
- [9] __, __ **FreeBSD Handbook: The FreeBSD Documentation Project**. 2003
- [10] Kamienski, C. A. **Qualidade de Serviço na Internet** . Pernambuco 1999.
- [11] Leiner, B. M. et al., **The Past and Future History of the Internet**, Communications of the ACM, Fevereiro 1997.

- [12] Lu, Guojun. **Communication and Computing for Distributed Multimedia Systems**. Artech House Inc, 1996.
- [13] Melo, Edison Tadeu Lopes. **Qualidade se Serviço em Redes IP com DiffServ: Avaliação através de Medições**. Dissertação 2001
- [14] Maia, Luiz Neto - **Administração de Sistemas FreeBSD – Módulo 1: Serviços Básicos**. Florianópolis, 2002
- [15] Rosen, E. et al., **Multiprotocol Label Switching Architecture**, Internet Draft, Disponível em: <<http://snad.ncsl.nist.gov/nistswitch/drafts/draft-ietf-mpls-arch-05.txt>> Abril 1999.
- [16] Shenker, S.; Partridge, C. & Guerin, R. **Specification of Guaranteed Quality of Service**, Request For Comments 2212. Disponível: <<ftp://www.ietf.org/rfc/rfc2212.txt>> 1997.
- [17] Stardust Technologies, Inc. **The Need for QoS** . White Paper Disponível em: <http://www.qosforum.com/white-papers/Need_for_QoS-v4.pdf> 1999.
- [18] Stardust Technologies, Inc. **QoS Protocols & Architectures**. White Paper Disponível em: <http://www.qosforum.com/white-papers/qosprot_v3.pdf> 1999.
- [19] Tanenbaum, A. S. **Redes de computadores** Tradução da 3ª edição. Editora Campus. Rio de Janeiro, 1997
- [20] Teitlebaum, B. & Hans, T. QoS Requirements for Internet2 Internet2 Technical Paper. Disponível em: <<http://www.internet2.edu/qos/may98Workshop/html/requirements.html>> 1998.
- [21] Wroclawski, J. **Specification of the controlled-Load Network element Service**, Request for Comments 2211. Disponível em:<<http://www.ietf.org/rfc/rfc2211.txt>> 1997.
- [22] Xiao, XiPeng; Ni, Lionel. **Internet QoS: A Big Picture**, IEEE Network

Magazine, Março/Abril, pp. 8-18, 1999.

- [23] Silva, Adelmo Jerônimo . **Arquiteturas Híbridas de QoS em Redes IP**. Dissertação 2003.
- [24] Cisco System Inc. **Internetworking Technology Overview**, Capítulo 26 - Disponível em: <<http://www.cisco.com>> 1999.
- [25] Vegesna, S. **IP Quality of Service**, Cisco Press, 2002.
- [26] Rizzo, Luigi **Dummynet: a simple approach to the evaluation of network protocols** Artigo 1997.

ANEXOS

Anexo I - BSD Copyright

BSD Copyright

Redistribution and use in source (SGML DocBook) and 'compiled' forms (SGML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (SGML DocBook) must retain the above copyright notice, this list of conditions and following disclaimer as the first lines of this file unmodified.
2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribuion.

Important: THIS DOCUMENTATION IS PROVIDED BY THE FREEBSD DOCUMENTATION PROJECT "ASIS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PUPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD DOCUMENTATION PROJECT BE LIABLE FOR ANYY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Anexo II – Geração de tráfego para congestionamento da rede com MGEN

Para geração do tráfego que congestiona a rede utilizou-se o software MGEN. O tráfego gerado é um stream UDP e tem como origem a máquina M1 na porta 5001 e destino M2 na porta 5002 conforme os arquivos de configuração:

GeraM1.cfg – Na máquina M1

```
0.0 ON 1 UDP SRC 5001 DST M2/5002 PERIODIC [800 1024]
300.0 OFF 1
```

Os valores em PERIODIC indica que serão gerados 800 pacotes por segundo e cada pacote com tamanho de 1024 bits. Isso gera uma tráfego de 8192Kbps.

CaptaM2.cfg – Na máquina M2

```
0.0 LISTEN UDP 5002
300.0 IGNORE UDP 5002
```

Para executar a geração de tráfego:

Em M1:

```
M1 # /MGEN/mgen input GeraM1.cfg
```

Em M2:

```
M2 # /MGEN/mgen input CaptaM2.cfg
```

Vale lembrar que o gerador de tráfego MGEN possui muitos outros recursos que não foram explorados por estar fora do escopo desse trabalho.

Anexo III – Geração de tráfego com NetPerf

Para gerar tráfego com a ferramenta NetPerf instala-se o software nas máquinas (M1 e M2). Existem várias opções de configuração, porém , optou-se pelos valores padrões, que geram um stream TCP que se adapta afim de obter a maior taxa possível.

Deve-se então executar em M1 (receptor):

```
M1 # /usr/local/netperf/netserver
```

Um processo ficará escutando na portaTCP 12865 esperando o transmissor.

E em M2 (transmissor):

```
M2 # /usr/local/netperf/netperf -H M1 - t 300
```

Onde “-H M1” indica o endereço IP da máquina receptora e “-t 300”, a duração da transmissão em segundos.

Anexo IV – Instalação do Dummynet

Para instalar o DUMMYNET deve-se recompilar o kernel do sistema e adicionar algumas opções:

A tarefa de recompilar o kernel só poderá ser feita se os arquivos fontes do mesmo estiverem presentes no diretório /usr/src/sys directory. Caso esse diretório não estiver presente, o primeiro passo é instalar os fontes. A maneira mais fácil é através do utilitário sysinstall.

```
# /stand/sysinstall
```

Este comando deve ser executado com o usuário root. Escolher a opção “Configure > Distributions > src > sys “. Outra maneira é através do download do pacote com os fontes ou cópia de um cd-rom

```
# mount /cdrom
# mkdir -p /usr/src/sys
# ln -s /usr/src/sys /sys
# cat /cdrom/src/ssys.[a-d]* | tar -xzf -
```

Após a instalação dos fontes do kernel, deve-se alterar o arquivo de configuração com as opções desejadas. Recomenda-se fazer uma cópia do arquivo genérico de configuração e alterá-lo de modo que o novo kernel atenda às necessidades.

```
# cp /usr/src/sys/i386/conf/GENERIC /usr/src/sys/i386/conf/MEUKERNEL
```

Editar o arquivo MEUKERNEL e incluir as novas opções. Para o Dummynet funcionar duas opções são necessárias:

```
# vi /usr/src/sys/i386/conf/MEUKERNEL
```

```
options    IPFIREWALL # Esta opção habilita o firewall ipfw
options    DUMMYNET # Esta opção habilita o Dummynet
```

Outras opções recomendadas são:

```
options    IPFIREWALL_VERBOSE          - habilita saída no firewall
```


options IPFIREWALL_VERBOSE_LIMIT - limita saída no firewall
options NMBCLUSTERS - define a quantidade de buffers de pacotes
options HZ=n - onde n define o timer da granulosidade

Após adicionar as opções, executar o utilitário config para gerar o código fonte do kernel:

```
# /usr/sbin/config MEUKERNEL
```

Mudar para o diretório de compilação:

```
# cd ../../compile/MEUKERNEL
```

Compilar o kernel.:

```
# make depend  
# make
```

Instalar o novo kernel:

```
# make install
```

Após a instalação do novo kernel deve-se reiniciar o computador para que as novas opções estejam habilitadas. As seguintes linhas aparecerão durante a inicialização da máquina.

*IP packet filtering initialized, divert disabled, rule-based forwarding enabled, default to deny,
logging limited to 1 packets/entry by default
DUMMYNET initialized (011031)*

Anexo V – Instalação do AltQ

Para realizar a instalação do AltQ também é necessário recompilar o kernel. Para isso os fontes devem estar instalados (operação já descrita na instalação do Dummynet). Presumindo que os fontes estão instalados, deve-se mudar para seu diretório:

```
# cd /usr/src
```

Através dos utilitários fetch ou wget realiza-se o download dos arquivos fontes do AltQ e o patch específico para a versão do FreeBSD utilizada :

```
# fetch ftp://ftp.csl.sony.co.jp/pub/kjc/altq-3.1.tar.gz  
# fetch ftp://ftp.csl.sony.co.jp/pub/kjc/sys-altq-freebsd-4.9-RELEASE.patch.gz
```

Deve-se então descompactar e extrair os arquivos:

```
# gzip -d altq-3.1.tar.gz  
# tar -xvf altq-3.1.tar
```

Aplica-se as correções (patch) necessárias:

```
# cp sys-altq-freebsd-4.9-RELEASE.patch.gz altq-3.1/sys-altq/  
# mkdir sys-altq  
# cd sys  
# tar cvf - . | (cd ../sys-altq; tar xf -)  
# cd /usr/src/sys-altq  
# patch -p < /usr/src/altq-3.1/sys-altq/ sys-altq-freebsd-4.9-RELEASE.patch.gz
```

Após a aplicação das correções, a seguinte mensagem é mostrada:

```
Hunk #1 succeeded at 924.  
Hunk #2 succeeded at 942.  
Hunk #3 succeeded at 1609.  
Hunk #4 succeeded at 2396.  
Hunk #5 succeeded at 2566.  
Hunk #6 succeeded at 2695.
```

Hunk #7 succeeded at 3199.

Cria-se o diretório altq a copia-se os arquivos gerados:

```
# mkdir altq
# cp /usr/src/altq-3.0/sys-altq/altq/* altq/
```

Um arquivo de configuração do kernel padrão para o altq é fornecido: ALTQ. Deve-se então juntar as configurações desse arquivo com as configurações atuais do kernel:

```
# cd i386/conf/
# cp ALTQ ALTQ.minhacopia
```

Mudar maxusers para 64

```
# vi ALTQ
```

Realizar to o procedimento de compilação e instalação do kernel (não esquecendo de fazer um backup do kernel atual):

```
# /usr/sbin/config ALTQ
# cd ../../compile/ALTQ
# make depend
# make clean
# make
# cp /kernel /kernel.minhacopia
# make install
```

Após a instalação deve-se reiniciar o computador:

```
# reboot
```

Terminar de configurar o altq:

```
# cd /usr/src/altq-3.1
# sh MAKEDEV.altq all
# make
# make install
```

Agora, é necessário criar um arquivo de configuração com políticas para cada interface de rede e salva-lo com o nome padrão "altq.conf" no diretório /etc.

Após a criação do arquivo de configuração o altq está apto a ser executado.

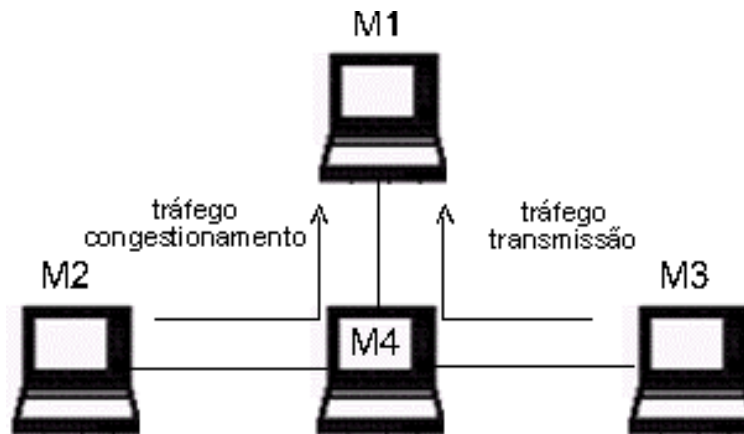
Para controle de banda e priorização basta executar o processo fantasma ou daemon.

```
# /usr/local/sbin/altqd
```

Para verificar as estatísticas usa-se o comando:

```
# /usr/local/bin/altqstat
```

Anexo VI – Sugestão de topologia



Com a adição de M4 consegue-se evitar problemas causados por colisões pois o uso do hub é dispensado. Para esse caso, as políticas de QoS devem ser implementadas em M4 que atuará com papel de roteador, podendo controlar as prioridades no tráfego no sentido M4-M1.

Anexo VII – Artigo sobre o Projeto

Software para Controle de Qualidade de Serviço no FreeBSD

Leonardo Luiz Ribeiro de Toledo
toledo@inf.ufsc.br

Universidade Federal de Santa Catarina - UFSC
Centro Tecnológico / Departamento de Informática e Estatística – CTC/INE
Curso de Bacharelado em Ciências da Computação - CCO
88.040-970 Florianópolis SC Caixa Postal: 479
Fone: (048) 231-9739 Fax: (048) 231-9770
Telex: (048) 240 UFSCBR

Resumo – Neste trabalho são apresentados alguns experimentos que envolvem ferramentas de software para controle de qualidade de serviço no sistema operacional FreeBSD. Alguns conceitos básicos de qualidade de serviço são apresentados assim como algumas características do sistema operacional FreeBSD. No final se apresenta os resultados obtidos com os experimentos.

Palavras-Chave – qualidade de serviço, FreeBSD, AltQ, Dummysnet

Abstract - In this work some experiments are presented that involve tools of software for quality of service control in the operational system FreeBSD. Some basic concepts of quality of service are presented as well as some characteristics of the operational system FreeBSD. In the end it presents the results gotten with the experiments.

Keywords - QoS, FreeBSD, AltQ, Dummysnet

1-Introdução

A cada dia cresce o número de serviços oferecidos via Internet. Dentre estes serviços pode-se citar: sites de conteúdo, comércio eletrônico, e-mails, troca de arquivos, streams de áudio e vídeo, entre outros. Com todas essas novas funcionalidades, a Internet está deixando de ser apenas uma ferramenta de auxílio para a produtividade da empresa e está se tornando parte do negócio.

Para transmitir todas essas informações, a Internet utiliza-se, desde seu início, do modelo do melhor esforço (Best Effort). Este modelo é baseado em filas do tipo FIFO (First In First Out) e não estabelece nenhum critério de tipo ou prioridade.

A até poucos anos, o modelo do melhor esforço conseguia atender as necessidades dos usuários da rede, mas com o surgimento de aplicações de tempo real, videoconferência e telefonia IP, este modelo se tornou obsoleto. Este tipo de aplicação necessita de uma série de requisitos para que possa funcionar corretamente. Em uma aplicação de transferência de arquivo não pode haver perda de pacotes pois isto comprometeria a integridade do arquivo, já se tratando uma conversação por VoIP, pequenas perdas são aceitas, já um atraso muito grande, inviabiliza a aplicação [11]. Tendo isto em mente pode se afirmar que aplicações diferentes requerem tratamentos diferentes.

Com a necessidade de suprir estas deficiências, novos protocolos e padrões estão sendo adotados. Eles são chamados de Arquiteturas de Qualidade de Serviço [15]. Dentre essas arquiteturas destacam-se, por sua grande aceitação, a de Serviços Integrados (IntServ) e a de

Serviços Diferenciados (DiffServ). Ambas arquiteturas contam com grupos de trabalho no IETF (Internet Engineering Task Force).

Esta tecnologia deve estar presente nos roteadores, pois são estes, os responsáveis pelo encaminhamento dos pacotes no nível inter-rede. O grande problema é que fora de um organização o acesso aos roteadores é restrito às companhias de telecomunicação.

Como implementar as arquiteturas de QoS definidas pela IETF é impraticável sem o controle dos pontos intermediários, pode-se adotar o uso de ferramentas de software para classificar, restringir e/ou priorizar o tráfego de certas informações no computador de borda. Pode-se dizer, então, que estas são ferramentas para controle de qualidade do serviço, QoS.

Paralelo a isto está o sistema operacional FreeBSD, desenvolvido pela Universidade da Califórnia, Berkeley, e que vem se popularizando cada vez mais. Considerado uma referência na implementação dos protocolos da família TCP/IP, o sistema é tem uma implementação robusta, focada para a área de redes de computadores. Não exige muito poder de processamento e é compatível com os SO's da família Linux. Essas características o torna uma alternativa muito interessante para prover soluções da área de redes.

Esse artigo mostra um pequeno estudo das ferramentas de software disponíveis e que vêm sendo utilizadas no sistema operacional FreeBSD com fins de prover qualidade de serviço.

2 Qualidade de Serviço

Já tem algum tempo que o modelo do melhor esforço não atende mais as necessidades de todas aplicações que se utilizam do protocolo IP. É preciso fornecer mecanismos à rede para que esta possa atender as novas necessidades das aplicações, oferecendo uma boa qualidade nos serviços oferecidos.

Qualidade de serviço é algo difícil de definir. Em geral, assume significados diferentes para pessoas distintas. A pouco tempo, QoS tinha conotações voltadas para algumas tecnologias de rede, principalmente a ATM, porém o termo já esta sendo mais amplamente utilizado. Hoje já se fala em QoS em sistemas de arquivo, gerência de memória, além de outros.

Novas aplicações têm a tendência de requererem grandes taxas de transmissão o que provocará um aumento de carga nos backbones Internet provocando um certo congestionamento. De fato, existe uma relação entre as aplicações e a rede. As aplicações estimulam cada vez mais melhorias na rede, e a rede, a medida que surgem melhorias, estimulam o surgimento de novas aplicações. Esta relação permite supor que no futuro as aplicações exigiram redes com alto desempenho e com qualidade.

São inúmeras as definições de qualidade de serviço encontradas na literatura, dentre elas pode-se citar:

De maneira semelhante Ferguson [4] definiu QoS como: "Capacidade da rede de fornecer tratamento especial a certos tipos de tráfego previsivelmente".

"Habilidade de um elemento da rede, seja uma aplicação, host, roteador ou outro dispositivo, ter algum nível de garantia que seu tráfego e exigências de serviços podem ser satisfeitas." [6]

"QoS é um termo que freqüentemente possui duplo significado: refere-se ao desempenho de uma rede em relação as necessidades de uma aplicação e ao conjunto de tecnologias que permitem a rede satisfazer estas garantias de desempenho" [13]

"Termo empregado para definir os parâmetros específicos necessários para uma determinada aplicação do usuário. Estes parâmetros de serviço podem ser definidos em termos de largura de banda, latência e jitter, visando que a aplicação possa obter uma melhor qualidade ao longo da rede" [3]

Neste trabalho, serão analisadas ferramentas que de alguma forma contribuam com a diferenciação do tratamento do tráfego para permitir melhor utilização da rede e garantir a funcionalidade de aplicações mais críticas. Em outras palavras, ferramentas que se opõem a idéia do melhor esforço serão consideradas ferramentas de QoS.

2.1 Parâmetros de QoS

Ao proporcionar garantias de transmissão para certos tipos de serviço, pode se dizer que se está provendo QoS. Deve-se, então, definir alguns parâmetros que serão utilizados como requisitos para uma aplicação que deseje uma transmissão com QoS [5]. Dentre esses parâmetros destacam-se:

- Atraso fim-a-fim: É o tempo despendido entre o envio de um pacote do emissor, através da rede, até o receptor. Em um roteador, o atraso é o tempo entre a recepção de um pacote e sua transmissão, também referido como atraso de propagação [8].
- Variação do atraso (jitter): É a variação no atraso fim-a-fim em redes de pacotes. Variações muito acentuadas do retardo prejudicam a qualidade do serviço oferecido a algumas aplicações.
- Largura de banda: É uma medida de capacidade de transmissão de dados ao longo do tempo, ou seja, a máxima taxa de transmissão de dados que pode ser sustentada entre dois pontos finais.
- Confiabilidade: Como uma propriedade dos sistemas de transmissão, pode ser vista como a taxa de erros do meio físico.

Um serviço com qualidade pode ser visto como aquele que provê baixo atraso e variação do atraso, grande quantidade de banda e muita confiabilidade. Quando se refere a QoS na Internet, no entanto, a questão diz respeito à diferenciação em uma ou mais dessas métricas de qualidade para uma determinada categoria de tráfego. Na prática, os provedores e operadoras de telecomunicação baseiam a qualidade dos serviços oferecidos no preço que o cliente pode pagar, quanto maior a qualidade exigida pelo cliente, mais ele terá que pagar.

2.2 Melhor esforço (Best Effort)

A Internet atual utiliza um modelo de serviço de melhor esforço [7], que significa que todos os usuários e aplicações têm o mesmo tratamento nos roteadores no caminho entre origem e destino dos pacotes. Em situações de congestionamento, roteadores guardam pacotes em filas na ordem estrita de chegada (FIFO). Quando a capacidade da fila se esgota, os pacotes são simplesmente descartados. Esse modelo apresenta uma grande simplicidade e robustez, que foi o motivo do sucesso da Internet. No entanto, não permite o desenvolvimento de aplicações avançadas e a diferenciação de serviços entre usuários.

2.3 Mecanismos IETF para QoS em Redes IP

A IETF tem dado grande atenção à introdução de QoS na Internet e propôs alguns modelos de serviços e mecanismos para atenderem essa demanda. Entre eles, com mais destaque estão: Arquitetura de Serviços Integrados, Arquitetura de Serviços Diferenciados e MPLS apresentados a seguir. Uma característica comum deste tipo de arquitetura é o fato delas proverem QoS fim-a-fim⁴. Isto significa que é preciso que todos os roteadores entre origem e destino implementem a arquitetura.

2.3.1 Serviços Integrados (IntServ)

A Arquitetura de Serviços Integrados (IntServ) [2] sugere uma extensão para a atual arquitetura da Internet de modo a oferecer novos serviços. O foco principal de IntServ são as aplicações de

⁴ QoS fim-a-fim é a habilidade da rede em prover o serviço requerido por um tráfego específico de uma extremidade a outra da rede (CISCO, 1999)

tempo real, que necessitam de garantias rígidas de QoS para funcionar corretamente. Para isso, foram propostas duas novas classes de serviços, além do serviço de melhor esforço existente:

- Serviço garantido (Guaranteed Service): [10] fornece limites rígidos aos quais os pacotes estarão condicionados nos roteadores, garantindo assim, um limite do atraso e da taxa de transmissão.
- Serviço de Carga Controlada (Controlled Load Service): [14] fornece um fluxo de dados com QoS muito próxima que o mesmo fluxo receberia de uma rede não sobrecarregada. Usa, porém, um controle de admissão para assegurar que este serviço será recebido mesmo que a rede esteja congestionada.

2.3.2 Serviços Diferenciados (DiffServ)

A arquitetura Serviços Diferenciados (DiffServ) [1] se destaca pela sua escalonabilidade que pretende ser obtida através da agregação de fluxos e separação das funções dos roteadores de borda e de núcleo nas grandes redes de backbone. As redes que implementam serviços diferenciados são chamadas Domínios DS.

Domínios DS negociam entre si contratos que visam o provimento de garantias mínimas de QoS para as aplicações dos usuários. Todos os pacotes que fluem de um domínio para outro são fiscalizados (policiados) nos roteadores de borda para verificar sua conformidade com os contratos.

2.3.3 Multiprotocol Label Switching (MPLS)

Quando um roteador recebe um pacote ele faz uma busca na sua tabela de roteamento e então, baseado no endereço IP do pacote, ele decide para onde enviá-lo. Essa busca pode levar bastante tempo, dependendo do tamanho da tabela de cada roteador. MPLS rompe com esse paradigma, usando um rótulo de tamanho fixo a partir do qual o roteador decide por onde enviar os pacotes.

MPLS [9] é na realidade a padronização de várias implementações existentes no mercado da técnica de encaminhamento de pacotes baseado em rótulos (label switching).

2.4 Qualidade de Serviço em Redes Corporativas

Habilitar QoS requer a cooperação de todos os níveis da rede, desde o topo até a base, bem como de todos os elementos da rede de fim-a-fim. Qualquer garantia de QoS será no máximo tão boa quanto o elo mais fraco na “cadeia” entre origem e destino [12].

Em redes corporativas, geralmente, tem-se a contratação de um link dedicado com alguma companhia de telecomunicação e/ou provedor de Internet. São vários o tipos de contrato, seus valores variam de acordo com a tecnologia utilizada e taxa de transmissão disponibilizada. O contrato estabelece uma taxa máxima de transmissão para envio (upload) e uma taxa máxima para recepção (download), números IP válidos além de outros parâmetros. A esse tipo de contrato dá-se o nome de SLA (Service Level Agreement).

Com esse tipo de acesso, não se tem garantia de QoS utilizando arquiteturas fim-a-fim. Para fora da corporação a Internet é vista como uma grande nuvem onde persiste o modelo do melhor esforço. Sem garantias é possível controlar QoS de duas maneiras [8]:

- **Reserva de Recursos:** os recursos da rede são divididos de acordo com os requisitos da aplicação, e sujeitos à política de administração de largura de banda.
- **Priorização:** o tráfego da rede é classificado e os recursos de rede são divididos de acordo com critérios de políticas de administração da largura de banda.

Essas duas maneiras podem ser implementadas juntas ou separadas acomodando diferentes exigências operacionais em diferentes contextos de redes.

2.5 Mecanismos provedores de QoS

Os mecanismos usados para prover QoS atuam de duas maneiras: preventiva e reativa. Na forma preventiva são implementados controle de admissão, adequação, políticas de tráfego e disciplina de filas. Na forma reativa, os mecanismos executam ações pré-programadas ao momento em que se detecta qualquer situação que provoque desvanecimento do nível de qualidade dos serviços, tais como os mecanismos de controle de fluxo do TCP e o gerenciamento ativo de filas.

2.5.1 Algoritmos de tratamento de filas

A política de tratamento de filas é um ponto fundamental para se obter êxito na busca de QoS. A eficiência no processamento do tráfego na rede deve estar adequada para situações de grande tráfego, buscando sempre evitar congestionamentos. Um congestionamento é quando a capacidade de saída de pacotes é menor do que a taxa de chegada, ou seja, um nó da rede não consegue processar seu tráfego de forma que vão acumulando pacotes a serem processados. Esses pacotes ficam em fila esperando seu processamento.

Uma fila é um espaço na memória reservada para armazenar pacotes a serem processados. Como é um espaço limitado, se a fila encher, alguns pacotes serão perdidos. Além disso, o tempo que um pacote fica na fila significa atraso na transmissão. Existem uma série de algoritmos para tratamento das filas, a seguir, é apresentado um resumo do funcionamento de alguns dos principais algoritmos [8]:

2.5.2 Mecanismos de Controle de Congestionamento

Nas redes atuais pode se dizer que é mais provável que as perdas de pacotes ou os atrasos ocorram por congestionamento do que por algum defeito no hardware. O fato é tanto as perdas como atrasos prejudicam as aplicações chegando até ao ponto de torná-las inviáveis. Com fins de amenizar as conseqüências causadas pelos picos de tráfego, que congestionam a rede, surgiram alguns mecanismos.

2.5.3 Políticas de Tráfego, Adequação e Controle de Admissão

Existem os casos onde o congestionamento se dá por períodos mais longos, e logo, alguns pacotes impreterivelmente serão descartados. Através de ações preventivas pode-se chegar a uma solução que possibilite um amplo controle do tráfego que será admitido para o interior da rede. Dentre essas ações destacam-se as Políticas de Tráfego, Adequação e Controle de Admissão.

3 O Sistema Operacional FreeBSD

Toda aplicação necessita de uma base sólida para ser executada com correção. Os serviços de redes na maioria das vezes devem estar disponíveis a todo momento. Para isso o sistema operacional precisa oferecer um ambiente robusto e flexível garantindo maior segurança, disponibilidade e confiabilidade do sistema. Este capítulo apresenta, de maneira breve, características do sistema operacional FreeBSD, um S.O. que fornece, além do que já foi mencionado, uma solução gratuita, com pouca exigência de hardware e boa performance. Dentre as várias características desse sistema, destacam-se:

- Código aberto: acesso completo ao seu código fonte
- Multitarefa: capacidade de realizar várias tarefas ao mesmo tempo com menos recursos de hardware
- Multi-usuário: suporta vários usuários compartilhando recursos e aplicativos simultaneamente.
- Capacidade de Rede TCP/IP: foi o sistema operacional pioneiro na implementação desse protocolo. É capaz de atuar como um servidor de serviços de Internet, estabelecer rotinas de roteamento e firewall.

- Memória Protegida: diferentes aplicações ou usuários não interferem entre si. Ou seja, mesmo que uma aplicação trave, isso não atrapalhará os outros processos ou usuários que estão utilizando o sistema.
- Módulos de compatibilidade Binária Linux, AT&T SysVR4 e SCO Unix: com a adição desses módulos o sistema será compatível com os demais sistemas relacionados.
- Multi processamento Simétrico: implementa suporta a hardware com vários processadores.
- Livre distribuição: software e documentação com distribuição livre, devendo apenas obedecer o BSD Copyright .

Outra característica importante do FreeBSD é a forma que são tratadas suas versões. A versão que esta sendo desenvolvida recebe o nome de CURRENT. Em intervalos de tempo (3 ou 4 vezes ao ano) é lançada uma versão em que as pendências mais críticas são arrumadas e então é disponibilizada para download com o nome RELEASE. Após vários testes realizados sobre a versão RELEASE, os erros são reportados e corrigidos. Quando não se encontra mais erros em uma nesta versão, é então disponibilizada uma nova versão chamada STABLE. Como a versão STABLE já foi testada de forma intensiva, têm-se uma garantia de que o sistema é estável e seguro.

4 Experimentos e Resultados

Esse projeto tem como objetivo aprofundar conceitos no que diz respeito a qualidade de serviço (QoS) em redes de computadores em seu âmbito geral e também a exploração prática desse tema no sistema operacional FreeBSD.

Espera-se realizar um estudo aprofundado dos recursos do sistema operacional FreeBSD no que diz respeito a parte de conectividade utilizando o protocolo TCP/IP e aplicação de parâmetros de QoS em suas configurações.

Através de uma pesquisa feita no site do S.O. FreeBSD e na Internet pretende-se verificar a utilização das ferramentas que fornecem suporte à qualidade de serviço.

4.1 Ferramentas

Existem muitas ferramentas disponíveis, ou sendo desenvolvidas, que atendam às necessidades de se prover QoS Seguindo as idéias básicas dos modelos IntServ e DiffServ, escolhe-se respectivamente os softwares Dummynet e AltQ. O Dummynet implementa pipes (dutos) com características que permitem reservar recursos. O AltQ implementa uma série de algoritmos de gerenciamento de filas que traz a possibilidade priorizar determinados tipos de tráfego. Nas próximas sessões procura-se apresentar de forma mais completa estes dois softwares.

4.1.1 DUMMYNET

O Dummynet foi proposto por Luigi Rizzo como uma alternativa aos simuladores de redes existentes até então. A justificativa dada [16], era que, parâmetros operacionais como largura de banda, atrasos e tamanho de filas eram fáceis de se controlar nos simuladores, porém, eles só ofereciam essas possibilidades no modelo simulado não havendo possibilidade de simular em um sistema real.

- O Dummynet implementa algumas modificações na pilha do protocolo o que permite experimentos em um sistema real. Ele trabalha através de interceptação das comunicações na camada em teste e simulação de efeitos de filas finitas, limitação de banda e atrasos de comunicação. Como é implementado na pilha real do sistema, permite o uso de geradores de tráfego além do uso de tráfego real. Ele trabalha em conjunto com o firewall padrão da maioria das versões 4.x do FreeBSD, o IPFW.

4.1.2 ALTQ (Alternate Queueing)

Como foi visto na seção 2.5, o controle das filas é um elemento essencial para o controle de tráfego, acontece que a maioria dos sistemas utilizam somente o algoritmo FIFO. O ALTQ, é um framework que permite o uso de uma série de outras disciplinas de controle de fila.

Tendo como sistema de origem o OPENBSD, o ALTQ já foi implementado como uma simples extensão do kernel do FreeBSD, além de algumas correções nos drivers de alguns dispositivos. Vários algoritmos de gerência de fila são implementados tais como CBQ, RED, WFQ, HFSC, PRIQ, JoBS, RIO e Blue.

4.2 Experimentos

Afim de verificar o funcionamento das ferramentas e também observar o comportamento em diferentes situações realiza-se alguns experimentos.

4.2.1 Ambiente

Para a realização dos experimentos, foram utilizados três computadores:

M1 – Pentium III 450 MHz com 96 Mbytes de memória RAM, placa de rede VIA VT6105 Rhine III 10/100BaseTX, SO FreeBSD v 4.9 STABLE

M2 – Pentium I 133MHz com 64Mbytes de memória RAM, placa de 3Com 3c905-TX Fast Etherlink, SO FreeBSD v 4.9 STABLE

M3 – Pentium III 600MHz com 128Mbytes de memória RAM, placa de rede SIS 900 Fast Ethernet, SO MS Windows 98 SE.

A Figura 1 apresenta a topologia do ambiente

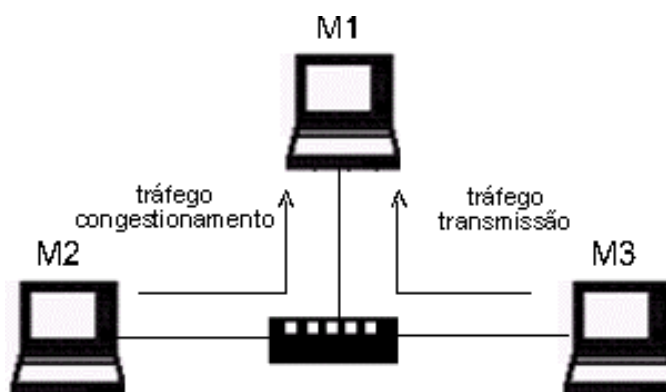


Figura 1 – Topologia do ambiente de experimentos

As ferramentas serão observadas em M1. M2 servirá para, em algumas circunstâncias, gerar um tráfego afim de ocupar ou congestionar a rede. M3 será usado para realizar as transmissões via SFTP. Os três computadores estão conectados, via par-trançado, a um hub padrão ethernet, o que forçara as placas de rede a trabalharem a até 10Mbps. Configurou-se as placas fornecendo a cada uma um endereço IP e através do utilitário ping, verificou-se as conexões. Foram instaladas em M1 e M2 as ferramentas MGEN e NetPerf, que geram tráfego entre as duas máquinas.

4.2.2 Experimento com DUMMYNET

O experimento consiste em realizar a transmissão de um arquivo de 10 Mbytes de M3 até M1 utilizando o software SSH Secure Shell observando diversas situações. Cada transmissão foi repetida 7 vezes para cada situação, o maior e o menor tempo são descartados. O tempo de transmissão então é calculado com a média aritmética simples dos 5 tempos restantes. Com esse procedimento tenta-se excluir quaisquer anormalidades que possam ocorrer em determinadas transmissões ficando com um valor médio mais próximo possível do real.

Situações:

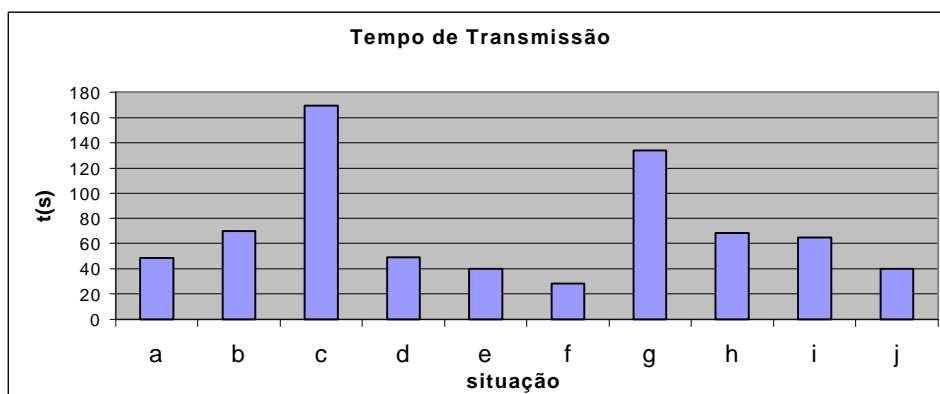
- a) Rede livre e sem pipe
- b) Rede congestionada e sem pipe
- c) Rede livre e com pipe de 800Kbps

- d) Rede livre e com pipe de 2400Kbps
- e) Rede livre e com pipe de 4000Kbps
- f) Rede livre e com pipe de 8000Kbps
- g) Rede congestionada e com pipe de 800Kbps
- h) Rede congestionada e com pipe de 2400Kbps
- i) Rede congestionada e com pipe de 4000Kbps
- j) Rede congestionada e com pipe de 8000Kbps

Resultados

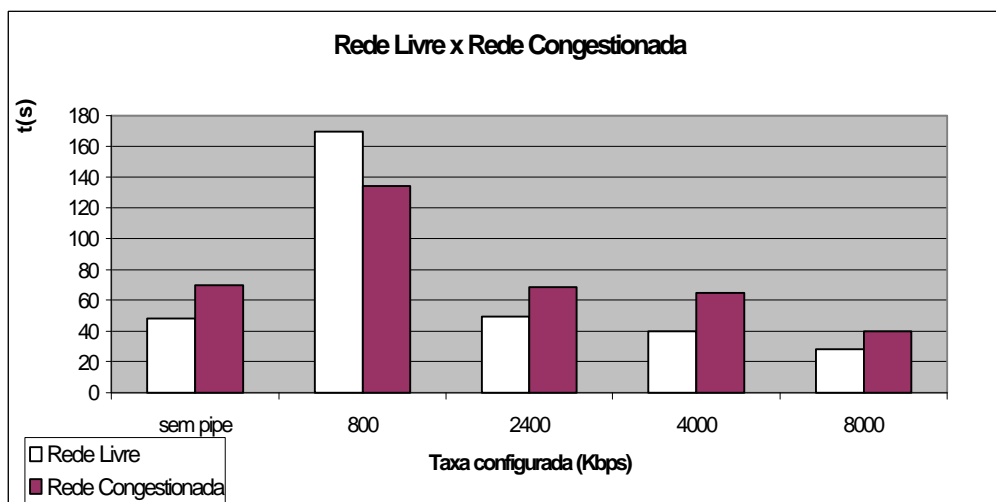
A figura 2 demonstra graficamente o comportamento geral dos tempos de transmissão.

Figura 2 – Tempos de transmissão para cada situação



A figura 3 trás uma comparação entre tempos em que a rede não tinha tráfego e os tempos que estava congestionada para os mesmo valores atribuídos aos pipes.

Figura 3 – Comparação dos tempos de transmissão com “rede livre” versus “Rede Congestionada”



Se considerarmos que cada bit transmitido seja um bit de dado (payload) pode-se afirmar que as taxas de transferência ideais são iguais às taxas nominais especificadas ou configuradas. Ou seja, uma rede padrão ethernet teria uma taxa uma taxa de transmissão 10Mbits por segundo. Claro que além de ignorar as informações de cabeçalho dos pacotes, não está se levando em conta outra série de fatores que afastam essa taxa da realidade (perdas, colisões, handshake do protocolo, etc). Utiliza-se esses valores apenas para observar o aumento que seria esperado com as diferenças entre as taxas configuradas e as taxas obtidas. A figura 4 mostra essa comparação.

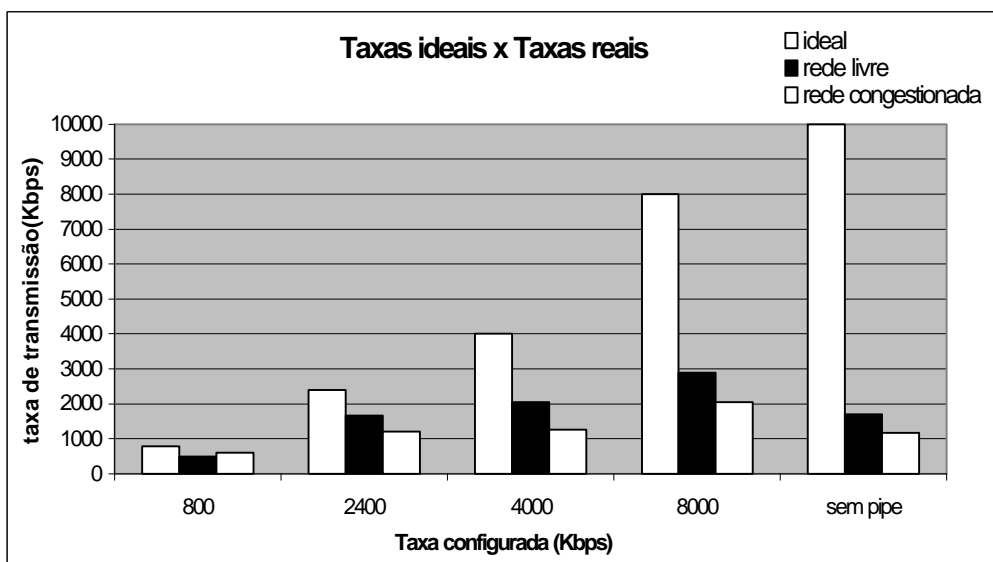


Figura 4 – Comparação de taxas de transmissão ideais com as taxas obtidas

O que se pode observar nessa comparação é o comportamento irregular da taxa real quanto à taxa ideal. Por exemplo, na transmissão com rede livre com o pipe configurado a 4000Kbps a taxa média encontrada foi 2056Kbps. Supõem-se então que ao dobrar o valor do pipe para 8000Kbps, a taxa real aumente proporcionalmente, ou seja, também dobre. Isso não acontece, a taxa da transmissão foi de 2888Kbps, um aumento de 40%, onde teoricamente, se esperava 100%.

Apesar da reserva de recursos ser a mesma, observou-se uma variação nas taxas de transferência em caso de congestionamento na rede. Acredita-se que este fato está relacionado com o protocolo de acesso ao meio, pois durante os experimentos com a rede congestionada observou-se um alto índice de colisões indicados pelo hub.

4.2.3 Experimentos com AltQ

Para utilizar o AltQ deve-se configurar o arquivo `/etc/altq.conf`. Nesse arquivo pode-se configurar para uma mesma rede uma infinidade de opções diferentes. O AltQ permite o uso de diversos algoritmos para tratamento de fila (BLUE, CBQ, FIFO, HFSC, PRIQ, RED, RIO, WFQ, JoBS e CDNR). Além disso, permite atribuir largura de banda, definir prioridades para determinados tráfegos, tamanho do pacotes, atraso máximo permitido entre outros.

Esta fora do escopo deste trabalho analisar detalhadamente a ferramenta e explorar todas essas características do software.

Para verificar o funcionamento da ferramenta, faz-se transmissões de um arquivo de 10Mbytes em diversas situações da máquina M3 para a máquina M1. Instala-se a ferramenta AltQ em M1 e cria-se um arquivo `/etc/altq.conf`.

As situações observadas:

- a) 0,5% da banda ou 50kbps com rede livre
- b) 0,5% da banda ou 50kbps com tráfego Netperf
- c) 1% da banda ou 100kbps com rede livre
- d) 1% da banda ou 100kbps com tráfego Netperf
- e) 3% da banda ou 300kbps com rede livre
- f) 3% da banda ou 300kbps com tráfego Netperf
- g) 5% da banda ou 500kbps com rede livre
- h) 5% da banda ou 500kbps com tráfego Netperf
- i) 10% da banda ou 1Mbps com rede livre

j) 10% da banda ou 1Mbps com tráfego Netperf

Resultados

Observando a figura 5, gráfico comparativo dos tempos de transmissão com e sem o tráfego gerado pelo NetPerf, observa-se que o tempo de transmissão, quando não se tem tráfego NetPerf, permanece praticamente constante. Isto se deve ao fato de que a reserva de uma classe pode ser usada por outra se esta não estiver sendo utilizada, ou seja, os recursos não ficam estagnados à espera de utilização.

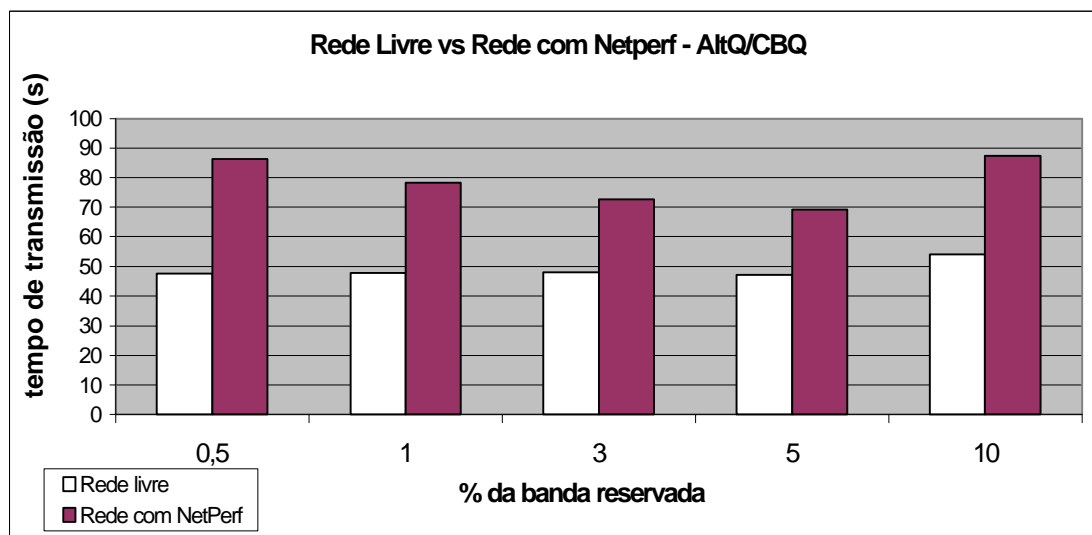


Figura 5- Comparação de tempos de transmissão utilizando AltQ

Pode-se também, perceber um comportamento inesperado quando o valor da reserva atingiu 10%, a tendência de diminuir o tempo da transmissão quando em uma situação de reserva maior não foi observada. Repetindo várias vezes e não alterando resultados percebeu-se que o led indicador de colisão do hub permanecia praticamente aceso durante toda transmissão.

5 Conclusões

As ferramentas escolhidas foram observadas em características diferentes. No Dummysnet observou-se sua semelhança ao paradigma do IntServ (reserva de recursos, de forma bem simplificada). A possibilidade de fazer reservas de banda garantindo que serviços mais importantes não sejam prejudicados por um congestionamento na rede é um ponto forte. Seu grande problema é o fato de ser estático, ou seja, os valores ficam fixos aos configurados, não importando a situação que a rede se encontre. A reserva de recursos pode causar congestionamento mesmo com banda disponível, como em uma situação em que um tipo de tráfego com grande reservas não estar sendo utilizado e outro tipo, com reservas menores, estar sendo muito requisitado. Nesse caso recomenda-se evitar pequenas reservas deixando o melhor esforço tratar o tráfego.

Dentre as vantagens observadas do Dummysnet pode se ressaltar sua facilidade de instalação e configuração. O software foi desenvolvido para o FreeBSD e nas versões mais atuais basta somente adicionar o modulo e recompilar o kernel do sistema. Além do uso de pipes para priorizar certos tipos de tráfego, pode se utiliza-los para deteriorar outros tipos. Isso se torna útil quando a intenção é desestimular o uso de alguma aplicação, o que é muito interessante para empresas que têm problemas com funcionários acessando vídeos, rádios ou fazendo uso de software que compartilham arquivos (Kaaza, E-Donkey, Morpheus, etc).

O AltQ demonstrou uma característica muito interessante quando se fala em reserva de recursos, o fato de um recurso reservado para uma classe poder ser utilizado por uma outra classe quando a primeira não estiver precisando, nos traz uma idéia de prioridade e uso eficaz da rede. Isso também evita desperdícios.

Pela documentação observada da ferramenta AltQ, percebe-se esta é uma ferramenta de grande flexibilidade e certamente indispensável para trabalhos com maior complexidade no que se diz respeito ao provimento de QoS. O ponto fraco em relação ao Dummynet é quanto a instalação que, de certa forma, é mais trabalhosa e é necessária de mais atenção para o perfeito funcionamento. Isto já foi observado e versões mais recentes do FreeBSD deverão trazer maiores facilidade.

O principal problema encontrado foi quanto ao ambiente de testes. Ao longo dos experimentos percebeu-se que a topologia era totalmente inadequada, pois os problemas gerados por conflitos (ou colisões) prejudicaram muito os resultados. Com esse fato torna-se mais visível a necessidade de mais laboratórios e equipamentos a serem disponibilizados aos alunos de graduação.

6 Trabalhos Futuros

Como trabalhos futuros sugere-se:

1. Repetir os experimentos em um ambiente mais adequado.
2. Um estudo mais detalhado de cada ferramenta observando seus vários outros recursos.
3. Uma análise das outras ferramentas disponíveis e que não foram escolhidas.
4. Utilizar meios mais precisos de medição e geração de tráfego além de isolar o problema quanto a fatores externos.
5. Testar, não só largura de banda e priorização, com outros parâmetros de qualidade de serviço como tamanho dos buffers das filas de entrada e saída, algoritmos de tratamento de fila, controle de admissão, atrasos e variação do atraso (jitter), dentre outros.
6. Fazer um estudo estatístico, em ambientes reais, em grupos com características de tráfego semelhantes afim de encontrar padrões nas definições dos parâmetros de QoS para cada grupo. Esses padrões auxiliariam na configuração de um novo ambiente que se encaixe em um determinado grupo.

Referências Bibliográficas

- [1] Black, D et al **An Architecture for Differentiated Services**, Request for Comments 2475. Disponível em:<<http://www.ietf.com/rfc/rfc2475.txt>> 1998.
- [2] Braden, R. ; Clark, D. & Shenker, S. **Integrated Services in the Internet Architecture: an Overview**. Request for Comments 1633. Disponível em: <<ftp://www.ietf.org/rfc/rfc1633.txt>> 1994.
- [3] Dantas, M. A. R. **Tecnologias de Redes de Comunicação e Computadores**. Axcel Books. Rio de Janeiro, 2002
- [4] Ferguson, P. & Huston, G **Quality of Service**. Wiley Computer Publishing,1998.
- [5] Ferguson, P. & Huston, G., **Quality of Service in the Internet: Fact, Fiction, or Compromise?**, INET'98, julho 1998.
- [6] Ferguson, P. & Huston, G **Quality of Service: Delivering QoS on the Internet and Corporate Networks**. Wiley Computer Publishing,1999.
- [7] Leiner, B. M. et al., **The Past and Future History of the Internet**, Communications of the ACM, Fevereiro 1997.

- [8] Melo, Edison Tadeu Lopes. **Qualidade se Serviço em Redes IP com DiffServ**: Avaliação através de Medições. Dissertação 2001
- [9] Rosen, E. et al., **Multiprotocol Label Switching Architecture**, Internet Draft, Disponível em: <<http://snad.ncsl.nist.gov/nistswitch/drafts/draft-ietf-mpls-arch-05.txt>> Abril 1999.
- [10] Shenker, S.; Partridge, C. & Guerin, R. **Specification of Guaranteed Quality of Service**, Request For Comments 2212. Disponível: <<ftp://www.ietf.org/rfc/rfc2212.txt>> 1997.
- [11] Stardust Technologies, Inc. **The Need for QoS** . White Paper Disponível em: <http://www.qosforum.com/white-papers/Need_for_QoS-v4.pdf> 1999.
- [12] Stardust Technologies, Inc. **QoS Protocols & Architectures**. White Paper Disponível em: <http://www.qosforum.com/white-papers/qosprot_v3.pdf> 1999.
- [13] Teitlebaum, B. & Hans, T. QoS Requirements for Internet2 Internet2 Technical Paper. Disponível em: <<http://www.internet2.edu/qos/may98Workshop/html/requirements.html>> 1998.
- [14] Wroclawski, J. **Specification of the controlled-Load Network element Service**, Request for Comments 2211. Disponível em: <<http://www.ietf.org/rfc/rfc2211.txt>> 1997.
- [15] Xiao, XiPeng; Ni, Lionel. **Internet QoS: A Big Picture**, IEEE Network Magazine, Março/Abril, pp. 8-18, 1999.
- [16] Rizzo, Luigi **Dumynet: a simple approach to the evaluation of network protocols** Artigo 1997.