UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO - ESCOLA DE ENGENHARIA
MESTRADO EM ENGENHARIA DE TELECOMUNICAÇÕES

MARCUS VINICIUS DE ALMEIDA FERREIRA

CLASSIFICAÇÃO DE FLUXOS DE VOZ BASEADA NA IMPORTÂNCIA DA CHAMADA

NITERÓI
2009

MARCUS VINICIUS DE ALMEIDA FERREIRA

CLASSIFICAÇÃO DE FLUXOS DE VOZ BASEADA NA IMPORTÂNCIA DA CHAMADA

Dissertação apresentada ao Curso de Pós-Graduação “Stricto Sensu” em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de concentração: Sistemas de Telecomunicações.

Orientadora: Profa. Dra. DÉBORA CHRISTINA MUCHALUAT SAADE

Niterói
2009

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

F383 Ferreira, Marcus Vinicius de Almeida.
Classificação de fluxos de voz baseada na importância da chamada / Marcus Vinicius de Almeida Ferreira. – Niterói, RJ : [s.n.], 2009.
120 f.

Orientadora: Débora Christina Muchaluat Saade.
Dissertação (Mestrado em Engenharia de Telecomunicações) - Universidade Federal Fluminense, 2009.

1. Sistema de telecomunicação. 2. Voz. 3. Qualidade de serviço. 4. Telefonia por Internet. 5. Mecanismos de escalonamento. 6. Rede em malha sem fio. I. Título.

CDD 621.3811

CLASSIFICAÇÃO DE FLUXOS DE VOZ BASEADA NA IMPORTÂNCIA DA CHAMADA

MARCUS VINICIUS DE ALMEIDA FERREIRA

Dissertação apresentada ao Curso de Pós-Graduação “Stricto Sensu” em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de concentração: Sistemas de Telecomunicações.

Aprovada em Abril/2009:

_____
Profa. Dra. Débora Christina Muchaluat Saade
Universidade Federal Fluminense
(Orientadora)

_____
Prof. Dr. Carlos Alberto Malcher Bastos
Universidade Federal Fluminense

_____
Prof. Dr. Antônio Tadeu Azevedo Gomes
Laboratório Nacional de Computação Científica

Niterói, 29 de abril de 2009.

Dedicatória

Dedico esta dissertação aos meus pais, à minha irmã, à minha avó e à minha esposa.

Agradecimentos

Agradeço primeiramente a Deus por tudo, pela vida que me foi dada e pelos caminhos que são percorridos. Sem a fé que tenho, não conseguiria atingir nenhum objetivo. Este é mais um passo, em uma longa estrada, que foi construída e desenhada pelo maior dos Engenheiros que já existiu.

Agradeço à minha esposa Bianca, pelo amor, companheirismo, compreensão e incentivo nas horas difíceis.

Agradeço aos meus pais por toda a motivação, educação e ensinamentos. Ainda, mais que tudo, pela vida que tenho.

Agradeço à minha avó, meu anjo da guarda que guia e protege meu caminho e me enche de felicidade a cada vez que ouço sua voz.

Agradeço à minha irmã, pela torcida e pelo incentivo nas conquistas.

Agradeço aos meus colegas de trabalho da VIVO, que me ajudaram e se fizeram presentes em minhas ausências nas tarefas do dia a dia.

Agradeço aos professores e colegas do CEFET-RJ e da UFF que ajudaram a construir minha carreira tecnológica. Na UFF, em especial ao colega Bruno Wanderley por me ajudar e estar sempre presente todas as vezes que precisei.

Dentre todos os professores, agradeço em especial a Professora Débora, pela sua paciência, incentivo, objetividade e vontade de ensinar e transformar o ensino brasileiro. Pelas correções criteriosas, porém sempre focadas no aprendizado e na correta informação. Deixo aqui meu muito obrigado.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

RESUMO

Esta dissertação propõe uma diferenciação de serviços para chamadas *VoIP* baseada nos números de destino e em classificações dos usuários originadores. A proposta se baseia na marcação de pacotes feita do lado cliente para que possa haver um tratamento diferenciado nos roteadores da rede em caso de congestionamento. Com isso, pode-se obter uma melhor utilização dos recursos e equipamentos do sistema, encaminhando somente fluxos considerados prioritários. A proposta de diferenciação de serviços para chamadas de voz foi implementada e testada em uma rede em malha sem fio.

Palavras-chaves: Qualidade de Serviço (QoS), *VoIP*, mecanismos de escalonamento, classificação de fluxos, marcação de pacotes.

ABSTRACT

This work proposes a differentiated service for VoIP calls based on destinations numbers and classification of the originating subscribers. The proposal is based on client-side packet marking in order to enable different QoS treatment in network routers in case of congestion. Using this proposal, an optimization of system resources and equipment can be achieved, forwarding only high priority flows when congestion occurs. This proposal was implemented and tested over a wireless mesh network.

Keywords: Quality of Service (QoS), VoIP, scheduling mechanisms, flow classification, packet marking.

1. INTRODUÇÃO

No mundo atual, de forma rápida e massiva, revoluções tecnológicas vêm acontecendo em curtos espaços de tempo. Tecnologias que antes eram consideradas de ponta, em poucos meses tornam-se obsoletas, dando espaço a novos produtos, com valores agregados e de melhor desempenho. Todos procuram estar conectados ao mundo através da Internet, acessar portais, fazer transações bancárias, ler notícias, trocar correspondências, enviar arquivos e até falar pela grande rede através de voz sobre IP (*VoIP*) [Goode, 2002]. Enfim, muitos caminhos podem ser abertos e muitas facilidades podem estar ao alcance de todos, bastando apenas um computador e um acesso a Internet.

A indústria das telecomunicações está em plena fase de convergência para o protocolo IP (*Internet Protocol*). A evolução das redes e dos sistemas de comutação de circuitos para as redes de comutação de pacotes foi potencializada pelas novas oportunidades surgidas em torno de serviços mais ricos e inovadores, e das possibilidades de redução de custos que eles permitem. Esta é a continuação natural da evolução das redes. Todos os elos da cadeia, desde a concepção do produto até a oferta para o usuário final, estão evoluindo. As prestadoras de serviço de telefonia fixa, por exemplo, procuram tirar partido da redução dos custos de implementação e manutenção de suas infra-estruturas de rede. Por sua vez, as prestadoras de serviços celulares esperam poder acelerar o ritmo de substituição dos serviços fixos pelos móveis, oferecendo aos clientes móveis novos serviços que anteriormente estavam disponíveis apenas nas redes fixas. Desta forma, as transformações nas topologias de rede e ofertas de serviços são imensas e todos, desde a operadora ao cliente final, esperam os benefícios dos serviços avançados oferecidos pelas plataformas IP, combinados com a facilidade de utilização e personalização dos dispositivos. Ainda, pode-se tomar como exemplo, TV sobre IP, acessos à Internet, etc. Com tantos serviços sendo agregados em uma única infra-estrutura, há a necessidade de dar tratamento diferenciado a esses serviços.

Uma frase de grande impacto foi: "*IP over everything*" ou IP sobre tudo ou qualquer coisa. Esta frase expressava a vontade e a idéia de operar com o protocolo IP sobre praticamente qualquer meio de transmissão e comunicar-se através de qualquer plataforma de sistema. Com o grande crescimento da Internet nos últimos

anos, o surgimento de novas aplicações, a convergência de outras redes, telefone, rádio e televisão tendo a Internet como meio de comunicação, "*Everything Over IP"*, ou tudo sobre IP, é certamente a frase mais adequada para os dias atuais.

Com o conceito de IMS (IP *Multimedia Subsystem*) [Poikselka, 1994] começando a ser implementado e testado em alguns países, toda a indústria de telecomunicações está atualmente deslocando seus estudos e esforços na direção de sistemas IP, com o objetivo de reduzir custos, criar novos serviços geradores de receita e proteger o modelo de negócio das operadoras. Em curto prazo, a expectativa é a troca total de acessos móveis e fixos para uma única plataforma baseada totalmente em IP. Esse modelo (Subsistema Multimídia IP) é definido pelo 3GPP/3GPP2 [3GPP, 2008] com novas funcionalidades, suporte a sofisticados serviços multimídia, rede orientada a pacotes e serviços que permitem a convergência de tecnologia de rede, dados e voz sobre uma infra-estrutura baseada em IP, como ilustrado na Figura 1.

Do lado do usuário, os serviços que serão adicionados ou incorporados ao IMS permitirão comunicações combinadas, ou seja, qualquer união entre dados, voz, fotos e vídeo, de maneira personalizada e segura.

Figura 1: Convergência na Rede IMS.

Fonte [Ericsson White Paper, 2003]

Quando se transporta os serviços de voz para um mundo originalmente pensado e projetado para dados, deve-se levar em consideração que o tráfego oriundo de tal modalidade é bastante intenso. Milhares de ligações telefônicas são efetuadas a cada segundo. A engenharia de tráfego deve ser estudada todos os dias. Os equipamentos utilizados são projetados e especificados para tal finalidade. Há pontos de acesso com características semelhantes e com capacidade intensa de processamento e gerenciamento de fluxos de voz.

1.1. MOTIVAÇÃO

Atualmente, nos sistemas de telefonia, um dos grandes questionamentos e problemas levantados em ocasiões de grande expressividade, tais como Natal, Ano Novo, Carnaval, chamadas para *reality shows*, etc., diz respeito à falta de controle nas chamadas originadas ou terminadas. Não só nestas datas ou ocasiões, há este tipo de problema. Todo o dia existe o chamado HMM (Horário de Maior Movimento). Toda a engenharia de tráfego utilizada, todos os cálculos e estudos levam em consideração este horário que, na verdade, pode ser considerado um limiar que se espera que ocorra. Além da inexistência de controle sobre os dígitos de destino, há a falta de priorização quando se leva em conta a classificação do usuário. Assinantes que geram mais receitas e, portanto, poderiam ter uma prioridade mais alta, disputam os recursos com outros que não trazem retorno de investimento para as operadoras. Pensando não só no retorno de receita gerado, mas também na importância da chamada, temos ainda os usuários que necessitam de atenção total e deveriam ser priorizados. Numa situação de incêndio, catástrofe ou acidente, os usuários envolvidos deveriam ter sempre à disposição todos os recursos necessários, e deveriam até poder impedir que outros utilizassem determinados serviços tais como, transmissão de vídeo e dados, para que suas ligações fossem completadas com sucesso.

Em redes de telefonia fixa, este estudo de tráfego é feito de maneira segura. Sabe-se exatamente a quantidade de terminais conectados a cada central e, com isso, há um estudo de probabilidade e demanda de recursos. Mesmo assim, não se tem a total certeza de que a rede estará livre de congestionamentos em ocasiões de muita utilização.

No caso de telefonia móvel, temos características que podem variar por causa de uma situação inesperada, na qual todos os usuários podem se mover na mesma direção. Com isso, toda a expectativa de crescimento seria desfeita. Com os protocolos legados, há algum tempo existentes nas redes, não podemos garantir acessos especiais ou separados por determinados usuários, ou seja, no acesso de voz, não temos como diferenciar a *QoS* (*Quality of Service*) dependendo da chamada.

No software de algumas centrais telefônicas, temos a possibilidade de restringir tráfego baseado na rota de origem. Este tipo de serviço procura apenas garantir um determinado nível de integridade para que problemas maiores não se propaguem para outros elementos.

A principal motivação deste trabalho diz respeito à qualidade de serviço oferecida ao usuário final quando existe o compartilhamento da rede por vários fluxos simultâneos de um mesmo tipo básico, ou seja, com requisitos similares em relação ao uso da rede, como por exemplo, diversos fluxos de voz. Por exemplo, em uma rede que oferece serviços de telefonia, numa situação de catástrofe, é desejável oferecer um serviço com qualidade aos usuários envolvidos que precisem entrar em contato com o atendimento de emergência.

1.2. OBJETIVOS

Esta dissertação propõe uma diferenciação de serviços para chamadas *VoIP* baseada nos números de destinos e em classificações dos usuários originadores. A proposta se baseia na marcação de pacotes feita do lado cliente para que possa haver um tratamento diferenciado nos roteadores da rede em caso de congestionamento. Com isso, pode-se obter uma melhor utilização dos recursos e equipamentos do sistema, encaminhando somente fluxos considerados prioritários.

Com o intuito de validar a proposta de classificação de fluxos baseada na importância da chamada, uma implementação da proposta foi realizada sobre o *testbed* de redes em malha sem fio da Universidade Federal Fluminense (UFF) [Projeto ReMesh, 2008]. A rede em malha sem fio da UFF é constituída de roteadores IP programáveis, o que permite a implementação de mecanismos específicos para controle de tráfego, como o proposto nesta dissertação.

1.3. ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está organizada como segue. O Capítulo 2 explica conceitos básicos de qualidade de serviço para voz sobre redes IP. Descreve um histórico das tentativas de transmissão de áudio em redes de pacotes e apresenta os protocolos que servem de base para a transmissão e sinalização de fluxos de voz.

O Capítulo 3 apresenta os principais mecanismos de escalonamento de pacotes e policiamento de tráfego, fundamentais para oferecer diferenciação de serviços em redes de pacotes.

No Capítulo 4 é apresentada a proposta de classificação de fluxos de voz baseada na importância da chamada. O capítulo mostra como os pacotes são marcados na origem, quais as classes e subclasses geradas nessas marcações e quais os campos do cabeçalho IP utilizados para representar estas informações, bem como a arquitetura da proposta.

O Capítulo 5 apresenta a implementação da proposta feita na rede em malha sem fio da UFF. O capítulo comenta as ferramentas IPERF e CallGen, usadas neste trabalho para emular tráfego de aplicações VoIP. Além disso, mostra a ferramenta de controle de tráfego TC (*Traffic Control*), a qual permitiu que os pacotes fossem priorizados de acordo com a proposta de classificação de fluxos de voz baseada na importância da chamada. Por fim, apresenta o resultado dos testes de transmissão de fluxos com diferentes subclasses e as análises acerca dos gráficos construídos a partir dos resultados.

O Capítulo 6 compara a proposta com trabalhos relacionados e o Capítulo 7 apresenta as considerações finais, destacando as principais contribuições desta dissertação e apontando trabalhos futuros.

2. QUALIDADE DE SERVIÇO PARA VOZ

2.1. A TELEFONIA IP

A tentativa de transportar áudio em redes de pacotes iniciou-se na década de 70, por Danny Cohen [Schulzrinne, 2006], numa experiência de transmissão de voz em pacotes e em tempo real entre o USC/ISI (*University of Southern California/Information Sciences Institute*) e o *MIT's Lincoln Lab*. As amostras de áudio eram comprimidas utilizando um codificador descrito como *Continuously-Variable Slope Differential* (CVSD) e o transporte feito através do protocolo *Network Voice Protocol* (NVP).

Em 1977, foi apresentada a primeira versão de um protocolo de transporte de voz sobre Internet pelo mesmo autor. O trabalho evoluiu no sentido de aproximar a qualidade oferecida pela rede de comutação de pacotes da qualidade oferecida nas redes de comutação de circuitos com incidência nos aspectos relacionados com entrega de pacotes assíncrona, alta taxa de perda, latências elevadas e *jitter* (variação entre o tempo em que o pacote é esperado e o tempo em que é recebido).

Em 1992, Henning Schulzrinne começou a desenvolver o *Real-Time Transport Protocol* (RTP) [Schulzrinne et al., 1996], de modo a normalizar uma camada de transporte para sistemas com requisitos de tempo real, sendo este protocolo publicado em 1995 como *IETF Proposed Standard* [IETF, 2008].

Também na década de 90, têm origem os protocolos H.323 e o SIP quando investigadores procuraram resolver os problemas de sinalização para transmissão de áudio e vídeo entre dois pontos. O H.323 obteve o seu primeiro sucesso comercial devido ao fato da equipe de trabalho do ITU envolvida no projeto ter obtido rapidamente resultados, e publicado o primeiro *standard* em 1996, pela *International Telecommunications Union* (ITU), a primeira versão da recomendação H.323 [ITU-T Recommendation H.323, 1998]. O *framework* H.323, também conhecido como uma recomendação guarda-chuva, é uma recomendação para a comunicação de áudio, vídeo e dados em redes de pacotes. Esta recomendação tem como objetivo a definição de protocolos ou a utilização de protocolos já existentes e a definição de procedimentos para a comunicação multimídia.

O SIP por outro lado, progrediu muito mais lentamente no IETF, onde o seu primeiro *draft* foi publicado em 1996, mas o primeiro *standard* foi aprovado e publicado muito mais tarde, em fevereiro de 1999 [Handley et al., 1999]. O protocolo SIP foi aceito como norma pelo IETF como um protocolo de sinalização para a criação, modificação e finalização de sessões com um ou mais participantes. O SIP foi várias vezes revisto ao longo dos anos e foi republicado em 2002 (RFC 3261), que é hoje reconhecido como a referência para o SIP [Schulzrinne e Rosenberg, 1999]. Esses atrasos no processo de aprovação resultaram em atrasos na adoção do SIP pelo mercado.

2.2. PROTOCOLOS E CODECS PARA APLICAÇÕES *VoIP*

Com a utilização de redes de pacotes para tráfego de voz, como em uma rede de voz sobre IP (VoIP), elimina-se a necessidade da presença de um circuito físico dedicado. Dentro destes conceitos, a voz é empacotada e transmitida em redes de computadores juntamente com os dados [Hassan, 2002]. Para a realização de uma chamada são necessários protocolos de controle e sinalização para executar algumas tarefas como localização do interlocutor, notificação de chamada, início de transmissão de voz, finalização de transmissão de voz e desconexão.

Uma das arquiteturas mais difundidas no mercado, conforme visto anteriormente, é o padrão H.323, proposto pelo ITU-T. Este padrão especifica uma pilha de protocolos que está focalizada na conexão e controle da chamada, que são separados da transmissão de conteúdo (voz, vídeo e dados) entre os computadores, como pode ser visto na Figura 2.

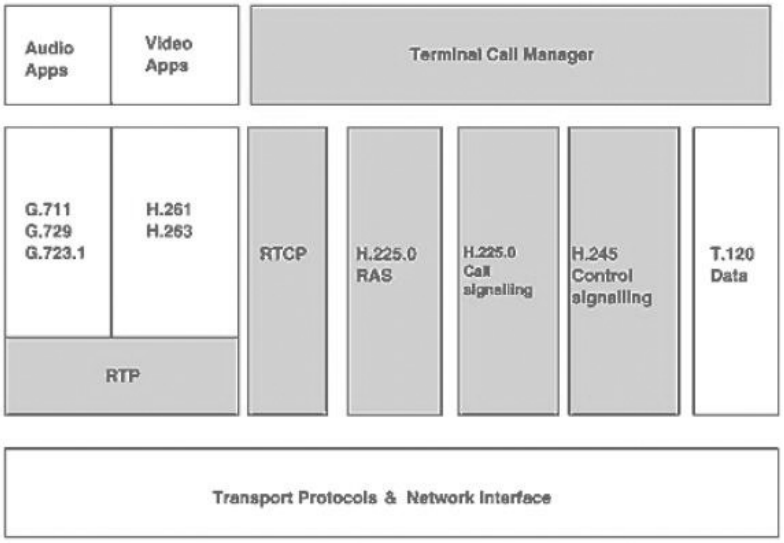

Figura 2: Pilha H.323

Fonte [Trilium Digital Systems, 2005]

O protocolo RTP é indicado para transmitir pacotes de áudio e vídeo entre sistemas terminais. O RTP é definido pela RFC 3550 do IETF [IETF, 2008]. O formato do *payload* para uma série de codecs está definido na RFC 3551, embora haja especificações que estão definidas em documentos também publicados pelo ITU e em outras RFCs. O RTP também providencia as informações necessárias para o receptor realizar a reordenação de pacotes e ajuda em questões sobre atrasos e *jitter* através do protocolo de controle *Real Time Control Protocol* (RTCP) também definido pela RFC 3550. Uma das áreas de preocupação para comunicações entre pessoas através da Internet é a possibilidade de terceiros interceptarem a conversação. Para resolver estas questões, o RTP foi revisto e melhorado resultando no *secure* RTP (SRTP) definido na RFC 3711 [Baugher et al., 2004]. O SRTP providencia criptografia, autenticação e integridade para pacotes de áudio e vídeo transmitidos entre duas entidades que se comunicam.

Uma vez entendidos os protocolos de transporte e suas variantes, deve-se lembrar que, antes do áudio ou vídeo poderem fluir entre dois extremos na rede, vários protocolos são empregados para encontrar o equipamento remoto e para negociar os meios que a informação utilizará para chegar ao seu destino [Goode, 2002]. Estes protocolos, conhecidos como protocolos de sinalização de chamada ou protocolos centrais, funcionam sobre uma infra-estrutura comum: *registration*, *admission and status* (RAS, ITU-T Rec H.225.0), *domain name service* (DNS),

telephony routing over IP (TRIP), *telephone mapping service* (ENUM) e outros protocolos para encontrar usuários [Protocolo ENUM, 2008]. A Figura 3 mostra a pilha de protocolos, dividindo as camadas e relacionando as dependências de cada um.

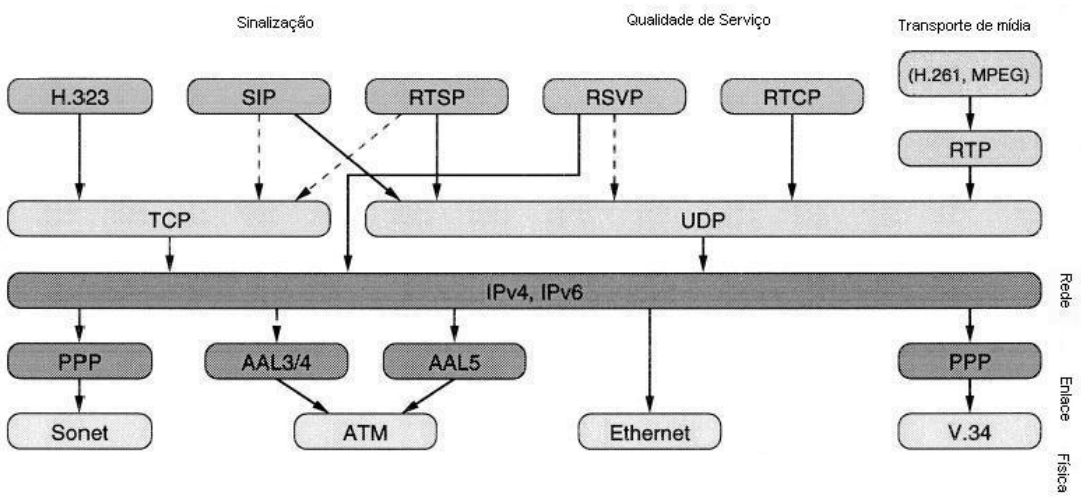

Figura 3: Pilha de Protocolos

Fundamentalmente, o H.323 e o SIP fornecem um conjunto de facilidades similares para comunicação multimídia (áudio, vídeo e dados), embora o H.323 e o SIP sejam diferentes conceitualmente, com o H.323 sendo um protocolo binário, e o SIP um protocolo baseado em texto. Ambos conseguem executar bem o trabalho a que se propõem, entretanto, pode-se salientar as seguintes vantagens do H.323 em relação ao SIP:

• melhor interoperabilidade com a Rede Pública de Telefonia Comutada (PSTN);

• Suporte a gerenciamento de largura de banda. O tráfego dos fluxos de vídeo e áudio é “consumidor” de largura de banda em uma rede. O H.323 provê mecanismos de gerenciamento que permitem delimitar a quantidade de conferências simultâneas e a quantidade de largura de banda destinada às aplicações. Além do mais, o H.323 também prevê facilidade de contabilidade de uso dos recursos da rede que podem ser usadas para fins de cobrança. Isto é possível através da utilização do componente *gatekeeper*, que é importante em uma estrutura H.323,

pois atua como ponto central para todas as chamadas dentro de sua zona, que é o conjunto de todos os terminais, gateways e MCUs gerenciados por um único *gatekeeper*.

O SIP, sendo um protocolo de "início de sessão", não foi projetado para resolver muitos dos problemas que foram levantados e resolvidos nos sistemas tradicionais de comunicação. O SIP foi popularizado e abraçado pelo mercado principalmente pela concepção de que seria um protocolo de fácil implementação e depuração. Quando comparado com o H.323, o SIP apresenta grande vantagem não pela qualidade, ponto que pode ser observado em ambas as soluções, mas sim na simplicidade de implementação. Como resultado, os proponentes do SIP definiram algumas variações do SIP (SIP-T [Vemuri e Peterson, 2002] e SIP-I [ITU, 2008]), assim como algumas extensões não padronizadas de forma a complementar o protocolo com a informação necessária para satisfazer as funcionalidades exigidas na prática. Pode-se dizer que há tantas variações do SIP como implementações [Hersent et al., 2002]. O H.323 possui uma grande parcela no mercado de *VoIP* dos serviços prestados pelos provedores, em particular para o transporte de voz para chamadas internacionais. O H.323 é igualmente amplamente utilizado para sistemas de videoconferência e sistemas de vídeo baseados em IP.

O SIP por sua vez, tem gradualmente vindo a adquirir popularidade pelo uso de sistemas de mensagens instantâneas (*instant messaging systems*). Tanto o H.323, como o SIP, pode ser considerado como protocolo com a inteligência incidente sobre os terminais (*endpoints*). Toda a inteligência necessária para localizar o destinatário e estabelecer fluxos de mídia entre o equipamento local e remoto é uma parte integrante do protocolo.

Existem ainda outros protocolos que são complementares ao H.323 e SIP referidos como protocolos de controle de equipamentos. Estes protocolos são o H.248 e o *Media Gateway Control Protocol* (MGCP) [Foster e Sivachelvan, 2003]. Alguns prestadores de serviço disponibilizam aos seus usuários equipamentos que implementam H.248 ou MGCP, ou outros protocolos do gênero. No coração da rede, alguns equipamentos que utilizam o MGCP providenciam a lógica H.323 e SIP necessária para poder concluir chamadas *VoIP*. Fora do domínio H.323/SIP e H.248/MGCP, existem algumas implementações não padronizadas de protocolos introduzidos por várias companhias que foram bem sucedidas no mercado. O

SKYPE [SKYPE, 2008], por exemplo, é uma dessas aplicações que foi extremamente bem sucedida e usa um protocolo proprietário.

Entretanto, vale a pena observar que protocolos ou soluções padronizadas são mais facilmente absorvidos e difundidos no mercado. Com soluções abertas, pode-se facilmente interoperar entre diferentes fabricantes, estimulando a competitividade. Todo produto com escala de mercado acaba sendo mais barato e, por fim, mais vantajoso para o usuário final.

Um outro parâmetro importante a ser considerado em um ambiente VoIP é a seleção do codec a ser utilizado. A seleção deste parâmetro para representar cada pacote com informação de um fluxo de voz é um compromisso entre qualidade do sinal de voz, taxa de bits gerada e retardo na codificação de cada pacote. Na Tabela 1, podemos observar alguns diferentes tipos de codec utilizados nos serviços VoIP, bem como seus respectivos parâmetros que caracterizam a taxa, o retardo e a qualidade do sinal, indicado pelo valor de MOS. A medida de qualidade MOS é um método subjetivo de testes de qualidade de voz, que varia em uma escala de 1 a 5, sendo o valor mais baixo de qualidade RUIM e o valor mais alto de qualidade EXCELENTE. Nas escalas intermediárias tem-se POBRE, RAZOÁVEL e BOA.

Tabela 1: Tabela de codecs de áudio.

Codec	Técnica de compressão	Kbps	Atraso de codificação (ms)	MOS
G.711	PCM - *Pulse Code Modulation*	64	0,125	4,1
G.726-32	ADPCM - Adaptative Differencial PCM (16,24,32,40 Kbps)	32	0,125	3,8
G.728	LD-CELP - Low-Delay Code-Excited Linear-Prediction	16	3-5	3,6
G.729 A	CS-ACELP - Conjugate-Structure Algebraic code Excited Linear Prediction	8	10	3,7
G.723.1	MP-MLQ	6,3	30	3,6
G.723.1	ACELP - Algebraic Code Excited Linear Prediction	5,3	30	3,1

O tamanho dos pacotes VoIP é um compromisso entre o tempo de processamento e transmissão na rede e o efeito que as perdas têm no desempenho. Pacotes de tamanho grande reduzem a largura de banda necessária à transmissão (pois o cabeçalho tem sempre o mesmo tamanho), mas aumentam o atraso devido ao tempo necessário para transmitir cada pacote de dados [Goode, 2002]. Pacotes de dados pequenos permitem que, em caso de perda de um pacote, não se perca

uma parte significativa da mensagem a transmitir (visto que os pacotes em falta não serão retransmitidos). O tamanho dos pacotes varia com o codec utilizado e com o tamanho e quantidade das amostras de voz utilizadas.

Numa compressão a 8 Kbps, com um envio de pacotes de dados a uma freqüência de 20 ms, temos que cada pacote de dados terá um tamanho de 20 bytes. No entanto, se a transmissão for feita numa rede local IP usando o protocolo RTP, teremos de acrescentar ainda um cabeçalho *Ethernet* de 18 bytes, um cabeçalho IP de 20 bytes, um UDP de 8 bytes e ainda 12 bytes para o cabeçalho RTP. Ou seja, para cada pacote de voz enviado, por exemplo, em um pacote de 20 bytes temos um overhead de 58 bytes.

2.3. QUALIDADE DE SERVIÇO

O que se tem visto até agora é que a convergência para IP é uma realidade e, em conseqüência, toda a concepção e estrutura da Internet precisará mudar para acomodar as demandas das novas aplicações multimídia. Um dos pontos a serem discutidos é sobre a ampliação da largura de banda. Todos concordam que este parâmetro de redes deve ser altamente levado em conta, porém não é o suficiente. Para termos uma rede eficiente, é necessário que esta seja bem utilizada e administrada.

Até o momento, tem-se fornecido o serviço de “melhor esforço”, no qual todos os recursos da rede são divididos igualmente. Com isso, a adição de qualidade de serviço (QoS) representa uma mudança necessária e significativa, pois possibilita o tratamento diferenciado de serviços. Entretanto, modifica o princípio fundamental da simplicidade que fez o sucesso da Internet. De qualquer forma, esta modificação não implica em desvantagens para o usuário. Pelo contrário, trará novos benefícios e possibilitará um novo modelo de negócios para as operadoras.

Ainda pensando em convergência, novos serviços sobre IP estão sendo criados, gerando novas oportunidades e incentivando o suporte a QoS, que é essencial em aplicações de comunicação em tempo real envolvendo áudio e vídeo. Em [Szigeti e Hattingh, 2004], uma relação mais completa sobre conceitos, termos e definições associadas a QoS pode ser encontrada.

Nas ligações telefônicas tradicionais, para cada chamada é estabelecido um circuito de modo a ligar os dois extremos. Este circuito possui uma largura de banda fixa. No entanto, esta solução implica em um desperdício enorme da rede, pois a largura de banda disponibilizada para o canal não é eficientemente utilizada, correspondendo a uma capacidade de pico [Sanneck e Mohr, 2001]. A comunicação de Voz sobre IP (*VoIP*), pelo contrário, compartilha o canal de comunicação com outras aplicações, que geram diferentes tipos de tráfego, como o tráfego de dados. Deste modo deve manter uma qualidade de serviço adequada, de forma a manter a qualidade perceptível pelo usuário.

Depois de discutidas as dificuldades e os desafios que se colocam com a transmissão de voz em redes IP, serão analisados agora os mecanismos que permitem garantir uma qualidade de serviço equivalente à prestada pela PSTN [Hassan e Atiquzzaman, 2000]. Nas redes de dados a presença do *VoIP* obriga a coexistência do trafego de voz com tráfego sem requisitos de tempo real, ambos sobre IP. A necessidade de garantia de qualidade de serviço (QoS) deve ser atendida então por todos os elementos que compõem e interligam os intervenientes de uma conversação. Isto inclui os diferentes equipamentos de comunicação. As técnicas de QoS para VoIP devem ser aplicadas de forma a dar prioridade aos fluxos de voz quando há uma “disputa” com fluxos de dados concorrentes, de forma a permitir que estes possam satisfazer necessidades específicas de qualidade como requisitos de tempo-real e de atraso.

Para que possamos controlar o tráfego de pacotes que passa em um determinado roteador, necessitamos estabelecer regras. O controle de tráfego é o nome dado ao conjunto de sistemas e mecanismos de escalonamento através dos quais os pacotes são recebidos ou transmitidos nos equipamentos [Martins, 1999]. Isto inclui tanto as decisões sobre quais pacotes devem ser aceitos e a que taxa, na entrada de uma interface, como a determinação de quais os pacotes a transmitir e com que retardo, na saída de uma interface.

Na situação mais simples, o controle de tráfego reduz-se a uma única fila de espera, que recebe os pacotes e os encaminha o mais rapidamente possível. Este tipo de fila de espera é conhecido como FIFO (*First In - First Out*). As filas de espera são locais onde os pacotes de uma rede esperam para serem processados, ou seja, são uma forma de organização de dados pendentes com a qual se pretende gerir os fluxos de dados que circulam numa rede, recorrendo-se para isso aos algoritmos de

escalonamento. Os algoritmos de escalonamento decidem qual o próximo pacote que será servido na fila de espera, sendo assim, um dos mecanismos responsáveis por distribuir a largura de banda do enlace pelos diferentes fluxos.

Estes algoritmos podem ser do tipo *work-conserving* ou *non-work-conserving* [Chen e Xuan, 2003]. No primeiro caso, o nó "trabalha" sempre, enquanto que no segundo caso, um nó pode transmitir um pacote apenas quando este se torna elegível (pacote cujo tempo necessário para se manter em espera terminou). Se no nó apenas se encontrarem pacotes não elegíveis em espera, então o nó se manterá inativo. Estes tipos de algoritmos são projetados para aplicações que não toleram variações no atraso de transmissão.

A necessidade para este tipo de solução vem com o fato de que algumas redes baseadas em comutação de pacotes não têm estado, ao contrário das redes baseadas em comutação de circuitos (tais como as redes telefônicas), que têm de manter um estado dentro da própria rede. Apesar desta "falta" de estado ser uma das vantagens principais do protocolo IP, o ponto fraco desta implementação é a falta de diferenciação entre tipos de fluxos de pacotes [Hassan e Atiquzzaman, 2000]. A maneira de introduzir essa diferenciação é através do controle de tráfego, que fornece a possibilidade de tratar os pacotes de forma diferente, baseando-se nos atributos do pacote. As situações mais freqüentes que requerem controle de tráfego e escalonamento de pacotes são as seguintes:

- ✓ Limitar a largura de banda total a uma determinada taxa;
- ✓ Limitar a largura de banda que um determinado serviço ou aplicação cliente pode utilizar;
- ✓ Maximizar a vazão de um protocolo, como por exemplo o TCP, dando prioridade à transmissão dos pacotes de reconhecimento (ACK);
- ✓ Reservar largura de banda para uma determinada aplicação ou serviço;
- ✓ Dar prioridade a tráfego sensível a variações de atraso;
- ✓ Distribuir a largura de banda de forma justa por diversos fluxos de tráfego;
- ✓ Assegurar que determinado tipo de tráfego não passe em um roteador.

Para adicionar recursos de qualidade de serviço à Internet, dois modelos de classes de serviços para tráfego são considerados e estudados, não só pela IETF (*Internet Engineering Task Force*), mas também por diversas instituições de

pesquisa. O primeiro refere-se aos serviços diferenciados, denominado *Differentiated Services*, comumente conhecido como DIFFSERV ou ainda de *Soft* QoS, que provê uma tratamento diferenciado a determinados tipos de fluxo. O segundo refere-se aos serviços integrados ou *Integrated Services*, comumente conhecido como INTSERV, também chamado de *Hard* QoS, que fornece uma garantia absoluta na alocação dos recursos da rede.

Nos itens a seguir, serão mostradas as características principais destes dois modelos de QoS, onde serão descritas vantagens e desvantagens de cada um.

2.3.1. INTSERV

Os Serviços Integrados (*Integrated Services* ou IntServ) [Braden et al., 1994], propostos pelo IETF (*Internet Engineering Task Force*), foram projetados para prover um conjunto de extensões ao modelo de entrega de tráfego de melhor esforço atualmente utilizado nas redes com tecnologia IP em geral, incluindo a Internet.

Tal modelo é caracterizado pela reserva de recursos, ou seja, antes de iniciar uma comunicação, o receptor solicita ao emissor a alocação de recursos necessária para definir-se uma boa qualidade na transmissão dos dados. O protocolo RSVP (*Resource Reservation Protocol*) [Braden, 1995] é utilizado, nesse modelo, para troca de mensagens de controle de alocação dos recursos e é baseado na noção de *soft-state*. Este termo foi inicialmente proposto por [Clark, 1998], cuja definição é entendida como o “estado”' que um determinado elemento, pertencente ao percurso de dados de um determinado par fonte-destino, se encontra quando uma reserva está estabelecida. O início do *soft-state* ocorre quando uma mensagem de reserva é recebida e realizada no elemento; este estado é periodicamente realimentado pelos receptores. Ao invés de entregar à rede a responsabilidade em detectar e responder a falhas, o RSVP delega aos receptores o trabalho de reenviar periodicamente suas requisições de serviços. Caso uma falha ocorra, somente uma nova requisição do serviço restabelecerá o *soft-state* nos roteadores [Schmidt, 2000].

O IntServ é caracterizado pela alocação de recursos para dois novos tipos de serviços que são os serviços garantidos para aplicações que necessitam de limites fixos de atraso, e serviços de carga controlada, cujo desempenho pode ser equiparado ao do melhor esforço sob condições de não congestionamento.

✓ Serviço Garantido (*Guaranteed Service*) [Shenker et al., 1997]: oferece rígidos limites, matematicamente prováveis em termos de atrasos de enfileiramento, aos quais os pacotes estarão condicionados nos roteadores. Ele garante tanto o atraso quanto a taxa de bits. Basicamente, uma sessão requisitando Serviço Garantido está requerendo que os bits em seus pacotes tenham uma taxa de transmissão garantida. Deve-se notar que este serviço não tenta minimizar a variação de atraso, mas controlar o atraso máximo de enfileiramento. Para este tipo de serviço, todos os nós intermediários devem implementar os serviços garantidos. Este serviço pode ser útil para aplicações requerendo limites de atraso fixos, tais como as de áudio e vídeo em tempo-real.

✓ Serviço de Carga Controlada (*Controlled Load Service*) [Wroclawski, 1997]: uma sessão requerendo tal serviço receberá uma Qualidade de Serviço muito próxima daquela que um fluxo poderia receber de uma rede não sobrecarregada. Em outras palavras, a sessão pode assumir que um "percentual muito alto" de seus pacotes passará com sucesso através dos roteadores sem ser descartada e com um atraso de enfileiramento pequeno. O Serviço de Carga Controlada não fornece garantias quantitativas acerca do desempenho – ele não especifica o que constitui um "percentual muito alto" de pacotes, nem que qualidade de serviço aproximada será fornecida por um elemento de rede não sobrecarregado. Este tipo de serviço é dirigido para aplicações em tempo-real adaptativas como as que estão sendo desenvolvidas atualmente na Internet. Estas aplicações funcionam razoavelmente bem quando a rede não está sobrecarregada, mas elas se degradam rapidamente quando a rede fica congestionada.

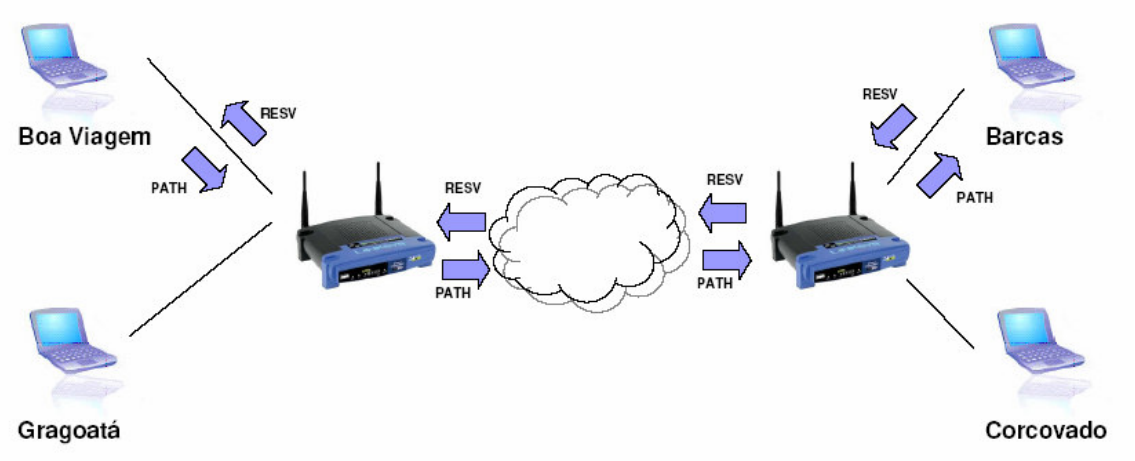

Figura 4: Reserva IntServ.

Na Figura 4, suponhamos que a máquina Boa Viagem deseje fazer uma comunicação de voz através da Internet (VoIP) com a máquina Barcas. Esta aplicação requer baixo atraso e baixa variação do atraso (*jitter*) para que sejam mantidos os requisitos de qualidade. Então, a máquina Boa Viagem envia uma mensagem especificando as características para o tráfego (*path*) para Barcas. Quando a mensagem de *path* chega para a máquina Barcas, inicia-se o procedimento de reserva de recursos (RESV) por todo caminho entre este dois nós da rede. Todos os roteadores entre os dois pontos passam pelo processo de alocação de recursos e qualquer um deles pode rejeitar a solicitação, informando para a máquina Boa Viagem que a solicitação não foi aceita. Caso todos os roteadores tenham condições de disponibilizar os recursos solicitados, é alocada a largura de banda e buffer necessários para a aplicação.

Durante a transmissão dos pacotes, são feitas classificações nos roteadores para cada fluxo, colocando-os em filas específicas para a aplicação. Como o controle é feito, basicamente, nos roteadores, isso exige grande capacidade de processamento, armazenamento e bons algoritmos para tratamento de filas. Ou seja, aumenta o grau de complexidade nos roteadores.

O modelo IntServ é implementado por quatro componentes principais:

- ✓ protocolo de sinalização: as configurações de caminhos e as reservas de recursos devem ser feitas antes da transmissão dos dados pelas aplicações

que necessitem de Serviço Garantido ou Serviço de Carga Controlada. Para tal, devem usar o protocolo de sinalização RSVP;

✓ rotina de controle de admissão: é a decisão tomada para assegurar o pedido de alocação de recursos. Implementa o algoritmo de decisão que um roteador ou *host* usa para determinar se um novo fluxo pode ter sua *QoS* garantida sem afetar os fluxos anteriormente admitidos;

✓ classificador: ao receber um pacote, o classificador do roteador executa uma classificação Multi-Campo (MF – *Multi-Field*) em um pacote IPv4 e coloca o pacote em uma fila específica em função do resultado da classificação. A classificação MF é o processo de classificar pacotes, baseado no conteúdo dos seus campos tais como: endereço de origem, endereço de destino, campo TOS (*Type of Service*) e identificador do protocolo. Cada pacote que chega deve ser mapeado em alguma classe, onde recebem o mesmo tratamento do escalonador de pacotes.

✓ escalonador de pacotes: após a classificação, o escalonador seleciona para transmissão o pacote de modo a satisfazer os requisitos de *QoS*. O escalonador de pacotes gerencia a retransmissão dos diferentes pacotes usando um conjunto de filas e outros mecanismos tais como temporizadores.

2.3.2. DIFFSERV

O modelo de serviços diferenciados, conhecido como DiffServ [Black et al., 1998], implementa QoS com base na definição de tipos de serviços. Diferentemente do serviço de melhor esforço e da arquitetura IntServ, que funciona com reservas de recursos e há uma considerável transação de sinalização, na arquitetura DiffServ não existe alocação explícita de recursos, tendo em vista que a prioridade do pacote é transmitida no cabeçalho IP, permitindo desta forma maior escalabilidade e baixa sobrecarga de sinalização.

Existe no cabeçalho de um pacote IP, um campo denominado TOS (*Type of Service*) que pode representar o tipo do serviço. No entanto, serviços diferenciados

ampliam a representação de serviços e o tratamento que pode ser dado para encaminhar um pacote, redefinindo o conteúdo do campo TOS, passando a chamá-lo de DS Field (*Differentiated Services Field*). No DS Field, são codificadas as classes para serviços diferenciados. O campo TOS já existia na definição do pacote IP, mas o DIFFSERV definiu uma nova utilização para o mesmo [Nichols et al., 1998].

O diagrama da Figura 5 ilustra onde os elementos de uma rede DiffServ são empregados. A arquitetura Diffserv insere o conceito de domínio DS, cuja definição pode ser entendida como um conjunto adjacente de nós DS que aplicam políticas comuns sobre o tráfego que atravessa um determinado domínio. Um domínio DS é constituído de nós de borda e nós de interior. Os nós DS de borda têm a responsabilidade de classificar e condicionar o tráfego que entra no domínio DS.

A cada fluxo de tráfego que entra no domínio pelos nós de borda, a política de QoS define qual deles terá um serviço diferenciado, como será tratado pelos nós interiores e como este deverá ser marcado nos nós de borda. Os nós interiores, por sua vez, examinam a marcação dos pacotes e atuam de acordo com as regras definidas ou seu perfil de tráfego.

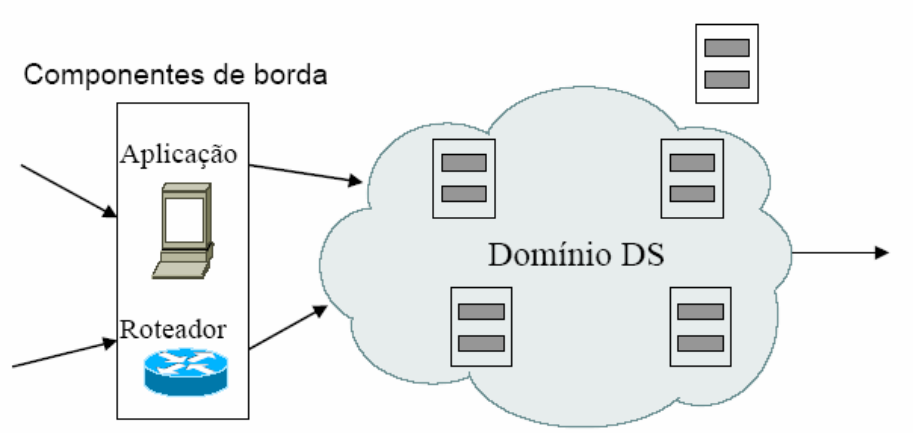

Figura 5: DiffServ – visão geral.

Dois importantes componentes nos nós DS são definidos na arquitetura DiffServ, conforme mostra a Figura 6: os componentes de classificação e de condicionamento de tráfego. Estas funções são mais complexas em nós DS de borda que em nós de interior, mas ambos os nós têm um classificador.

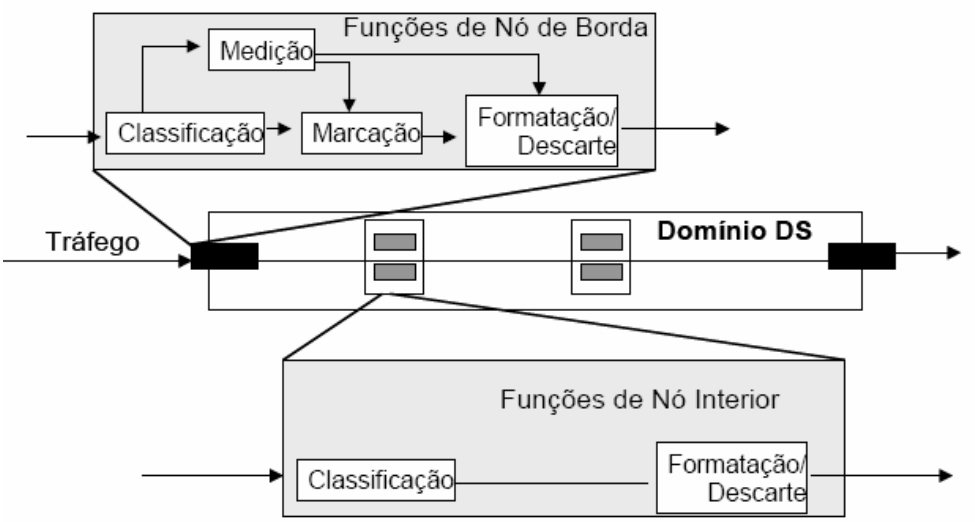

Figura 6 Roteador que implementa arquitetura DiffServ

Estes classificadores são classificados em dois tipos: um que classifica o fluxo baseado apenas na classificação DS e outro que verifica múltiplos campos no cabeçalho IP. Tais classificadores são conhecidos como classificador de comportamento agregado (BA – *behaviour agregate*) e classificador multi-campo (MF – *multifield classifier*), respectivamente. A marca de classificação DS é conhecida como DS *codepoint* ou DSCP.

Em nós DiffServ, o classificador é o componente que divide o fluxo de entrada em um conjunto de fluxos de saída por meio de filtros de tráfego baseados no conteúdo do cabeçalho do pacote e/ou em diferentes atributos do pacote que podem ser implicitamente derivados.

No momento de chegada do pacote o medidor verifica se o pacote está de acordo com um perfil de tráfego pré-definido. Dependendo da conformidade várias ações podem ser executadas, através dos elementos de ações. Três tipos de ações, e a combinação delas, podem ser executadas: marcação, moldagem de tráfego (*traffic shapping*) e descarte.

Podemos entender escalonamento como o processo de decidir no momento da transmissão qual fila entre o conjunto de candidatas deve ser servida, dependendo de alguma propriedade da fila e/ou do pacote. Geralmente ele é empregado no caso de contenção de recursos na saída. A ação de moldagem de tráfego ou policiamento modifica o tráfego de entrada para forçar um determinado perfil de saída.

Uma fila, por sua vez, é uma área de memória necessária para executar escalonamento e/ou moldagem. Condicionadores de tráfego são empregados em um determinado estágio do caminho dos dados para forçar uma determinada política. Eles podem ser implementados através da combinação de um ou mais componentes de DiffServ definidos anteriormente (classificadores, medidores, elementos de ação e filas) ou, alternativamente, através da combinação de condicionadores de tráfego existentes. Marcação é um exemplo de condicionamento de tráfego.

2.3.2.1 ENCAMINHAMENTO DE TRÁFEGO – PHBs

Como comentado anteriormente, o fluxo de tráfego é classificado e marcado nos nós de borda. Os campos DSCP (*DiffServ Code Point*) [Baker, 1995] são mapeados para os PHBs (*Per Hop Behaviors*) [Nichols et al., 1998] definidos na arquitetura DiffServ. O comportamento de encaminhamento de um pacote em um nó DiffServ é definido pelos PHBs, que são identificados através de um rótulo (*label*) de 6 bits, do campo TOS (*Type Of Service*), do cabeçalho do pacote IPv4, e o campo *Class*, do cabeçalho do pacote IPv6, agora chamados de *Differentiated Services Code Point* (DSCP) conforme mostra a Figura 7.

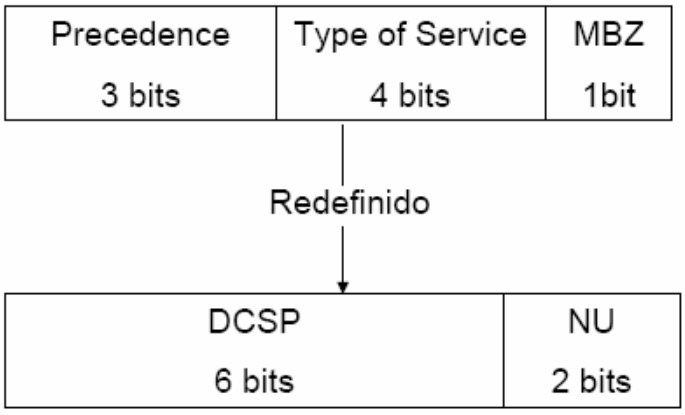

Figura 7: Campo TOS versus Byte DS

Os seis bits do campo DS são usados para selecionar o PHB que o pacote terá em cada nó. Este campo é tratado como um índice em uma tabela usada para selecionar o mecanismo de manipulação de pacotes implementado em cada

dispositivo e ainda, é definido como um campo não estruturado para facilitar a definição de futuros PHBs. Outros comportamentos podem especificar que, para determinados pacotes, serão dadas certas prioridades relativas a outros, em termos de vazão média (*throughput*) ou de preferência para descarte, mas sem ênfase particular em atrasos.

Os PHBs são implementados utilizando mecanismos de enfileiramento e são comportamentos individuais aplicados em cada roteador, por isso isoladamente não garantem QoS fim-a-fim [Kurose e Ross, 2004]. Entretanto, a interligação de roteadores com os mesmos PHBs e a limitação da taxa em que os pacotes são enviados para um PHB, possibilita o uso de PHBs para construir QoS fim-a-fim. Por exemplo, a concatenação de PHBs EF (*Expedited Forwarding*) [Jacobson et al., 1999] ao longo de uma rota preestabelecida, com um cuidadoso controle de admissão, pode prover um serviço similar ao de uma linha dedicada, que é satisfatório para voz. Uma outra concatenação de PHBs pode ser utilizada para transmissão de vídeo armazenado, e assim por diante.

O PHB *default* serve para o tráfego BE (*best effort*), ou de melhor esforço. Ele assegura compatibilidade com o encaminhamento melhor esforço padrão em todos os roteadores e é documentado no RFC 1812. PHBs de tráfegos com DSCP mais alto devem ter encaminhamento preferencial sobre aqueles com valor mais baixo. Por exemplo, o PHB 111000 deve ter prioridade mais alta que 000000 [Melo, 2001].

2.3.2.2. PHBs PADRÕES

Foram padronizados dois importantes PHBs: o PHB EF (*Expedited Forwarding*) e o PHB AF (*Assured Forwarding*) [Heinanen et al., 1999].

O PHB EF, também referido como serviço *premium* ou de canal dedicado, pode ser usado para tráfego com requisitos de baixa perda, baixo atraso, baixo *jitter* (variação de atraso) e garantia de largura de banda. Estes requisitos são alcançados assegurando-se que os agregados de tráfego encontram nenhum ou pouco enfileiramento. As implementações devem prover meios de limitar o dano que o tráfego EF possa infligir sobre o outro tráfego. Os dispositivos de borda do domínio DS devem policiar o tráfego EF para assegurar a taxa de bits definida. Pacotes em excesso devem ser descartados.

O PHB AF tem por objetivo fornecer entrega de pacotes IP, com largura de banda assegurada, em quatro classes de transmissão, mas não oferece garantias quanto ao atraso. Cada classe tem três precedências de descartes (*Drop Precedence*), as quais são utilizadas para determinar a importância do pacote. Assim, um nó congestionado dá preferência para serem descartados, entre os pacotes de uma mesma classe, aqueles com maiores valores de precedência de descarte.

Os três primeiros bits do DSCP identificam a classe de transmissão, 001, 010, 011 e 100 e os três últimos bits definem a precedência de descarte: 010 para a precedência mais baixa (ou seja, último a ser descartado), 100 para precedência média e 110 para a mais alta precedência de descarte (ou seja, primeiro a ser descartado). Na Tabela 2, Tabela 3 e Tabela 4 são apresentadas os DSCPs recomendados para as classes de tráfego AF, BE e EF.

Tabela 2: PHB AF – RFC 2597 – quatro classes de tráfego independentes.

Precedência de descarte	Classe AF1		Classe AF2		Classe AF3		Classe AF4	
	Binário	Decimal	Binário	Decimal	Binário	Decimal	Binário	Decimal
Baixa	001010	10	010010	18	011010	26	100010	34
Média	001100	12	010100	20	011100	28	100100	36
Alta	001110	14	010110	22	011110	30	100110	38

Tabela 3: PHB Default – Tráfego melhor esforço.

Precedência de descarte	Classe BE	
	Binário	Decimal
N/A	000000	0

Tabela 4: PHB EF – RFC 2598 – Tráfego *premium*.

Precedência de descarte	Classe EF	
	Binário	Decimal
N/A	101110	46

A marcação da precedência de descartes para tráfegos AF pode seguir dois enfoques: marcador com taxa simples e marcador com taxa dupla [Heinanen e Guerin, 1999]. Ambos usam um regulador do tipo duplo balde furado (*Dual Leaky Bucket*), sendo que no marcador simples, os dois baldes são preenchidos na mesma taxa e no marcador com taxa dupla, os baldes são preenchidos com duas taxas diferentes. Ambos os enfoques medem o tráfego e marcam o pacote, dependendo se o fluxo excede a taxa acordada ou contratada CIR (*Committed Information Rate*).

Um pacote é marcado, por exemplo, como verde, amarelo ou vermelho dependendo em quanto o CIR foi excedido. Um pacote verde tem a menor prioridade de descarte, enquanto que um pacote vermelho, a mais alta [Melo, 2001]. O principal objetivo é fornecer a marcação para algum mecanismo de descarte baseado na precedência de descartes, tal como o mecanismo RED (*Random Early Detect*).

2.3.2.3. SERVIÇOS EM UM DOMÍNIO DS

A arquitetura parte do princípio que domínios adjacentes tenham um acordo sobre os serviços que serão disponibilizados entre os mesmos. Esse acordo denomina-se SLA – *Service Level Agreement*. Um SLA determina as classes de serviços suportadas e a quantidade de tráfego na banda entre os domínios. Os domínios podem definir um SLA estático ou dinâmico, sendo que, neste último caso, um protocolo de sinalização e controle será necessário para o estabelecimento do SLA.

Com DiffServ, os próprios clientes podem marcar seus DS Fields e enviar para o receptor. No entanto, dessa forma, não há como saber se há recursos disponíveis para a comunicação, fazendo com que, por exemplo, o pacote chegando em um roteador que não provê QoS com o DS Field marcado, seja remarcado e passe a ser um pacote de um serviço de melhor esforço.

Existem tipos de serviços que não podem conviver com essa incerteza. Por isso, um componente mediador foi inserido nesse modelo para gerenciar recursos no domínio com QoS. Este componente foi denominado *Bandwidth Broker* (BB). O BB trabalha como um gerente de recursos do domínio que tem como função básica controlar a largura de banda, as políticas e prioridades dentro e entre as organizações. Quando há uma solicitação de um fluxo, o BB é o componente que verifica a disponibilidade de recursos e a autorização do cliente para conexão dentro do domínio com QoS. Ele também se encarrega de fazer as alocações necessárias para a comunicação dentro do seu domínio e solicita ao BB do domínio adjacente, caso o pedido de conexão seja para fora do domínio, para fazer o mesmo. O processo de solicitação de alocação de recursos é continuado entre BBs adjacentes até que se chegue ao BB do domínio do receptor. O BB pode usar o RSVP para priorizar aplicações em casos de congestionamento.

Serviços diferenciados têm sido o modelo mais utilizado para implementação de QoS. Ele exige menos dos roteadores para prover bons métodos de classificação, policiamento, montagem e remarcação de pacotes.

No Capítulo 3, veremos alguns diferentes algoritmos de classificação e escalonamento de pacotes, usados no controle do tráfego e as suas características associadas, e a sua influência nos diferentes fluxos de dados.

2.3.3. COMPARAÇÃO ENTRE DIFFSERV E INTSERV

Com a arquitetura IntServ, temos uma forte mudança na arquitetura atual da Internet, a qual é fundamentada na idéia de que todas as informações de estado relacionadas aos fluxos deveriam estar nos sistemas finais. Tendo em vista este conceito, podemos identificar alguns problemas com a arquitetura de Serviços Integrados:

- ✓ A quantidade de informações de estado cresce proporcionalmente à quantidade de fluxos. Isto causa uma sobrecarga de armazenamento e processamento nos roteadores. Portanto esta arquitetura é considerada pouco escalável;

- ✓ Os requisitos nos roteadores são altos: todos os roteadores devem implementar RSVP, controle de admissão, classificação MF e escalonamento de pacotes;

- ✓ Para Serviço Garantido, toda a rede deve suportar IntServ. Uma instalação gradativa de Serviço de Carga Controlada é possível;

- ✓ IntServ e RSVP não são satisfatoriamente aplicáveis em alguns casos, como por exemplo, aplicações do tipo navegadores WWW, onde a duração de um fluxo típico é apenas de poucos pacotes.

A sobrecarga causada pela sinalização RSVP poderia facilmente deteriorar o desempenho da rede sendo percebida pela aplicação.

Por outro lado, na arquitetura DiffServ, podemos destacar as seguintes características:

- ✓ Existência de apenas um número limitado de classes de serviço indicadas no campo DS. Desde que o serviço seja alocado na granularidade de uma classe, o conjunto de informações de estado é proporcional apenas ao número de classes e não proporcional ao número de fluxos. O modelo de Serviços Diferenciados é, portanto, mais escalável;

- ✓ As operações de classificação, marcação, policiamento e retardo são necessárias somente nas fronteiras das redes e dependendo das políticas definidas, os fluxos de tráfego com requisitos similares são agregados. Assim, os roteadores intermediários necessitam apenas implementar a classificação Comportamento Agregado (BA – *Behavior Aggregate*), que é uma classificação baseada apenas no byte DS. Portanto, DiffServ é mais fácil de implementar e usar;

- ✓ No modelo de Serviços Diferenciados, um Serviço Assegurado pode ser fornecido por um sistema que suporta parcialmente os Serviços Diferenciados. Os roteadores que não suportam DiffServ simplesmente ignoram os campos DS dos pacotes e fornecem serviço assegurado ou serviço de melhor esforço. Como os pacotes de serviço assegurado têm menos probabilidade de serem descartados em roteadores compatíveis com DS, o desempenho total do tráfego de Serviço Assegurado será melhor que o tráfego de Melhor Esforço.

As facilidades apontadas na comparação entre os dois modelos de provisão de QoS analisados motivaram a escolha do modelo DiffServ no desenvolvimento desta dissertação. Percebeu-se que estas facilidades apontadas em termos de implementação e escalabilidade viabilizariam a implantação imediata dos serviços de priorização de pacotes baseados na importância do chamador e no destino final. Entretanto, como a quantidade de classes oferecida pelo DiffServ é muito limitada, esta dissertação propõe uma nova definição do campo ToS para realizar a diferenciação de chamadas de voz considerando a importância da chamada, como será detalhado no Capítulo 4. Em [Hilario, 2006] é apresentada e implementada a

proposta de um *framework* de suporte a QoS para redes *mesh*. Tal proposta baseia-se na reserva de recursos, utilizando-se do protocolo Insignia. Entretanto, como exposto anteriormente, mecanismos de reserva de recursos não foram utilizados nesta dissertação.

2.4. REQUISITOS ESSENCIAIS EM REDES COM QoS

Existe um conjunto de requisitos que são considerados essenciais para redes que implementam mecanismos de QoS. Dentre estes requisitos, inclui-se a minimização do atraso fim-a-fim e da variação do atraso (*jitter*), a minimização da taxa de perda de pacotes e a vazão de dados e largura de banda consistente conforme detalhado nos itens a seguir.

2.4.1. ATRASO FIM-A-FIM

Considera-se atraso fim-a-fim, o tempo entre o envio de uma mensagem por um nó e a recepção desta mensagem pelo nó destino [Shenker, 1997]. Este atraso inclui o tempo de percurso ao longo do caminho de transmissão e o atraso nos dispositivos intermediários. Em cada roteador, o atraso é o total de tempo entre a recepção do pacote e a sua transmissão. Este tempo é também referido como atraso ou retardo de transmissão.

2.4.2. VARIAÇÃO DO ATRASO (*JITTER*)

O *jitter* é um parâmetro de engenharia de redes que é definido como uma diferença ocorrida nos tempos de chegada entre pacotes comparados aos tempos originais de transmissão entre pacotes. É uma distorção que acontece, por exemplo, quando fluxos de voz ou vídeo são transmitidos em uma rede e os pacotes não chegam em seu destino dentro da ordem sucessiva ou em uma determinada cadência. Ou seja, existe uma variação no retardo entre chegadas em relação ao intervalo existente entre o envio dos pacotes. Esta distorção é particularmente prejudicial ao tráfego multimídia, que precisa manter uma taxa de reprodução

constante, fazendo com que o sinal de áudio ou vídeo tenha uma qualidade distorcida ou fragmentada na recepção.

2.4.3. PERDA DE PACOTES

A taxa de perda de pacotes representa o percentual de pacotes que foram transmitidos na rede, mas não chegaram ao destino, sobre o total de pacotes. As aplicações multimídia são tolerantes a perda, porém é necessário, em alguns casos, limitar a taxa para garantir a qualidade desejada.

2.4.4. LARGURA DE BANDA E VAZÃO

Largura de banda é uma medida de capacidade de transmissão de dados, normalmente expressa em kilobits por segundo (Kbps) ou megabits por segundo (Mbps). A largura de banda indica a capacidade máxima de transmissão teórica de uma conexão [Szigeti, 2004].

Entretanto, na medida em que a taxa de transmissão utilizada se aproxima da largura de banda teórica máxima, fatores negativos como atraso na transmissão das informações podem causar deterioração na qualidade. A largura de banda de uma rede pode ser vista como um tubo que transfere dados. Quanto maior o diâmetro do tubo, mais dados podem ser enviados através dele simultaneamente.

A vazão é o montante de tráfego de dados movidos de um nó da rede para outro em um determinado período de tempo. A vazão também é expressa em Kbps ou Mbps.

2.4.5. CONTROLE DE ADMISSÃO

O controle de admissão de fluxos tem como finalidade, determinar se um fluxo pode ser aceito sem prejudicar a QoS das transmissões em andamento. O mecanismo deve então verificar se existem recursos disponíveis suficientes e

balancear a probabilidade de congestionamento com a utilização eficiente dos recursos da rede para determinar se um novo fluxo pode ou não ser admitido.

2.5. RESUMO DO CAPÍTULO

O Capítulo 2 explicou conceitos básicos de qualidade de serviço para voz sobre redes IP. Descreveu um histórico das tentativas de transmissão de áudio em redes de pacotes e apresentou protocolos e codecs que servem de base para a transmissão e sinalização de fluxos de voz. Além disso, explorou a questão de qualidade de serviço, apresentando as idéias e conceitos relacionados aos modelos DIFFSERV e INTSERV. Concluindo o Capítulo 2, pudemos analisar quais os requisitos básicos essenciais para que tenhamos qualidade de serviço em uma rede.

O capítulo a seguir apresenta os principais mecanismos de escalonamento de pacotes e policiamento de tráfego, fundamentais para oferecer diferenciação de serviços em redes de pacotes.

3. MECANISMOS DE ESCALONAMENTO E POLICIAMENTO

Em uma rede de comunicação comutada por pacotes, os pacotes transmitidos pelos nós de origem passam por nós intermediários que podem controlar a retransmissão dos mesmos através de diversos mecanismos de controle de tráfego [Goode, 2002]. Como dito anteriormente, controle de tráfego é o nome dado ao conjunto dos sistemas de filas de espera e mecanismos associados pelos quais os pacotes são recebidos e transmitidos em um roteador.

As filas de espera são o conceito fundamental por trás dos mecanismos de escalonamento. Em uma rede de pacotes, uma fila é um local que contém um conjunto finito de pacotes a espera de serem processados.

Os mecanismos de escalonamento são algoritmos que gerem a forma como os pacotes são processados dentro das filas de espera. No modelo mais simples, os pacotes são servidos na mesma ordem em que são recebidos numa interface. Este modelo é designado por FIFO (First In - First Out).

Sem mecanismos de escalonamento de pacotes, uma fila de espera não permite fazer qualquer tipo de controle de tráfego. Uma fila torna-se muito mais útil quando associada a outros mecanismos que permitem atrasar pacotes, reordená-los, descartá-los e separar os pacotes por diversas classes de acordo com a sua prioridade.

Alguns mecanismos de escalonamento podem ser combinados com mecanismos de policiamento de tráfego (*traffic policing*) para estabelecer limites superiores ao retardo de uma fila em nós intermediários e garantir a qualidade serviço aos diferentes fluxos. Estes mecanismos serão descritos nas próximas seções.

3.1. MECANISMOS DE ESCALONAMENTO DE PACOTES

3.1.1. FIFO – *FIRST IN FIRST OUT*

FIFO (*First In – First OUT*) [Ferguson e Huston, 1999] é o mecanismo de escalonamento de pacotes mais simples, onde todos os pacotes são tratados de

maneira igual, sendo colocados numa fila única e servidos por ordem de chegada. Tal conceito pode ser observado na Figura 8.

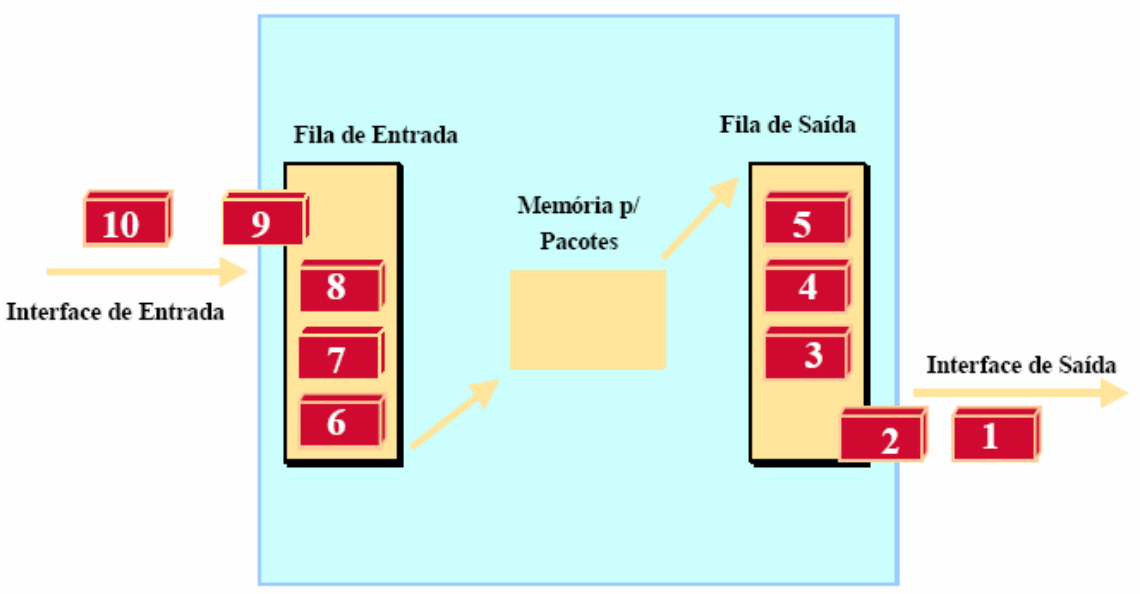

Figura 8: Uma fila FIFO.

Apesar de ser simples, não permite diferenciação por tipos de fluxos, assim como limites máximos nos atrasos [Schulzrinne, 2006]. Sendo assim, os fluxos de tráfego que recebem um serviço melhor são os que geram mais tráfego. As suas principais vantagens são:

- ✓ Para roteadores baseados em software, este algoritmo não sobrecarrega as máquinas, comparativamente com outros algoritmos de escalonamento de pacotes mais sofisticados;

- ✓ O comportamento do algoritmo é previsível, dado que os pacotes não são reordenados e o atraso máximo corresponde ao tamanho da fila de espera e a taxa de transmissão da interface;

- ✓ Enquanto o tamanho da fila permanecer curto, o FIFO fornece um método de contenção simples para os recursos da rede, sem acrescentar uma diferença significativa ao atraso da fila em cada nó da rede.

Tem, no entanto, bastantes limitações:

- ✓ Uma fila FIFO única não permite a organização dos pacotes em diferentes categorias, de forma que os roteadores possam servir uma classe de tráfego de forma independente das outras classes;
- ✓ Como é uma fila única, o impacto do atraso influencia de forma idêntica todos os fluxos. É possível assim, aumentar o atraso, o *jitter* e as perdas nas aplicações de tempo-real;
- ✓ Em períodos de congestionamento de rede, as aplicações que usam UDP são favorecidas quando comparadas com as que usam TCP: quando existe perda de pacotes nesta situação, as aplicações baseadas em TCP reduzem a sua taxa de transmissão, adaptando-se às condições atuais da rede, enquanto que as baseadas em UDP continuam a transmitir pacotes à mesma taxa, dado que não estão "informadas" sobre as perdas de pacotes. Assim, a largura de banda ocupada pelo TCP é reduzida, aumentando o atraso e o *jitter*;
- ✓ Um fluxo "mal comportado" pode consumir todo o espaço de memória de uma fila de espera FIFO, o que provoca a negação do serviço a todos os outros fluxos até que este acabe. Mais uma vez, a conseqüência é o aumento do atraso, *jitter*, e perdas para todos os outros fluxos UDP e TCP que passem por esta fila;

Ainda assim, o FIFO é o mecanismo de escalonamento utilizado na maioria dos roteadores, devido à sua simplicidade e eficiência.

3.1.2. SFQ – *Stochastic Fairness Queueing*

O mecanismo de escalonamento *Stochastic Fairness Queueing* pertence à família de algoritmos baseado no algoritmo *Fair Queueing*, proposto por John Nagle em 1987 [Nagle, 1987]. Estes foram desenhados para assegurar que cada fluxo tenha um acesso justo aos recursos da rede, evitando que um fluxo "mal-comportado" consuma mais do que a sua quota de largura de banda [McKenney, 2000].

Os pacotes são classificados através de diversas características (normalmente, através de um valor numérico gerado a partir da tupla [endereço de origem, porta de origem, endereço de destino, porta de destino], ou, no caso das implementações mais sofisticadas, [endereço de origem, porta de origem, endereço de destino, protocolo, porta de destino]) em filas de fluxos, e colocados numa fila (FIFO) dedicada a esse fluxo, conforme pode ser visto na Figura 9. As filas são então servidas, um pacote de cada vez, por um algoritmo *round-robin*; filas vazias não são consideradas. Sendo assim, quanto maior for o número de filas, maior será a "justiça" do algoritmo.

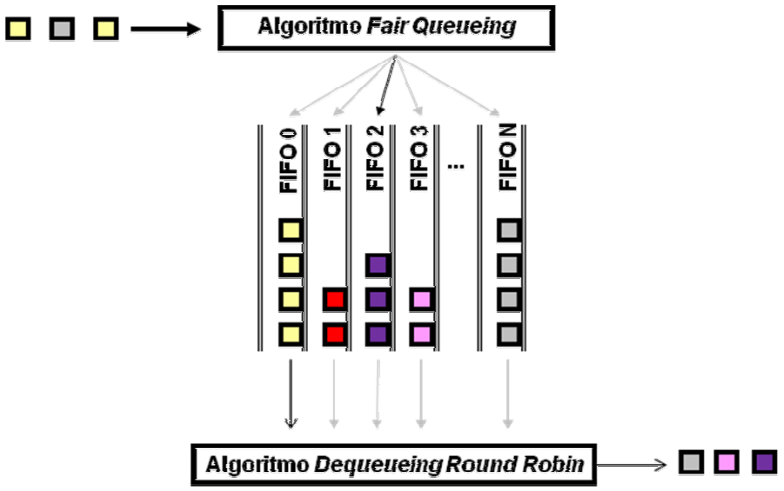

Figura 9: A estrutura do *Stochastic Fairness Queueing*.

No entanto, como não é prático haver uma fila para cada categoria de pacotes, o SFQ utiliza uma função *hash* para dividir os fluxos. Devido ao fato de se utilizar uma função *hash*, vários fluxos podem ser classificados na mesma fila, o que iria diminuir a oportunidade de cada fluxo mandar pacotes, diminuindo a largura de banda efetiva. Para prevenir esta situação, aumentando a "justiça" do algoritmo, a função é trocada periodicamente.

O *Stochastic Fairness Queueing* tem a vantagem de ser bastante simples e fácil de ser configurado. Permite também um acesso justo aos recursos da rede, apesar de não permitir diferenciação de pacotes. A sua principal vantagem em relação aos outros algoritmos da família *Fair Queueing* deve-se ao fato de que a sua complexidade computacional é inferior, apesar de ser menos preciso. Este mecanismo de escalonamento apenas pode ser utilizado em situações que não

requeiram diferenciação (exceto quando utilizado em conjunção com outros mecanismos). É implementado quando se quer assegurar o mesmo acesso a vários fluxos.

3.1.3. RED – *Random Early Detection*

O mecanismo de escalonamento de pacotes implementado nos roteadores da Internet originalmente foi o *First In First Out DropTail*. Como já visto anteriormente, com este mecanismo os pacotes são servidos por ordem de chegada, sendo colocados no fim da fila quando chegam e retirados da cabeça da fila quando há capacidade disponível no enlace. O *DropTail* é um mecanismo que consiste em descartar os pacotes que chegam quando a fila está cheia (ou, em algumas implementações, descartar o pacote que está no início da fila). Esta técnica, apesar de ser de simples utilização e implementação, origina dois problemas graves:

- ✓ *Starvation* - é o fenômeno que acontece quando um recurso (neste caso a largura de banda do enlace) é consumido excessivamente por um pequeno número de fluxos. Os fluxos restantes são bloqueados (*locked-out*), isto é, não conseguem ascender à fila e, conseqüentemente, é negado o acesso ao enlace. Neste caso, a fila é ocupada apenas pelos pacotes de um pequeno número de fluxos, enquanto que os pacotes associados à maioria dos fluxos são constantemente descartados. Este fenômeno acontece devido a efeitos de temporização, o que resulta no fato de que os pacotes de alguns fluxos encontrarem sempre a fila de espera cheia. Considerando uma situação na qual muitas fontes mandam rajadas de pacotes periodicamente que no conjunto excedem a capacidade da fila, se estas fontes estiverem sincronizadas (mandando os pacotes quase simultaneamente), os primeiros pacotes que chegam à fila (normalmente, os da fonte mais próxima do "gargalo" do enlace) irão encontrar a fila ainda com capacidade de serviço, enquanto que os que chegarem a seguir irão ser descartados. Se esta sincronização se mantiver entre as fontes, as fontes que enviam os pacotes primeiro irão fazer progresso, enquanto que todas as outras irão ver os seus pacotes sistematicamente descartados;

- ✓ *Full-queues* – As *full-queues* (filas cheias) são filas de espera que estão na grande maioria do tempo ocupadas ao máximo. Se rajadas de pacotes chegarem a uma fila de espera cheia, muitos pacotes são descartados simultaneamente, o que pode levar a grandes oscilações na utilização da rede. Se os pacotes descartados forem de fluxos diferentes, pode haver *backoffs* sincronizados entre vários fluxos. O *backoff* sincronizado é um fenômeno no qual muitas fontes recebem simultaneamente notificações de congestionamento, reduzindo conseqüentemente a quantidade de informação enviada. Como resultado, a taxa de ocupação da rede pode descer abaixo da capacidade do enlace, subindo novamente mais tarde para vir a exceder a mesma - levando mais uma vez a descartes simultâneos devido ao excesso de pacotes nas filas. Este fenômeno acontece, por exemplo, com o protocolo TCP, dado que este utiliza mecanismos de congestionamento tais como o *Slow-Start*. Este fenômeno tem o nome de sincronização global [Braden et al., 1998].

O mecanismo de escalonamento RED (*Random Early Detection*) foi criado com o objetivo de prevenir esses fenômenos [Floyd e Jacobson, 1993], fornecendo o mais rapidamente possível uma resposta a fluxos que implementem controle de congestionamento (tais como o TCP) antes que a fila encha, indicando assim que o congestionamento da rede está próximo, em vez de esperar que esse se torne excessivo. Os descartes de pacotes são também distribuídos de forma mais equilibrada entre todos os fluxos. Este efeito é obtido controlando o tamanho médio da fila de espera no roteador. Este é comparado a dois limiares (*threshold*), um mínimo e um máximo, conforme pode ser visto na Figura 10.

Quando o tamanho médio da fila é inferior ao limiar mínimo, nenhum pacote é marcado. Já se este tamanho for superior ao limiar máximo, todos os pacotes que chegam são marcados; entre estes dois valores cada pacote é marcado com probabilidade “p” (valor calculado em função do tamanho médio da fila).

O RED tem assim, dois algoritmos distintos: um para calcular o tamanho médio da fila, que vai determinar a quantidade de pacotes que vão ser descartados na fila de espera, e outro para calcular a probabilidade de marcação de pacotes, que determina qual a freqüência de marcação de pacotes do roteador, de acordo com o nível atual de congestionamento.

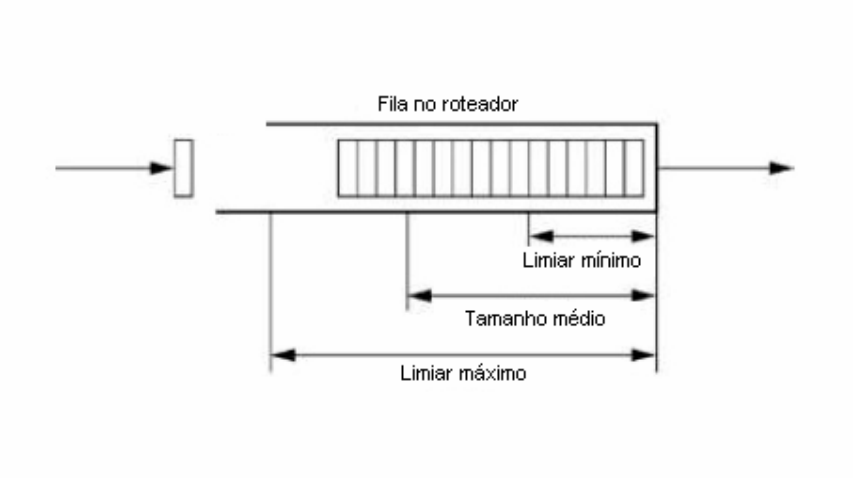

Figura 10: A fila do *Random Early Detection*, com os limites mínimo e máximo.

O objetivo é não só marcar pacotes em intervalos periódicos, de forma a evitar a sincronização global, mas marcá-los também com a frequência suficiente para controlar o tamanho médio da fila.

Existem três "modos" de descarte de pacotes:

- ✓ Quando o tamanho médio da fila de espera é inferior ao limiar mínimo, não existe congestionamento, portanto nenhuma ação é tomada - a probabilidade de descarte de pacotes vai ser zero;

- ✓ No outro extremo, quando o tamanho médio da fila de espera é superior ao limiar máximo, ou se a fila está cheia, o algoritmo assume que existe um grave congestionamento da rede - a grande maioria dos pacotes que chegam à fila de espera vão então ser descartados. A probabilidade de descarte é, assim, muito próxima de 1;

- ✓ Finalmente, quando o tamanho médio da fila de espera está entre os dois limiares, o algoritmo opera num modo probabilístico de descarte de pacotes. Assim, os pacotes vão ser descartados aleatoriamente, sendo que a probabilidade de um determinado pacote ser descartado oscila entre zero e um em função de três parâmetros: maxp, que determina a probabilidade máxima de dois pacotes consecutivos serem descartados quando se está neste modo (de descarte probabilístico); o tamanho médio da fila de espera e

o parâmetro *count*, que indica quantos pacotes foram servidos desde que o último pacote foi descartado.

Se os pacotes marcados forem efetivamente descartados, isto assegura que o tamanho médio da fila de espera não excede significativamente o limiar máximo definido.

Por outro lado, a probabilidade de um pacote de um determinado enlace ser marcado é diretamente proporcional ao percentual de largura de banda ocupada nesse enlace no roteador. Estes dois fatos asseguram que o RED previna o congestionamento na rede.

O RED é assim, um mecanismo de escalonamento bastante engenhoso, com a grande vantagem de assegurar um melhor serviço ao protocolo TCP e de impedir a ocupação excessiva das filas por pequenos fluxos. Para suportar a diferenciação de serviços, deve-se utilizar o GRED - *Generalized* RED [Almesberger et al., 1999].

3.1.4. WFQ – *Weighted Fair Queueing*

Tal como o SFQ e todos os algoritmos da família “FQ”, o WFQ (*Weighted Fair Queueing*) foi desenvolvido para assegurar que cada fluxo tenha um acesso justo aos recursos de rede, prevenindo que os fluxos “mal comportados” consumam mais do que a largura de banda que lhes é atribuída [Stiliadis e Varma, 1998].

Apesar de não ser um algoritmo de escalonamento *Classful* (os escalonadores *Classful* utilizam os mecanismos de classificação de pacotes para dividir os vários fluxos em classes, de forma a servir de forma diferente cada tipo de fluxo) e de ser orientado para garantir a “justiça” na partilha de recursos entre fluxos, como dito anteriormente, o WFQ permite fazer diferenciação entre tipos de pacotes.

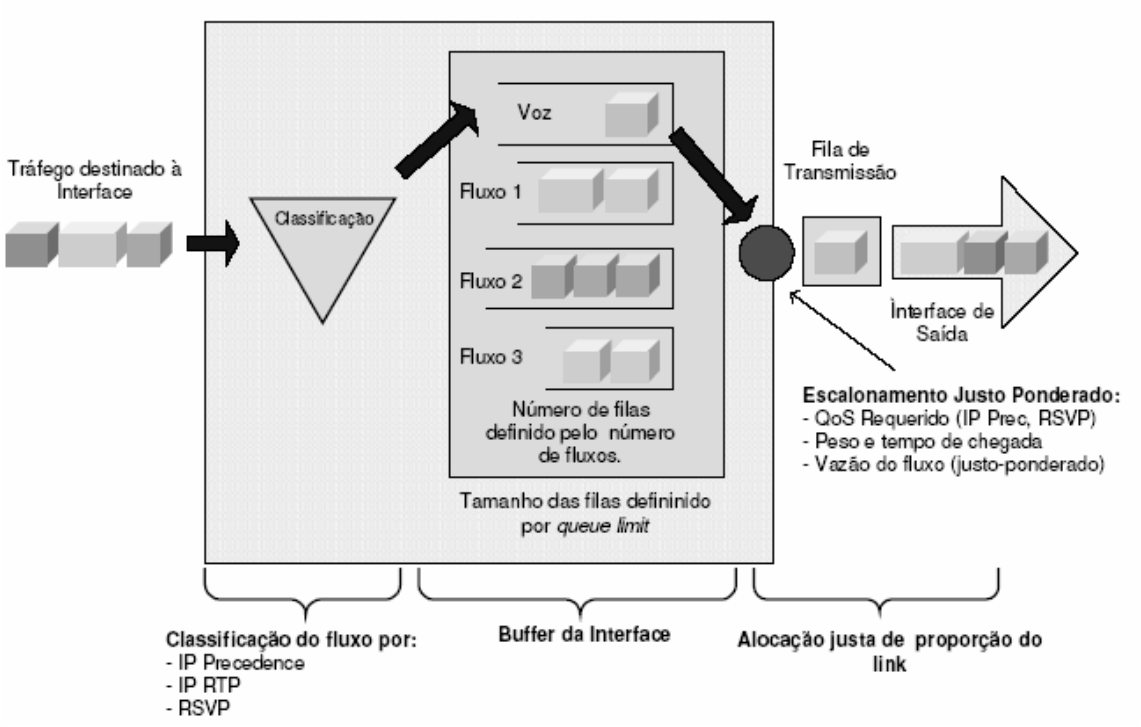

Figura 11: O funcionamento do *Weighted Fair Queueing*

Quando um pacote chega, é avaliado pelo mecanismo de classificação e atribuído a uma fila. A ordem do serviço é *weighted round-robin*, ou seja, *round-robin* com pesos, determinados por vários parâmetros como, por exemplo, uma representação numérica baseada em informação retirada do cabeçalho do pacote (endereço e porta de origem, endereço e porta de destino, protocolo, etc), conforme pode ser visto na Figura 11. Mais uma vez, tal como o SFQ, o WFQ utiliza uma função *hash*, juntamente com os parâmetros de classificação, para atribuir um pacote a uma fila. Assim, quanto maior for o número de filas, melhor é o funcionamento do algoritmo.

3.1.5. PRIO – Prioridade Exata ou PQ - *Priority Queueing*

No mecanismo PRIO (*Priority Queueing*) [Law, 1997; Ronngren e Ayani, 1997], o escalonador é constituído por diversas filas de espera, cada uma com uma prioridade diferente, sendo o tráfego servido por ordem de prioridade, como pode ser observado na Figura 12. É a forma mais simples de permitir diferenciação de tipos de fluxos. No PRIO, os pacotes são classificados pelo sistema, sendo em seguida

colocados em filas (FIFO) com diferentes prioridades [Brown, 1988]. Os pacotes das filas de prioridade seguinte são servidos apenas se todas as filas de prioridade superior estiverem vazias.

O PRIO tem a vantagem de ser a forma mais simples de implementar a diferenciação de tipos de pacotes. Por outro lado, tem uma desvantagem: no caso de não existir nenhum mecanismo de controle de admissão dos fluxos com maior prioridade, uma grande quantidade de pacotes de elevada prioridade pode impedir completamente o serviço de pacotes com menor prioridade - fenômeno designado por *starvation*.

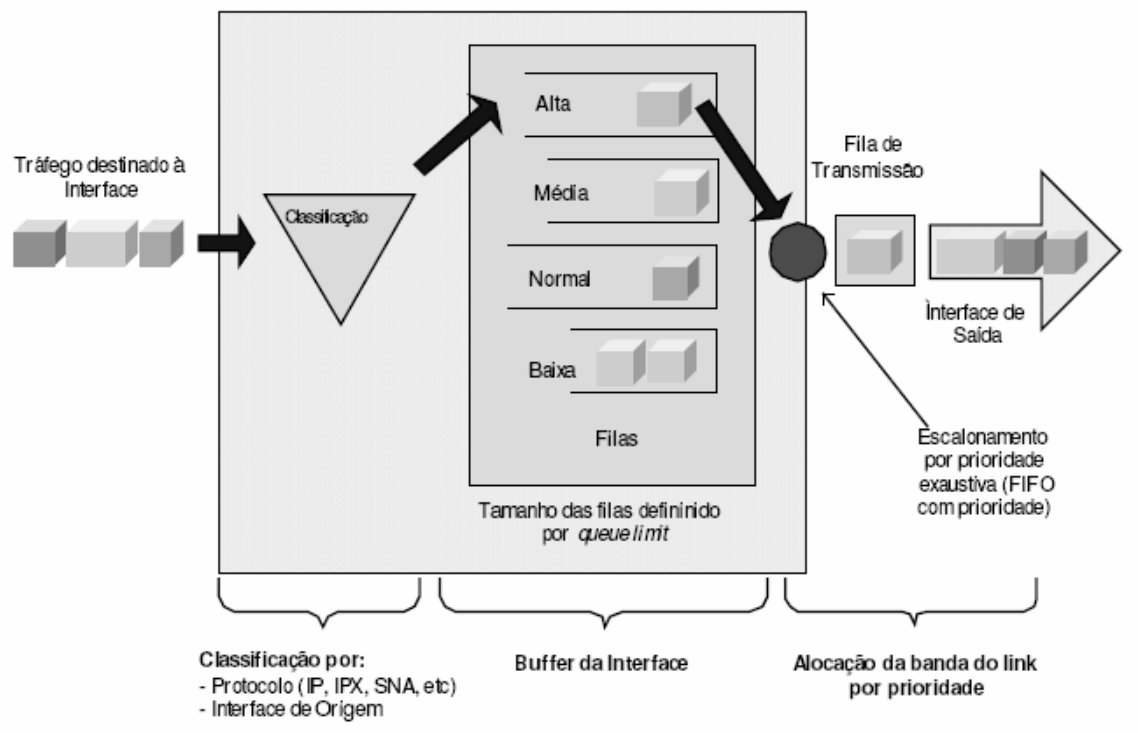

Figura 12: Priority Queueing

No entanto, isto pode ser prevenido ao se utilizar, por exemplo, o TBF (*Token Bucket Filter* – não é um mecanismo de escalonamento, mas sim um mecanismo de moldagem de tráfego (*traffic shaper*) - controla, assim, a taxa máxima na qual os pacotes são processados de uma fila) para garantir que os fluxos de maior prioridade impeçam os seguintes de serem servidos.

3.1.6. CBQ – *Class-Based Queueing*

O CBQ (*Class-Based Queueing*) [Fragouli et al., 1998] é provavelmente o mecanismo de escalonamento mais complexo existente atualmente. É constituído por dois escalonadores: um escalonador genérico e um escalonador de partilha de enlace. O escalonador genérico tenta garantir aos fluxos com requisitos de tempo real, baixos atrasos na fila de espera, enquanto que o escalonador de partilha impede que as classes de tráfego de tempo real monopolizem a utilização do enlace, e faz também a distribuição do débito excedente [Floyd e Jacobson, 1995].

Baseia-se numa estrutura hierárquica em árvore de classes de tráfego, sendo cada pacote classificado na entrada dos nós. Entre as suas principais vantagens, está o fato de ser um método de diferenciação de pacotes em classes e gerenciamento de recursos de filas. No entanto, existem problemas como a sobrecarga computacional gerada pela reordenação dos pacotes. Os principais usos são nos serviços diferenciados e redes complexas, com requisitos específicos para vários fluxos heterogêneos e/ou requisitos de tempo real.

3.1.7. HTB – *Hierarchical Token Bucket*

O HTB (*Hierarchical Token Bucket*) [Devera, 2006] é um mecanismo de escalonamento criado por Martin Devera, sendo considerado o sucessor do CBQ. O HTB implementa um escalonador *classful* para o controle de tráfego, fornecendo métodos para controlar a largura de banda que cada classe de tráfego pode utilizar, assim como indicar a razão de distribuição de largura de banda quando existe largura de banda extra disponível. O *Hierarchical Token Bucket* faz então, a partilha hierárquica de uma conexão, obedecendo a duas regras:

• Cada classe interior ou classe folha deve receber aproximadamente a sua largura de banda alocada durante os intervalos de tempo apropriados, se houver pedidos para essa classe;

• Se todas as classes folha e classes interiores com pedidos suficientes tiverem recebido pelo menos a sua largura de banda alocada, a distribuição de largura de banda excedente não deve ser arbitrária, mas sim seguir um conjunto de regras.

Conceitualmente, o HTB é constituído por um número arbitrário de *token buckets* organizados numa hierarquia. Cada classe HTB tem dois parâmetros principais – *rate* (taxa) e *ceil* (teto). A largura de banda usada entre a taxa e o teto é tomada emprestada do pai da classe. A taxa é assim a largura de banda garantida disponível para uma dada classe, enquanto que o teto é a largura de banda máxima que uma classe pode consumir. Uma parte fundamental do escalonador HTB é o mecanismo de *borrowing* (empréstimo).

Assim, o HTB assegura que a quantidade de serviço fornecida a cada classe é, no mínimo, a quantidade que esta pede, e no máximo o teto. Quando uma classe pede menos do que a quantidade a que tem direito, a largura de banda em excesso é distribuída pelas outras classes que façam pedidos de largura de banda. Para evitar que haja empréstimo de largura de banda, o que pode ser necessário, por exemplo, num provedor de serviços Internet (ISP), em que cada usuário paga pela sua conexão, basta colocar a taxa com o mesmo valor que o teto na configuração do HTB. Entre as inúmeras vantagens que advêm do uso do HTB, destaca-se o fato de ter a mesma capacidade de moldagem de tráfego que o *Token Bucket Filter* (TBF).

A configuração e o uso de classes hierárquicas são fáceis, permitindo assim a partilha de um enlace entre fluxos heterogêneos.

3.1.8. CQ – *Custom Queueing*

No mecanismo de escalonamento CQ (*Custom Queueing*) [Plachecki e Przylucki, 2003] o tráfego é classificado em múltiplas filas com limites configuráveis. Através da média do tamanho do pacote, do tamanho da unidade máxima de transmissão (MTU) e do percentual de largura de banda a ser alocado, pode-se calcular os limites das filas.

O tráfego é controlado através da alocação de uma determinada parte da fila para cada fluxo classificado. Usa um esquema *round-robin*, no qual temos, para cada fila, enviada a quantidade de pacotes referente à parte da banda alocada antes de passar para a fila seguinte, como podemos observar na Figura 13. Um contador configurável é associado a cada fila para estabelecer quantos bytes devem ser enviados antes de passar para a próxima fila.

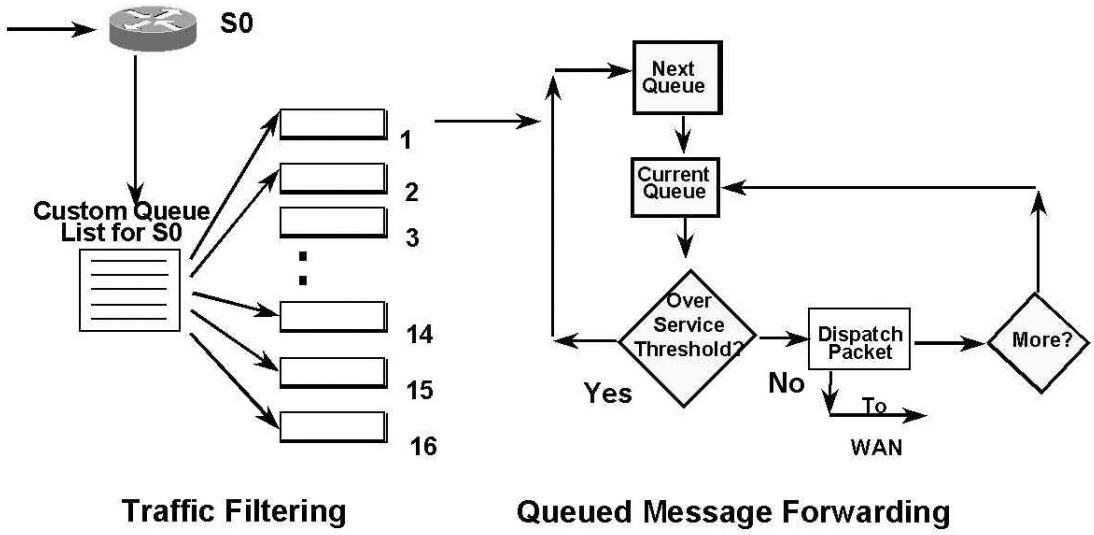

Figura 13: Operação do enfileiramento do *Custom Queueing*

Fonte [RNP, 2008]

Uma percentagem da banda pode ser alocada para uma determinada aplicação (alocação absoluta da banda). A banda reservada é compartilhada proporcionalmente, no percentual pré-definido, entre as aplicações e os usuários. O restante da banda é compartilhado entre os outros tipos de tráfego.

Como pode ser observado na Figura 14, podemos ter até 16 filas definidas. Contudo, a fila zero deve ser reservada para mensagens do sistema como sinalização, *keep alive*, etc. Pode-se fazer a classificação através do endereço fonte ou destino, por protocolo (IP, IPX, etc), por precedência IP, por interface de entrada ou ainda por listas de acesso.

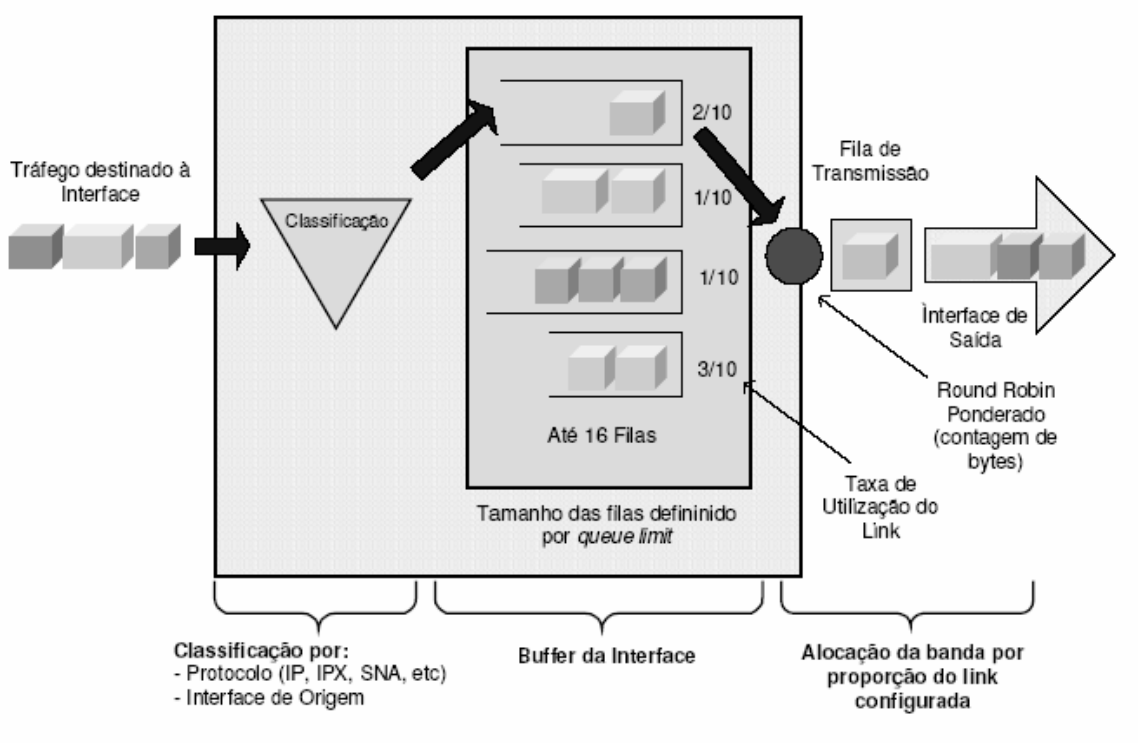

Figura 14: Filas do *Custom Queueing*

Fonte [CISCO, 2007]

Como limitação, apresenta o fato de não possuir prioridades. Além disso, os cálculos de garantias de largura de banda são aproximados e há uma limitação no número de filas.

3.1.9. CB-WFQ – *Class-Based* WFQ

O mecanismo CB-WFQ (*Class-Based WFQ*) [Davidson et al., 2004] combina as características de outras políticas de escalonamento. Do *Custom Queueing* é herdada a garantia de banda. Quanto à distribuição justa entre diversos fluxos, este mecanismo utiliza a mesma idéia do *Weighted Fair Queueing*. Por fim, para atendimento às filas, o WRR, mecanismo explicado a seguir é utilizado [Xiuzhong e Xuan, 2003].

O CB-WFQ é bastante similar ao LLQ (comentado adiante), com exceção de que não há fila de prioridade. É um dos tipos mais flexíveis de técnicas de escalonamento, porém, por não ser tão simples de implementar, é recomendado

apenas quando os requisitos de QoS o exijam. A Figura 15 mostra o princípio de funcionamento do mecanismo de escalonamento CB-WFQ (*Class-Based WFQ*).

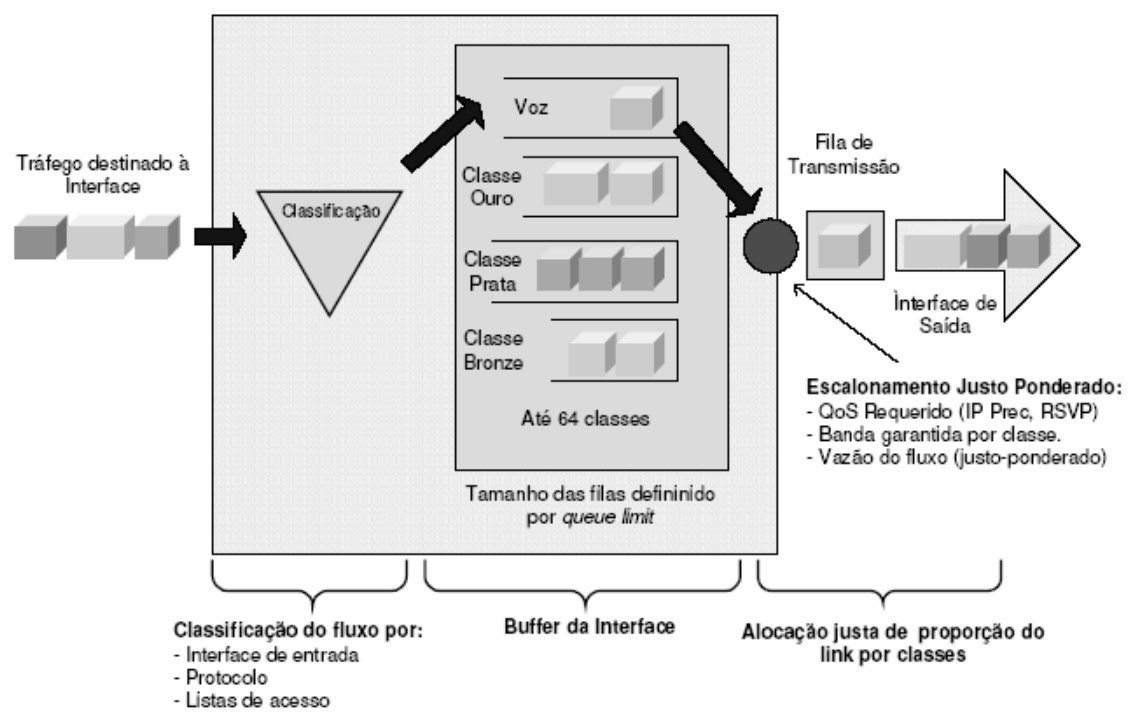

Figura 15: O mecanismo de escalonamento *CB-WFQ*

Fonte [CISCO, 2007]

Como limitação, temos o fato de que não é adequado para garantir prioridade absoluta para aplicações de tempo real, como voz e vídeo.

3.1.10. WRR – *Weighted Round-Robin*

O mecanismo WRR (*Weighted Round-Robin*) serve aos fluxos proporcionalmente aos pesos atribuídos [Xiuzhong e Xuan, 2003].

- ✓ Caso 1: pesos diferentes e pacotes com comprimento fixo; é servido mais do que um pacote por fila e por ciclo, após normalização para obter pesos inteiros;

- ✓ Caso 2: pesos diferentes e pacotes com comprimento variável; os pesos são normalizados pelo tamanho médio dos pacotes, como exemplificado a seguir:
  - – pesos: 0.5, 0.75, 1.0
  - – tamanho médio dos pacotes: 50, 500, 1500
  - – pesos normalizados: 0.5/50, 0.75/500, 1.0/1500 = 60, 9, 4

Uma das limitações do algoritmo WRR deve-se ao fato de que o mesmo pressupõe o conhecimento do tamanho médio dos pacotes gerados por cada fonte. Na maioria dos casos isto é imprevisível, e impede uma distribuição de recursos de acordo com o critério max-min *fairness*.

Além da limitação mostrada acima, temos ainda a "justiça" do algoritmo apenas em escalas temporais superiores a duração de um ciclo. Durante ciclos longos, podemos ter "injustiça" para fluxos com pequeno peso ou quando existe um número elevado de fluxos, como podemos ver no exemplo a seguir:

- ✓ Ligação T3 (45 Mbit/s) com 500 ligações (250 com peso 1 e 250 com peso 10); tamanho médio dos pacotes: 500 bytes

- ✓ Tempo de transmissão de um pacote: $500 \times 8 / (45 \times 10^6) = 0{,}0000888$ s

- ✓ Duração de um ciclo: $(250 \times 10 + 250 \times 1) \times 0.0000888 = 244{,}2$ ms

- ✓ Em intervalos muito menores do que 244,2 ms, algumas ligações obtêm uma quota de serviço muito superior a de outras.

3.1.11. WRED – *Weighted RED*

O mecanismo de escalonamento WRED (*Weighted RED*) é uma implementação proprietária do fabricante Cisco [Cisco, 2007]. Combina as funcionalidades de classificação de pacotes por precedência de IP com as funcionalidades do RED. Através desta classificação, o algoritmo descarta pacotes

seletivamente, descartando inicialmente os pacotes de menor prioridade, com diferentes pesos para cada classe, como pode ser visto na Figura 16.

Existe a possibilidade de desabilitarmos a classificação por precedência IP e habilitar o descarte com base apenas no tamanho do buffer da fila. Podemos também utilizar o RED em conjunto com o RSVP, caso desejemos um descarte mais seletivo. Nesse caso, antes que ocorra uma situação de congestionamento, os fluxos de menor prioridade nas sessões RSVP serão descartados antes dos outros de maior prioridade.

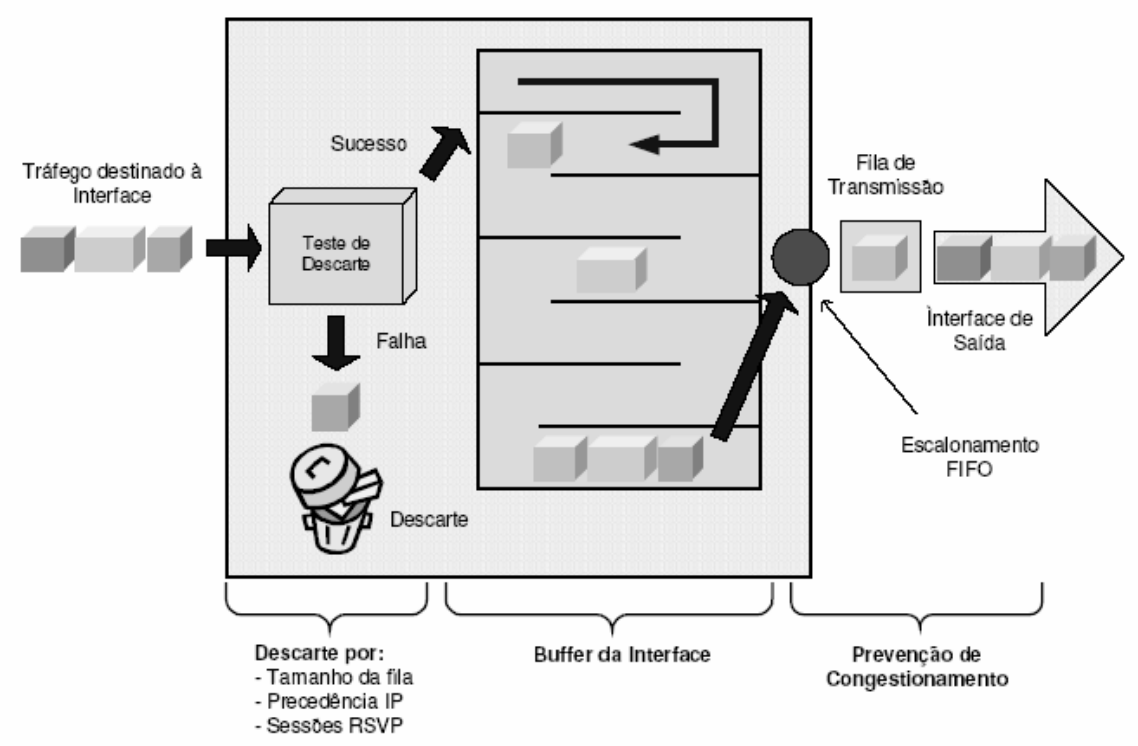

Figura 16: Filas de funcionamento do WRED

Fonte [CISCO, 2007]

WRED é útil em qualquer interface na qual a possibilidade de congestionamento seja iminente. Entretanto, é geralmente utilizado em roteadores centrais de *backbone* (*core routers*), com a precedência IP habilitada pelos roteadores de borda (*edge routers*).

3.1.12. LLQ ou PQ-CBWFQ – *Low Latency Queueing*

A técnica de escalonamento LLQ (*Low Latency Queueing*) ou PQ-CBWFQ [Cisco, 2007] é utilizada na interface de saída de dispositivos de rede e consiste em habilitar várias filas para transmissão, como pode ser observado na Figura 17. Estas filas são disponibilizadas de acordo com as classes dos pacotes, além de uma fila de prioridade.

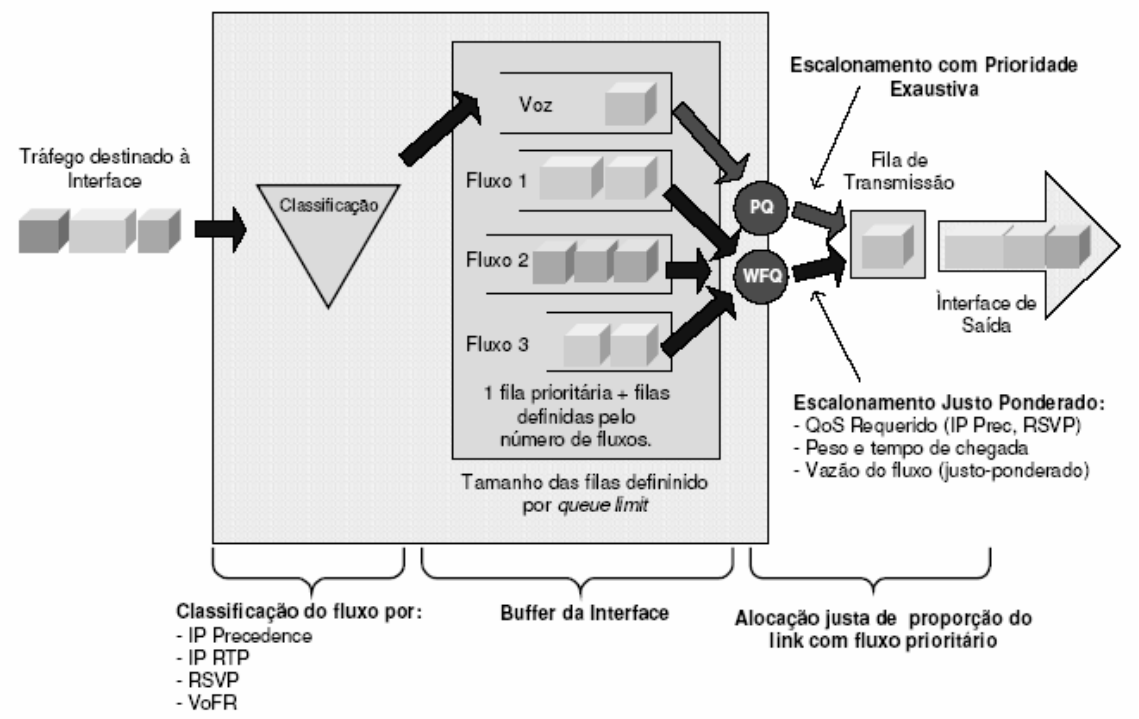

Figura 17: O mecanismo de escalonamento LLQ

Fonte [CISCO, 2007]

O algoritmo de enfileiramento de pacotes LLQ assegura baixa latência e *jitter* reduzido para aplicações de tempo real, garantindo também que o tráfego de menor prioridade seja enviado ao longo do tempo, evitando assim o chamado s*tarvation* (possibilidade do tráfego de menor prioridade nunca ser enviado) [Kurose e Ross, 2004].

3.1.13. GPS – *Generalized Processor Sharing*

O objetivo do algoritmo GPS (*Generalized Processor Sharing*) [Parekh e Gallager, 1993] é a atribuição de recursos de acordo com o critério max-min *fairness*. Existe uma fila por fluxo. Durante qualquer intervalo de tempo em que existam N filas não vazias, o servidor serve simultaneamente os N pacotes do início das filas, cada um com um débito igual a 1/N da capacidade do enlace físico (N varia à medida que se completam e/ou iniciam serviços). De forma equivalente, pode-se dizer que é servida uma quantidade de dados infinitesimal em cada visita a uma fila (uma aproximação consistiria em servir rotativamente um bit de cada fila não vazia). A designação FFQ (*Fluid Fair Queueing*) é também usada como sinônimo de GPS.

3.1.14. VC – *Virtual Clock*

O algoritmo VC (*Virtual Clock*) [Zhang, 1991] tem como objetivo emular um sistema TDM (*Time Division Multiplexing*). A cada pacote é atribuído um instante virtual de transmissão (*virtual transmission time*), correspondente a um serviço TDM, porém sem garantias quanto ao atraso fim-a-fim. Os pacotes são servidos por ordem crescente desses tempos. Entretanto, por se tratar de um sistema TDM, onde eliminamos completamente a interferência entre fluxos, a capacidade do canal é perdida quando em um determinado instante de tempo o fluxo correspondente não possui dados a serem transmitidos.

3.2. MECANISMOS DE POLICIAMENTO DE TRÁFEGO

O conceito de policiamento de tráfego (*Traffic Policing*) pode ser entendido como um mecanismo para controlar o montante e o volume de tráfego que é enviado para a rede e a taxa em que este está sendo enviado. Além disso, pode também servir para identificar diferentes fluxos de tráfego no ponto de entrada da rede com uma granularidade que permita ao método utilizado, separar o tráfego em fluxos individuais e tratá-los diferentemente.

Destacam-se dois métodos para o policiamento de tráfego nas redes de pacotes, os quais são descritos em [Ferguson e Huston, 1999] e [Linney, 1999] que são: o método do balde furado (*leaky bucket*) e o método do balde de *tokens* (*token bucket*).

3.2.1. O MÉTODO DO BALDE FURADO (*Leaky Bucket*)

A principal idéia do algoritmo do balde furado (*Leaky Bucket*) tem como base uma balde com um pequeno furo embaixo, conforme Figura 18.

Figura 18: O método do balde furado – água

A velocidade com que a água entra no balde não tem influência no fluxo de saída, que fica em uma taxa constante e, uma vez o balde cheio, toda água que nele entrar escorrerá pelos lados e será perdida. Porém, ao invés de água, utiliza-se pacotes conforme Figura 19.

Cada *host* é conectado à rede por uma interface que contém um “balde furado”. Caso um pacote chegue e a fila esteja cheia, ele será descartado.

Além disso, o *host* pode inserir um pacote a cada pulso de relógio na rede, o que transforma um fluxo de pacotes irregular dos processos de usuário dentro do *host* em um fluxo de pacotes regular para a rede. Desta forma, haverá suavização de rajadas e serão reduzidas as chances de ocorrência de um congestionamento. Quando temos pacotes do mesmo tamanho (células ATM, por exemplo), esse algoritmo pode ser usado da forma descrita acima. Se forem utilizados pacotes de tamanho variável, a melhor opção é permitir um número fixo de bytes por pulso, em vez de apenas um pacote.

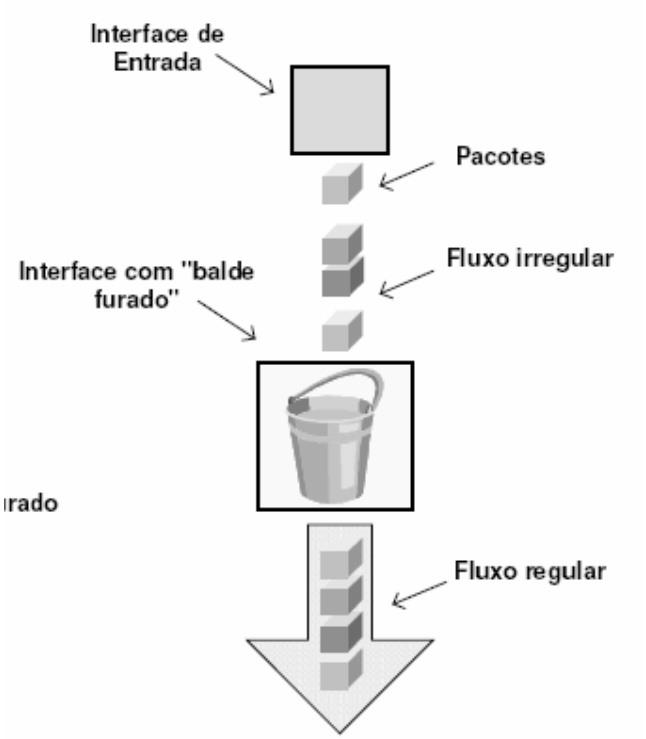

Figura 19: O método do balde furado – pacotes

O tamanho do balde e a taxa de transmissão geralmente são configuráveis pelo usuário e medidos em bytes.

3.2.2. O MÉTODO DO BALDE DE TOKENS (*Token Bucket*)

A idéia do algoritmo do balde de tokens (*Token Bucket*) é modelar e controlar as taxas de transmissão. A análise do algoritmo pode ser feita através de uma comparação com um balde com fichas (tokens), onde cada ficha significa permissão para transmitir uma certa quantidade de tráfego, que pode ser medida em bytes ou em pacotes. Em muitas aplicações, é melhor permitir que a saída aumente um pouco sua velocidade quando chegarem rajadas maiores. Para que haja este controle, existe o algoritmo do balde de *tokens*, onde o balde retém *tokens*, gerados por um *clock* na faixa de um *token* a cada ΔT segundos.

Na Figura 20 vemos um balde com três *tokens*, com cinco pacotes aguardando para serem transmitidos. Para que um pacote seja transmitido, ele deve capturar e destruir um *token*.

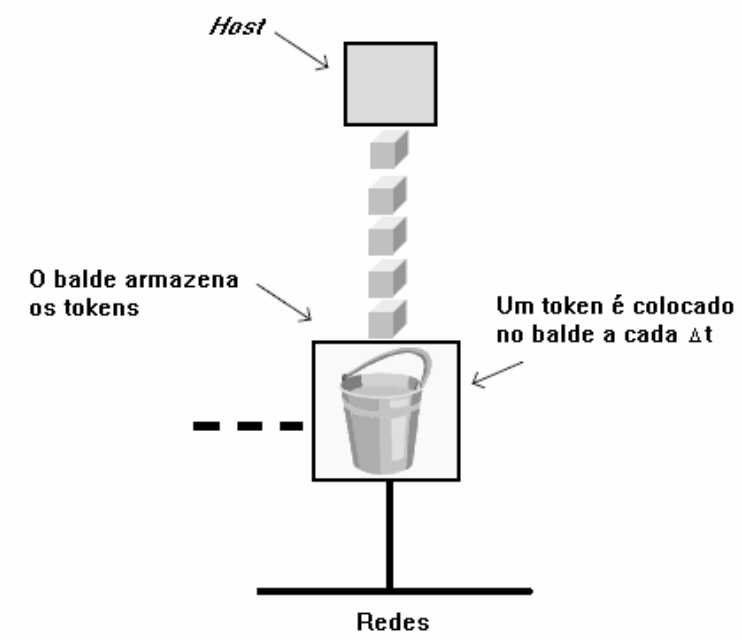

Figura 20: O método do balde de *tokens*

Na Figura 21, vemos que três dos cinco pacotes são transmitidos, mas os outros dois estão aguardando que dois outros *tokens* sejam gerados.

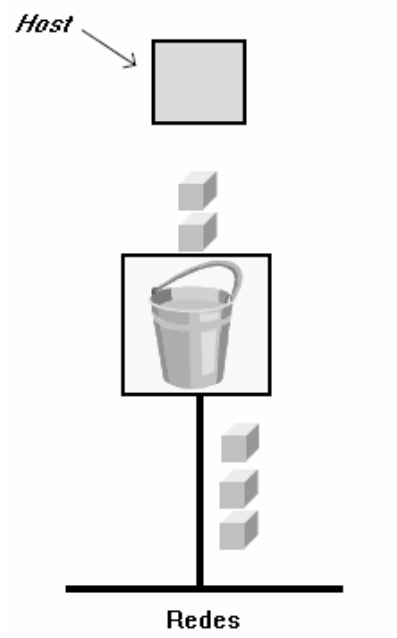

Figura 21: O método do balde de *tokens*

O algoritmo do balde de *tokens* joga *tokens* fora quando o balde enche, mas nunca descarta pacotes. Pelo fato de o balde de *tokens* permitir rajadas em pequenos intervalos de tempo, o tráfego pela rede será mais regular se for colocado um balde furado após o balde de *tokens*.

3.3. RESUMO DO CAPÍTULO

O Capítulo 3 mostrou os mecanismos de escalonamento de pacotes e policiamento de tráfego. A primeira parte abordou diversos mecanismos de escalonamento, desde os mais simples, como o FIFO, por exemplo, até o HTB, que apresenta-se como um mecanismo bastante eficaz e que traz grandes vantagens no uso. A segunda parte explicou sobre os mecanismos de policiamento de tráfego. Descreveu dois tipos de policiamento de tráfego bastante utilizados, que podem ser resumidos em mecanismos que controlam o montante e o volume de tráfego que é enviado para a rede e a taxa em que este está sendo enviado.

O capítulo a seguir apresenta a proposta para priorização de fluxos de voz, objeto do trabalho desta dissertação.

4. PROPOSTA DE CLASSIFICAÇÃO DE FLUXOS DE VOZ BASEADA NA IMPORTÂNCIA DA CHAMADA

Com a expectativa da rede e da operadora de serviços em atender prioritariamente aos destinos mais importantes e, posteriormente, aos de maior retorno, podemos fazer uma classificação das chamadas de voz de acordo com os números de destino, como apresentado a seguir.

A. Números especiais.

- ✓ Números de emergência - polícia (190), ambulância (192), bombeiro (193), polícia rodoviária federal (195) e defesa civil (199);
- ✓ Números de acesso à telefonista;

B. Números locais, longa distância (LD) e internacionais.

- ✓ Números locais - 8 dígitos ou com 0 CN + 8 dígitos, sendo CN (código nacional) igual à área de origem;
- ✓ Números de longa distância - 0 CS CN + 8 dígitos, sendo CS o código da prestadora de serviços de longa distância e o CN diferente da área de origem;
- ✓ Internacionais - 00 CS + dígitos, sendo CS o código da prestadora de serviços internacionais.

C. Chamadas a cobrar locais e de longa distância.

- ✓ Chamadas a cobrar locais - 9090 + 8 dígitos;
- ✓ Chamadas a cobrar de longa distância - 90 CS CN + 8 dígitos, sendo CS o código da prestadora de serviços de longa distância e o CN diferente da área de origem;

D. Chamadas para serviços de terceiros

- ✓ Numeração 0800 – ligação gratuita destinada geralmente às centrais de atendimento;

- ✓ Numeração 0300 – ligação cobrada. Destinada geralmente aos serviços de consulta ou promoções;
- ✓ Numeração 0500 – ligação cobrada. Destinada geralmente às campanhas beneficentes.

E. Outros tipos de chamadas

- ✓ Chamadas utilizando as teclas "*" e "#" no início. Geralmente para acesso de serviços da operadora. Tais ligações podem ser incluídas em uma mesma classe.

Como podemos observar, existe uma gama enorme de dígitos que podem ser utilizados pelo cliente sem que a operadora possa diferenciar a QoS oferecida. Mesmo usando modelos para diferenciação de serviços em redes IP, como *Diffserv*, por exemplo, a classificação de fluxos disponível não é suficiente para separá-los por tipo de fluxo, tal como voz ou vídeo, e ainda subclassificar as chamadas de voz pela sua importância. A proposta apresentada nesta dissertação sugere uma classificação principal dos fluxos, baseada no tipo de serviço oferecido, e ainda uma subclassificação para cada tipo, dependendo da importância da chamada e/ou do usuário chamador [Ferreira et al., 2008]. Apesar de esta proposta ser genérica para qualquer tipo de serviço, esta dissertação foca apenas em serviços de voz.

A idéia principal é marcar cada datagrama IP que carrega informação de voz com uma subclasse baseada no número de destino e na categoria do usuário que está realizando a chamada. Isto é necessário para diferenciar as chamadas e oferecer qualidades de serviço distintas para fluxos de voz. Como a definição do campo ToS (*Type of Service*) pelo DiffServ apresenta um número muito limitado de classes, esta dissertação propõe uma redefinição do campo ToS do protocolo IPv4 [DARPA, 1981], cujos 8 bits são utilizados da seguinte forma: 2 bits para determinação da classe, 4 bits para análise do número de destino e 2 bits para análise do usuário originador, conforme apresentado na Tabela 5.

Tabela 5: Classificação dos fluxos e destinos

		TOS								Prioridade no geral		Prioridade no grupo	
		DSCP						User **					
		Tipo											
		0	1	2	3	4	5	6	7				
Sinalização		0	0	0	0	0	0	0	0	4		0	
Dados		0	1	0	0	0	0	0	0	3		0	
Vídeo		1	0	0	0	0	0	0	0	2	*	0	*
Voz	Especiais	1	1	1	1	1	1	0	0	1		1	
	Locais, LD e Internacionais	1	1	1	1	1	0	0	0	1		2	
	Chamadas a cobrar locais e LD	1	1	1	1	0	1	0	0	1		3	
	Chamadas para destinos 0800	1	1	1	1	0	0	0	0	1		4	
	Chamadas para destinos 0300	1	1	1	0	1	1	0	0	1		5	
	Chamadas para destinos 0500	1	1	1	0	1	0	0	0	1		6	
	Outros tipos de chamadas	1	1	1	0	0	1	0	0	1		7	
	Reservado	1	1	1	0	0	0	0	0	1		0	

* As setas representam maior prioridade

** Os bits destinados a "User" identificam a importância do usuário.
11 - Especiais para emergência
10 - VIP
01 - Normal
00 - Pré-pago

A coluna “prioridade no geral” estabelece o grau de atendimento de serviço da classe, onde temos: sinalização, dados, vídeo e voz. Como mostrado na Tabela 5, a classe voz prevalece sobre as demais classes. A coluna “prioridade no grupo” estabelece o grau de atendimento dentro daquela classe. Ou seja, na classe voz, temos a subclasse “Especiais” com maior prioridade dentre as demais.

Para marcar os pacotes de voz, de acordo com a importância da chamada, a marcação precisa ser feita no cliente que a inicia e identifica o número de destino. Para dar tratamento diferenciado a fluxos de voz distintos, priorizando os mais importantes, cada roteador (de borda ou de núcleo) deve implementar um mecanismo de escalonamento que permita tal diferenciação. Em nossa proposta, é necessário um mecanismo que diferencie classes e subclasses de maneira hierárquica. Dentre os mecanismos de escalonamento existentes, detalhados no Capítulo 3, o mecanismo HTB (*Hierarchical Token Bucket*) [Devera, 2006] foi selecionado. Tal mecanismo foi escolhido por implementar um escalonador *classful* (escalonadores *classful* são aqueles que utilizam os mecanismos de classificação de pacotes para dividir os vários fluxos em classes para servir de forma diferente cada tipo de fluxo), fornecendo métodos para controlar a largura de banda que cada

classe e/ou subclasse pode utilizar, assim como indicar a razão de distribuição de largura de banda quando existe largura de banda extra disponível. Uma parte fundamental do escalonador HTB é o mecanismo de *borrowing* (empréstimo). Assim, o HTB assegura que a quantidade de serviço fornecida a cada classe é, no mínimo, a quantidade que esta pede, e no máximo o teto. Quando uma classe usa menos do que a quantidade a que tem direito, a largura de banda em excesso é distribuída pelas outras classes que façam uso de largura de banda. A Figura 22 ilustra a proposta de classificação de fluxos de voz baseada na importância da chamada.

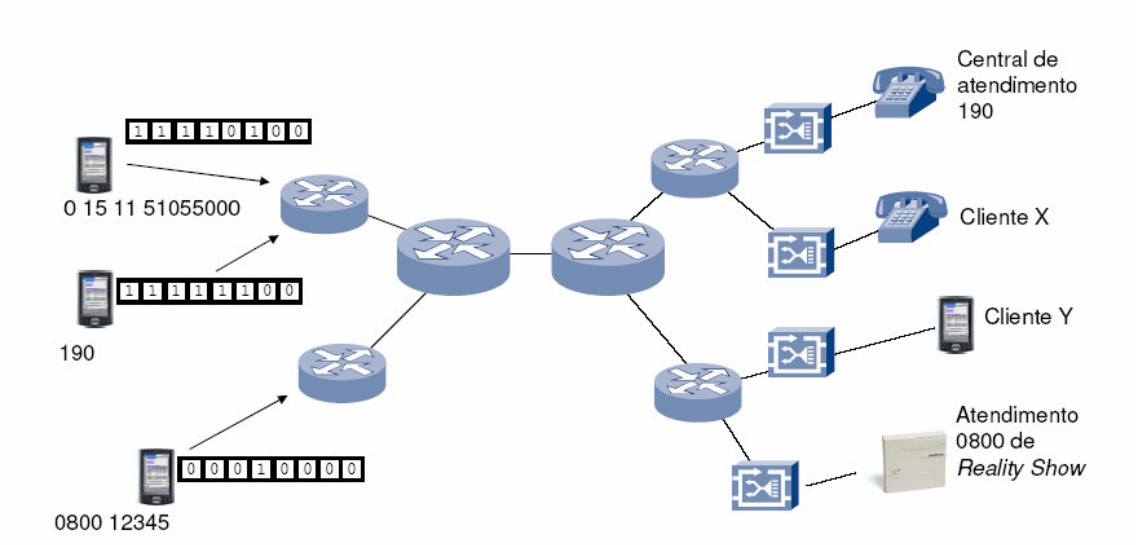

Figura 22: Proposta de classificação de fluxos de voz.

Na ilustração foi montada uma topologia de rede onde temos três usuários discando para destinos diferentes. Os dígitos discados pelo primeiro usuário referem-se a uma ligação de longa distância, utilizando-se do código de seleção de prestadora 15. Ainda, compartilhando o acesso no mesmo roteador de borda, existe outro usuário ligando para um destino de emergência que, neste caso é o 190 da polícia. Por último, o terceiro usuário liga para um destino 0800, que será atendido por uma central designada como “Atendimento 0800 de *Reality Show*”. Os dois primeiros, por estarem acessando um mesmo roteador, já sofrerão uma classificação de pacotes. Baseando-se na premissa anterior mostrada na Tabela 5, temos que, por se tratarem de fluxos de voz, ambos terão a mesma prioridade geral. Entretanto, dentro da classe voz, o fluxo de voz cujo destino é um número de emergência terá prioridade total. Esta priorização valerá para todos os roteadores da rede, tanto os

de borda quanto os de núcleo. A Figura 23 ilustra um esquema onde podemos ver as filas nos roteadores.

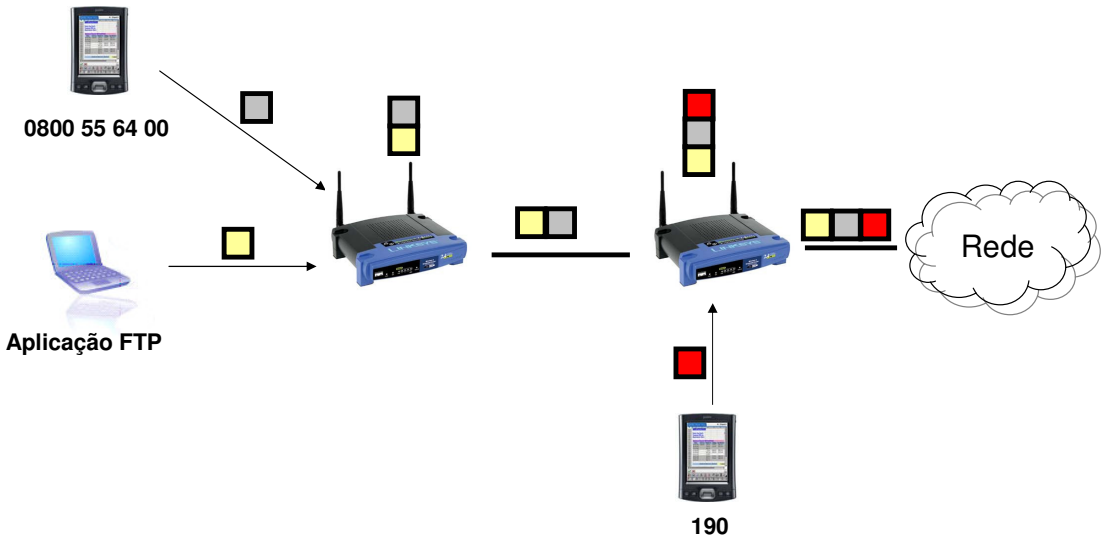

Figura 23: Exemplo ilustrando as filas nos roteadores.

Pode-se notar que, mesmo os pacotes entrando no instante posterior na fila do roteador, os de mais alta prioridade sempre terão prioridade de encaminhamento para atingir o seu objetivo. Na ilustração da Figura 23, temos um fluxo de uma aplicação FTP sendo enviado juntamente com um fluxo de voz cujo destino é um número de emergência. Por ser utilizado o mecanismo de escalonamento HTB, temos esta priorização e o mecanismo de empréstimo. Uma vez que cessemos um dos fluxos acima, o valor reservado para o mesmo poderá ser emprestado para outros fluxos. Além de garantir um determinado valor, observamos também que há a priorização de encaminhamento, o que torna a proposta interessante para garantirmos acessos diferenciados e importantes no sistema. Ainda, temos a oportunidade de controlarmos o núcleo da rede, manipulando a prioridade e a banda reservada para os respectivos fluxos. Deve-se observar ainda que, a priorização sozinha não garante uma política de QoS totalmente efetiva se não for associado a ela um mecanismo de controle de admissão (para os fluxos de mídia contínua, em particular). Para o estudo realizado, não foram considerados mecanismos de controle de admissão.

5. IMPLEMENTAÇÃO NA REDE MESH

A proposta de classificação de fluxos de voz baseada na importância da chamada apresentada no Capítulo 4 é genérica e pode ser implementada em qualquer rede IP. Para a implementação da proposta, fornecendo tratamento diferenciado aos fluxos, dependendo da subclasse, é necessário um ambiente de rede real em que se possam programar os roteadores, implementando a marcação dos pacotes e modificando os mecanismos de escalonamento.

A UFF desenvolve um projeto de rede em malha sem fio [Passos et al., 2006] utilizando roteadores configuráveis, com sistema operacional *OpenWRT*, baseado em Linux. Em particular, a rede *mesh* implantada pela UFF consiste de roteadores sem fio, instalados tipicamente em locais altos, como topo de edifícios, comunicando-se entre si em modo *ad-hoc* através de múltiplos saltos, de forma a encaminhar mensagens aos seus destinos. Redes *mesh* possuem a vantagem de serem redes de baixo custo, fácil implantação e bastante tolerantes a falhas. Por sua vez, os usuários nos locais onde se tem ponto de acessos da rede *mesh* podem se conectar de forma cabeada, tipicamente via *Ethernet*, ou de forma sem fio, através de redes IEEE 802.11. A infraestrutura de um *backbone* típico de uma rede *mesh* pode ser observada na Figura 24.

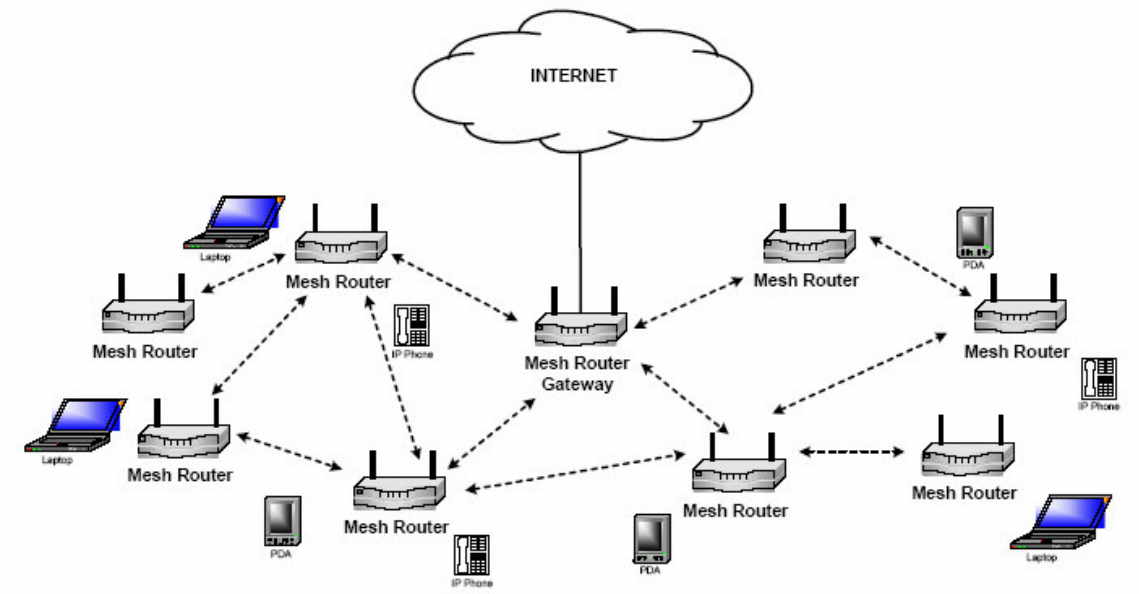

Figura 24: Infra-estrutura / backbone de uma rede mesh.

Fonte [Abelem, 2007]

A rede *mesh* da UFF foi usada como *testbed* para a implementação desta proposta. A rede *mesh* interna da UFF está ilustrada na Figura 25. Em um ambiente de rede sem fio, pelo fato de ter largura de banda mais escassa e grande variabilidade de qualidade em seus enlaces, o congestionamento da rede pode acontecer mais frequentemente, o que reforça a necessidade de um mecanismo que dê tratamento diferenciado aos diversos fluxos que utilizam a rede a cada momento.

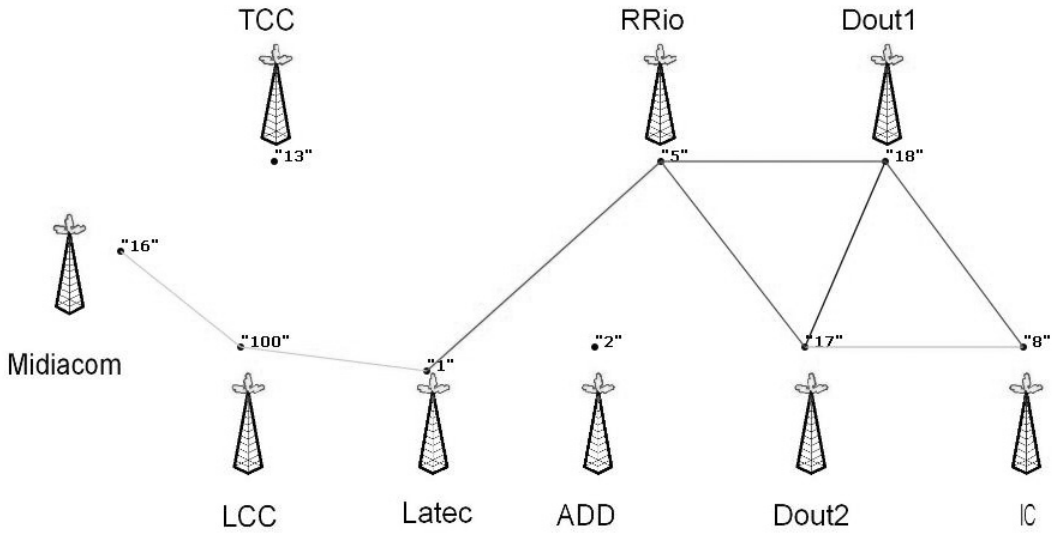

Figura 25: Rede *mesh* interna UFF.

A Figura 25 mostra a localização dos nós da rede *mesh* em laboratórios do Departamento de Engenharia de Telecomunicações e do Departamento de Ciência da Computação da UFF. Os números acima dos nós indicam o ID do elemento de rede.

O *backbone* da rede *mesh* [IEEE, 2007] montada na UFF [Projeto ReMesh, 2008] está implementado com roteadores modelo WRT54G e WRT54GS da Linksys, que se comunicam através do padrão IEEE 802.11g. Cada roteador teve o seu *firmware* original substituído pelo sistema operacional *OpenWRT*.

Para a implementação do mecanismo de escalonamento, foi utilizada a ferramenta *Traffic Control* (TC) [Hubert, 2005], disponível no *OpenWRT* e em diversas versões do Linux. Esta ferramenta possui um sistema bastante eficiente de

gerenciamento de largura de banda, classificação, priorização, divisão de recursos e limitação de tráfego, já oferecendo a implementação do mecanismo de escalonamento HTB. Com a ferramenta TC instalada, foi necessária a inserção de dois *patches* para que o HTB pudesse ser utilizado[1].

O sistema de filtros da ferramenta TC oferece uma variedade de possibilidades para classificação de pacotes. Basicamente, a classificação pode ser feita através da verificação dos campos do pacote, como por exemplo, o cabeçalho IP, através das rotas existentes na tabela de rotas ou ainda através das marcações feitas por um firewall no pacote [Hilario, 2006]. De acordo com [Hubert, 2005], a classificação mais utilizada é a primeira, a qual classifica de acordo com os valores dos campos do pacote. Tal filtro é chamado de u32 e possui formato simples, que consiste de dois campos: o seletor e a ação. Os seletores fazem a comparação do cabeçalho IP do pacote que está sendo processado com as diversas regras existentes, até que a primeira combinação seja encontrada. Cada regra de filtro está associada a uma ação que deve ser executada.

A proposta de classificação de fluxos de voz indica que a estação cliente realize a marcação dos pacotes de voz de acordo com a importância da chamada e/ou do usuário chamador. Para implementar essa função, pode-se modificar um cliente VoIP, ou então, eleger um aplicativo de geração de tráfego que abra possibilidades de alterações dos campos. Para a realização dos testes de validação da proposta de classificação de fluxos de voz foram utilizados os softwares IPERF [IPERF, 2007] e CallGen [H323 Call Generator, 2009]. A utilização do aplicativo IPERF foi feita de forma a realizar a marcação dos pacotes na origem, informando a subclasse correspondente, conforme a importância da chamada. Tendo o cliente marcado um destino longa distância, por exemplo, e sendo o mesmo um usuário pré-pago, o campos ToS (*Type of Service*) do cabeçalho IP será preenchido como pode ser visto na Figura 26.

[1] Estas correções/*updates* foram instaladas a partir do *site* http://downloads.openwrt.org/whiterussian/rc5/packages/. Para a instalação, os *patches* foram inseridos no diretório raiz dos roteadores e executados com o comando *ipkg install arquivo.ipk*. Os arquivos a serem instalados são: kmod-sched_2.4.30-brcm-3_mipsel.ipk e tc_2.6.11-050330-1_mipsel.ipk.

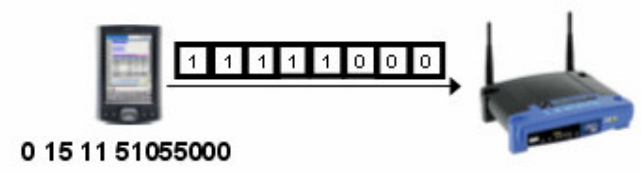

Figura 26: Pacote Classificado.

5.1. *TRAFFIC CONTROL* (TC)

O TC (*Traffic Control*) é uma ferramenta oferecida nos sistemas operacionais convencionais Linux 2.2 e superiores, e também está disponível no sistema operacional *OpenWRT*, usado nos roteadores da rede *mesh* da UFF. Entre suas funcionalidades, oferece métodos para classificação, priorização, divisão de recursos e limitação de tráfego.

Nesta ferramenta, são criadas disciplinas de enfileiramento para gerenciamento de largura de banda, conhecidas como *Queueing Disciplines* ou simplesmente qdiscs. As mesmas apresentam a flexibilidade de mudar a maneira de como os dados são tratados.

Explicando melhor alguns termos encontrados na ferramenta *Traffic Control*, temos as seguintes definições:

1 – qdisc é um algoritmo que gerencia a fila de um dispositivo (nó). As filas podem ser de entrada (*ingress*) ou saída (*egress*). Como dito anteriormente, é possível manusear os dados que chegam ou que saem de um nó;

2 – Uma *root* qdisc é uma qdisc associada ao dispositivo ou nó;

3 – Existem duas classificações para as disciplinas de enfileiramento. A primeira é conhecida como *classless* qdisc e caracteriza-se por não possuir subdivisões internas configuráveis. Como exemplo de *classless* qdisc temos os mecanismos Pfifo_fast, TBF (*Token Bucket Filter*) e SFQ (*Stochastic Fairness Queueing*) [Hubert, 2005]. A outra classificação, conhecida como *classful* qdisc, contém múltiplas classes internas. Como exemplo de tais filas, temos os mecanismos PRIO, HTB (*Hierarquical Token Bucket*) e CBQ (*Class Based Queueing*).

Os diferentes tipos de tráfego que passam por um dispositivo são categorizados pelas várias classes de uma *classful* qdisc. Estas classes podem conter outras qdiscs ou até mesmo outras classes. Desta forma, uma classe pode ter uma qdisc ou outra classe como pai. Uma classe folha é uma classe que não possui filhas. Esta classe possui uma qdisc associada a ela, a qual é responsável por enviar os dados da classe. Quando uma classe é criada, uma disciplina de enfileiramento FIFO (fifo qdisc) é associada a ela por padrão. Quando se adiciona uma classe filha, esta qdisc é removida. Para as classes folha, o mecanismo fifo qdisc pode ser substituído com um outro tipo de qdisc.

Para determinarmos para qual classe uma determinada qdisc deverá enviar um pacote que entrou pela rede, deve-se utilizar classificadores. Essas classificações podem ser feitas usando filtros, que contêm um número de condições que devem ser analisadas quando um pacote precisa ser classificado e, caso os dados do pacote casem com as informações do filtro, o mesmo aponta para a classe na qual o pacote deve ser enviado.

Além das definições mencionadas acima, a ferramenta TC utiliza a idéia de *scheduling* (escalonamento), *shaping* e *policing*. Por fim, são apresentados os termos *work-conserving* e *non-work-conserving*. Uma disciplina de enfileiramento *work-conserving* sempre entrega um pacote caso ele esteja disponível. Em outras palavras, nunca atrasa a entrega do pacote se o adaptador de rede estiver pronto para enviá-lo. Já a disciplina de enfileiramento *non-work-conserving* segura o pacote por um certo tempo com o objetivo de limitar a largura de banda, como por exemplo, o mecanismo TBF (*Token Bucket Filter*) [Clark et al., 1992], já explicado anteriormente.

Na Figura 27, é mostrado o fluxo de encaminhamento de pacotes onde podemos visualizar algumas das definições explicadas.

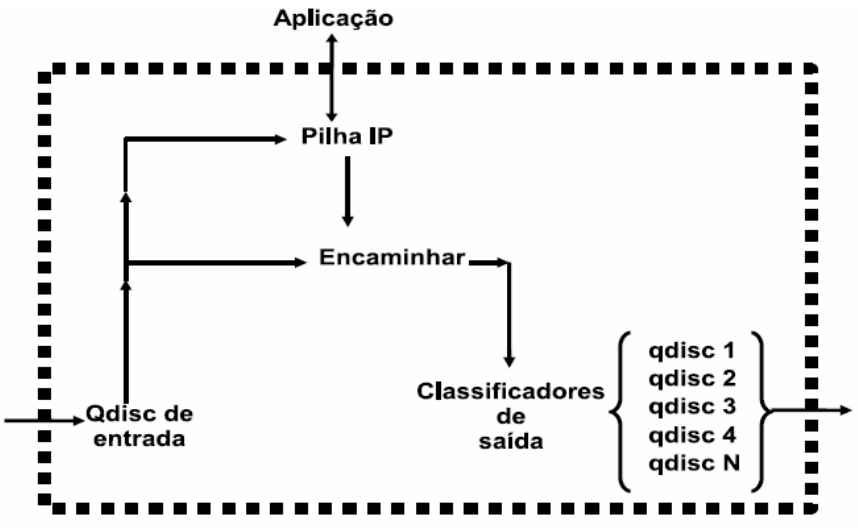

Figura 27: Disciplinas de filas nos dispositivos.

O bloco maior representa o *kernel*. A seta mais à esquerda representa o tráfego da rede entrando no dispositivo. O nó é então "alimentado" pelas qdisc *ingress*, as quais aplicam filtros aos pacotes, e decidem se irão ou não descartá-los (através da definição de políticas – *policies*). Caso o pacote seja liberado para continuar, o mesmo será destinado para uma aplicação local e é enviado para a pilha IP. O pacote pode também ser encaminhado sem que entre em uma aplicação, caso este que será destinado para a saída diretamente para um nó. Antes de serem enviados, os pacotes passarão pelas filas de saída, que poderão conter diferentes disciplinas. Da mesma forma que na fila de entrada, os filtros também deverão ser consultados para saber em qual classe e em qual qdisc cada pacote deverá ser enfileirado.

5.2. IPERF

Como emulador de fluxos foi utilizado o IPERF [IPERF, 2007]. O IPERF é um software de análise de desempenho de banda e cálculo de perda de datagramas na rede. Tal programa é mantido pela Universidade de Illinois sob licença GPL (*General Public License*).

O software IPERF baseia-se na estrutura cliente/servidor para que sejam feitas medições ativas nos cenários desejados. Além disso, esse software pode ser

usado com um gerador de carga na rede, quando não há preocupação com o perfil do tráfego que está sendo gerado. Parâmetros como *jitter* e a perda também são possíveis de serem medidos. O IPERF é capaz de usar tanto o protocolo UDP quanto TCP, e pode lidar com múltiplas conexões simultâneas.

Quando utilizado com protocolo UDP, necessita apenas da especificação do parâmetro –u para que os pacotes sejam marcados no cabeçalho IP como protocol = 0x11. Para a modificação do campo ToS (*Type of Service*) do datagrama IP [RFC 791, 1981], ilustrado na Figura 28, qualquer valor é conseguido quando o utilizamos. No caso do protocolo TCP, somente os valores especificados na RFC 1349 [RFC 1349, 1992] (0x02, 0x04, 0x08 e 0x10).

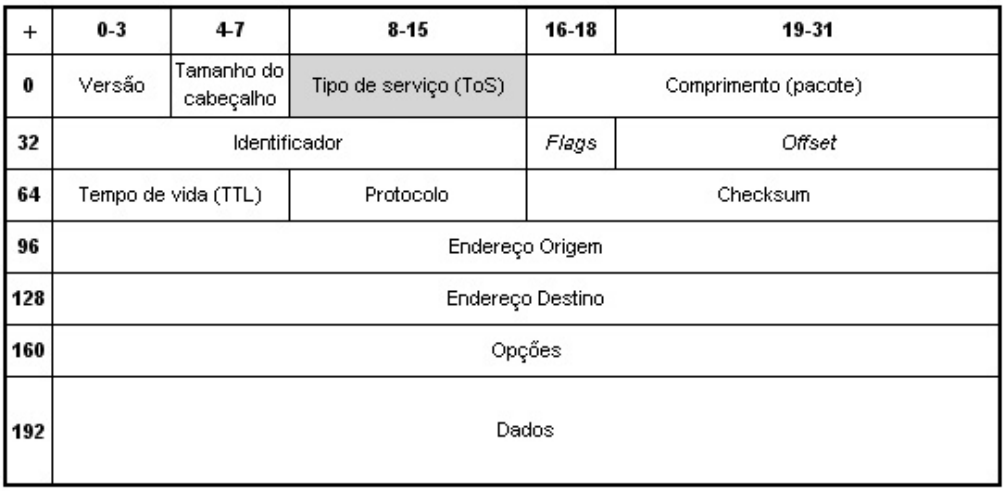

Figura 28: Datagrama IP.

Em [IPERF, 2007], pode-se encontrar diversos parâmetros para a utilização do software. Na Tabela 6, segue a relação de alguns parâmetros que foram utilizados nos testes para alterações nos cabeçalhos dos pacotes.

Tabela 6: Parâmetros básicos da ferramenta IPERF.

Opção de parâmetros de comandos	Variável de opções (outras opções de entrada dos parâmteros)	Descrição
Opções para modo *Client* e *Server*		
-i, --interval #	$IPERF_INTERVAL	Define o intervalo de tempo em segundos entre a largura de banda, *jitter* e perdas. *Default* é zero.
-u, --udp	$IPERF_UDP	Para definir o uso do protocolo UDP. Quando esta opção é omitida, o protocolo TCP é gerado.
Opções específicas para o modo *Server*		
-s, --server	$IPERF_SERVER	Para executar o IPERF em modo *server*.

Opções específicas para o modo *Client*		
-b, --bandwidth *#[KM]*	$IPERF_BANDWIDTH	Largura de banda UDP em bps. Isto implica no uso da opção -u. *Default* é 1 Mbps.
-c, --client *host*	$IPERF_CLIENT	Para executar o IPERF em modo *client*, conectando a um *server* que esteja rodando o aplicativo IPERF.
-t, --time *#*	$IPERF_TIME	Especifica o tempo de transmissão em segundos. *Default* é 10 segundos.
-S, --tos *#*	$IPERF_TOS	Definir o *Type of Service* (ToS) dos pacotes a serem enviados. Podem ser especificados valores em Hexadecimal (Hex), com o prefixo '0x' prefix, especificados valores em Octal (Oct), com o prefixo '0', ou especificado em decimal. Por exemplo, '0x10' hex = '020' octal = '16' decimal.

5.3. CALLGEN

Além do emulador de fluxos IPERF, foi utilizado em testes finais o software CallGen [DCC-UFAM, 2009] para verificação da qualidade dos fluxos transmitidos através de valores de MOS (*Mean Opinion Score*), descrito nas Recomendações ITU-T (ITU P.800 [ITU P.800, 1996] e P.830 [ITU P.830, 1996]), quando aplicamos QoS nos fluxos de voz. O CallGen323 é um gerador de chamadas VoIP utilizando o protocolo de sinalização H.323 [ITU-T Recommendation H.323, 1998]. Originalmente, ele apenas gera fluxo de sinalização, mas não pode gerar por si só o fluxo de mídia. Para tanto, é necessário especificar um arquivo no formato .wav, o qual fornecerá a mídia a ser empacotada pelo protocolo RTP. Não possui interface gráfica, é executado em linha de comando, o que facilita a criação de scripts para automatização de experimentos.

O CallGen323 oferece as seguintes funcionalidades:

- ✓ Gerar um número configurável de chamadas simultâneas;
- ✓ Executar um arquivo de voz previamente gravado;
- ✓ Ajustar a duração das chamadas, seja em um valor fixo, seja em um valor situado em intervalo especificado;
- ✓ Aguardar durante um tempo ajustável a realização de uma nova chamada;
- ✓ Armazenar um arquivo de trace contendo informações de qualidade sobre a chamada;

- ✓ configurar o número de quadros de codec por pacote RTP (duração do pacote de voz);

Em [DCC-UFAM, 2009], pode-se encontrar diversos parâmetros para a utilização do software bem como o procedimento completo de instalação. Na Tabela 7, segue a relação de alguns parâmetros que foram utilizados nos testes para o desejado funcionamento da ferramenta.

Tabela 7: Parâmetros básicos da ferramenta CALLGEN.

Opção	Ação correspondente
-l	aguarda recebimento de chamadas
-m <num>	executa <num> chamadas simultâneas
-r <num>	repete a chamada <num> vezes
-t	gera relatório de informações sobre as chamadas geradas
-o <arquivo>	armazena o relatório de informações no arquivo especificado
-O <arquivo>	especifica um arquivo .wav, codificado em PCM de 16 bits, como conteúdo da chamada
-P <codec>	seleciona o codec de preferência
--g711frames <num>	especifica o número de quadros por pacote G.711
-tmincall <tempo>	duração mínima da chamada (em segundos)
-tmaxcall <tempo>	duração máxima da chamada (em segundos)
-tminwait <tempo>	intervalo mínimo entre chamadas (em segundos)
-tmaxwait <tempo>	intervalo máximo entre chamadas (em segundos)

Após as rotinas executadas, os relatórios foram coletados e posteriormente manipulados com o intuito de aplicar o modelo E para a obtenção do fator R das chamadas e posterior conversão para o valor referente de MOS. Mais informações sobre o modelo E, o fator R e o cálculo de MOS estão disponíveis em [DCC-UFAM, 2009].

5.4. TESTES COM IPERF

Para assegurar a eficácia da ferramenta TC (*Traffic Control*) e verificar se a mesma atendia plenamente aos requisitos para que fosse utilizada na implementação da proposta desta dissertação, alguns testes básicos foram feitos.

Com a ferramenta IPERF, tentou-se alterar pacotes TCP e pôde-se verificar que os mesmos foram modificados. Conforme citado anteriormente, somente os valores especificados na RFC 1349 puderam ser utilizados. Como o protocolo TCP não foi objeto desta dissertação, não serão demonstrados os campos alterados.

Os mesmos testes descritos no parágrafo anterior foram feitos para o envio de pacotes UDP. Neste caso, pudemos observar que qualquer valor de ToS informado na ferramenta IPERF alcançava o destino. Com isso, baseado na tabela de classificação de fluxos e destinos, conseguiremos modificar qualquer valor de ToS para que os resultados sejam atingidos. Como pode-se observar nas linhas a seguir, para enviarmos pacotes UDP, as únicas modificações a serem feitas em relação ao envio de pacotes com o protocolo TCP são as inserções do parâmetro –u no comando executado do lado cliente e no comando executado do lado servidor.

No cliente: iperf –c 200.20.10.76 –u –i1 –t30 –S0x01
No servidor: iperf –s –u –i1

Na Figura 29, podemos verificar que, o cabeçalho foi marcado com valor 0x01, conforme especificado no comando do lado cliente. Os bits dos campos DSCP, ECT (*Explicit Congestion Notification - Capable Transport*) e ECN (*Explicit Congestion Notification - Congestion Experienced*) foram modificados de acordo com o esperado. A concatenação destes três campos forma o *Differentiated services field*, que na verdade é uma redefinição do Type of Service (RFC 2474) [Nichols et al., 1998]. O campo ECT, também conhecido como ECN-CT é usado para indicar aos roteadores que os *host* finais estão trabalhando com notificação de congestionamento explícita ao trocar dados com o protocolo TCP. Assim, sempre que a origem e o destino decidirem trabalhar com controle explícito de congestionamento, eles deves definir este parâmetro no cabeçalho IP. Já no do campo ECN, também conhecido como ECN-CE, quando a rede está congestionada, datagramas são descartados pelos roteadores. Se algum desses datagramas possuir o parâmetro ECT, o roteador envia um datagrama para o receptor (destino do datagrama descartado) com este *flag* ECN-CE. Com isso, pode-se mostrar ao receptor que houve congestionamento e que um datagrama foi descartado,

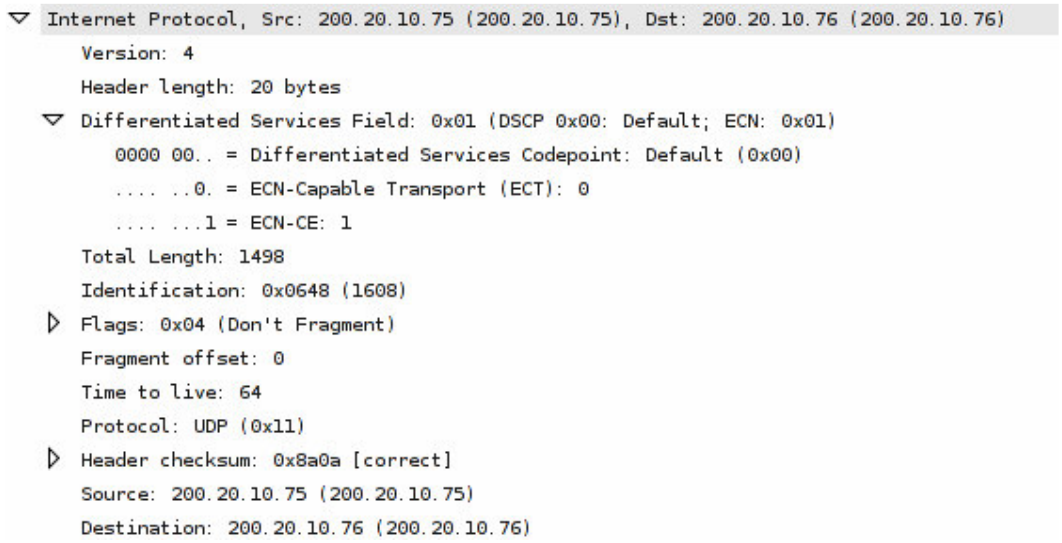

Figura 29: Campo ToS modificado no cliente com valor 0x01.

Na Figura 30, temos o pacote enviado acima, ao chegar no lado servidor, cujo endereço IP é o 200.20.10.76. Os valores do campo *Differentiated services field* permaneceram inalterados.

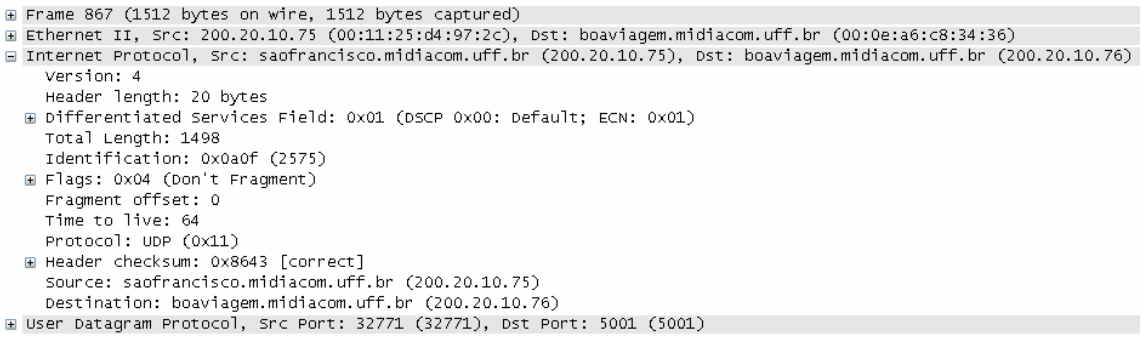

Figura 30: Campo ToS modificado no servidor com valor 0x01.

Com o campo ToS podendo ser alterado para qualquer valor de entrada, apenas no protocolo UDP, foram testados todos os valores possíveis para verificar o limite de alteração dos bits *Differentiated Services Code Point*. Na Figura 31, pode-se observar que a combinação 10000000 (0x80) pode ser validada no simulador de tráfego quando enviado o pacote a partir do cliente. Tal combinação refere-se à classe vídeo, conforme foi mostrado anteriormente.

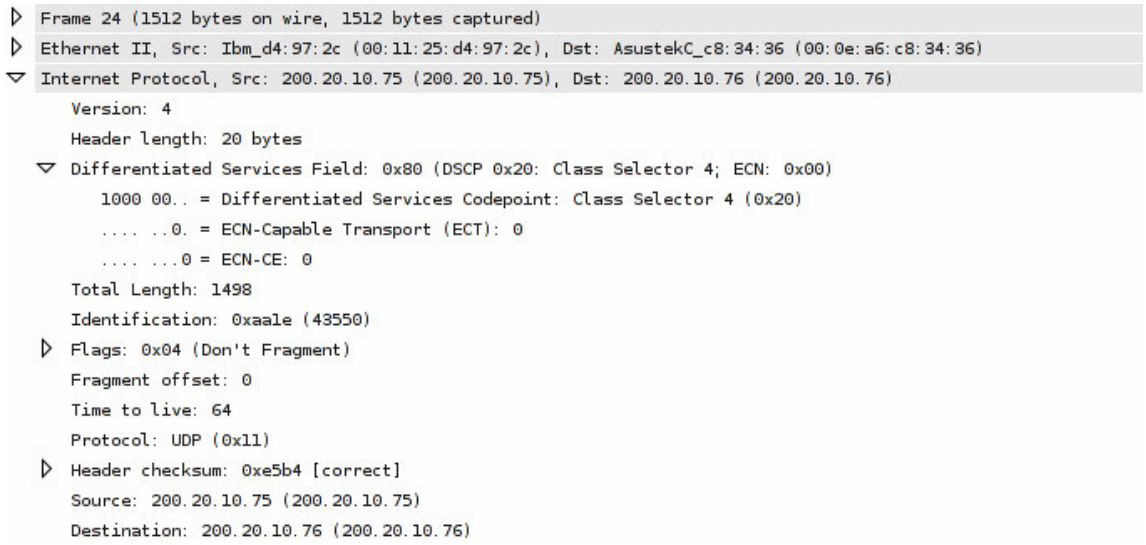

Figura 31: Campo ToS modificado no cliente com valor 0x80.

Na seqüência, a Figura 32 mostra que os bits alterados pelo cliente não sofreram modificações ao chegar no servidor.

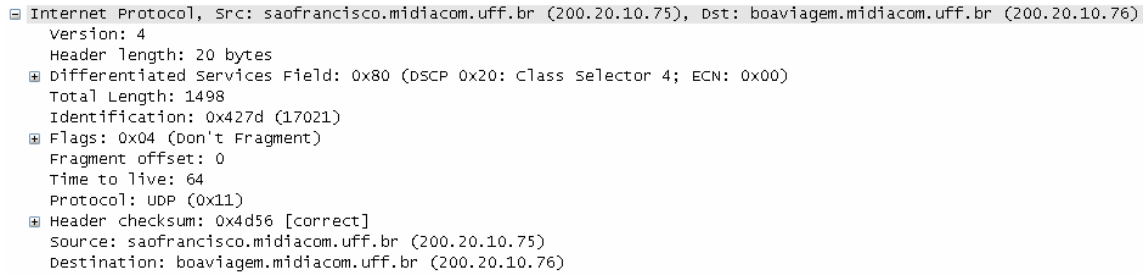

Figura 32: Campo ToS modificado no servidor com valor 0x80.

5.5. TESTES COM A FERRAMENTA TC

Com os parâmetros do IPERF ajustados e validados, foi montado um script para que pudesse ser assegurada a confiabilidade da ferramenta TC (*Traffic Control*). O teste, conforme pode ser visto na Figura 33, baseou-se em estabelecer dois fluxos, um TCP e outro UDP, através de um roteador WRT54G da rede *mesh*. Com este teste, a validação de priorização de pacotes e alguns ajustes no mecanismo de empréstimo puderam ser feitos e levados para os estudos que são objeto desta dissertação.

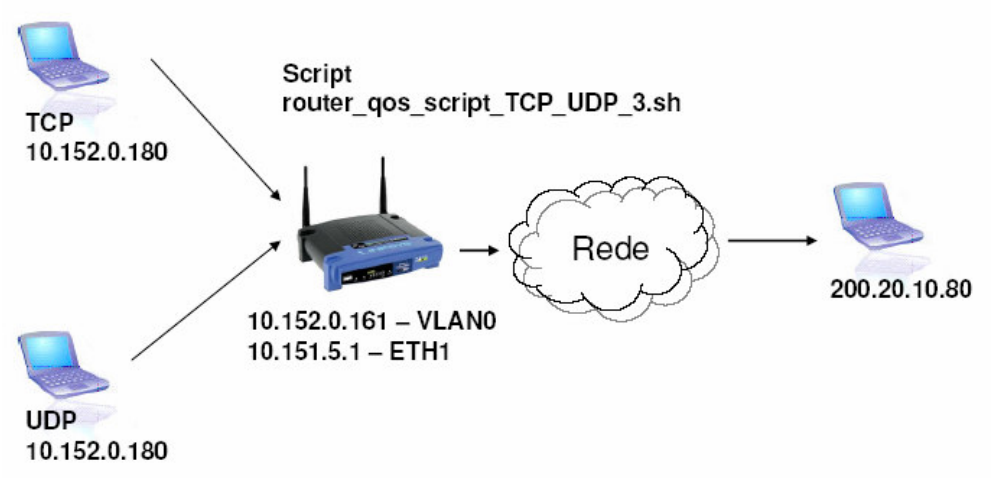

Figura 33: Validação TCP x UDP.

O script utilizado para o teste pode ser observado no Anexo 9.1.

De posse dos resultados dos testes, pôde-se observar os resultados exibidos na Figura 34.

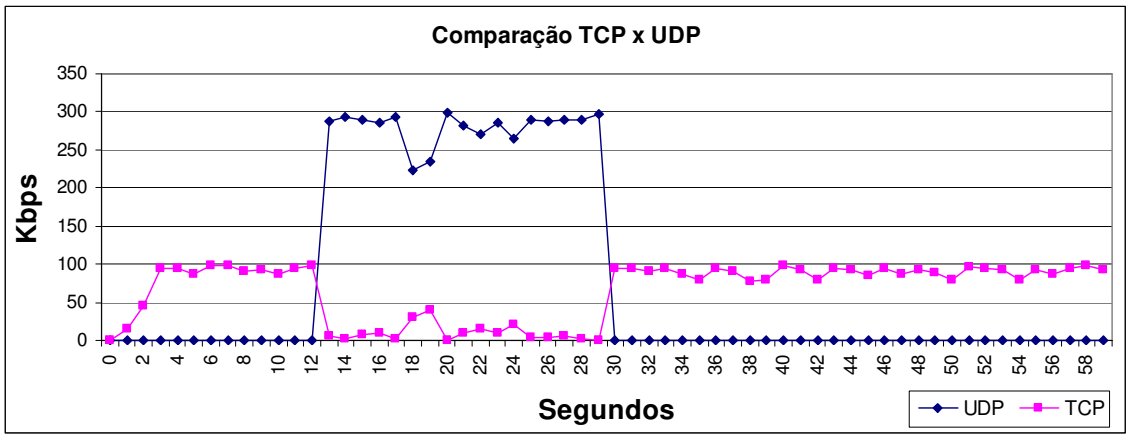

Figura 34: Gráfico comparativo TCP x UDP – prioridades diferentes.

Analisando a Figura 34, podemos observar que, durante 60 segundos, que foi o período estipulado para o teste, tivemos o comportamento exatamente como desejado, ou seja, os pacotes UDP com total prioridade em relação aos pacotes TCP. Uma vez a transmissão dos pacotes UDP cessada, há a retomada do fluxo TCP. Tal comportamento se justifica por parâmetros modificados no script, ou seja, o protocolo UDP com prioridade superior ao protocolo TCP.

Quanto à largura de banda, foi estabelecido que o protocolo UDP teria 300 kbps e o protocolo TCP teria 100 kbps, para uma largura total de 300 kbps.

O valor máximo que pode ser utilizado quando o outro protocolo não estiver sendo utilizado foi definido em zero (0), o que é caracterizado como empréstimo para o mecanismo de escalonamento HTB. Pode-se observar ainda que, em alguns momentos que tivemos a queda na transmissão dos pacotes UDP, os pacotes TCP foram enviados para o servidor, demonstrando assim a ocupação da banda ociosa quando temos alguma folga na transmissão.

No teste seguinte, foram alterados os parâmetros de prioridade. Foram colocadas prioridades iguais nos dois protocolos, ou seja, ambos os protocolos ficaram com prioridade zero. Além deste, foram alterados os valores máximos (valores de teto = *CEIL*) dos dois protocolos, para o valor total da conexão. Ou seja, no momento que houver disponibilidade de banda na conexão, qualquer um dos protocolos poderá atingir o valor estipulado no parâmetro *BANDWIDTH.* Os resultados podem ser observados na Figura 35.

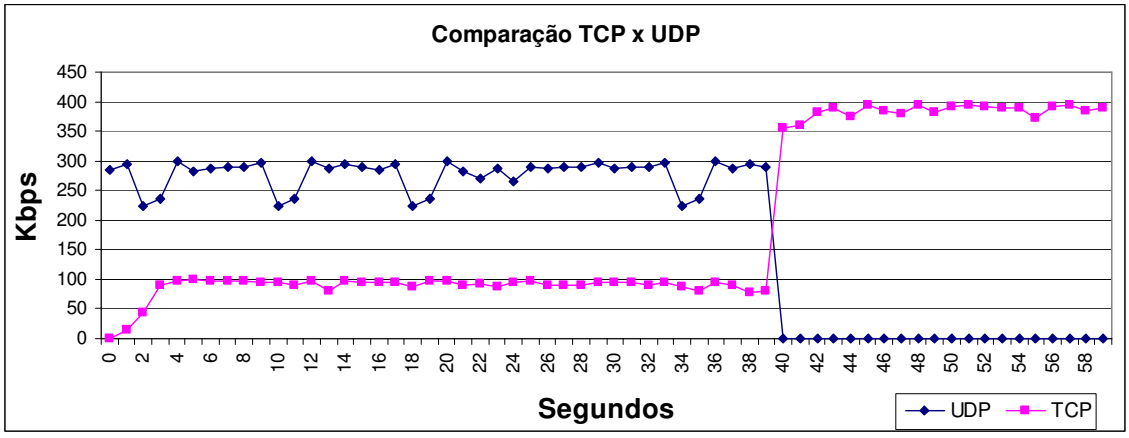

Figura 35: Gráfico comparativo TCP x UDP – prioridades iguais.

Na análise dos resultados, podemos observar o mecanismo de empréstimo do HTB. Cada protocolo comportou-se conforme havia sido programado. Os fluxos UDP foram enviados por 40 segundos. Já os fluxos TCP, foram enviados por 60 segundos. No momento do término do envio dos pacotes UDP, pode-se observar o imediato aumento na largura de banda do outro protocolo, uma vez que o valor máximo desta vez foi aumentado para 400 kbps.

5.6. PRIORIZAÇÃO DE CLASSES

Uma vez validados os resultados e os scripts do *Traffic* Control para uso com HTB, foram executados fluxos marcados de acordo com a Tabela de Classificação de Fluxos e Destinos. A Tabela 8 mostra a modificação de bits em cada pacote (colunas a a h), de acordo com as classes e subclasses. Além disso, traz os valores em Hexadecimal (coluna Hex) e Decimal (coluna Dec), os quais foram utilizados na marcação dos pacotes para o envio com o aplicativo IPERF.

Tabela 8: Classificação dos fluxos e destinos com conversões

a	b	c	d	e	f	g	h		Dec	Hex	Tipo do fluxo	Tipo de usuário
0	0	0	0	0	0	0	0	00000000	0	0	Sinalização	
0	1	0	0	0	0	0	0	01000000	64	40	Dados	
1	0	0	0	0	0	0	0	10000000	128	80	Vídeo	
1	1	1	1	1	1	1	1	11111111	255	FF	Especiais	Emerg
1	1	1	1	1	1	1	0	11111110	254	FE	Especiais	VIP
1	1	1	1	1	1	0	1	11111101	253	FD	Especiais	Normal
1	1	1	1	1	1	0	0	11111100	252	FC	Especiais	Pré-pago
1	1	1	1	1	0	1	1	11111011	251	FB	Locais, LD e Internacionais	Emerg
1	1	1	1	1	0	1	0	11111010	250	FA	Locais, LD e Internacionais	VIP
1	1	1	1	1	0	0	1	11111001	249	F9	Locais, LD e Internacionais	Normal
1	1	1	1	1	0	0	0	11111000	248	F8	Locais, LD e Internacionais	Pré-pago
1	1	1	1	0	1	1	1	11110111	247	F7	Cobrar local e LD	Emerg
1	1	1	1	0	1	1	0	11110110	246	F6	Cobrar local e LD	VIP
1	1	1	1	0	1	0	1	11110101	245	F5	Cobrar local e LD	Normal
1	1	1	1	0	1	0	0	11110100	244	F4	Cobrar local e LD	Pré-pago
1	1	1	1	0	0	1	1	11110011	243	F3	Chamadas para destinos 0800	Emerg
1	1	1	1	0	0	1	0	11110010	242	F2	Chamadas para destinos 0800	VIP
1	1	1	1	0	0	0	1	11110001	241	F1	Chamadas para destinos 0800	Normal
1	1	1	1	0	0	0	0	11110000	240	F0	Chamadas para destinos 0800	Pré-pago
1	1	1	0	1	1	1	1	11101111	239	EF	Chamadas para destinos 0300	Emerg
1	1	1	0	1	1	1	0	11101110	238	EE	Chamadas para destinos 0300	VIP
1	1	1	0	1	1	0	1	11101101	237	ED	Chamadas para destinos 0300	Normal
1	1	1	0	1	1	0	0	11101100	236	EC	Chamadas para destinos 0300	Pré-pago
1	1	1	0	1	0	1	1	11101011	235	EB	Chamadas para destinos 0500	Emerg
1	1	1	0	1	0	1	0	11101010	234	EA	Chamadas para destinos 0500	VIP
1	1	1	0	1	0	0	1	11101001	233	E9	Chamadas para destinos 0500	Normal
1	1	1	0	1	0	0	0	11101000	232	E8	Chamadas para destinos 0500	Pré-pago
1	1	1	0	0	1	1	1	11100111	231	E7	Outros tipos de chamadas	Emerg
1	1	1	0	0	1	1	0	11100110	230	E6	Outros tipos de chamadas	VIP
1	1	1	0	0	1	0	1	11100101	229	E5	Outros tipos de chamadas	Normal
1	1	1	0	0	1	0	0	11100100	228	E4	Outros tipos de chamadas	Pré-pago
1	1	1	0	0	0	1	1	11100011	227	E3	Reservado	Emerg
1	1	1	0	0	0	1	0	11100010	226	E2	Reservado	VIP
1	1	1	0	0	0	0	1	11100001	225	E1	Reservado	Normal
1	1	1	0	0	0	0	0	11100000	224	E0	Reservado	Pré-pago

Para uma distribuição de largura de banda, foram definidos percentuais para cada tipo de classe e, dentro de cada subclasse, divididos percentualmente de

acordo com a importância da chamada. A Figura 36 mostra a divisão percentual de banda.

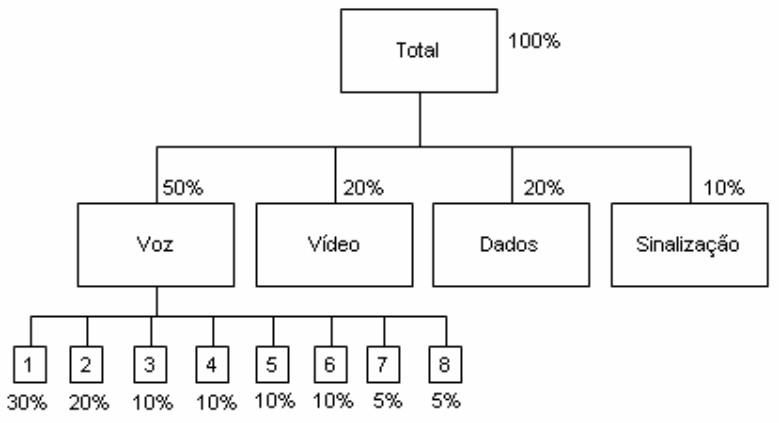

Figura 36: Divisão percentual de largura de banda.

Os valores encontrados a seguir referem-se a pacotes marcados com os valores 0xFF, 0xF7, 0x40 e 0xEF, que são assinantes especiais com destinos a numerações de emergência, usuários especiais com destinos a chamadas a Cobrar local e LD, pacotes de dados e usuários especiais com destinos a chamadas 0300, respectivamente. Na Tabela 9, podemos ver os tempos que foram utilizados por cada usuário na rede e as portas para diferenciar os fluxos na ferramenta IPERF. Estas portas foram utilizadas apenas para que pudessem ser identificados no lado servidor, não tendo, portanto, nenhuma influência na obtenção e análise dos resultados.

Tabela 9: Tipo do pacote e tempo de envio

Tipo do pacote	Marcação em Hex	Tempo (seg)	Porta
Assinantes especiais com destinos a numerações de emergência	0xFF	15 seg	5001
Usuários especiais com destinos a chamadas a Cobrar local e LD	0XF7	60 seg	5002
Pacotes de dados	0x40	120 seg	5003
Usuários especiais com destinos a chamadas 0300	0xEF	30 seg	5004

A Figura 37 ilustra a priorização dos pacotes. Para a classe de voz, vídeo, dados e outros foram estabelecidos os valores máximos (*CEIL*) de 90 kbps. Quanto à largura de banda que cada classe tem direito, foram divididas da seguinte forma:

- ✓ Classe Voz – 50 kbps;

- ✓ Classe Vídeo – 20 kbps;
- ✓ Classe Dados – 15 kbps;
- ✓ Classe *Others* – 5 kbps.

Ainda, dentro da “Classe Voz”, foram especificadas a largura de banda das subclasses, conforme itens a seguir:

- ✓ Classe Voz_LEVEL_0 – 16 kbps;
- ✓ Classe Voz_LEVEL_1 – 10 kbps;
- ✓ Classe Voz_LEVEL_2 – 5 kbps;
- ✓ Classe Voz_LEVEL_3 – 5 kbps;
- ✓ Classe Voz_LEVEL_4 – 5 kbps;
- ✓ Classe Voz_LEVEL_5 – 5 kbps;
- ✓ Classe Voz_LEVEL_6 – 5 kbps;
- ✓ Classe Voz_LEVEL_7 – 2 kbps;

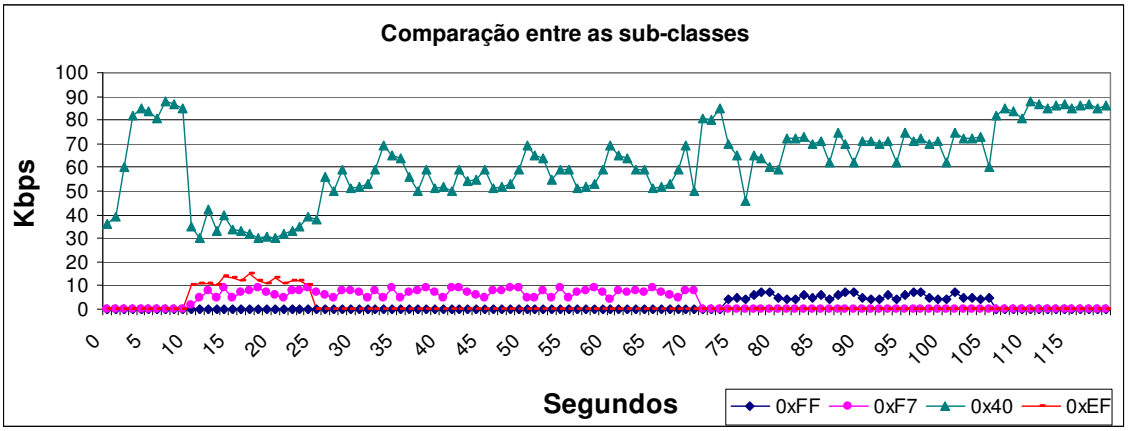

Figura 37: Gráfico com priorização de pacotes.

A largura de banda máxima utilizada para este teste foi de 90 kbps, com um empréstimo de 90kbps para cada classe, caso haja largura de banda disponível. As subclasses de voz foram caracterizadas como VOZ_LEVEL_0_BW, para a marcação 0xFF, DADOS_BW, para a marcação 0x40, VOZ_LEVEL_5_BW, para a marcação 0XF7 e VOZ_LEVEL_3_BW, para a marcação 0xEF.

Foram enviados pacotes de dados durante os 120 segundos. Nos períodos que havia somente este tipo de pacote trafegando na rede, pode-se observar que a

totalidade de banda disponível foi utilizada, uma vez que foi configurado o valor de CEIL para 90 kbps. No momento de descarga de pacotes de outros fluxos, o fluxo de dados foi diminuído, respeitando as distribuições estabelecidas. Todos os pacotes puderam então, ser distribuídos de acordo com o que lhes foi estabelecido, sem que outros fluxos concorrentes os interrompesse ou degradasse a qualidade do serviço oferecido.

Outro teste que foi feito consiste na verificação da influência de diferentes fluxos em uma rede sem QoS. Foram enviados fluxos, conforme topologia da Figura 38. Os testes de transmissão simularam chamadas de voz prioritárias, como, por exemplo, para a polícia (número 190), compartilhando a mesma infraestrutura de rede com chamadas de voz menos prioritárias, como, por exemplo, as realizadas por usuários do serviço de voz de um determinado *reality show* (número iniciado por 0800).

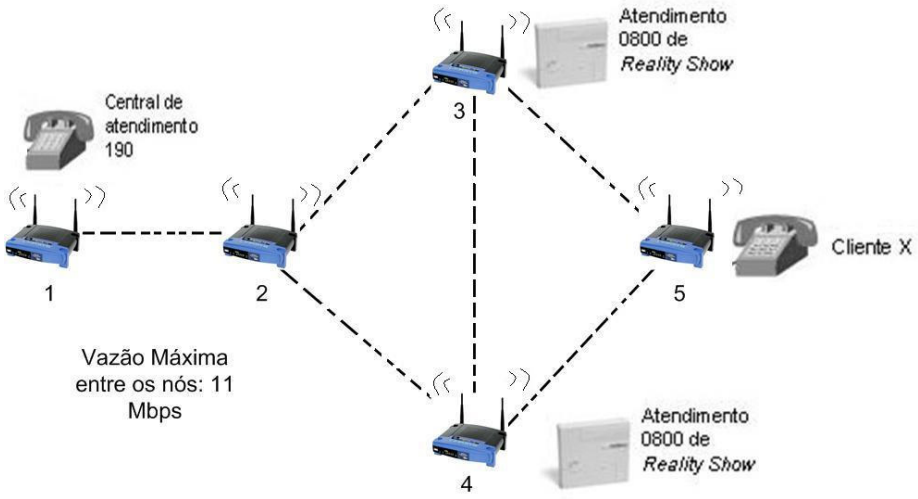

Figura 38: Topologia de testes na rede em malha sem fio.

Foram criados fluxos entre os nós 3-5, 4-5 e 1-5. Os fluxos entre os nós 3-5 e 4-5 simulam chamadas não prioritárias. Tais fluxos foram dimensionados para usar toda a banda disponível nos enlaces sem fio. Neste cenário, qualquer outro serviço, seja de voz ou de dados poderia ficar comprometido se não houvesse nenhum tipo de diferenciação de serviços. O fluxo entre os nós 1-5 simula uma chamada de emergência, que neste caso necessitam obrigatoriamente trafegar por um dos enlaces congestionados por chamadas não prioritárias. Os resultados dos testes na rede sem mecanismos de QoS são exibidos na Figura 39.

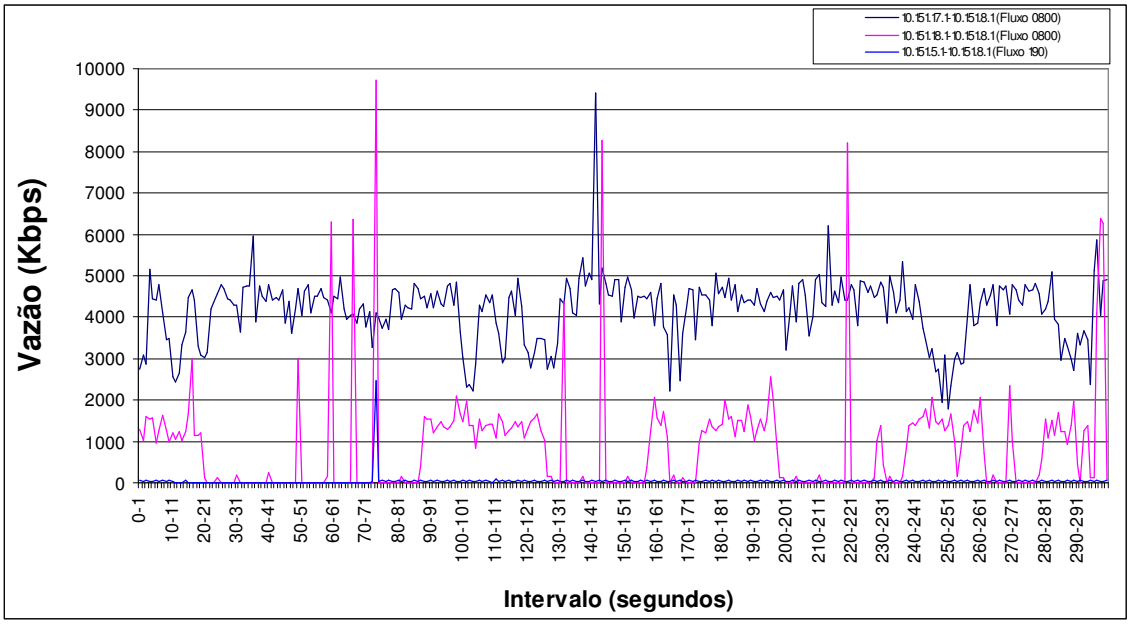

Figura 39: Influência dos fluxos 0800 no fluxo 190 (rede sem QoS).

A Figura 39 mostra a vazão dos diferentes tipos de fluxo transmitidos. Os fluxos 1 e 2, que simulam ligações 0800, ocupam a banda disponível nos enlaces sem fio do nó 5 (destino dos três fluxos) e limitam a vazão do fluxo 1, que simula uma chamada prioritária. Note que, independente do nó originador, como os três fluxos são destinados ao mesmo nó 5, ocorre o compartilhamento dos enlaces sem fio 3-5 e 4-5 pelas diferentes chamadas. Como os fluxos 1 e 2 não prioritários (0800) ocupam a capacidade máxima teórica dos enlaces sem fio IEEE 802.11b (11 Mbps), o fluxo 3, referente a uma chamada de emergência 190, não consegue ser atendido adequadamente em uma infraestrutura de serviços de voz que não contemple a importância da chamada. Tal fato fica evidente no gráfico exibido na Figura 39 entre o intervalo de 15 a 72 segundos, onde o fluxo 3 sofre uma perda de pacotes que compromete a qualidade do serviço. Desta forma, a possibilidade de tratamento diferenciado para fluxos de voz de acordo com a importância da chamada é fundamental para garantir a qualidade do serviço.

Na Figura 40 são exibidos resultados de vazão de outro teste realizado, com descrição similar ao teste anterior, mas que usa a implementação da proposta de classificação dos fluxos de voz e escalonamento baseado em HTB com prioridade para chamadas de emergência perante as outras classes de serviço. Com a

diferenciação de serviços implementada, fica evidente a melhora na vazão do fluxo 3 (chamada de emergência para 190), onde não houve nenhuma perda de pacotes em todos os experimentos realizados, mesmo com enlaces congestionados por outros fluxos menos prioritários (chamadas para números 0800). Através destes resultados, fica comprovada a eficiência do mecanismo de diferenciação de serviços para chamadas de voz baseada na importância da chamada.

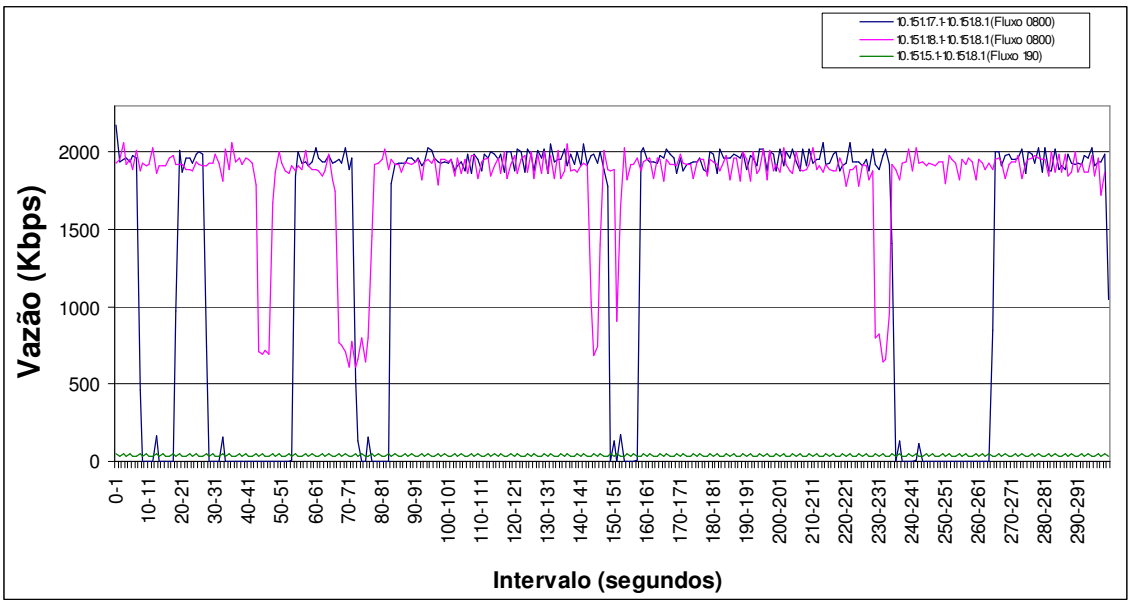

Figura 40: Influência dos fluxos 0800 no fluxo 190 (rede com QoS).

5.7. TESTES COM A FERRAMENTA CALLGEN

Os testes com o IPERF e o TC mostraram que as priorizações de pacotes de voz, dados e vídeo foram feitas de acordo com as análises e programações executadas. Para uma melhor análise da solução, utilizou-se a ferramenta de emulação de chamadas CallGen, com o CODEC G.711 servindo de suporte aos testes, para que pudéssemos verificar o valor de MOS e medir a qualidade dos fluxos de voz transmitidos. Na mesma topologia apresentada na Figura 38, utilizamos o script apresentado no Anexo 9.2, porém com priorizações de pacotes UDP e TCP incluídas no código. Para o início dos testes, foram gerados arquivos de voz sintética distintos (arquivos em formato .wav) para o emissor e receptor da chamada. Os valores destes arquivos variam conforme a metodologia de

caracterização adotada para produção da conversação artificial [DCC-UFAM, 2009], com característica de conversação e silêncio. Colocou-se um laptop no ponto 2 e outro laptop no ponto 5 (vide Figura 38) e foram gerados fluxos no sentido de 2 para 5. Foram criados fluxos UDP (gerados com a ferramenta CallGen) simulando duas conversações para destinos de emergência e fluxos TCP/UDP (gerados com a ferramenta IPERF). Após os arquivos de áudio gerados, foram inseridos os arquivos de classificação de pacotes nos roteadores do caminho. Os fluxos TCP/UDP gerados pela ferramenta IPERF possuíam o campo ToS, com valores diferentes dos apresentados pelos fluxos gerados pela ferramenta CallGen. Foram geradas 10 rotinas, uma com QoS na rede e outra sem, para que pudessem ser identificados ganhos de qualidade de voz com a classificação de pacotes.

Os testes sem QoS nos roteadores do caminho, utilizando as ferramentas acima descritas para a geração de fluxos TCP e UDP, forneceram o gráfico mostrado na Figura 41. Fluxos concorrentes foram gerados e enviados para a rede para que pudesse ser observado o comportamento dos fluxos de áudio gerados pelo CallGen. Pudemos notar que, numa arquitetura sem priorização de fluxos baseados na importância da chamada, obtivemos valores de MOS médio variando entre 2,5 e 3,4. Os valores das barras mostram o valor MOS para cada fluxo de voz em cada sessão de testes e a curva apresenta o valor médio.

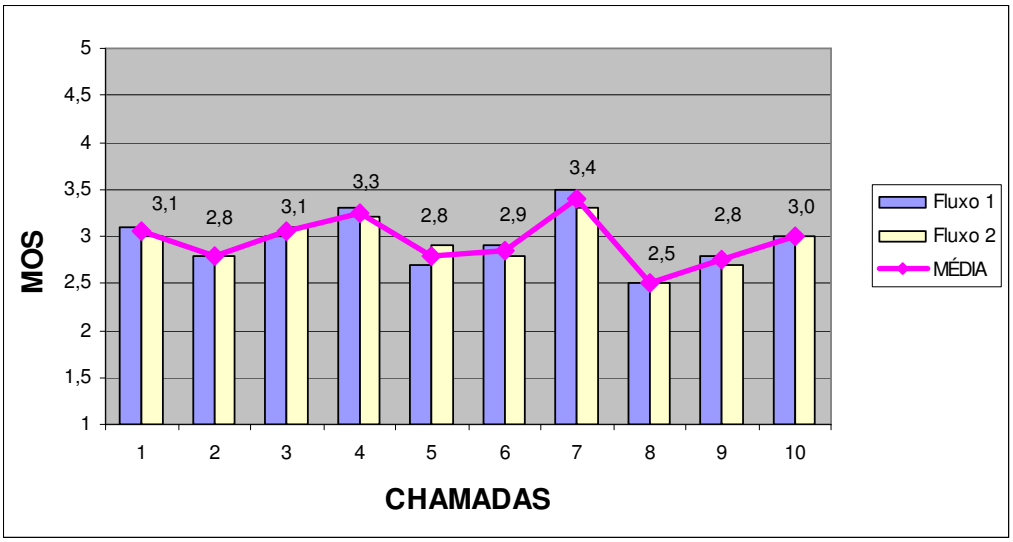

Figura 41: Valores de MOS obtidos com 10 chamadas de teste (rede sem QoS).

Após a execução dos testes na rede em malha sem QoS, ou seja, sem levar em consideração a priorização de fluxos baseados na importância da chamada, foram inseridos classificadores nos roteadores da rede, de modo a termos diferenciações de pacotes e com isso obter as priorizações já anteriormente descritas. Os mesmos comandos nas ferramentas IPERF e CallGen foram gerados e resultados melhores foram obtidos, conforme pode ser visto na Figura 42. A ilustração mostra valores médios de MOS entre 3,7 e 4,2. Com os mesmos princípios utilizados nos testes sem QoS, pudemos observar que obteve-se um ganho quando foi utilizado o script que prioriza os fluxos nos roteadores da rede.

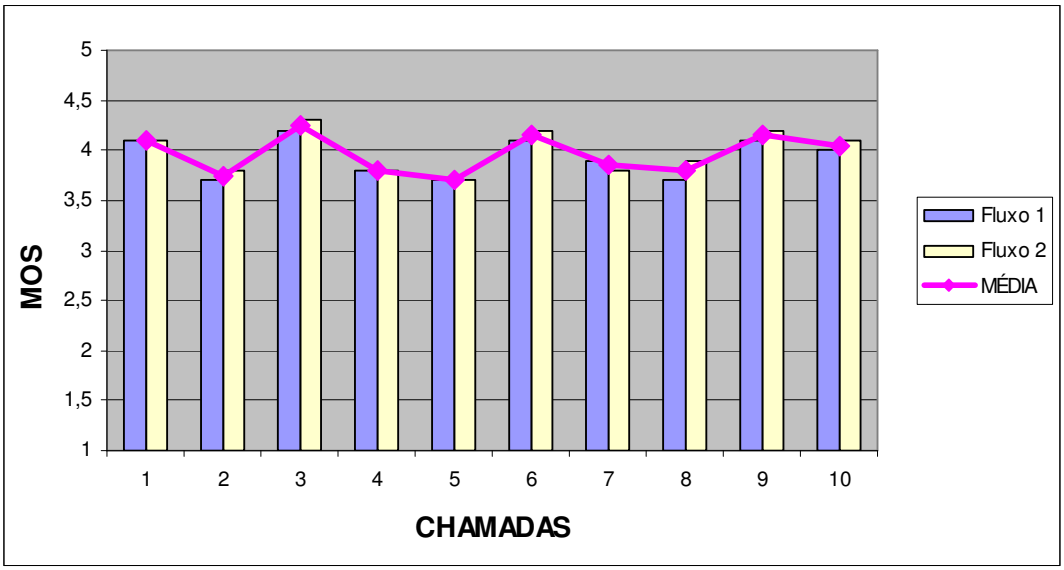

Figura 42: Valores de MOS obtidos com 10 chamadas de teste (rede com QoS).

Com os resultados descritos anteriormente, vemos que em uma rede sem QoS não conseguimos obter valores bons de MOS pois há concorrência de diferentes fluxos. Podemos concluir que a priorização de pacotes baseada na importância da chamada traz ganhos na qualidade de voz das ligações.

5.8. RESUMO DO CAPÍTULO

O Capítulo 5 mostrou a implementação da proposta de classificação de fluxos de voz baseada na importância da chamada na rede *mesh* da UFF. Abordou os conceitos da ferramenta conhecida como *Traffic Control* (TC), mostrando suas limitações e funcionamento. Além desta ferramenta, abordou as características das

ferramentas que funcionam como gerador de tráfego, conhecidas como IPERF e CALLGEN. Nestes itens, foram mostrados claramente os comandos e parâmetros utilizados nos testes. Após a introdução das ferramentas que serviram como base na execução dos testes, foram feitas simulações para a validação dos *scripts* e comandos. Uma vez validados todos os programas e entendidos os conceitos do funcionamento em conjunto do TC, CALLGEN e do IPERF, partiu-se para a execução dos testes em cima de priorização de classes e subclasses.

O capítulo a seguir comenta alguns trabalhos relacionados e compara-os a proposta apresentada nesta dissertação.

6. COMPARAÇÕES COM TRABALHOS RELACIONADOS

Em [EHAS, 2006], é mostrado o projeto “Enlace Hispano Americano de Salud”. Esta fundação é uma entidade sem fins lucrativos, cuja finalidade é a melhora dos sistemas públicos de assistência de saúde em zonas rurais dos países hispano-americanos e todos aqueles próximos que se encontram em dificuldade e/ou em fases de desenvolvimento, através do uso de novas tecnologias da informação e comunicação. Esta fundação foi constituída inicialmente pela Universidade Politécnica de Madrid e a ONG (Organização Não Governamental) Engenharia sem fronteiras. As áreas de atuação da entidade EHAS abrangem há alguns anos o Peru, Colômbia, Cuba e acaba de iniciar suas operações no Equador. Recentemente, foi instalada uma rede sem fio em Cuzco, que é uma cidade no Peru situada no sudeste do Vale de Huatanay ou Vale Sagrado dos Incas, na região dos Andes, com população de 300.000 habitantes. Esta rede provê conectividade a 12 nós, cujas localizações são os centros de assistência médica. Dentre os serviços oferecidos pela rede, há o monitoramento, via *Web Cam*, de pacientes, VoIP e dados que são trafegados. Ao invés de utilizarem *soft phones* modificados, está sendo estudada a modificação no próprio PABX dos centros médicos para que os pacotes sejam marcados. Uma vez que temos como ponto prioritário o serviço de voz, algumas priorizações são feitas. Entretanto, não há a diferenciação de pacotes marcados em função do número de destino, nem a estruturação de subclasses de acordo com a importância do usuário.

Em [RFC 3953, 2005], é apresentada uma idéia de mapeamento de dígitos E.164 [ITU, 2007], conhecida como ENUM (*Eletronic Number* ou *Telephone Number Mapping*). Essa referência apresenta um mapeamento de números telefônicos para nomes de domínio, usando uma arquitetura baseada no DNS (*Domain Name System*). Este protocolo facilita serviços como VoIP, e permite que elementos da rede encontrem serviços na Internet usando somente um número telefônico. O protocolo ENUM não altera o plano de numeração e não altera a numeração telefônica ou sua administração já existente. Além disso, não irá gastar fontes já escassas de numeração porque ele usa números existentes. Apesar de trabalhar com os dígitos discados, não oferece qualquer suporte a QoS como é apresentado neste trabalho. Prover serviços de emergência em redes VoIP é um dos fatores fundamentais para o sucesso absoluto e a integração na rede existente. Não

somente a lógica de funcionamento e os desafios de implementação, mas também a abertura de oportunidades de aumentar a infraestrutura do tratamento de chamadas de emergência. Em [Schulzrinne e Arabshian, 2002], é proposta uma arquitetura para tratar serviços de emergência em redes VoIP baseadas em SIP. Esta arquitetura oferece a flexibilidade de identificar as localizações dos chamadores, rotear as chamadas de emergência para os apropriados pontos de atendimento (PSAP - *Public Safety Answering Points*) e apresentar algumas informações para os operadores de atendimento das centrais. Entretanto, a idéia abordada no artigo em questão não oferece suporte a qualidade de serviço baseado em dígitos discados.

Em [Hilario, 2006], há o estudo de problemas intrínsecos relativos a redes *mesh*, como interferência na comunicação causada por ruídos, colisões de pacotes ou quebra de conectividade, gerada possivelmente por causa da mobilidade dos seus nós. Tais fatores devem ser levados em consideração no momento da especificação dos requisitos de QoS. Tendo em vista o exposto, é apresentada uma proposta e a implementação de um *framework* de suporte a QoS para redes *mesh*, e a sua avaliação em um ambiente real. O *framework* de suporte a QoS apresentado utiliza uma solução híbrida, ou seja, são utilizados mecanismos de reserva de recursos a medições de requisitos. O sistema de reserva de recursos trabalha com classes diferentes para cada tipo de tráfego e, é disponibilizada para cada classe, uma parte da capacidade nominal de largura de banda do roteador. Ao chegar no roteador, o pacote é encaminhado para a classe na qual se encaixa. A Figura 43 mostra esta divisão de recursos.

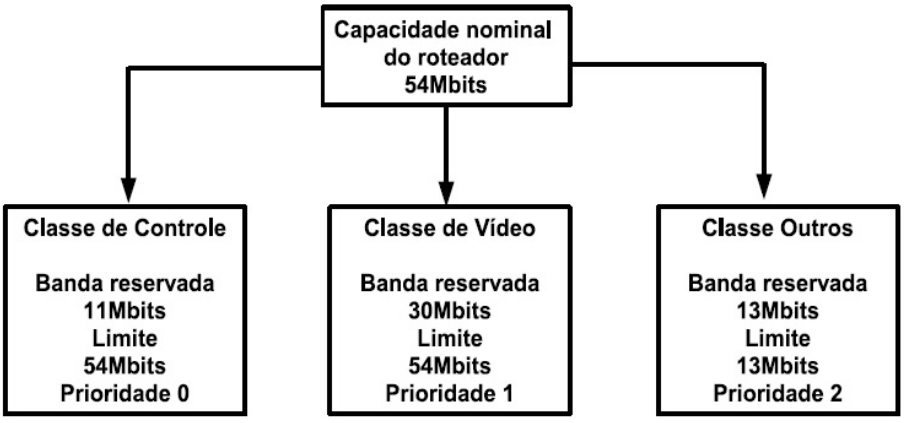

Figura 43: Estrutura de classes do mecanismo de reservas de recursos.

O trabalho utilizou a ferramenta TC (*Traffic Control*), disponível no *OpenWRT* para determinar a priorização do tráfego e o gerenciamento da largura de banda e também utilizou o mecanismo de escalonamento HTB (*Hierarchical Token Bucket*). Mecanismos similares foram utilizados na implementação da proposta desta dissertação.

Em [TERENA, 2004], é apresentado um estudo feito por um grupo, denominado TERENA (*Trans-European Research and Education Networking Association*), e tem como objetivo, estudar os benefícios econômicos de se adotar VoIP em diversos cenários, a fim de se obter redução de custo nas ligações. Aborda diferentes topologias de rede e roteamento de chamadas, bem como a integração deste mundo VoIP com a telefonia tradicional existente. Além disso, considerações de órgãos regulatórios europeus são mostrados. Apesar do estudo se mostrar bastante completo, não aborda a qualidade de serviço a ser oferecida.

7. CONCLUSÕES

Este trabalho apresentou uma solução para fornecer qualidade de serviço para chamadas de voz. O desenvolvimento e a implementação da proposta em redes em malha permitiram conhecer, na prática, o comportamento dos fluxos priorizados.

Quando uma arquitetura é proposta e apenas simulações são feitas, os resultados obtidos nem sempre correspondem à realidade. Entretanto, como a implementação dos scripts e testes foram executados em um ambiente real, pudemos observar todo o comportamento e resultados reais dos fluxos. Pôde-se verificar se os resultados estavam de acordo com o esperado. Os resultados da nossa proposta para QoS baseada na análise de dígitos reafirmam alguns conceitos e a eficácia dos mecanismos propostos no trabalho. A priorização de recursos num ambiente em que existe escassez de largura de banda, por exemplo, em redes sem fio, e em que todos os participantes da rede precisam disputá-los, torna-se importante principalmente para fluxos com maior necessidade de acesso à rede. Os testes na rede mesh interna da UFF comprovaram que a priorização desse tipo de tráfego resulta na melhor qualidade do serviço.

7.1. CONTRIBUIÇÕES

O trabalho apresentou uma diferenciação de serviços para chamadas VoIP, baseada nos números de destino e em classificações dos usuários originadores. A proposta baseou-se na marcação de pacotes feita do lado cliente para que possa haver um tratamento diferenciado nos roteadores da rede em caso de congestionamento. Com isso, pode-se obter uma melhor utilização dos recursos e equipamentos do sistema, encaminhando somente fluxos considerados prioritários. A proposta deve ser entendida como uma adequação da rede, onde tínhamos há pouco tempo, apenas fluxos de dados, sem que houvesse uma preocupação com a convergência de serviços. Esta rede estava baseada em serviços de melhor esforço, onde não havia preocupação com o pacote a ser entregue. Com a entrada de outros serviços, tais como voz e *streaming* de vídeo, por exemplo, toda a arquitetura da rede não pode ser mais ser concebida sem que haja uma priorização de tráfego.

Com a idéia descrita neste trabalho, espera-se que haja uma racionalização de serviços, de acordo com o esperado pela operadora / prestadora de serviços, para que os destinos prioritários sejam alcançados e, além disso, para que usuários com maior penetração na economia de uma empresa possam ser beneficiados pelo serviço que está sendo contratado.

7.2. TRABALHOS FUTUROS

Neste trabalho, foi utilizada a ferramenta de emulação de pacotes de rede conhecida como IPERF. Em trabalhos futuros, pretende-se aplicar a idéia de marcação de pacotes do cabeçalho IP, diretamente em *soft phones*. Alguns deles apresentam desenvolvimento em SIP e outros em H.323. A seguir, temos alguns que poderão ter o código fonte alterado para que sejam enviados pacotes já marcados.

- ✓ Bone Phone – O *soft phone* BonePhone [Bone Phone, 2008] é uma aplicação telefônica desenvolvida para rodar em plataformas Linux 2.4 usando X11. Foi desenvolvido pelo iptel.org, que é um portal de estudos de voz sobre IP e é mantido pelo instituto de pesquisa da Alemanha *Fraunhofer Fokus*. Baseia-se em uma arquitetura modular, desta forma, a modificação do código fonte torna-se mais fácil;

- ✓ K-Phone – O soft phone K-Phone [K-Phone, 2008] é uma aplicação baseada em SIP e roda em ambiente Linux. Originalmente, foi desenvolvido por Billy Bigs, no WirLab, até 2005. Este laboratório está localizado em Seinäjoki, na Finlândia e tem como principal foco, a pesquisa em aplicações de rede em tempo real. Atualmente, este projeto é conduzido por voluntários.

- ✓ Open H.323 – O projeto Open H.323 [OpenH323Project, 2007] desenvolveu uma aplicação aberta, baseada nas recomendações do ITU para o protocolo H.323. Pode ser alterado e utilizado, tanto para usos pessoais ou comerciais, sem custo.

Além da utilização de programas de originação de chamadas, conhecidos como *soft phones*, como trabalho futuro, espera-se desenvolver, nos roteadores de borda, a capacidade de análise de pacotes das camadas de aplicação para que seja feito o reconhecimento do que está sendo enviado. Tal análise só será feita quando o campo ToS não vier marcado pelo aplicativo das estações ou dispositivos móveis. Então, da mesma forma que temos os aplicativos dos usuários marcando os pacotes, sejam de voz, vídeo ou dados, teremos um trabalho dos elementos de borda. A idéia, neste caso, é não repassar esta função para os elementos de núcleo da rede para que não haja uma sobrecarga desnecessária no funcionamento dos mesmos.

Outro trabalho futuro para aprimorar a priorização dos pacotes baseada na importância das chamadas diz respeito à definição dinâmica das classes/subclasses pelo elemento controlador da rede de telefonia IP, ou seja, pelo *gatekeeper* ou *SIP Server*, durante a sinalização entre cliente e *gatekeeper* / *SIP Server*. Tal idéia implica em estender o protocolo de sinalização utilizado para que o elemento controlador possa informar ao cliente a prioridade que deve ser usada para marcar pacotes de um determinado fluxo.

8. REFERÊNCIAS BIBLIOGRÁFICAS

3GPP – 3rd Generation Partnership Project, http://www.3gpp.org/, acessado em Junho/2008.

Abelém, A. J. G., C. V. N. de Albuquerque, D. C. M. Saade, E. S. Aguiar, J. L. Duarte, J. E. M. da Fonseca e L. C. S. Magalhães. Redes mesh: Mobilidade, qualidade de serviço e comunicação em grupo. Minicurso do SBRC 2007, cap.2. 2007.

Almesberger, W., J. Salim e A. Kuznetsov. Differentiated Services on Linux. Proceedings of Globecom. 1999.

Almquist, P. Type of Service in the Internet Protocol Suite. RFC 1349. 1992.

Baker, F. Requirements for IP Version 4 Routers. RFC 1812. 1995.

Baugher, M., D. McGrew, M. Naslund, E. Carrara e K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711. 2004.

Black, D., S. Blake, M. Carlson, E. Davies, Z. Wang e W. Weiss. An Architecture for Differentiated Services. RFC 2475. 1998.

Bone Phone. http://old.iptel.org/products/bonephone/, acessado em Fevereiro/2008.

Brandl, M., D. Daskopoulos e E. Dobbelsteijn. IP Telephony Cookbook. Projeto TERENA. http://www.terena.org, acessado em Janeiro/2009.

Braden, B., D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski e L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309. 1998.

Braden, R., D. Clark e S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633. 1994.

Braden, R. Resource ReSerVation Protocol (RSVP)-Version 1. RFC 2205. 1995.

Brown, R. Calendar queues: A fast priority queue implementation for the simulation event set problem. Communications of the ACM, v.31, p.1220–1227. 1988.

Chen, C. W. X. e D. Xuan. Survey on QoS management of VoIP. International Conference of Computer Networks and Mobile Computing. ICCNMC 2003, p.20–23. 2003.

CISCO – User Guide for CiscoWorks QoS Policy Manager, http://www.cisco.com/, acessado Julho/2007.

Clark, D., S. Shenker e L. Zhang. Supporting real-time applications in an Integrated Services Packet Network: architecture and mechanism. Conference proceedings on Communications architectures & protocols, p.14–26. 1992.

Clark, D. The Design Philosophy of the DARPA Internet Protocols. Proceedings of ACM SIGCOMM 88. 1988.

Davidson, J., J. Peters, M. Bhatia, S. Kalindindi e S. Mukherjee. Voice Over IP Fundamentals. Editora Ciscopress.com -USA, p.179. 2004.

DCC UFAM – Departamento de Ciência da Computação da Universidade do Amazonas. http://grcm.dcc.ufam.edu.br/index.php/Callgen323, acessado em Janeiro/2009.

DARPA – Defense Advanced Research Projects Agency. Internet Protocol. RFC 791. 1981.

Devera, M. Hierarchical token bucket theory, http://luxik.cdi.cz/devik/qos/htb/manual/theory.htm, acessado em Agosto/2006.

EHAS – Enlace Hispano Americano de Salud. http://www.ehas.org/, acessado em Julho/2006.

Ericsson white paper. Combinational services – the pragmatic first step toward all-IP, Ericsson Review n.2. 2003.

Ferguson, P. e G. Huston. Quality of Service: Delivering QoS on the Internet and in Corporate Networks. Editora Wiley Computer Publishing. 1999.

Ferreira, M. V. A., B. L. Wanderley e D. C. M. Saade. Diferenciação de Serviços para Chamadas Multimídia Baseada na Importância da Chamada. I2TS. 2008.

Floyd, S. e V. Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking (TON), v.1, p.397–413. 1993.

Floyd, S. e V. Jacobson. Link-sharing and resource management models for packet networks. IEEE/ACM Transactions on Networking, v.3, p.365–386. 1995.

Fragouli, C., V. Sivaraman e M. Srivastava. Controlled multimedia wireless link sharing via enhanced class-based queuing with channel-state-dependent packet scheduling. Procedures of IEEE INFOCOM, p.572–580. 1998.

Foster, B. e C. Sivachelvan. Media Gateway Control Protocol (MGCP). RFC 3661. 2003.

Goode, B. Voice Over Internet Protocol (VoIP), Proceedings of the IEEE, v.90, p.1495-1517. 2002.

H323 Call Generator. http://callgen323.sourceforge.net/, acessado em Janeiro/2009.

Handley, M., H. Schulzrinne, E. Schooler e J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543. 1999.

Hassan, M. e M. Atiquzzaman. Internet telephony: services, technical challenges and products. IEEE Communications Magazine, p.96–103. 2000.

Hassan, J. Engineering Internet QoS. Artech House Inc. 2002.

Heinanen, J., F. Baker, W. Weiss e J. Wroclawski. Assured Forwarding PHB Group. RFC 2597. 1999.

Heinanen, J. e R. Guerin. A Single Rate Three Color Marker. RFC 2697. 1999.

Heinanen, J. e R. Guerin. A Two Rate Three Color Marker. RFC 2698. 1999.

Hersent, O., D. Guide e J. P. Petit. Telefonia IP – Comunicação multimídia baseada em pacotes. Editora PEARSON Addison Wesley. 2002.

Hilário, L. E. N. Qualidade de serviço em redes Mesh. Dissertação de Mestrado - Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil. 2006.

Hubert, B. LARTC - Linux Advanced Routing and Traffic Control, http://www.lartc.org/, acessado em Dezembro/2005.

IEEE 802.11s (Redes Mesh). http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm, acessado em Agosto/2007.

IETF – Internet Engineering Task Force, http://www.ietf.org, acessado em Junho/2008.

IPERF. http://sourceforge.net/projects/iperf/, acessado em Maio/2007.

ITU - International Telecommunication Union. http://www.itu.int/, acessado em Dezembro/2007.

ITU–T Recommendation P.800. Methods for subjective determination of transmission quality. 1996.

ITU–T Recommendation P.830. Subjective performance assessment of telephone band and wideband digital codecs. 1996.

ITU–T Recommendation H.323. Packet–based multimedia communications systems. 1998.

Jacobson, V., K. Nichols e K. Poduri. An Expedited Forwarding PHB. RFC 2598. 1999.

Kurose, J. e K. Ross. Redes de Computadores e a Internet - Uma Nova Abordagem. Editora PEARSON Addison Wesley. 2004.

K-Phone. http://sourceforge.net/projects/kphone/, acessado em Fevereiro/2008.

Linney, L. Differentiated Services on IBM 221x Routers. IBM Co, 1999.

Law, K.L.E. The bandwidth guaranteed prioritized queuing and its implementations. Global Telecommunications Conference, IEEE GLOBECOM'97, v.3, p.1445–1449. 1997.

Martins, J. Qualidade de Serviço (QoS) em Redes IP, Princípios Básicos, Parâmetros e Mecanismos. Network Designers, 1999.

McKenney, P. Stochastic fairness queuing - the multiple facets of integration. INFOCOM'90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies, v.2, p.733–740. 2000.

Melo, E. T. L. Qualidade de Serviço em Redes IP com DiffServ: Avaliação através de Medições. Dissertação de Mestrado - Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil. 2001.

Nagle, J. On packet switches with infinite storage, IEEE Transactions on Communications, v. 35, p.435–438. 1987.

Nichols, K., S. Blake, F. Baker e D. Black. Definition of the Differentiated Services Filed (DS Field) in the Ipv4 and Ipv6 Headers. RFC 2474. 1998.

OpenH323Project. http://sourceforge.net/projects/openh323, acessado em Dezembro/2007.

Parekh, A. e R. Gallager. A generalized processor sharing approach to flow control integrated services networks: the single-node case. IEEE/ACM Transactions on Networking, p.344–357. 1993.

Passos, D., D. V. Teixeira, D. C. M. Saade, L. C. Schara e C. Albuquerque. Mesh Network Performance Measurements. 5th International Information and Telecommunicatios Technologies Symposium, Cuiabá, MT, Brasil. 2006.

Peterson, J. Telephone Number Mapping (ENUM) Service. RFC 3953. 2005.

Plachecki, K. e S. Przylucki. Algorithms of buffering packets in computer network IP. Proceedings of the SPIE, v.5125, p.398–401. 2003.

Poikselkä, M., G. Mayer, H. Khartabik e A. Niemi. The IMS – IP Multimedia Concepts and Services in the Mobile Domain. John Wiley and Sons LTD. 1994.

Projeto ReMesh – Grupo de Pesquisa em Redes Mesh, http://mesh.ic.uff.br/, acessado em Junho/2008.

Protocolo ENUM (Electronic Numbering), http://www.enum.org/, acessado em Julho/2008.

RNP – Rede Nacional de Ensino e Pesquisa. http://www.rnp.br/, acessado em Dezembro/2008.

Rönngren, R. e R. Ayani. A comparative study of parallel and sequential priority queue algorithms. ACM transaction Model. Comput. Simul., p.157–209. 1997.

Sanneck, H. e W. Mohr. Selective packet prioritization for wireless voice over ip (VoIP). Fourth International Symposium on Wireless Personal Multimedia Communication, p.621–630. 2001.

Schmidt, A. L. P. O Protocolo RSVP e o Desempenho de Aplicações Multimídia. Newsgeneration, v.4, n.3. 2000.

Schulzrinne, H. Historical notes, http://www.cs.columbia.edu/hgs/rtp/history.html, acessado em Janeiro/2006.

Schulzrinne, H e J. Rosenberg. Internet telephony: Architecture and protocols – an IETF perspective. Computer Networks and ISDN Systems, v.31, p.237–255. 1999.

Schulzrinne, H. e K. Arabshian. Providing emergency services in Internet telephony. Internet Computing IEEE, v.6, p.39–47. 2002.

Schulzrinne, H., V. Jacobson, S. Casner e R. Frederick. RTP: A Transport Protocol for Real–Time Applications. RFC 1889. 1996.

Shenker, S., C. Partridge e R. Guerin. Specification of Guaranteed Quality of Service. RFC 2212. 1997.

Skype, http://www.skype.com/, acessado em Julho/2008.

Stiliadis, D. e A. Varma. Efficient fair queuing algorithms for packet-switched networks. IEEE/ACM Transactions on Networking (TON), v.6, p.175-185. 1998.

Szigeti, T. e C. Hattingh. End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs. Editora Cisco Press, 2004.

Trillium Digital Systems, Inc. H.323 Tutorial. 2005.

Vemuri, A. e J. Peterson. Session Initiation Protocol for Telephones (SIP-T): Context and Architectures. RFC 3372. 2002.

Wroclawski, J. Specification of the Controlled-Load Network Element Service. RFC 2211. 1997.

Zhang, L. Virtual clock: A new traffic control algorithm for packet-switched networks. ACM Transaction on Computer Systems, v.9, n.2, p.96–103. 1991.

9. ANEXOS

9.1. SCRIPT TCP x UDP

```
#!/bin/sh
#
# 1st parameter MAX_CLIENTS
# 2nd parameter BASE_PORT

DEV=eth1
# Bandwidth values (in kbits)
BANDWIDTH=300
CEIL_UDP=300
CEIL_TCP=300

# The following must sum $BANDWIDTH or less

# Banda UDP Kbits
UDP_BW=300

#Banda TCP Kbits
TCP_BW=100

#Prioridades
PRIO_UDP=0
PRIO_TCP=1

INSMOD=/sbin/insmod

# Load kernel modules
grep -q ^sch_htb /proc/modules || $INSMOD /lib/modules/`uname -r`/sch_htb.o
grep -q ^sch_sfq /proc/modules || $INSMOD /lib/modules/`uname -r`/sch_sfq.o
grep -q ^cls_u32 /proc/modules || $INSMOD /lib/modules/`uname -r`/cls_u32.o

# Clean existing qdiscs (redirects error output to /dev/null)
tc qdisc del dev $DEV root 2> /dev/null
tc qdisc del dev $DEV ingress 2> /dev/null

# Creates root qdisc with HTB policy. Default class for packets is 12 (less
prio)
tc qdisc add dev $DEV root handle 1: htb default 12
```

```
tc class add dev $DEV parent 1: classid 1:1 htb rate ${BANDWIDTH}kbit ceil
${BANDWIDTH}kbit

# Now creates 2 classes: (1:10 = max prio)
# 1:10 for UDP packets
# 1:11 for TCP packets
tc class add dev $DEV parent 1:1 classid 1:10 htb rate ${UDP_BW}kbit ceil
${CEIL_UDP}kbit
tc class add dev $DEV parent 1:1 classid 1:11 htb rate ${TCP_BW}kbit ceil
${CEIL_TCP}kbit

# Now creates qdiscs to handle the packets of each class
# Since all of them must treat equally concurrent connections we use sqf
everywhere

tc qdisc add dev $DEV parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev $DEV parent 1:11 handle 11: sfq perturb 10

echo "Vai comecar os match"
# Filters

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_UDP u32 \
      match ip protocol 17 0xff \
      flowid 1:10

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_TCP u32 \
      match ip protocol 6 0xff \
      flowid 1:11
```

9.2. SCRIPT COM PRIORIZAÇÃO DE FLUXOS

```
#!/bin/sh
#
# 1st parameter MAX_CLIENTS
# 2nd parameter BASE_PORT

DEV=eth1
# Bandwidth values (in kbits) 54Mbits
BANDWIDTH=100
CEIL_VOZ=0
CEIL_VIDEO=0
```

```
CEIL_DADOS=0
CEIL_OTHERS=0

# The following must sum $BANDWIDTH or less

# Banda de voz 27 Mbits - 50%
VOZ_BW=50

# Banda de video 10,8 Mbits - 20%
VIDEO_BW=20

# Banda de Dados 10,8 Mbits - 20%
DADOS_BW=15

# Banda de outros 5,4 Mbits - 10%
OTHERS_BW=5

# Banda de voz distribuida por importancia da chamada

VOZ_LEVEL_0_BW=16
VOZ_LEVEL_1_BW=10
VOZ_LEVEL_2_BW=5
VOZ_LEVEL_3_BW=5
VOZ_LEVEL_4_BW=5
VOZ_LEVEL_5_BW=5
VOZ_LEVEL_6_BW=2
VOZ_LEVEL_7_BW=2

# Banda de voz distribuida por usuario de emergencia

#VOZ_LEVEL_0_0_BW=
#VOZ_LEVEL_0_1_BW=
#VOZ_LEVEL_0_2_BW=
#VOZ_LEVEL_0_3_BW=
#VOZ_LEVEL_0_4_BW=
#VOZ_LEVEL_0_5_BW=
#VOZ_LEVEL_0_6_BW=
#VOZ_LEVEL_0_7_BW=

# Banda de voz distribuida por usuario VIP

#VOZ_LEVEL_1_0_BW=
#VOZ_LEVEL_1_1_BW=
#VOZ_LEVEL_1_2_BW=
#VOZ_LEVEL_1_3_BW=
#VOZ_LEVEL_1_4_BW=
#VOZ_LEVEL_1_5_BW=
#VOZ_LEVEL_1_6_BW=
#VOZ_LEVEL_1_7_BW=

# Banda de voz distribuida por usuario normal

#VOZ_LEVEL_2_0_BW=
#VOZ_LEVEL_2_1_BW=
#VOZ_LEVEL_2_2_BW=
#VOZ_LEVEL_2_3_BW=
#VOZ_LEVEL_2_4_BW=
#VOZ_LEVEL_2_5_BW=
#VOZ_LEVEL_2_6_BW=
#VOZ_LEVEL_2_7_BW=
```

```
# Banda de voz distribuida por usuario pré-pago

#VOZ_LEVEL_3_0_BW=
#VOZ_LEVEL_3_1_BW=
#VOZ_LEVEL_3_2_BW=
#VOZ_LEVEL_3_3_BW=
#VOZ_LEVEL_3_4_BW=
#VOZ_LEVEL_3_5_BW=
#VOZ_LEVEL_3_6_BW=
#VOZ_LEVEL_3_7_BW=

#Buffer length in k
#VIDEO_BUFFER=15000

# Priorities to get the remaining bandwidth
# 0 is max (can be repeated)
PRIO_VOZ=0
PRIO_VIDEO=1
PRIO_DADOS=2
PRIO_OTHERS=3

# Priorities to get the users

#PRIO_VOZ_EMERG=0
#PRIO_VOZ_VIP=1
#PRIO_VOZ_POS=2
#PRIO_VOZ_PRE=3

# Priorities to get the sub-classes

PRIO_VOZ_SC0=0
PRIO_VOZ_SC1=1
PRIO_VOZ_SC2=2
PRIO_VOZ_SC3=3
PRIO_VOZ_SC4=4
PRIO_VOZ_SC5=5
PRIO_VOZ_SC6=6
PRIO_VOZ_SC7=7

INSMOD=/sbin/insmod

# Load kernel modules
grep -q ^sch_htb /proc/modules || $INSMOD /lib/modules/`uname -r`/sch_htb.o
grep -q ^sch_sfq /proc/modules || $INSMOD /lib/modules/`uname -r`/sch_sfq.o
grep -q ^cls_u32 /proc/modules || $INSMOD /lib/modules/`uname -r`/cls_u32.o

echo "Vai comecar o script"

# Clean existing qdiscs (redirects error output to /dev/null)
tc qdisc del dev $DEV root 2> /dev/null
tc qdisc del dev $DEV ingress 2> /dev/null

echo "Passou pelo del"

# Creates root qdisc with HTB policy. Default class for packets is 22 (less
prio). O default 22 define que qualquer
# trafego que nao estiver classificado de outra forma sera inputado a
classe 1:22.

tc qdisc add dev $DEV root handle 1: htb default 22
```

```
echo "Passou pelo cria qdisc root"

# Creates a root class hanging from the root qdisc
# This allows other sub-classes of this one to borrow the remaining
bandwidth
# If we prescind of this one and hang subclasses directly from the root
qdisc
# then the remaining bandwidth is lost (!)

tc class add dev $DEV parent 1: classid 1:1 htb rate ${BANDWIDTH}kbps ceil
${BANDWIDTH}kbps

echo "Passou pela primeira classe"

# Now creates 4 classes: (1:10 = max prio)
# 1:10 for Voice packets
# 1:11 for Video packets
# 1:12 for Data packets
# 1:13 for Other packets

tc class add dev $DEV parent 1:1 classid 1:10 htb rate ${VOZ_BW}kbit ceil
${CEIL_VOZ}kbit prio $PRIO_VOZ
tc class add dev $DEV parent 1:1 classid 1:11 htb rate ${VIDEO_BW}kbit ceil
${CEIL_VIDEO}kbit prio $PRIO_VIDEO
tc class add dev $DEV parent 1:1 classid 1:12 htb rate ${DADOS_BW}kbit ceil
${CEIL_DADOS}kbit prio $PRIO_DADOS
tc class add dev $DEV parent 1:1 classid 1:13 htb rate ${OTHERS_BW}kbit
ceil ${CEIL_OTHERS}kbit prio $PRIO_OTHERS

echo "Passou por todas as classes"

# Creates eight sub-classes for Voz (level 0 (1:14), level 1 (1:15), level
2 (1:16), level 3 (1:17), level 4 (1:18),
# level 5 (1:19), level 6 (1:20) and level 7 (1:21))

tc class add dev $DEV parent 1:10 classid 1:14 htb rate
${VOZ_LEVEL_0_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC0
tc class add dev $DEV parent 1:10 classid 1:15 htb rate
${VOZ_LEVEL_1_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC1
tc class add dev $DEV parent 1:10 classid 1:16 htb rate
${VOZ_LEVEL_2_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC2
tc class add dev $DEV parent 1:10 classid 1:17 htb rate
${VOZ_LEVEL_3_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC3
tc class add dev $DEV parent 1:10 classid 1:18 htb rate
${VOZ_LEVEL_4_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC4
tc class add dev $DEV parent 1:10 classid 1:19 htb rate
${VOZ_LEVEL_5_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC5
tc class add dev $DEV parent 1:10 classid 1:20 htb rate
${VOZ_LEVEL_6_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC6
tc class add dev $DEV parent 1:10 classid 1:21 htb rate
${VOZ_LEVEL_7_BW}kbit ceil ${BANDWIDTH}kbit prio $PRIO_VOZ_SC7

echo "Passou pelas subclasses"

# Now creates qdiscs to handle the packets of each class
# Since all of them must treat equally concurrent connections we use sqf
everywhere

tc qdisc add dev $DEV parent 1:11 handle 11: sfq perturb 10
tc qdisc add dev $DEV parent 1:12 handle 12: sfq perturb 10
tc qdisc add dev $DEV parent 1:13 handle 13: sfq perturb 10
```

```
tc qdisc add dev $DEV parent 1:14 handle 14: sfq perturb 10
tc qdisc add dev $DEV parent 1:15 handle 15: sfq perturb 10
tc qdisc add dev $DEV parent 1:16 handle 16: sfq perturb 10
tc qdisc add dev $DEV parent 1:17 handle 17: sfq perturb 10
tc qdisc add dev $DEV parent 1:18 handle 18: sfq perturb 10
tc qdisc add dev $DEV parent 1:19 handle 19: sfq perturb 10
tc qdisc add dev $DEV parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev $DEV parent 1:21 handle 21: sfq perturb 10

echo "Vai comecar os match"

# Filters

#tem que especificar primeiro se é um pacote transferido por tcp ou udp (no
caso de pacotes de controle é usado tcp)
#depois, a partir do vigésimo byte, que é o campo de opções do cabeçalho
ip, deve-se ler os bytes mostrados
#depois disso, passar o fluxo para a classe de pacotes de controle
#protocol 6 e TCP
#protocolo 17 e udp

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC0 u32 \
      match ip tos 0x04 0xff \
      flowid 1:14

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC1 u32 \
      match ip tos 0x08 0xff \
      flowid 1:15

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC2 u32 \
      match ip tos 0xc 0xff \
      flowid 1:16

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC3 u32 \
      match ip tos 0x10 0xff \
      flowid 1:17

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC4 u32 \
      match ip tos 0x14 0xff \
      flowid 1:18

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC5 u32 \
      match ip tos 0x18 0xff \
      flowid 1:19

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC6 u32 \
      match ip tos 0x1c 0xff \
      flowid 1:20

tc filter add dev $DEV parent 1: protocol ip prio $PRIO_VOZ_SC7 u32 \
      match ip tos 0x20 0xff \
      flowid 1:21

echo "Passou pelo 1 match"

#tem que especificar primeiro se é um pacote transferido por tcp ou udp (no
caso de pacotes de vídeo é usado udp)
#depois, a partir do vigésimo primeiro byte, que é o campo de opções do
cabeçalho ip, deve-se ler os bytes mostrados
#depois disso, passar o fluxo para a classe de pacotes de controle
```

```
tc filter add dev $DEV protocol ip parent 1: prio $PRIO_DADOS u32 \
      match ip tos 0x40 0xff \
      flowid 1:12

echo "Passou pelo 2 match"

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_VIDEO u32 \
      match ip tos 0x80 0xff \
      flowid 1:11

echo "Passou pelo 3 match"

#Filtro da classe OTHERS

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip protocol 17 0xff \
      match u8 0xa0 0xff at 31 \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x00 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x01 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x02 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
     match ip tos 0x03 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
     match ip tos 0x24 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
     match ip tos 0x25 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
    match ip tos 0x26 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
     match ip tos 0x27 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
    match ip tos 0x28 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
     match ip tos 0x29 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x2A 0xff \
```

```
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x2B 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x2C 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x2D 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x2E 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x2F 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x30 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x31 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x32 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x33 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x34 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x35 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x36 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x37 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x38 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x39 0xff \
      flowid 1:13
```

```
tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x3A 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x3B 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x3C 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x3D 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x3E 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x3F 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x41 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x42 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x43 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x44 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x45 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x46 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x47 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x48 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x49 0xff \
      flowid 1:13
```

```
tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x4A 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x4B 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x4C 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x4D 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x4E 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x4F 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x50 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x51 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x52 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x53 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x54 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x55 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x56 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x57 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x58 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
```

```
      match ip tos 0x59 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x5A 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x5B 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x5C 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x5D 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x5E 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x5F 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x60 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x61 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x62 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x63 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x64 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x65 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x66 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x67 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x68 0xff \
```

```
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x69 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x6A 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x6B 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x6C 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x6D 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x6E 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x6F 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x70 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x71 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x72 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x73 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x74 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x75 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x76 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x77 0xff \
      flowid 1:13
```

```
tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x78 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x79 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x7A 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x7B 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x7C 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x7D 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x7E 0xff \
      flowid 1:13

tc filter add dev $DEV protocol ip parent 1: prio $PRIO_OTHERS u32 \
      match ip tos 0x7F 0xff \
      flowid 1:13

echo "Passou pelo 4 match"
```