JORGE LIMA DE OLIVEIRA FILHO

TECNOLOGIAS DIFFSERV COMO SUPORTE PARA A QUALIDADE DE SERVIÇO (QOS) DE APLICAÇÕES MULTIMÍDIA - ASPECTOS DE CONFIGURAÇÃO E INTEGRAÇÃO

SALVADOR
2006

JORGE LIMA DE OLIVEIRA FILHO

TECNOLOGIAS DIFFSERV COMO SUPORTE PARA A QUALIDADE DE SERVIÇO (QOS) DE APLICAÇÕES MULTIMÍDIA - ASPECTOS DE CONFIGURAÇÃO E INTEGRAÇÃO

Dissertação apresentada à Universidade Salvador, como parte das exigências do Curso de Mestrado Profissional em Redes de Computadores, área de concentração em Redes de Computadores, para a obtenção do título de Mestre.

Orientador: Prof. Dr. Joberto Sérgio Barbosa Martins

SALVADOR
2006

JORGE LIMA DE OLIVEIRA FILHO

TECNOLOGIAS DIFFSERV COMO SUPORTE PARA A QUALIDADE DE SERVIÇO (QOS) DE APLICAÇÕES MULTIMÍDIA - ASPECTOS DE CONFIGURAÇÃO E INTEGRAÇÃO

Dissertação apresentada à Universidade Salvador, como parte das exigências do Curso de Mestrado Profissional em Redes de Computadores, área de concentração em Redes de Computadores, para a obtenção do título de Mestre.

Banca Examinadora:

Prof. Dr.	Membro
Prof. Dr.	Membro
Prof. Dr. Joberto Sérgio Barbosa Martins	Orientador

SALVADOR
2006

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com o surgimento de novas aplicações multimídia em redes TCP/IP, surgiu a necessidade das redes IP suportarem requisitos diferenciados de qualidade de serviço (QoS) visando o funcionamento adequado dessas aplicações. A arquitetura DiffServ (Arquitetura de Serviços Diferenciados) surgiu para atender aos requisitos de QoS das aplicações multimídia, garantindo um serviço mais escalável e flexível comparado ao IntServ, baseado na diferenciação de serviços. A abordagem principal do DiffServ é o processamento de agregados de tráfego que, em termos práticos, são pacotes com necessidades semelhantes de QoS. O objetivo deste trabalho de dissertação consiste na especificação, desenvolvimento, implantação e validação de uma rede protótipo experimental (*testbed*) utilizando a solução DiffServ para a abordagem do problema de QoS em redes IP. O contexto básico para o qual a rede foi desenvolvida e validada consiste num cenário de redes IP suportando as aplicações médicas do Projeto Infravida[1]. Para a validação da Rede Protótipo Experimental foram desenvolvidas campanhas de teste com o objetivo de demonstrar a validade e a viabilidade da especificação e implantação desenvolvidas. O trabalho desenvolvido utiliza, na medida do possível, soluções de software livre.

Palavras-chave: qualidade de serviço, DiffServ, Aplicações para telemedicina, redes IP, Rede protótipo experimental.

[1] O "Projeto Infravida" foi um projeto de P&D desenvolvido pela UNIFACS, UFPE, UFBA e Hospital Português do Recife. O Infravida (Infra-estrutura de Vídeo Digital para Aplicações de Telesaúde) visava o desenvolvimento de um sistema de telemedicina, incluindo aplicações de Telediagnósticos, Segunda Opinião Médica e um Sistema de Educação Continuada a Distância para profissionais de saúde baseados na Internet e redes IP.

ABSTRACT

The new multimedia applications based on TCP/IP networks are becoming prevalent for communications and, in this context, there is an urgent need to support quality of service (QoS) on these networks. The DiffServ (Differentiated Services) architecture has been proposed as an efficient technological to support QoS requirements for these new multimedia applications, providing a more scalable and flexible service through service differentiation. The basic approach used by DiffServ is the aggregate processing of traffic that, in practical terms, corresponds to process packets with similar QoS needs. The main goal the work reported on this dissertation consists on the specification, development, implementation and validation of an "experimental prototype network" (testbed) using the DiffServ solution as the basic approach to support the QoS' problem in IP networks. The basic implementation context developed and validated consists of a scenario of an IP network supporting medical applications in the Infravida Project[1]. For the validation of the testbed", procedures have been developed with the objective of demonstrating the validity and viability of the developed specification and implementation. This dissertation work uses, as much as possible, free software solutions.

Keywords: quality of service, DiffServ, medical applications, IP Network, Experimental Prototype Network.

[1] The "Infravida Project" is a R&D project developed by UNIFACS, UFPE, UFBA and Portuguese Hospital of Recife. The Infravida (Infra-structure of Digital Video for Tele-Medicine Applications) aim the development of a Tele-Medicine system, including Telediagnosis applications, Second Medical Opinion and a system of continuos distance education for health professions based on the TCP/IP networks.

1. INTRODUÇÃO

O presente trabalho está contextualizo no ambiente das redes TCP/IP (*Transmission Control Protocol/Internet Protocol*) [Comer, 1998] que, com o advento da Internet evoluiu e, hoje, conecta milhões de usuários. As redes TCP/IP se popularizaram e foram efetivamente adotadas por milhares de organizações ao redor do mundo, sendo o protocolo mais utilizado para comunicação de dados.

Por outro lado, com a evolução na taxa de transmissão de dados, um grande número de novas aplicações surgiram explorando a maior disponibilidade das redes e, também, a viabilidade da utilização de múltiplas mídias. Dentre estas novas aplicações destacam-se as aplicações multimídia que, diferentemente das aplicações de dados tradicionais, apresentam novos requisitos para um funcionamento adequado como, por exemplo, a necessidade de garantia de banda, atraso e a priorização de tráfego de uma aplicação em relação a outras. O protocolo TCP/IP em sua forma original não tem prevista uma forma de garantir que esses requisitos sejam atendidos.

Esses novos requisitos diferenciados necessários para as aplicações multimídia em rede levam à noção e necessidade de garantias da qualidade de serviço (QoS) [Vegesna, 2001] em redes TCP/IP. Com objetivo de atender às necessidades de QoS deste novo conjunto de aplicações, surgiram diversas iniciativas de pesquisa e desenvolvimento e grupos de trabalhos espalhados pelo mundo como, por exemplo, a organização IETF *(Internet Engineering Task Force)* [IETF, 2002] possui vários grupos de trabalho (GTs). O IETF no contexto de suas ações de padronização em QoS trabalhou inicialmente na definição da arquitetura IntServ *(Integrated Services)* [Braden et al., 1994]. Nessa arquitetura, as aplicações sinalizam seus requisitos de QoS para a rede que encaminha essa sinalização até o destino final com o objetivo de reservar os recursos através do protocolo RSVP (*Resource Reservation Protocol*)

[Braden et al., 1997]. Ainda no contexto dos trabalhos do IETF, foi definida a arquitetura DiffServ *(Differentiated Services)* [Blake et al., 1998], com o objetivo principal de possibilitar a implementação de diferenciação de serviços na Internet. Uma outra ação de padronização do IETF foi responsável pela definição do MPLS *(Multiprotocol Label Switching)* [Rosen et al., 2001], uma forma de comutação de rótulo, como uma alternativa mais rápida e flexível que o roteamento tradicional das redes TCP/IP e que também apresenta uma aplicabilidade relevante no contexto da implantação de soluções suportando a qualidade de serviço em redes. O projeto Infravida, uma outra componente do contexto desta dissertação, foi desenvolvido pelas Universidade Salvador (Unifacs), Universidade Federal de Pernambuco (UFPE), Universidade Federal da Bahia (UFBA), Universidade Federal do Rio Grande do Norte (UFRN) e Universidade Federal da Paraíba (UFPB), e envolve ainda o Real Hospital Português do Recife, a Faculdade de Medicina da UFBA (FAMED) e o Hospital das Clínicas da UFPE. De maneira geral, e em relação aos aspectos de QoS o projeto Infravida teve os seguintes objetivos:

- Desenvolver um sistema de telemedicina, incluindo aplicações de telediagnósticos e segunda opinião médica.
- Desenvolver um sistema de educação continuada a distância para profissionais de saúde baseados na Internet.
- Integrar os sistemas desenvolvidos sobre uma infra-estrutura de vídeo digital que implementa QoS em redes IP.

O presente trabalho de dissertação se encaixa no contexto mais geral do projeto Infravida e tem como objetivo principal a especificação, implantação e desenvolvimento de uma rede protótipo experimental (*testbed)* e desenvolvimento de campanhas de teste que

visam fundamentalmente demonstrar a validação e a viabilidade do projeto Infravida com respeito à qualidade de serviço requerida pelo mesmo. De forma mais detalhada, a dissertação objetiva o estudo dos mecanismos de qualidade de serviço em redes IP, a avaliação e identificação das aplicações envolvidas no projeto Infravida que necessitam de QoS, a identificação dos requisitos de QoS destas aplicações e, finalmente, a especificação, desenvolvimento e teste de uma implantação de uma rede protótipo experimental que satisfaça as necessidades de QoS com relação aos requisitos de atraso fim-a-fim e largura de banda, dentre outros. O objetivo macro do trabalho desenvolvido é que o tráfego em tempo real das aplicações multimídia elencadas receba um tratamento diferenciado pela rede IP de forma que o tráfego concorrente de dados dos outros sistemas componentes da solução Infravida e de outras aplicações que façam uso concorrente das mesmas redes não comprometam o tempo de resposta e a qualidade dos serviços disponibilizados durante a execução das aplicações.

A dissertação proposta adota uma abordagem mais prática. Para tal, foi especificada, implantada e testada uma rede protótipo experimental para fins de avaliação e validação da proposta de QoS definida para o projeto Infravida no Núcleo Interdepartamental de Pesquisas em Redes de Computadores (Nuperc) da Universidade Salvador.

A rede protótipo experimental implantada disponibiliza uma infra-estrutura básica de rede IP (roteadores e PCs) em ambiente aberto GNU/ Linux [McLagan, 2003]. A rede protótipo experimental adotou a arquitetura DiffServ como plataforma básica para a implantação da solução de QoS.

Em outras palavras, o objetivo precípuo da dissertação é a emulação e criação de diversas situações que espelhem a realidade das aplicações e da rede no cenário do projeto Infravida e por fim, fazer a medição dos fluxos destas aplicações. Com este objetivo o trabalho de dissertação envolveu na sua fase de testes de implantação a geração de diversos

padrões de tráfego de aplicações e a aplicação de diferentes configurações de QoS nos roteadores visando espelhar os diversos cenários de aplicação do projeto Infravida.

A gerência dos roteadores foi também um aspecto operacional avaliado no trabalho desenvolvido na dissertação. No caso, uma solução desenvolvida em Java, para a invocação remota de *scripts* de configuração em cada roteador foram avaliadas.

1.1 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado como segue:

Este capítulo, a introdução, aborda o contexto no qual o trabalho está inserido, descreve em resumo o projeto Infravida e os principais componentes envolvidos em relação aos objetivos gerais, específicos e implantação realizada.

O capítulo 2 aborda os princípios de QoS. Trata-se de um resumo dos principais conceitos de qualidade de serviço, seus parâmetros e como a QoS é caracterizada. Apresenta também as alternativas de implantação da qualidade de serviço em redes IP. Neste capítulo são mostradas ainda as operações básicas dos roteadores, o algoritmo do balde de fichas (t*oken bucket)*. São abordadas também as funcionalidades básicas para a implantação da qualidade de serviço nos roteadores como técnicas de enfileiramento, policiamento de tráfego, suavização de tráfego, esquemas de classificação e marcação.

No capítulo 3 a arquitetura DiffServ é detalhada. O capítulo apresenta o modelo DiffServ com a marcação de pacotes, os principais comportamentos no roteador (*PHB*), explica como é o DiffServ para o GNU/Linux, mostra as ferramentas utilizadas para implantar QoS no GNU/Linux. Todas as técnicas e ferramentas utilizadas neste capítulo têm a finalidade de implantar uma rede protótipo experimental utilizada para a validação de QoS das aplicações do projeto Infravida.

No capítulo 4 são apresentados o cenário detalhado do projeto Infravida, as aplicações de telemedicina e seus requisitos de QoS e alternativas de QoS para alcançar os seus requisitos. Aborda também a construção da rede protótipo experimental (*testbed)*, ao tempo em que apresenta os *scripts* de configuração dos roteadores para a obtenção de QoS para as aplicações do projeto Infravida.

No capítulo 5 são apresentados os cenários de teste com suas respectivas campanhas de medição e a análise dos resultados obtidos.

No capítulo 6 é feito um resumo de todos os capítulos apresentando as considerações relativas aos resultados e objetivos do trabalho e sugerindo futuros trabalhos que complementem as contribuições deste estudo.

2. QUALIDADE DE SERVIÇO – PRINCÍPIOS BÁSICOS

O crescimento do número de aplicações que utilizam o conjunto de protocolo TCP/IP, principalmente das aplicações multimídia, as quais necessitam de requisitos mínimos para que apresentem um bom funcionamento, e, como o projeto inicial das redes IP não previa essa evolução, não atendendo os requisitos das novas aplicações, foram desenvolvidos vários mecanismos para contornar o problema de projeto inicial do conjunto de protocolo TCP/IP.

Neste capítulo serão discutidos os princípios básicos (parâmetros e caracterização) da qualidade de serviço para possibilitar o melhor aproveitamento das aplicações nas redes que necessitem de requisitos mínimos para o bom funcionamento.

Ultimamente na Internet e, por decorrência, nas redes privadas (locais, metropolitanas e de longa-distância) que utilizam como plataforma base os protocolos TCP/IP, houve um aumento significativo na utilização de aplicações multimídia. Estas aplicações utilizam diversos tipos de mídias como som, vídeo, imagem, gráficos e, obviamente, texto, de forma a tornar a interface com o usuário mais amigável. Observa-se então a necessidade de que a rede garanta determinados parâmetros de operação que, de maneira geral, são referenciados como sendo a "qualidade de serviço" (QoS) da rede.

O termo qualidade de serviço (QoS) pode ter várias definições diferentes, dependendo do autor. Segundo [Durand et al., 2001], qualidade de serviço é "conjunto de ferramentas disponível ao administrador da rede para assegurar certa garantia ao qual um nível mínimo de serviço irá ser provido a certo tráfego". Já [Martins et al., 2003] definem qualidade de serviço como sendo "requisito de uma aplicação expressada por um conjunto de parâmetros na qual deve ser provida pela rede sobre o fundamento fim-a-fim para preservar o comportamento adequado da operação de uma aplicação e uma "satisfação" do usuário final". Já [Westerinen et al., 2001] define qualidade de serviço como "a capacidade de entregar um serviço de rede

de acordo com os parâmetros especificados no Acordo de Nível de Serviço (*Service Level Agreement – SLA*)". Em suma, qualidade de serviço pode ser definida como um conjunto de ações a serem tomadas pelo administrador da rede para garantir um serviço que necessite de mínimas condições de funcionamento para um determinado fluxo de uma aplicação.

A qualidade de serviço é caracterizada para as aplicações (de todos os tipos) como sendo um conjunto de parâmetros de operação que, direta ou indiretamente, definem as características da aplicação em questão. Uma das questões importantes neste cenário consiste em "especificar" a QoS de forma clara e inequívoca em termo dos usuários de rede e dos provedores de serviço de rede. No caso, uma das alternativas possíveis de especificação de QoS é através do conceito de Acordo de Nível de Serviço (*Service Level Agreements – SLA*), Especificação de Nível de Serviço (*Service Level Specification – SLS*) e Especificação de Condicionamento de Tráfego (*Traffic Conditioning Specification – TCS*) [Westerinen et al., 2001].

Segundo [Blake et al., 1998], SLA é um contrato de serviço entre o cliente e um provedor de serviço que especifica o serviço de encaminhamento que será prestado do provedor para o cliente. Um cliente pode ser, por exemplo, uma organização, um provedor de serviço. Um SLA pode incluir regras de condicionamento de tráfego, garantias de disponibilidade, modelos de pagamento, mecanismos de autenticação além de outras questões legais e administrativas. Uma SLA pode especificar, por exemplo, que o provedor garantirá um mínimo de banda para certo tipo de tráfego ou que o provedor fornecerá um enlace com disponibilidade de 99,99%.

O termo Especificação de Nível de Serviço (*Service Level Specification – SLS*) é uma terminologia utilizada pelo DiffServ e detalha perfis de tráfego e seus comportamentos para cada nó da rede. Segundo [Grossman, 2002] SLS é usado quando o SLA é relacionado com o DiffServ para facilitar as especificações de QoS no contrato, define um conjunto de

parâmetros e seus respectivos valores para juntos especificar um serviço para um determinado fluxo de tráfego. Um SLS pode especificar, por exemplo, os fluxos (fonte, origem, destino) das aplicações ou definir uma ação a ser tomada por um roteador, por exemplo, descartar o excesso de tráfego de certa aplicação que desobedecer um determinado limite de volume de tráfego.

Um termo importante integrante do SLS é a Especificação de Condicionamento de Tráfego (*Traffic Conditioning Specification – TCS*). É uma nova terminologia introduzida em [Westerinen et al., 2001] para detalhar e especificar as regras do SLS quando se tratar de DiffServ. O TCS é um conjunto de parâmetros com seus respectivos valores que juntos especificam um conjunto de regras e um perfil de tráfego. Um esforço foi realizado para padronizar todos esses conceitos vistos anteriormente em relação à especificação de QoS, pelo projeto TEQUILA [TEQUILA, 2000].

Em seguida, discute-se os parâmetros básicos de caracterização da qualidade de serviço para as aplicações de maneira geral.

2.1 PARÂMETROS DE QoS – DEFINIÇÃO E DISCUSSÃO

Os parâmetros básicos de especificação da qualidade de serviço são as seguintes [Martins et al., 2003]:

- Atraso
- Variação de atraso (*jitter*)
- Largura de banda
- Perda de Pacote
- Confiabilidade

Em seguida, faremos uma discussão individual sobre cada um deles tentando ilustrar a sua importância no contexto das aplicações multimídia que são o foco da abordagem desta dissertação.

2.1.1 Atraso

O atraso é um parâmetro básico importante para a qualidade de serviço das aplicações. Pode ser definido como o tempo que os pacotes consomem enquanto são transmitidos até atingir seu destino final. No percurso do pacote, desde a transmissão até a recepção no seu destino podem haver diversos componentes que contribuam com o atraso.

Existem os seguintes tipos e/ou componentes de atraso:

- atraso de transmissão
- atraso de propagação
- atraso nas filas
- atraso de processamento nos equipamentos
- atraso fim-a-fim

O atraso de transmissão corresponde ao tempo gasto por um equipamento de rede (roteador, interface de rede etc.) para transmitir o pacote inteiro para o enlace. Considere um comprimento do pacote igual a “C” e a taxa de transmissão do equipamento igual a “T”. O atraso de transmissão é definido como sendo C/T. Hoje em dia as interfaces tendem a ser de alta velocidade (10, 100,1000 Mb/s) e, assim sendo, os atrasos de transmissão tendem a ser bastante pequenos.

O atraso de propagação é o tempo gasto para a propagação do sinal elétrico no meio que está sendo utilizado. Este meio, pode ser tipicamente, fibra opticas, coaxial. A figura 1 ilustra o atraso de propagação entre dois hospedeiros separados por um único enlace. O tempo que o primeiro *bit* transmitido leva da posição 1 até chegar na posição 3 (totalmente já recebido pelo receptor) é o atraso de propagação do meio ou seja o tempo que o sinal leva para se propagar do emissor até o receptor.

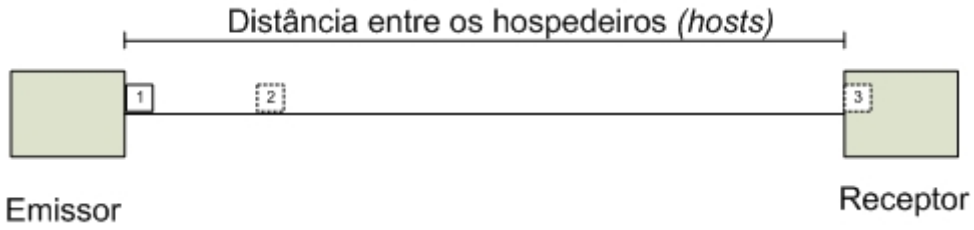

Figura 1: Atraso de propagação entre dois hospedeiros (*hosts)*

O atraso de propagação é calculado através da seguinte fórmula:

Atraso de propagação = distância entre dois pontos / velocidade de propagação

O atraso de propagação em enlaces longos (milhares de quilômetros) pode ser uma componente relevante para o atraso fim-a-fim.

O atraso nas filas é o tempo que os pacotes têm que esperar nas filas dos equipamentos antes de serem transmitidos no enlace. Esse atraso pode variar dependendo do estado das filas no momento que o pacote chega ao equipamento. A fila pode estar quase cheia, fazendo com que o pacote espere até o momento de ser transmitido ou a fila pode estar vazia fazendo com que o pacote não espere na fila e seja transmitido imediatamente.

O atraso de processamento é o tempo consumido pelos equipamentos de rede para examinar e processar os pacotes. Quando um pacote chega ao equipamento, por exemplo, ele

precisa examinar o cabeçalho do pacote e determinar onde enviá-lo de acordo com seu endereço de destino. Esse tempo consumido pelo equipamento para examinar e processar o pacote é conhecido como atraso de processamento.

Uma das componentes mais importantes na especificação e atendimento da qualidade de serviço de uma aplicação é o atraso fim-a-fim. O atraso fim a fim é a soma de todos os atrasos que o pacote sofre entre a origem (emissor) e o destino (receptor).

2.1.2 Variação de atraso (*jitter*)

Como descrito anteriormente, os pacotes sofrem diversos tipos de atrasos até chegar ao seu destino. Devido ao atraso nas filas dos roteadores serem muito variáveis, a soma dos atrasos, ou seja, o atraso fim-a-fim que os pacotes sofrem pode ser diferentes. Essa diferença de atraso fim-a-fim é conhecida como variação de atraso ou *jitter*.

O *jitter* é um parâmetro muito importante para as aplicações que necessitem reproduzir pacotes em ordem correta e em tempos definidos como as aplicações multimídias que exibem vídeos ou as de voz sobre IP (*Voice Over IP – VoIP).* Se a aplicação ignorar o *jitter* e reproduzir os pacotes no tempo e na ordem que eles chegam, a mídia pode ter sua qualidade degradada no receptor. O *jitter* pode ser removido com freqüência usando-se números de seqüência, marcas de tempo e atraso de reprodução como explica [Kurose and Ross, 2002].

2.1.3 Largura de banda

A banda (vazão) é a velocidade com que os pacotes são transmitidos na rede. É um parâmetro básico de QoS e é necessário para a operação adequada de qualquer aplicação. Cada aplicação geral utiliza diferente e, via da regra apresentam um requisito de banda

mínimo para o bom funcionamento. A tabela 1 [Martins, 1999] ilustra alguns valores de banda típicas de algumas aplicações.

Tabela 1: Banda típica de aplicações em rede

Aplicação	**Banda (Típica)**
Voz	1 kbps a 50 kbps
Aplicações Web	10 kbps a 500 kbps
Transferência de arquivos (grandes)	10 kbps a 1 Mbps
Vídeo (Streaming)	100 kbps a 10 Mbps
Aplicação conferência	500 kbps a 1 Mbps
Aplicação Imagens médicas	10 Mbps a 100 Mbps

2.1.4 Perda de pacotes

A perda de pacotes ocorre quando um pacote transmitido na rede não consegue chegar ao seu destino, ou seja, por algum motivo, em algum momento no trajeto o pacote falha, não chegando ao seu destinatário. Quando essa falha acontece, se diz que o pacote foi perdido ou houve uma perda de pacote. Existem diversos motivos para que aconteça a perda de pacotes. A seguir são mostradas as principais situações onde ocorrem perdas de pacotes:

- Perdas devido às filas dos roteadores estarem cheias. Um pacote quando chega numa fila cheia do roteador, automaticamente é descartado pelo roteador, ou seja, é perdido, pois não há espaço suficiente na fila para acomodá-lo.
- Falha em algum enlace no trajeto do pacote. Um enlace pode, por exemplo, se romper, fazendo com que todos os pacotes sejam perdidos.
- O pacote pode ser encaminhado pelo roteador de forma errada não chegando ao destino correto.
- Erros de transmissão também podem ocasionar perdas de pacotes.

Segundo [Kurose and Ross, 2002], o desempenho de um nó (roteador) é frequentemente medido não apenas em termos de atraso, mas também em termos da probabilidade de perda de pacotes.

2.1.5 Confiabilidade

A confiabilidade [Martins et al., 2003] é um parâmetro de qualidade considerado durante o projeto de uma rede. É influenciado por uma serie de fatores como, por exemplo, a qualidade dos componentes utilizados na fase de implantação de uma rede e a existência de caminhos alternativos em caso de falhas de alguns nós da rede. A confiabilidade de uma rede é um parâmetro importante para as aplicações críticas de empresas que utilizam as redes. Portanto, para qualidade de serviço das aplicações, a confiabilidade deve ser um parâmentro levado em consideração na fase de projeto.

2.2 ROTEADORES - OPERAÇÃO BÁSICA

Basicamente um roteador é composto por um processador, uma determinada quantidade de portas de entradas e uma certa quantidade de portas de saída. Cada porta tem associada à sua operação uma porção de memória - onde ficam armazenadas as filas - para acomodar os pacotes recebidos e a serem transmitidos. O processador executa os protocolos de roteamento, mantém as tabelas de roteamento e realiza funções de gerenciamento de rede, também é responsável, por exemplo, pela classificação e encaminhamento de pacotes [Kurose and Ross, 2002].

Em resumo, um pacote ao chegar num roteador, é processado, onde são tomadas as decisões como, por exemplo, de classificação e encaminhamento e, em seguida, pode ser repassado para a transmissão através de uma interface de saída ou pode ser descartado.

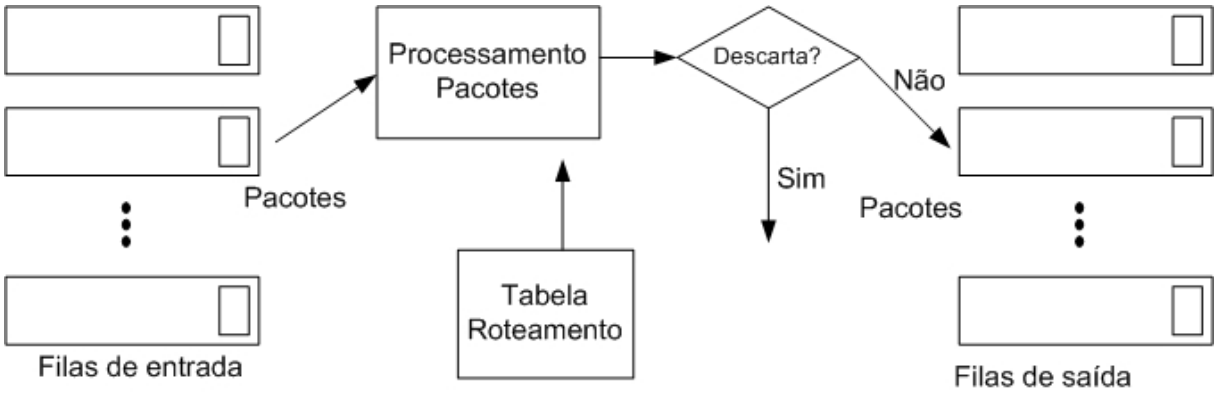

Figura 2: Resumo da operação de encaminhamento num roteador

Filas de pacotes podem se formar nos roteadores se a taxa de chegada dos pacotes for maior do que a taxa de saída. Ao preencher a memória das filas, os pacotes começam a ser descartados prejudicando o desempenho da aplicação.

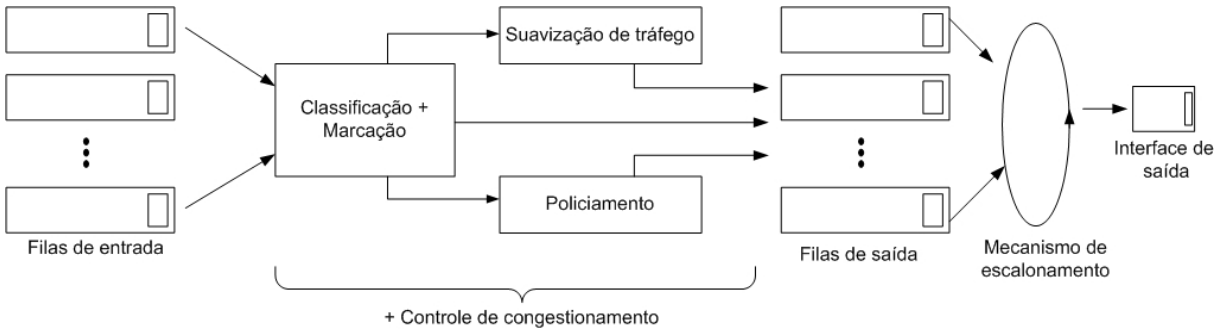

Figura 3: Funcionalidades necessárias para implantação de QoS nos roteadores [Martins et al., 2003]

As ações e funcionalidades que devem ser acionadas para fornecer qualidade de serviço para as aplicações são ilustradas na figura 3 e serão explicadas adiante, muitas delas relacionadas aos roteadores.

2.3 CLASSIFICAÇÃO

A função de classificação permite que um roteador faça distinção de pacotes pertencentes a diferentes classes de tráfego [Kurose and Ross, 2002]. A classificação é a ação de examinar o cabeçalho do pacote, seguindo alguns critérios como, por exemplo, número da porta, endereço de destino etc., para identificar o fluxo ou a aplicação a qual os pacotes pertencem. Tal identificação visa permitir a tomada de ações que modifiquem o comportamento dos pacotes. A classificação associada à funcionalidade de marcação permite, por exemplo, que os pacotes tenham um tratamento diferenciado nas filas de saída.

Um exemplo da importância da classificação poderia ser o seguinte: dois pacotes chegam à fila do roteador, um pacote de vídeo sensível ao atraso e outro de FTP tolerante ao atraso. O roteador precisa diferenciar entre esses pacotes de tal forma a dar o tratamento devido a cada tipo de pacote, nesse caso o pacote de vídeo, sensível ao atraso.

2.4 MARCAÇÃO

Classificação e marcação de pacotes são funções distintas, mas estão fortemente associadas. Para que aconteça a marcação, alguma classificação tem que ser realizada. A maneira especifica como essas funções operam dependem da estratégia de QoS usada [Martins et al., 2003].

Na estratégia DiffServ os pacotes classificados são marcados no campo ToS (*Type of Service*) do cabeçalho IP com o código DSCP (*Differentiated Services Code Point*). Esta marcação permite que um roteador trate pacotes de forma diferente de acordo com as classes de tráfego.

2.5 POLICIAMENTO

O policiamento no contexto de QoS consiste em um conjunto de regras associadas a serviços e descreve um conjunto de condições que devem ser satisfeitas para que uma determinada ação seja tomada baseada nessa condição [Martins et al., 2003].

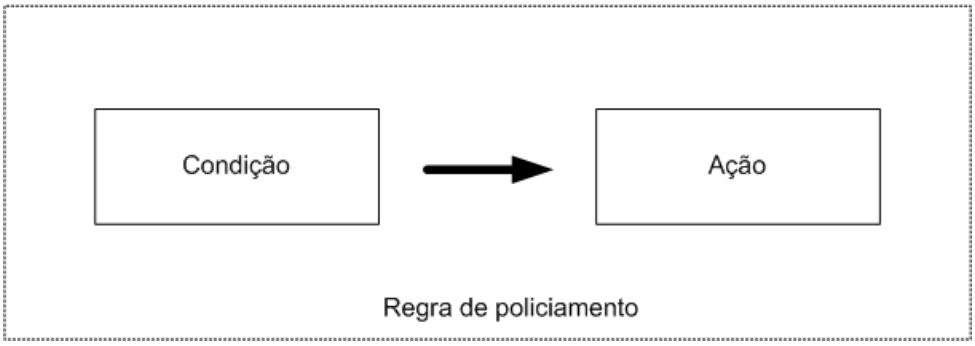

Figura 4: Para cada condição uma ação deve ser tomada

O policiamento monitora o tráfego da rede para verificar se esse tráfego está conforme com o SLA. Os pacotes que estão fora das condições definidas no contrato são processados conforme a regra de policiamento definida. Geralmente o policiamento é feito na borda da rede.

Um exemplo de regra de policiamento poderia ser: Marcar os pacotes que estão fora da taxa acertada no SLA para o descarte. A condição é "Pacotes fora da taxa acertada no SLA" e a ação é "marcar para o descarte" [Martins et al., 2003].

2.6 SUAVIZAÇÃO DE TRÁFEGO (*Traffic Shapping*)

A função de suavização (T*raffic Shapping)* fornece um mecanismo para controlar o volume de tráfego injetado na rede e a taxa em que este está sendo enviado. Assim sendo, a função de suavização efetivamente regula os fluxos de rede. Quando aplicado a um fluxo de

pacotes, a suavização de tráfego permite entre outras possibilidades, controlar a rajada de pacotes.

O esquema do balde de fichas (*token bucket*) é frequentemente usado para implementação da função de suavização nos roteadores, este esquema é explicado a seguir.

2.6.1 Balde de Fichas (*Token Bucket*)

A figura 5 ilustra o esquema do balde furado. Fichas representam permissões e são geradas a uma taxa de "t" permissões por segundo. Fichas são acumudadas no balde até o mesmo esgotar sua capacidade de armazenamento com tamanho "f" permissões. Caso o balde esteja cheio, novas fichas geradas são ignoradas. Para um pacote ser aceito na rede é necessário que o balde possua fichas, ou seja, permissões, que serão "consumidas"/removidas ao liberarem a transmissão de um dado pacote!

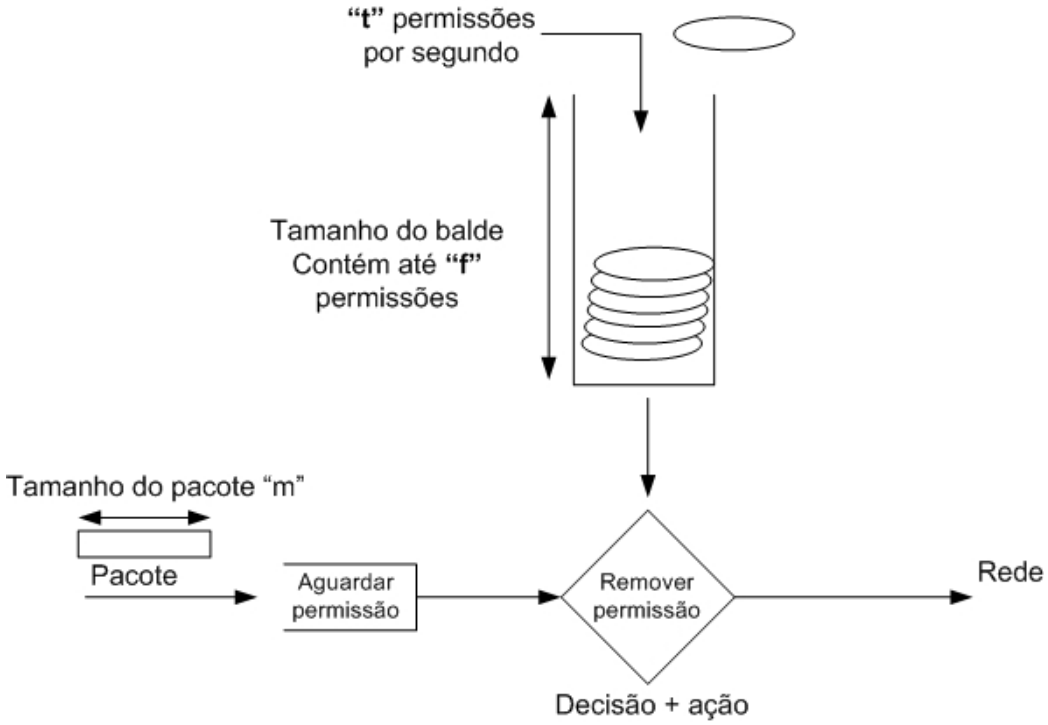

Figura 5: Balde de fichas (*token bucket)*

Segue um exemplo de como o esquema do balde furado é usado para fazer a suavização de um fluxo de pacotes. O tamanho máximo de uma rajada para um fluxo de pacotes que terá permissão para ser injetado na rede é determinado pelo tamanho do balde, ou seja, no máximo “f” pacotes. A taxa de geração de novas permissões é de “t” permissões por segundo, assim sendo, o número máximo de pacotes que serão injetados na rede para qualquer intervalo de tempo “T” é de “tT” + “f”. Um pacote que e não encontra fichas no balde para ser “liberado” pode ser submetido a diversas ações como por exemplo: aguardar por permissões, ser descartado etc.

2.7 ESCALONAMENTO DE FILAS

Quando os pacotes chegam no roteador, após serem processados, são repassados através dos elementos de comutação para as filas de saída e esperam até que chegue a sua vez de serem transmitidos num determinado enlace. O modo como esses pacotes são selecionados para a transmissão é conhecido como escalonamento de filas. Os principais objetivos dos mecanismos de escalonamento de filas nos roteadores IP são: compartilhar banda de uma forma justa, para garantir parâmetros de QoS como largura de banda e atraso para aplicações específicas e reduzir o *jitter* [Martins et al., 2003].

Existem diversos algoritmos de escalonamento de filas usados pelos roteadores, e, em termos desta dissertação, serão apresentados as opções de domínio público (abertas) e disponíveis em GNU/Linux, o ambiente utilizado na implantação resultante deste trabalho. As opções discutidas são:

- *First In First Out (FIFO)*
- *Priority Queueing (PQ)*
- *Class Based Queueing (CBQ)*
- *Weighted Fair Queueing (WFQ)*

2.7.1 *First In First Out* (*FIFO*)

No FIFO, os pacotes que chegam no roteador são enfileirados se houver algum pacote sendo transmitido no momento. A ordem de transmissão dos pacotes será de acordo com o instante de chegada, o primeiro pacote a chegar vai será o primeiro pacote a ser transmitido. Se a taxa de chegadas for maior que a taxa de transmissão, o *buffer* da fila se encherá e os pacotes que chegarem na fila e a encontrarem cheia serão descartados, pois não haverá mais espaço no *buffer*.

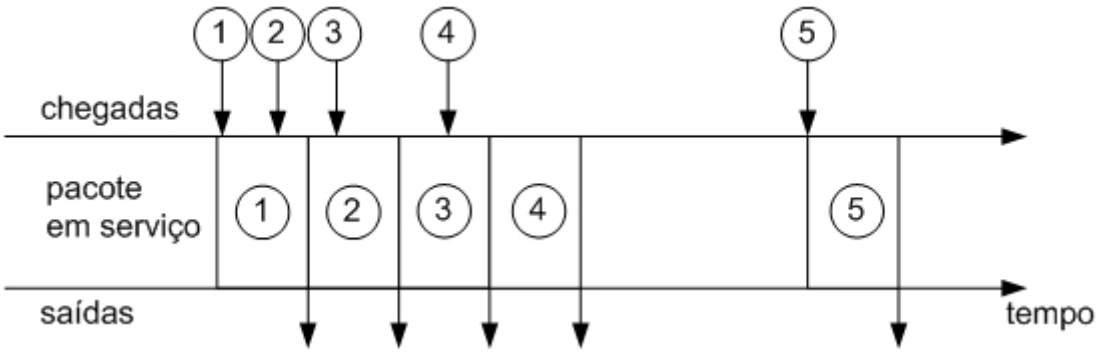

Figura 6: Algoritmo de escalonamento *First In Fist Out* (FIFO)

A figura 8 ilustra um exemplo de fila FIFO em operação. Os pacotes são representados pelos círculos numerados. A ordem de chegada é representada pelos números 1 - o primeiro a chegar a 5 - o último a chegar. À medida que os pacotes vão chegando eles vão sendo transmitidos em ordem. A figura indica um espaçamento entre a transmissão do pacote 4 e a chegada do pacote 5. Nesse espaço de tempo o enlace está ocioso, e assim que o pacote 5 chega ele é imediatamente transmitido.

Os benefícios e limitações de se usar o algoritmo de escalonamento FIFO são explicados por [Semeria, 2001].

Benefícios:

- O FIFO oferece baixa carga computacional quando comparado com outros algoritmos de escalonamento, é ideal para roteadores baseado em software, como é o caso da rede protótipo experimental montada.

- O comportamento do FIFO é previsível, pacotes não são ordenados e o máximo atraso que um pacote sofre é determinado pelo comprimento da fila.

Limitações:

- Uma simples fila FIFO não permite que roteadores organizem pacotes que estejam em sua memória (armazenados na fila) assim não pode priorizar pacotes em relação a outros.

- Uma fila FIFO prejudica todos os fluxos da mesma maneira, ou seja, quando há um congestionamento todos os pacotes de todos os fluxos sofrerão atrasos aumentando esse congestionamento. Como resultado o FIFO pode aumentar o atraso, a variação de atraso (*jitter*) e as perdas para as aplicações de tempo real que estão atravessando uma fila FIFO.

- Durante os períodos de congestionamentos o enfileiramento FIFO beneficia os fluxos UDP em relação aos fluxos TCP. Quando os pacotes de fluxos TCP se perdem as aplicações baseadas em TCP reduzem suas taxas de transmissão. Já quando ocorrem perdas nos pacotes UDP, as aplicações baseadas em UDP,

> podem não se preocupar com isso e continuam transmitindo pacotes normalmente. Como as aplicações TCP baixam suas taxas de transmissão para se adaptar às mudanças nas condições da rede, as filas FIFO podem aumentar o atraso dessas aplicações como também a variação de atraso (*jitter*) e a redução de banda para a aplicação TCP que está atravessando a fila.

Uma rajada de um fluxo pode consumir completamente o espaço do *buffer* da fila FIFO, podendo prejudicar todos os outros fluxos, normalizando somente depois que toda a rajada for servido pelo algoritmo de escalonamento. Isto pode resultar em atraso, no aumento de variação de atraso (*jitter*) e perda para outros fluxos bem comportados que estão atravessando a fila.

2.7.2 *Priority Queueing (PQ)*

No PQ, os pacotes que chegam são examinados, classificados e enfileirados em classes de prioridade. Neste esquema de escalonamento, enquanto houver pacotes na fila de maior prioridade, os mesmos serão transmitidos. As demais filas de classes de menor prioridade não são servidas enquanto houver pacotes nas filas mais prioritárias. De maneira geral, os pacotes com maior prioridade são transmitidos primeiro, onde cada classe de prioridade tem sua própria fila.

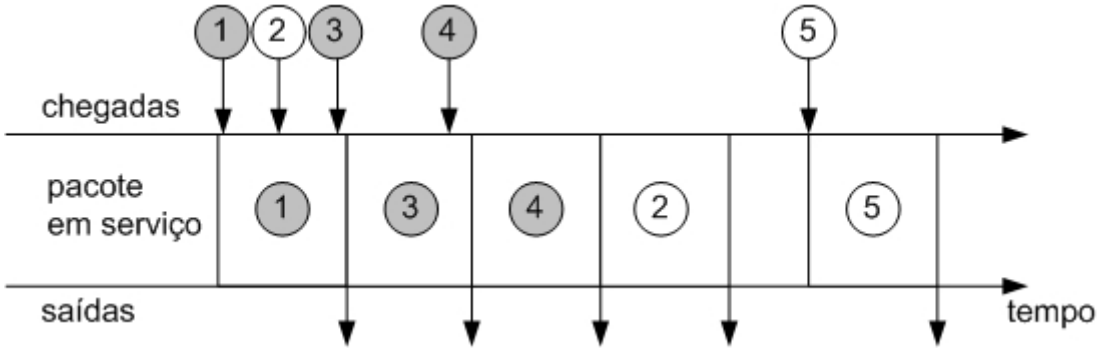

Figura 7: Algoritmo de escalonamento PQ (*Priority Queue*)

A figura 9 ilustra um exemplo de PQ em operação. Os pacotes de cor cinza são os de alta prioridade e os brancos são de baixa prioridade. Ao chegar, o pacote 1 é imediatamente transmitido, apesar de o pacote 2 ter chegado antes do pacote 3, ele tem menor prioridade e espera os pacotes de maior prioridade serem transmitidos primeiro. No caso, ambos os pacotes 2 e 5, só serão transmitidos depois que a fila de maior prioridade não tenha nenhum pacote para ser transmitido.

2.7.3 *Class Based Queueing (CBQ)*

O CBQ é uma variação do PQ onde várias filas de saída podem ser definidas. Pode-se também definir a prioridade de cada fila e a quantidade de banda passante utilizada medida em *bytes* para cada turno de serviço [Armitage, 2000]. Assim sendo, cada tipo de tráfego pode ter uma porção da banda disponível.

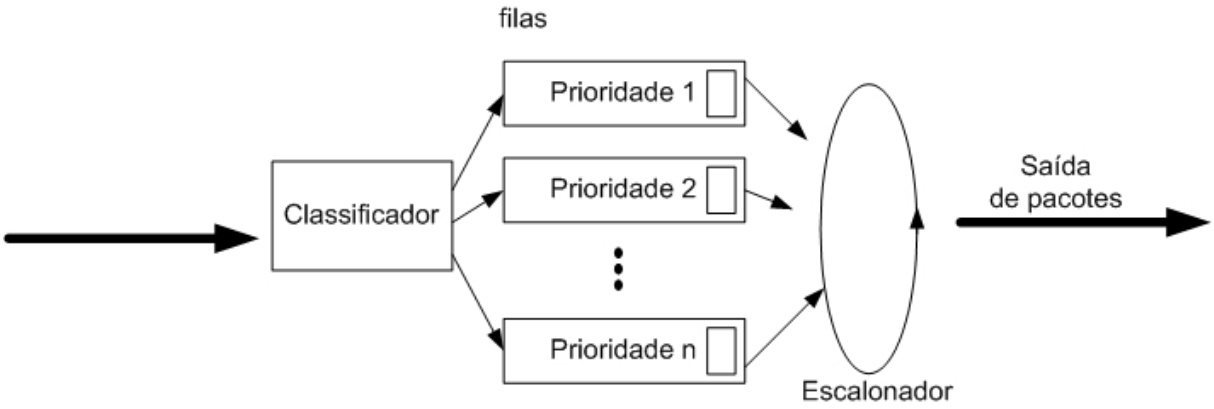

Figura 8: Algoritmo de escalonamento CBQ

Na figura 10 os pacotes são classificados e encaminhados para as filas, de acordo com sua prioridade. O escalonador, a título de exemplo, pode ser configurado para servir 400 *bytes* na fila com prioridade 1, 200 *bytes* na fila com prioridade 2 e assim por diante. O escalonador começa a servir a fila de prioridade 1 até completar 400 *bytes*, passa para a fila seguinte até servir os 200 *bytes* e assim por diante.

Desta forma o tráfego que foi categorizado e classificado para ser enfileirado em diversas filas tem boas chances de ser transmitido sem que uma latência significativa seja induzida, permitindo ao sistema evitar escassez de buffer [Armitage, 2000].

2.7.4 *Weighted Fair Queueing (WFQ)*

O mecanismo de escalonamento WFQ procura algoritmicamente prover comportamento previsível na entrega de pacotes e assegurar que os fluxos não tenham falta total de buffers [Huston and Ferguson, 1998].

Nesta disciplina, os pacotes que chegam são examinados e classificados em classes de prioridade, o escalonador alterna o serviço entre as filas transmitido um pacote em cada fila. Porém cada classe pode receber uma quantidade de serviço diferenciado em qualquer intervalo de tempo. Para cada classe é atribuído um peso “p”, o WFQ garante que a classe “i”

receberá uma fração de serviço igual a $pi/(\sum pj)$ onde o denominador é a soma de todas as classes que também têm pacotes enfileirados para transmissão. Considerando um enlace com taxa de transmissão R, a classe i terá uma vazão no mínimo R * $pi/(\sum pj)$. A figura 11 ilustra esse processo. Pacotes ao chegarem no classificador são classificados e colocados em suas devidas filas de acordo com a prioridade. O escalonador vai escolhendo os pacotes para ser transmitidos de forma cíclica obedecendo ao peso configurado de cada classe.

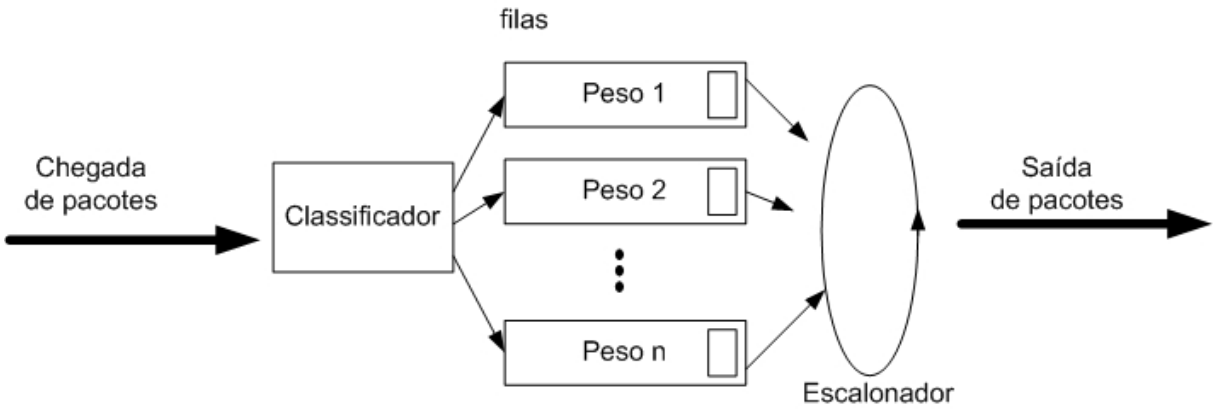

Figura 9: Algoritmo de escalonamento WFQ

3. A ARQUITETURA DiffServ E QUALIDADE DE SERVIÇO NO GNU/LINUX

A principal motivação para o desenvolvimento da arquitetura do DiffServ foi a implementação de um serviço de diferenciação para agregados de tráfego de uma forma flexível e escalável na Internet em relação ao IntServ. Um agregado de tráfego se caracteriza pelo agrupamento de pacotes que recebem um mesmo código, o DSCP (*Differentiated Service Code Point*). A arquitetura DiffServ é flexível em relação ao IntServ, pois não define serviços específicos apenas classes de serviços. Os pacotes são categorizados e alocados às classes de serviço do DiffServ através de marcação do DSCP no campo DS (*Differentiated Services*) do cabeçalho dos pacotes IP. Assim sendo, os pacotes que pertencerem a uma determinada classe receberão o mesmo tratamento por todos os roteadores DiffServ que eles atravessarem ao logo dos seus caminhos (*path*) entre origem e destino. Os recursos da rede (banda, escalonamento, outros) são alocados às classes de serviço do DiffServ de acordo com políticas de provisionamento de serviço [Blake et al., 1998].

O DiffServ é uma ferramenta eficiente para administradores gerenciarem os enlaces de rede. Dentre vários fluxos de pacotes circulando na rede pode-se dar prioridade aos mais importantes, como, por exemplo, as aplicações de VoIP e de vídeo.

Este capítulo também descreve algumas das tecnologias utilizadas para a implantação de uma rede protótipo experimental com a finalidade de implantar e avaliar os mecanismos de qualidade de serviço (QoS) no ambiente GNU/Linux. Em resumo, são apresentadas as funcionalidades utilizadas para a implantação das funções de roteamento com QoS (marcação, filtragem, escalonamento, outras) em computadores pessoais (PCs) dedicados à função de roteadores.

De forma a identificar a nomenclatura utilizada, a figura 12 ilustra um domínio DiffServ e seus principais componentes.

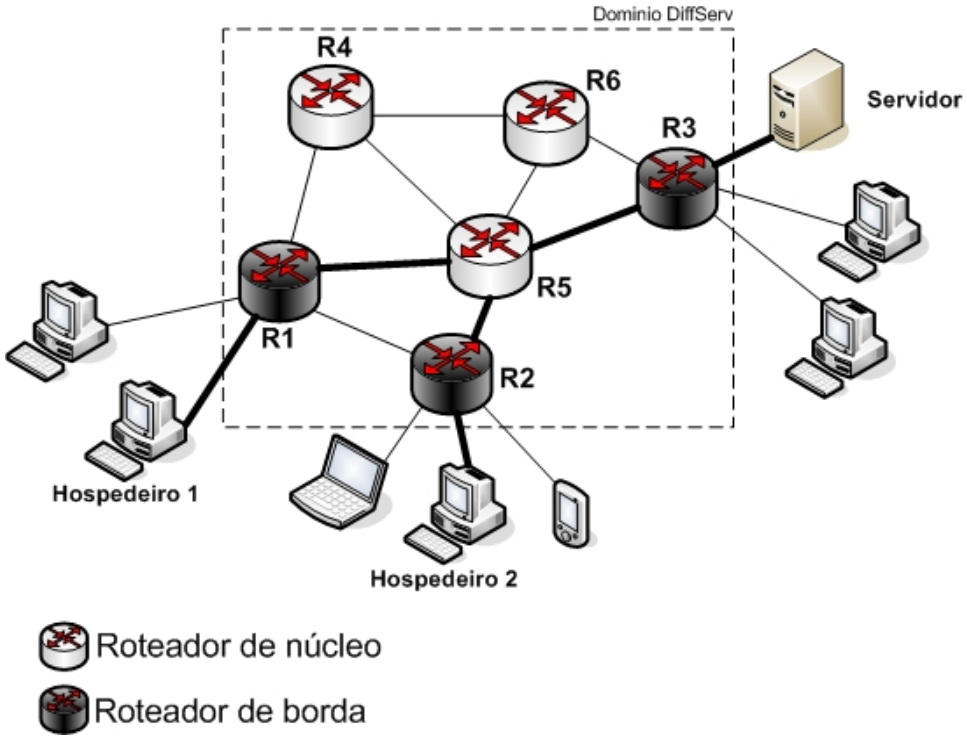

Figura 10: Dominio DiffServ

Os roteadores R1, R2 e R3 são os roteadores de borda e, no contexto DiffServ, eles desempenham as funções de classificação e marcação dos pacotes e o condicionamento de tráfego, Os roteadores R4, R5 e R6 são os roteadores de núcleo e, no DiffServ, eles são responsáveis por rotear os pacotes de acordo com um comportamento por salto (*Per Hop Behavior - PHB*) associado à classe do pacote.

Os agregados de tráfego são elementos chave para a escalabilidade do DiffServ. Como o DiffServ agrupa os pacotes em um pequeno número de classes, ele trata agregados de tráfego e não cada fluxo individualmente, resolvendo a questão da escalabilidade quando comparado ao IntServ. Os pacotes marcados com o mesmo DSCP receberão o mesmo tratamento (*Per Hop Behavior* – PHB) no roteador DiffServ. A marcação dos pacotes geralmente é feita no roteador de borda do domínio DiffServ, mas pode também ser feita nos

hospedeiros ou em um elemento intermediário ou de acesso (*gateway*) específico (VoIP, outros) [Martins et al., 2003].

Em seguida, descreve-se alguns aspectos técnicos de implantação relevantes para a arquitetura DiffServ.

3.1 DSCP (*Differentiated Services Code Point*)

O DSCP [Jha and Hassan, 2002] é um código padrão de 6 bits, originalmente definido nas RFCs 2474 e 2475 alocado no campo DS (*Differentiated Services*), uma atualização do campo ToS (*Type of Service*) do cabeçalho IP. O campo ToS foi redimensionado em duas partes:

- 6 bits – campo DS, que acomoda o código DSCP
- 2 bits – não são usados.

A figura 14 ilustra o campo DS.

O código DSCP é a marca que os pacotes recebem para possibilitar que os mesmos recebam um tratamento específico, o comportamento por salto *(Per Hop Behavior*), em cada roteador do domínio DiffServ. Como o campo DS tem 6 bits, até 64 classes podem ser suportadas. A figura 13 ilustra o cabeçalho do IPV4 e o campo tipo de serviço (ToS) que no contexto DiffServ é chamado de campo DS e é usado para armazenar o DSCP.

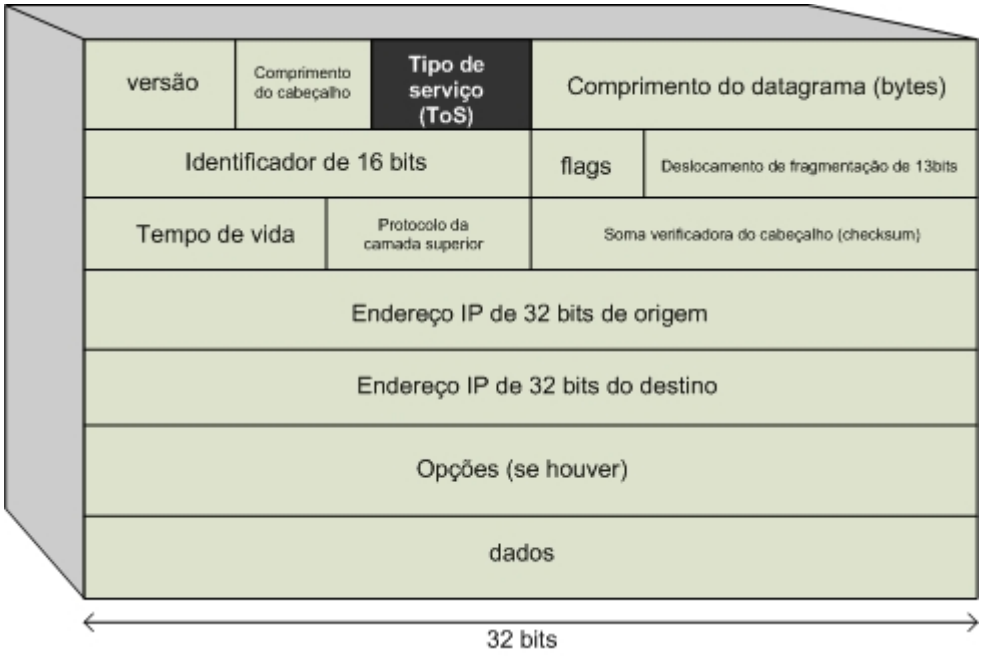

Figura 11: Cabeçalho IPV4

Figura 12: Código DSCP

3.2 COMPORTAMENTO POR SALTO *(Per Hop Behavior – PHB)*

No roteador, os serviços aplicados aos pacotes são chamados de PHB ou comportamentos. O PHB define o tratamento que os pacotes recebem quando marcados com um determinado DSCP. O PHB pode ser definido em termos de recursos, de prioridade sobre outros PHBs ou por características de tráfego, por exemplo, atraso ou perda.

Existem algumas padronizações para o PHB, mas nada impede a implantação de um PHB especifico de acordo com as necessidades de cada rede através do mapeamento local específico do DSCP.

O PHB padrão corresponde ao valor 000000 e equivale ao serviço tradicional de melhor esforço (*best effort)*, assegura compatibilidade com o encaminhamento aos roteadores não preparados para o DiffServ. O serviço de melhor esforço corresponde ao comportamento padrão aplicado aos pacotes. Além desse PHB padrão, a RFC 2744 define mais dois PHBs.

- Encaminhamento expresso (*Expedited Forwarding - EF*)
- Encaminhamento assegurado (*Assured Forwarding - AF*)

3.2.1 Encaminhamento expresso (*Expedited Forwarding - EF*)

O PHB de encaminhamento expresso representa o melhor serviço em um domínio DiffServ. Ele assegura baixa perda, baixo atraso, baixa variação de atraso (*jitter*), garantia mínima de banda. Este nível de serviço emula um circuito virtual dedicado dentro de um domínio DiffServ [Durand et al., 2001]. O PHB EF é usado para aplicações que exigem baixo atraso, baixa variação de atraso e banda fim-a-fim assegurada tais como aplicações de Voz sobre IP *(Voice Over IP - VoIP*) e aplicações de vídeoconferência. Os níveis desejados desses parâmetros só são obtidos fazendo com que o tráfego agregado desse PHB enfrente pouca ou nenhuma fila. Para isso, o PHB EF precisa garantir que a taxa de chegada de um agregado seja menor que a taxa de saída configurada para o mesmo.

3.2.2 Encaminhamento assegurado (*Assured Forwarding - AF*)

O PHB de encaminhamento assegurado é o mais flexível e o mais comum dentro de um domínio DiffServ. O *PHB* AF se divide em quatro grupos, chamados de classes. Cada classe pode ter três níveis de precedência de descarte e tem um tratamento preferencial sobre outras classes, permitindo ter níveis diferenciados de garantia de serviço a diferentes

agregados de tráfego. Administradores de rede podem fazer uso dessas classes para tratar seus diferentes tipos de tráfegos de maneira diferenciada dentro de um domínio DiffServ de acordo com seu SLA e requisitos de rede.

Para suportar a funcionalidade do PHB AF, cada nó no domínio DiffServ deve implementar alguma forma de alocação de banda para cada classe AF e alguma forma de priorização de filas para permitir o policiamento das classes [Durand et al., 2001].

Prioridade de tratamento ←

	Classe 1	Classe 2	Classe 3	Classe 4
Precedência De Descarte ↓	001010 (AF11)	010010 (AF21)	011010 (AF31)	100010 (AF41)
	001100 (AF12)	010100 (AF22)	011100 (AF32)	100100 (AF42)
	001110 (AF13)	010110 (AF23)	011110 (AF33)	100110 (AF43)

Figura 13: Representação das classes de serviço DiffServ e seus respectivos códigos DSCP

A figura 15 representa os valores DSCP para as classes AF. Os três primeiros dígitos representam a classe, os dois seguintes (4 e 5) representa a precedência de descarte para o caso de quando houver congestionamento na rede. Em caso de congestionamento os primeiros pacotes a serem descartados são da classe AFx3, depois os pacotes da classe AFx2 por fim os pacotes da classe AFx1. A precedência de descarte é usada para penalizar fluxos dentro de um mesmo agregado de tráfego que excedam a banda alocada à classe, os pacotes mal comportados, por exemplo, podem ser remarcados recebendo uma marca com maior precedência de descarte.

Em seguida serão apresentados os recursos existentes no ambiente operacional GNU/Linux, utilizado nessa dissertação, visando a implantação efetiva da qualidade de serviço baseada na arquitetura DiffServ.

3.3 QUALIDADE DE SERVIÇO NO GNU/LINUX

Esta seção ilustra a disponibilidade e as funcionalidades disponíveis no ambiente operacional GNU/Linux de forma a suportar a implantação da rede protótipo experimental.

Primeiramente é feita uma breve descrição do sistema operacional GNU/Linux, e em seguida são apresentadas funcionalidades do GNU/Linux como o controle de tráfego do Linux (*Linux Traffic Control)*. Posteriormente é apresentado o pacote de ferramentas IPRoute2 [Brown, 2003] que inclui as ferramentas **ip** que controla as configurações do IPV4 e IPV6 e **tc** que configura o controle de tráfego no GNU/Linux. Essas ferramentas possuem diversas funções, dentre outras pode-se citar:

- Configuração de políticas de roteamento;
- Configuração de NAT (*Network Address Translator*) e
- Configuração da qualidade de serviço (QoS).

Finalmente, esta seção identifica as alternativas existentes no GNU/Linux em termos dos algoritmos de escalonamento e ilustra a sua configuração para a qualidade de serviço (QoS) de forma genérica.

3.4 GNU/LINUX

O sistema operacional GNU/Linux é um sistema gratuito, multi-usuário, multitarefa e multiprocessado de livre distribuição. Apresenta inúmeras vantagens sobre os sistemas operacionais convencionais devido à sua alta estabilidade, baixo custo de implantação, flexibilidade, alto desempenho e robustez [McLagan, 2003].

Kernel é um software que possibilita a comunicação entre as aplicações do computador e o *hardware,* provendo serviços como gerenciamento de arquivos, de memória virtual, de dispositivos de entrada e saída (I/O) dentre outros. Um sistema operacional completo precisa mais do que um simples *kernel* para funcionar. Assim a organização GNU portou e desenvolveu diversas aplicações que combinadas com o *kernel* Linux passou a ser um verdadeiro ambiente operacional. Por isso que o ambiente operacional é chamado de GNU/Linux, termo adotado ao longo desse documento. A reunião dessas aplicações em conjunto com o *kernel* com configurações personalizadas e programas de instalação é conhecida, popularmente como “distribuição”.

Devido à sua alta flexibilidade, o GNU/Linux foi escolhido como o ambiente operacional a ser instalado nos roteadores da rede protótipo experimental. Contribuíram para esta escolha o fato de ser um sistema aberto onde pode-se fazer modificações em seu código fonte e devido a um amplo número de ferramentas voltadas para redes de computadores. No presente trabalho tem-se interesse especificamente em ferramentas com o propósito de emular as funcionalidades de um roteador habilitado com funcionalidades de qualidade de serviço (QoS) para a arquitetura DiffServ.

3.5 CONTROLE DE TRÁFEGO DO GNU/LINUX (*Linux Traffic Control*)

A funcionalidade de controle de tráfego no GNU/Linux foi introduzida a partir do *kernel* 2.2. Controle de tráfego, de maneira geral, é o conjunto de funcionalidades que processam pacotes IP com o objetivo de receber e transmitir esses pacotes, decidindo se esses pacotes serão aceitos ou rejeitados e também, controla a taxa de transmissão que esses pacotes irão receber. O controle de tráfego possui também funcionalidades para priorização de pacotes e decisão sobre a interface a ser usada para a transmissão de determinado pacote.

De maneira objetiva, as funcionalidades de controle de tráfego no GNU/Linux implementa um "roteador aberto" na medida em que implementa as funcionalidades básicas de roteamento com software livre [Dibona et al., 2005] como ilustrado na figura 16.

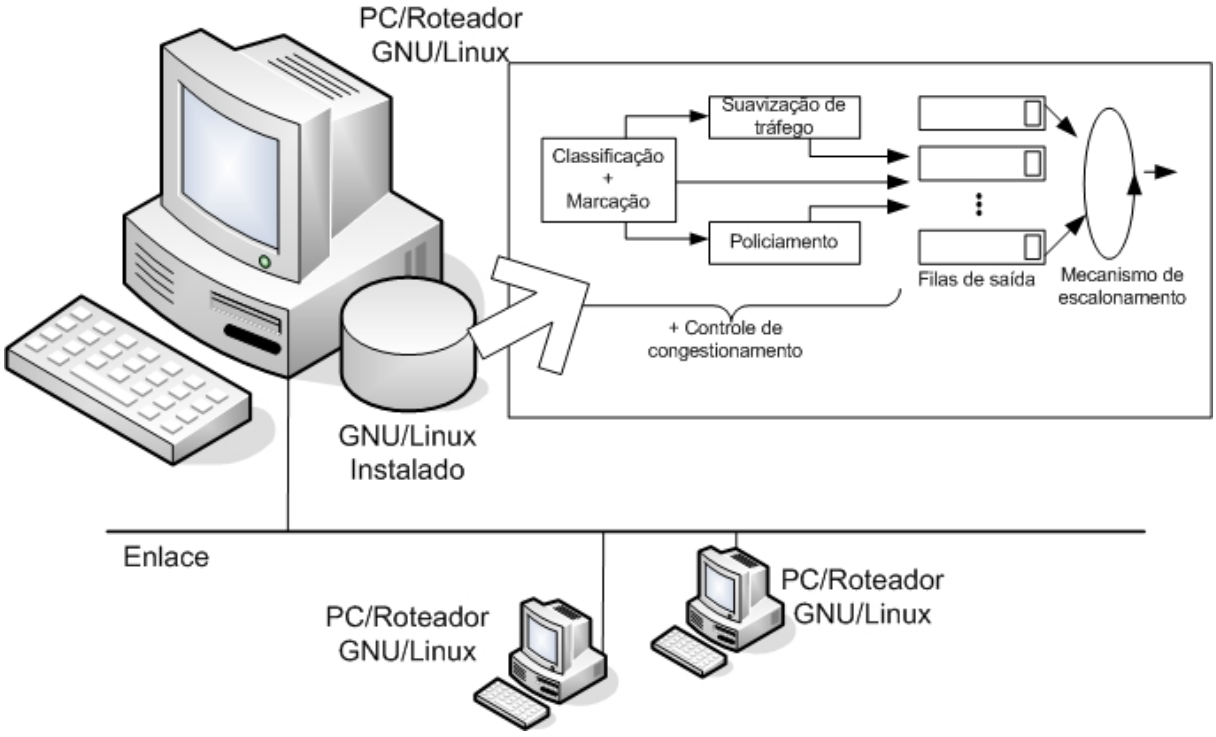

Figura 14: PC/Roteador GNU/Linux

Como descrito em [Brown, 2003], na maioria das vezes (Configuração padrão – "*default"*) o controle de tráfego consiste em uma fila simples que armazena pacotes recebidos e desenfileira esses pacotes usando a disciplina padrão de escalonamento FIFO do controle de tráfego GNU/Linux.

De forma a entender melhor o controle de tráfego e onde ele atua no GNU/Linux são mostrados a seguir os mecanismos envolvidos na manipulação dos pacotes, desde a sua recepção, até a transmissão quando os pacotes são efetivamente enviados para o meio fisico e deixam o roteador GNU/Linux.

Cada interface de rede instalada no GNU/Linux é controlada por um software de controle (d*river*) de rede que gerencia todas as funcionalidades da interface (*hardware*). Esse

driver é a ponte de comunicação entre o *hardware* da interface de rede e o *kernel* do sistema. Ele atua como um mecanismo de troca entre o módulo de funcionalidades de roteamento (rede) do GNU/Linux e a interface física. A figura 17 ilustra essa concepção.

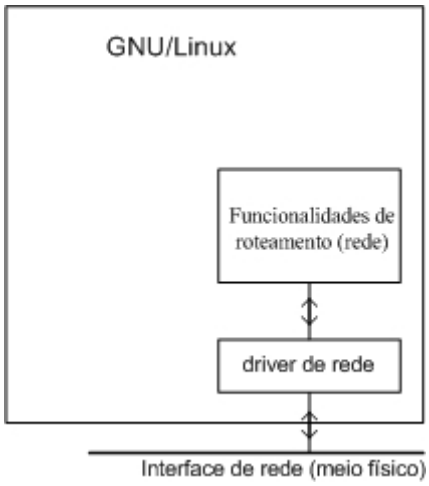

Figura 15: *Driver* de rede atuando como uma ponte de comunicação entre a interface de rede (meio físico) e as funcionalidades de roteamento do GNU/Linux.

Quando as aplicações do sistema querem transmitir um pacote, elas requisitam ao módulo de funcionalidades de roteamento (rede) que execute essa tarefa, esse módulo transmite o pacote através do *driver* de rede que por sua vez transmite esse pacote para a interface de rede.

Como o objetivo principal desse capítulo é explicar a implantação de roteadores usando GNU/Linux, e roteadores geralmente possuem múltiplas interfaces que encaminham pacotes de uma interface para outra e vice-versa, é necessário complicar o conceito ilutrado na figura 17 para o caso do PC / Roteador ter mais de uma interface. A figura 18 ilustra esse caso usando uma abordagem onde se tem o GNU/Linux com duas interfaces de rede funcionando com se fosse um roteador com duas portas. Considerando que um pacote vindo da rede1 deve ser encaminhado para a rede2, é feito o seguinte percurso:

- O pacote recebido entra pela interface de rede0 é entregue ao módulo de funcionalidades de roteamento do GNU/Linux pelo *driver* de rede da interface de rede0;
- O módulo de funcionalidades de roteamento do GNU/Linux processa o pacote e entrega para o *driver* de rede da interface de rede1, onde o mesmo transmite pela interface de rede1.

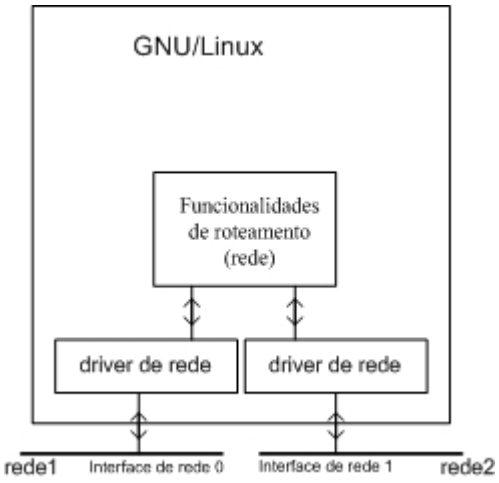

Figura 16: Múltiplas interfaces de rede atuando em conjunto com o *driver* de rede do GNU/Linux.

O princípio básico envolvido na implementação de qualidade de serviço (QoS) no GNU/Linux é ilustrado na figura 19 [Werner, 1999]. Como ilustra a figura 19, no GNU/Linux pacotes chegam no bloco denominado de "entrada de-multiplexador", o *kernel* então examina os pacotes recém chegados e determina se são destinados para este equipamento (destino é o roteador GNU/Linux). Se sim, os pacotes são enviados para a camada TCP, UDP, visando o processamento pelos protocolos de nível superior. Caso contrário, o *kernel* manda os pacotes para o bloco denominado "encaminhamento". O bloco de "encaminhamento" pode também receber pacotes gerados localmente pela camada superior. A tabela de roteamento é consultada, determinando o destino que o pacote irá seguir, o próximo salto. Assim sendo, o

"bloco de encaminhamento" corresponde em termos práticos às funcionalidades do protocolo IP implementadas no núcleo do sistema operacional. Depois disso, os pacotes são enfileirados para serem transmitidos em sua interface de saída. Neste ponto que o controle de tráfego do GNU/Linux entra em ação, podendo atrasar os pacotes, priorizar pacotes em relação a outros etc. Mais detalhes sobre as funcionalidades e atuação do controle de tráfego são mostrados a seguir.

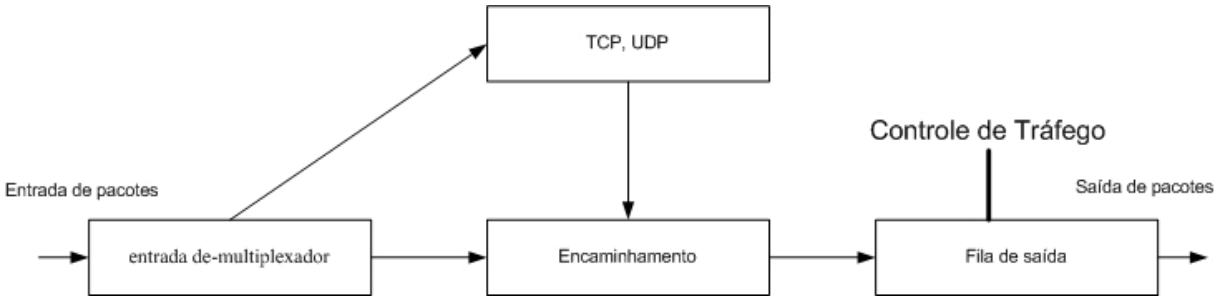

Figura 17: Controle de Tráfego do GNU/Linux [Werner, 1999].

Como os PCs usados na rede protótipo experimental são dedicados à função exclusivas de roteamento (roteadores), a camada superior não irá gerar tráfego localmente para ser injetado na rede passando pelo bloco de "encaminhamento", e os pacotes recebidos pelas interfaces de entrada não irão ser direcionados para as camadas TCP/UDP. O *kernel* do GNU/Linux desempenhará a função exclusiva de roteador, ou seja, encaminhar pacotes e eventualmente descartar pacotes seguindo o fluxo entrada de-multiplexador, encaminhamento, fila de saída como ilustrado na figura 20.

A figura 20 descreve de forma mais detalhada as funcionalidades acionadas quando um pacote é recebido via uma interface de entrada. A primeira funcionalidade aplicada aos pacotes de entrada é o policiamento. O policiamento descarta os pacotes que não são

conformes às especificações de tráfego definidas gerencialmente (SLA), como, por exemplo, tráfego de uma aplicação sendo recebido além da taxa configurada. Depois do policiamento os pacotes podem ter dois destinos:

- Podem ser encaminhados para uma outra rede, direcionados para uma outra interface, ou;
- Podem ser enviados para a camada superior na pilha de protocolo (transporte como UDP ou TCP).

As camadas superiores também podem gerar pacotes endereçados para elas mesmas e entregar esses pacotes para a camada abaixo para desempenhar tarefas como encapsulamento, roteamento e transmissão. O bloco "encaminhamento", equivalente ao protocolo IP, desempenha funções de roteamento como escolher a interface de saída para os pacotes, seleção do próximo nó, encapsulamento, dentre outras. Quando o bloco "encaminhamento" conclui seu processamento, os pacotes são enfileirados nas suas respectivas interfaces de saída. Neste ponto o controle de tráfego influencia no processamento dos pacotes. O controle de tráfego decide se os mesmos vão ser enfileirados ou descartados (ex. a fila pode alcançar certo limite ou o tráfego pode exceder alguma taxa limite). O controle de tráfego também pode decidir em qual ordem os pacotes vão ser transmitidos (priorização de tráfego), pode atrasar a transmissão dos pacotes (limitar a taxa de algum tráfego mal comportado). Uma vez que o controle de tráfego libera o pacote para ser transmitido, o *driver* da interface envia o pacote para a rede física.

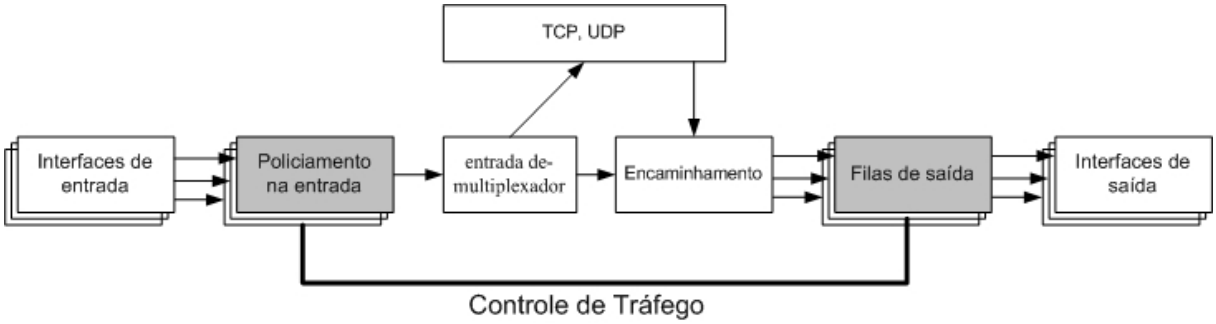

Figura 18: Controle de tráfego atuando no GNU/Linux [Werner, 1999]

Ainda com relação à figura 20, pode-se observar que o controle de tráfego do GNU/Linux atua nos blocos cinza, no "policiamento na entrada" e nas "filas de saída". O primeiro ponto que os pacotes sofrerão um controle é no bloco "policiamento na entrada" podendo descartar os pacotes não conformes às especificações de tráfego definidas gerencialmente (SLA) e limitar a taxa de entrada dos mesmos. O segundo ponto de controle a que os pacotes são submetidos é no bloco "filas de saída". Aqui os pacotes poderão ser enfileirados, descartados ou priorizados em relação a outros de acordo com a política implementada. Esses blocos são chamados de controle de tráfego do código do *kernel* do GNU/Linux (*Traffic Control Code of the Linux Kernel*) [Werner, 1999].

Os componentes principais no código do controle de tráfego do *kernel* do GNU/Linux consistem basicamente em:

- Policiadores.
- Filtros;
- Classes;
- Algoritmo de Escalonamento;

A seguir é explicada a sintaxe da ferramenta **tc** [Mortensen, 2000]**,** usada para configurar os componentes de QoS no Linux e cada um desses componentes detalhando como

eles funcionam e exemplificando cada componente e como eles interagem para fornecer qualidade de serviço (QoS) para as aplicações num roteador GNU/Linux de código aberto.

3.6 FERRAMENTA TC (*Traffic Control*)

Nesta seção é explicado o funcionamento da ferramenta **tc** usada para configurar os componentes no código do controle de tráfego do *kernel* do GNU/Linux.

A ferramenta **tc** faz parte do pacote Iproute2, utilizado para realizar diversas configurações de rede no GNU/Linux.

Todas as configurações feitas no *kernel* do GNU/Linux relacionada ao controle de tráfego são configuradas com a ferramenta **tc**. Combinada com diversos parâmetros pode-se selecionar as interfaces de redes a serem manipuladas, selecionar quais componentes do controle de tráfego sofrerão alteração como exemplo: qdisc (algoritmos de escalonamento), class (classes) ou filter (Filtros). A seguir é explicada a sintaxe da ferramenta **tc**.

Uso: tc [OPÇÕES] OBJETO {COMANDO | help}

Onde OBJETO pode ser: = {qdisc, class ou filter}.

OPÇÕES: = {-s[tatistics], -d[etails] ou -r[aw]}.

A seguir são mostrados alguns exemplos de configuração usando o **tc.**

Exemplo 1: Configuração de algoritmo de escalonamento HTB [Radhakrishnan, 1999] numa interface de rede.

tc qdisc add dev eth1 root handle 1:0 htb

tc qdisc add – Adiciona uma disciplina, poderia ser usado o parâmetro **del** para apagar uma disciplina.

dev eth1 – Especifica que está adicionando a disciplina na interface de rede eth1.

root – Significa "egresso" para o **tc**. Interfere nas filas de saída da interface. O termo ***root*** deve ser usado para isso. Um outro qdisc, o qdisc ***ingress*** poderia ser configurado na mesma interface para a funcionalidade de policiar o tráfego na entrada. A figura 20 mostra o bloco de "Policiamento na entrada" (parâmetro ***ingress***) e as "Filas de saída" (parâmetro ***root***).

handle 1:0 – O ***handle*** é usado pelo usuário na forma de maior:menor**.** O número menor para qualquer disciplina (qdisc) acompanhado com o termo ***handle*** deve ser sempre "0". Um atalho também aceito pelo **tc** para configurar a qdisc é "1:" onde o menor número é assumido como zero, nesse caso não especificado.

htb – A disciplina qdisc escolhida.

No exemplo 1 é mostrada a sintaxe do **tc** para configurar uma disciplina na interface eth1. No exemplo a seguir é mostrado como adicionar uma classe a uma classe pai já configurada.

Exemplo 2: Configuração de uma classe herdada do algoritmo de escalonamento HTB numa interface de rede.

tc class add dev eth1 parent 1:1 classid 1:3 htb rate 128kbit

tc class add – Adiciona uma classe, poderia também ser usado o parâmetro del para apagar uma classe.

dev eth1 - Indica que está adicionando a classe na interface de rede eth1.

parent 1:1 – Indica que que está adicionando uma classe herdada da disciplina identificada por 1:1.

classid 1:3 – É o identificador único da classe adicionada no formato (maior:menor). O número menor deve ser diferente de "0".

htb – Como cada classe filha deve ser do mesmo tipo da classe pai esse parâmetro é HTB, ou seja uma classe do tipo HTB.

rate 128kbit – é um parâmetro particular do HTB que faz com que a classe seja configurada com uma taxa de 128 kbps.

3.7 POLICIADORES

Os pacotes que chegam na interface de entrada podem sofrer alterações em seu comportamento, quem faz esse controle na entrada são os policiadores. Os policiadores podem descartar pacotes indesejados. Esse descarte de pacotes nas interfaces de entrada geralmente ocorre porque a taxa de chegada desses pacotes estão mais altas do que as configuradas no roteador. Por exemplo, um cliente de um provedor de serviço compra desse provedor um serviço de acesso onde terá 64 kbps de largura de banda. Esse cliente pode querer transmitir com uma taxa maior que 64 kbps, então os policiadores de entrada do provedor descarta os pacotes que estiverem acima da taxa contratada pelo cliente. A figura 20 mostra onde atuam os policiadores no controle de tráfego no GNU/Linux.

3.8 FILTROS

O filtro é um classificador que classifica e seleciona os pacotes baseados em suas características e envia-os para o algoritmo de escalonamento associado com essa característica. Essa classificação pode ser por:

- Endereço de destino;
- Endereço de origem;
- Tipo do protocolo de transporte (TCP ou UDP);
- Código DSCP

A figura 21 ilustra o conceito de algoritmo de escalonamento, classes e filtro do GNU/Linux. Um pacote ao chegar ao controle de tráfego do GNU/Linux é examinado primeiramente pelos filtros. O filtro então encaminha esse pacote de acordo com a regra de classificação configurada para as classes. Cada classe deve ter um algoritmo de escalonamento configurado dentro dela, que vai determinar o comportamento que o pacote receberá antes de ser encaminhado para o enlace de saída.

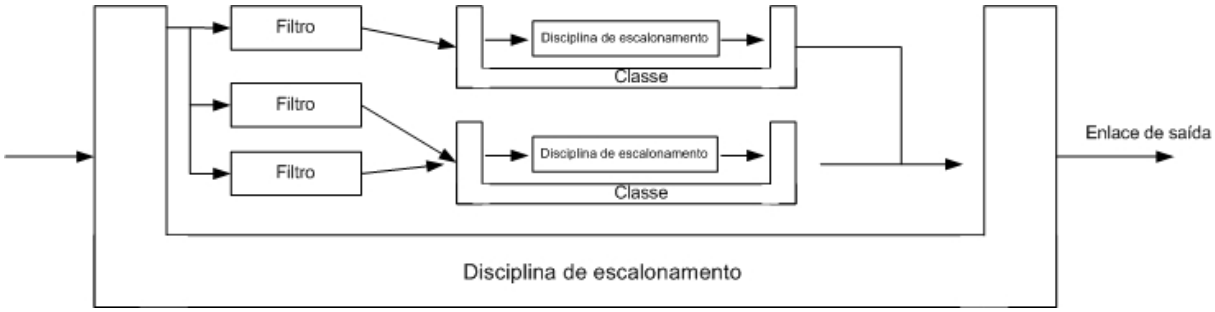

Figura 19: Algoritmo de escalonamento, filtro e classe

3.9 CLASSES

No controle de tráfego do GNU/Linux existe um tipo de fila que ao invés de armazenar pacotes, elas contêm outras filas dentro delas. Essas filas são chamadas de filas com classes. Esse conceito é explicado por [Mortensen, 2000] no exemplo abaixo:

Imagina-se um algoritmo de escalonamento baseada em prioridade com as seguintes propriedades:

- Cada pacote quando enfileirado é associado a uma prioridade. Por exemplo, a prioridade poderia ser baseada no endereço de destino do pacote. Considera-se que a prioridade seja um número entre 1 e 5.
- Quando o pacote é desenfileirado, ele sempre será selecionado de acordo com o número de prioridade mais baixa.

Uma maneira de se implementar esse algoritmo de escalonamento é fazer um algoritmo de escalonamento baseado em prioridade que contém dentro dela outros algoritmos de escalonamento numerados de 1 a 5. O algoritmo de escalonamento baseado em prioridades terá que fazer o seguinte:

- Quando um pacote for enfileirado, ela calcula o número da prioridade (entre 1 e 5), baseada em seu endereço de destino, então enfileira o pacote na disciplina de enfileiramento associada com esse número.
- Quando o pacote vai ser desenfileirado, ele sempre desenfileira de um algoritmo de escalonamento que não esteja vazia e que possua o menor número.

No GNU/Linux o conceito citado acima, é conhecido como classes, um algoritmo de escalonamento pode conter classes. No exemplo anterior, o algoritmo de escalonamento baseada em prioridade tem 5 classes. As classes normalmente não armazenam seus pacotes, elas usam outros algoritmo de escalonamento para armazenar seus pacotes.

3.10 ALGORITMO DE ESCALONAMENTO

Cada interface de rede do GNU/Linux tem um algoritmo de escalonamento associado à sua fila, que controla como os pacotes são processados visando a transmissão. Algoritmo de escalonamento são algoritmos que controlam como os pacotes que estão enfileirados nas filas das interfaces são servidos. A figura 22 ilustra um algoritmo de escalonamento FIFO simples.

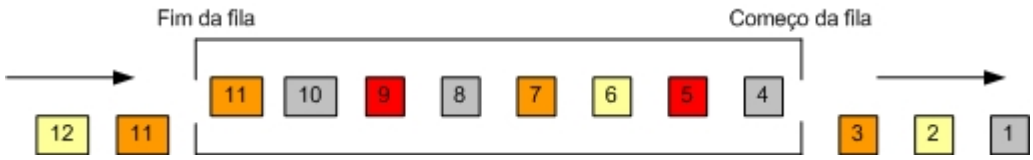

Figura 20: Fila com escalonamento FIFO

Existem diversos algoritmos de escalonamento do GNU/Linux [Radhakrishnan, 1999], os utilizados na rede protótipo experimental são:

- Class Based Queuing (CBQ)
- Hierarchical Token Bucket (HTB)
- First In First Out (FIFO)
- Priority Queuing (PQ)

A seguir são identificados os algoritmo de escalonamento utilizados na rede protótipo experimental para atender os requisitos de qualidade de serviço (QoS) das aplicações do projeto Infravida especificada na tabela 5. Na explanação de cada uma dessas disciplinas é mostrado como funciona a ferramenta **tc** utilizada para configurar essas disciplinas no

GNU/Linux. Foi utilizada uma combinação dos componentes principais do código do controle de tráfego do GNU/Linux (Algoritmo de escalonamento, Classes, Filtros e Policiadores) para atender os requisitos das classes de serviço do DiffServ que serão apresentadas na tabela 5 para as aplicações do projeto Infravida.

3.10.1 *First In First Out* (FIFO)

No algoritmo FIFO (Seção 2.7.1) todos os pacotes são tratados igualmente colocados em uma única fila em ordem de chegada, os primeiros a serem transmitidos são os primeiros que chegaram.

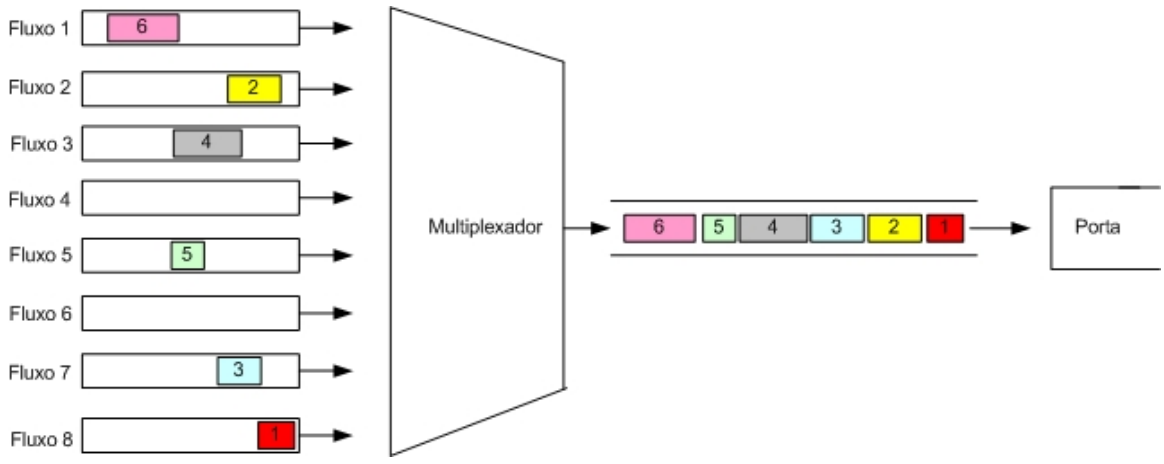

Figura 21: Algoritmo de escalonamento FIFO.

Para configurar uma disciplina FIFO no GNU/Linux deve-se usar o **tc** com a seguinte sintaxe:

tc qdisc add dev eth0 root pfifo limit 100

dev: indica a interface considerada

root: disciplina raiz

pfifo: o algoritmo configurado para a interface “dev”

limit: tamanho da fila

3.10.2 *Class Based Queuing* (CBQ)

O algoritmo de escalonamento CBQ (Seção 2.7.3) implementado no GNU/Linux é um algoritmo de escalonamento hierárquico de compartilhamento de enlace por fazer uso de um enlace físico para disponibilizar vários enlaces simulados de tamanhos menores. A seguir será explicada a configuração da disciplina no GNU/Linux e seus parâmetros.

Para explicar a implantação do CBQ no GNU/Linux é criado um exemplo hipotético levando em consideração os requisitos de QoS das aplicações de uma empresa “Y”. Primeiramente é explicado as necessidades de QoS das aplicações da empresa “Y”, depois é definida uma configuração baseado nessas necessidades e logo depois são apresentados os comandos da ferramenta **tc** para a configuração do CBQ no GNU/Linux.

A empresa “Y” possui um enlace com vazão de 1 Mbps e deseja dividir a banda desse enlace igualmente em 4 partes de 256 kbps para cada um de seus 4 departamentos (financeiro, pesquisa, vendas e administrativo). A empresa necessita que cada divisão tenha uma banda mínima garantida de 256 kbps e que o tráfego de um departamento não influencie no tráfego dos demais. As necessidades de QoS das aplicações por departamento são como segue na tabela 2.

Tabela 2: Necessidades das aplicações da empresa "Y"

Financeiro (256 kbps)	
VoIP	128 kbps
WWW	64kbps
outras aplicações	64 kbps
Pesquisa (256 kbps)	
VoIP	128 kbps
WWW	32 kbps
FTP	32 kbps
outras aplicações	64 kbps
Vendas (256 kbps)	
VoIP	128 kbps
WWW	64 kbps
Correio	32 kbps
outras aplicações	32 kbps
Administrativo (256 kbps)	
VoIP	128 kbps
WWW	32 kbps
Correio	32 kbps
Outras aplicações	64 kbps

A implantação CBQ para estes requisitos de aplicação é ilustrado na figura 24.

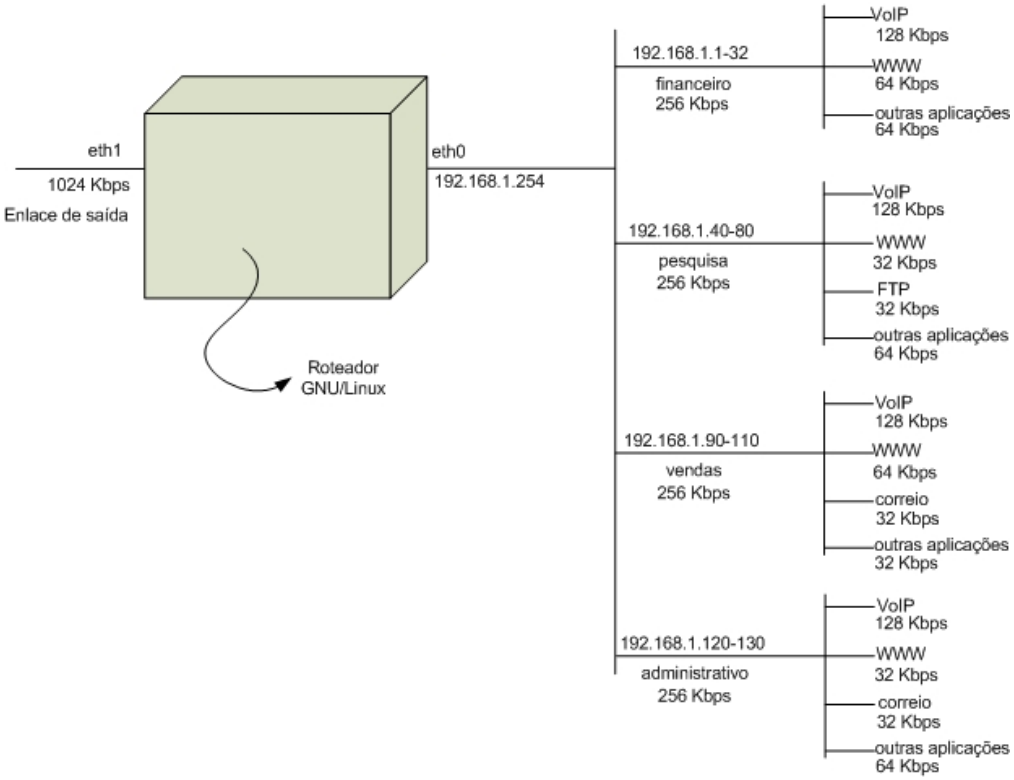

Figura 22: Configuração baseada nos requisitos de banda das aplicações e redes da empresa "Y"

A figura 25 ilustra a representação visual da configuração de algoritmo de escalonamento CBQ no GNU/Linux com base nas especificações de largura de banda feitas no diagrama da figura 24 no exemplo hipotético da empresa “Y”. Um algoritmo de escalonamento CBQ é configurado subdividido em várias classes.

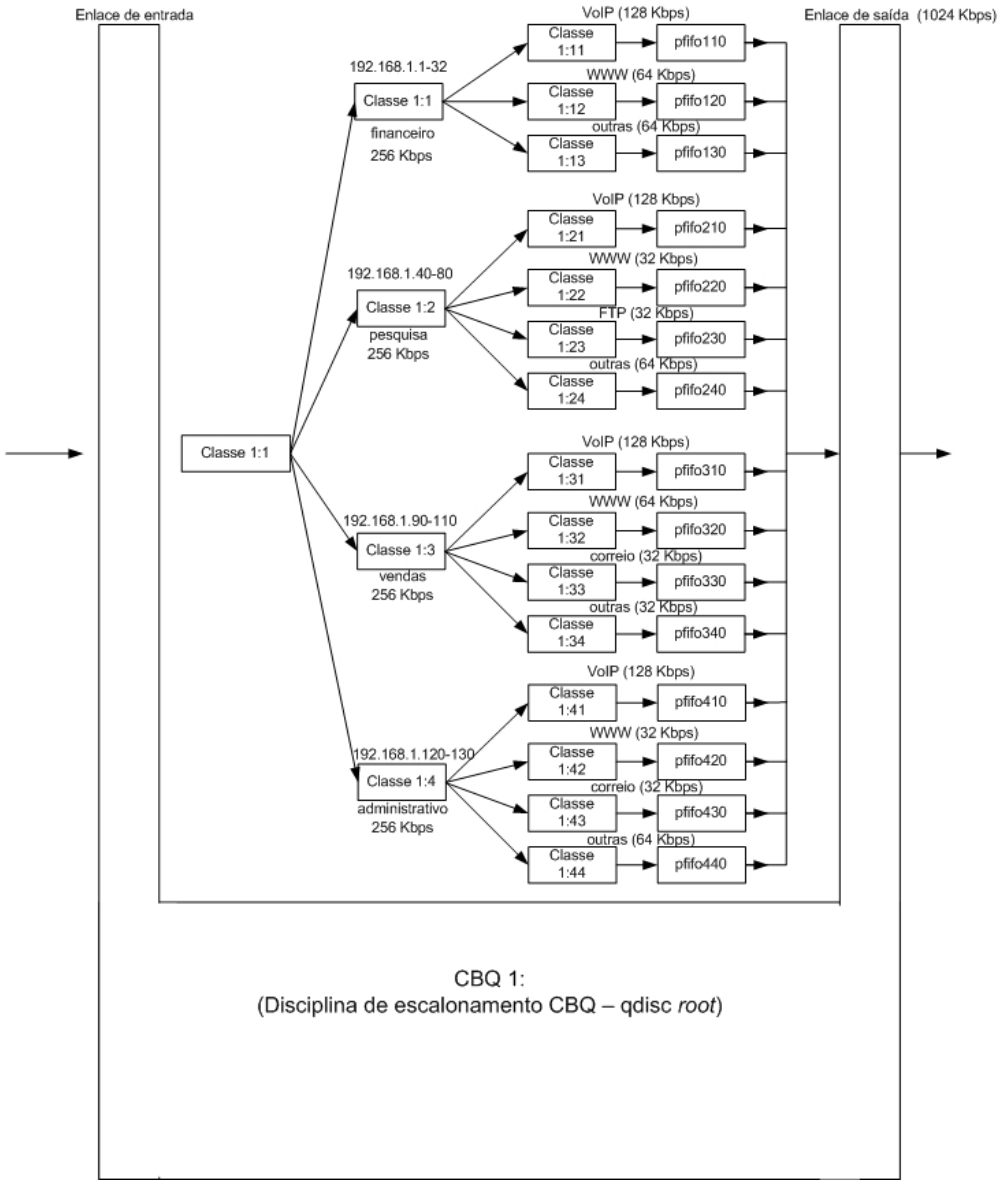

Figura 23: Diagrama de representação visual do algoritmo de escalonamento CBQ.

A seguir é mostrada a sintaxe da ferramenta **tc** para a configuração no GNU/Linux do algoritmo de escalonamento CBQ baseada na representação visual da figura 25.

tc qdisc add dev eth1 root handle 1: cbq bandwidth 1Mbit avpkt 1000

dev: indica a interface considerada, eth1.

root: disciplina raiz, onde as demais classes se derivarão.

handle 1: a identificação da disciplina.

cbq: o algoritmo configurado para a interface "dev"

bandwidth: A largura de banda máxima que a disciplina vai suportar/limitar.

avpkt: tamanho médio de um pacote, medido em *bytes*.

O comando acima cria uma disciplina CBQ raiz com banda total de 1Mbps na interface de rede eth1 identificada pelo parâmetro handle 1:

tc class add dev eth1 parent 1: classid 1:11 cbq bandwidth 1Mbit rate 256kbit avpkt 252 prio 1 bounded isolated

class: define uma classe, que na nomenclatura GNU/Linux é derivado do algoritmo de escalonamento.

parent 1: define quem é o algoritmo de escalonamento raiz.

classid: 1:1: a identificação da classe

rate: porção de banda que a classe será configurada, a soma dos das taxas de todas as classes não deve ultrapassar o parâmetro *bandwidth* da disciplina raiz.

prio 1: prioridade com relação a outras classes. 1 é a prioridade máxima.

bounded: a classe não "empresta" sua banda se não estiver utilizando na totalidade.

isolated: a classe não pede emprestado banda de outras classes se estiver precisando.

O comando acima cria uma classe derivada da raiz (parent 1:) com banda total de 256 kbps identificada pelo parâmetro classid 1:11.

tc qdisc add dev eth0 parent 1:11 handle 110: pfifo limit 10

handle 110: identificação da disciplina

pfifo: o algoritmo configurado

limit: tamanho da fila.

Toda classe do GNU/Linux deve ter associado um algoritmo de escalonamento qdisc. No comando acima um algoritmo de escalonamento pfifo identificado pelo comando handle 110 é associado com a classe 1:11.

3.10.3 *Hierarchical Token Bucket* (HTB)

O HTB [Devik, 2002] é um algoritmo de escalonamento especifico do GNU/Linux usada no compartilhamento de enlace entre as aplicações. Na verdade, essa disciplina faz uso de um enlace físico para simular diversos enlaces menores fazendo passar diversos tipos de tráfegos entre esses enlaces simulados com uma banda mínima garantida.

O HTB é uma variação do CBQ (Seção 2.7.3) e é conhecido por ser um algoritmo de escalonamento hierárquico de compartilhamento de enlace. Uma referência de algoritmo de escalonamento hierárquico de compartilhamento de enlace pode ser encontrada em [Floyd and Jacobson, 1995]. Segundo eles, um requisito para o algoritmo de escalonamento hierárquico de compartilhamento de enlace é compartilhar a banda total de um enlace entre múltiplos usuários, onde cada usuário da banda queira receber um compartilhamento desse enlace com

largura de banda garantida durante um possível congestionamento, mas quando a largura de banda estiver disponível, pode ser disponibilizada para outros usuários.

4. APLICAÇÕES MULTIMIDIAS COM QoS – CENÁRIO, IMPLANTAÇÃO DA REDE PROTÓTIPO EXPERIMENTAL (*TESTBED)*

Este capítulo define o cenário de implantação da estratégia de QoS, o projeto Infravida, descreve sua arquitetura, seus requisitos e onde o presente trabalho se encaixa neste projeto. Como descrito na introdução desta dissertação, o projeto Infravida foi criado para desenvolver um sistema de telemedicina incluindo áudio e vídeoconferência, um sistema de educação a distância e integrar todos esses sistemas através de uma rede IP que implementa um mecanismo de qualidade de serviço (QoS) para suportar as aplicações do projeto. A Figura 26 ilustra a arquitetura resumida do projeto Infravida [Lage, 2004].

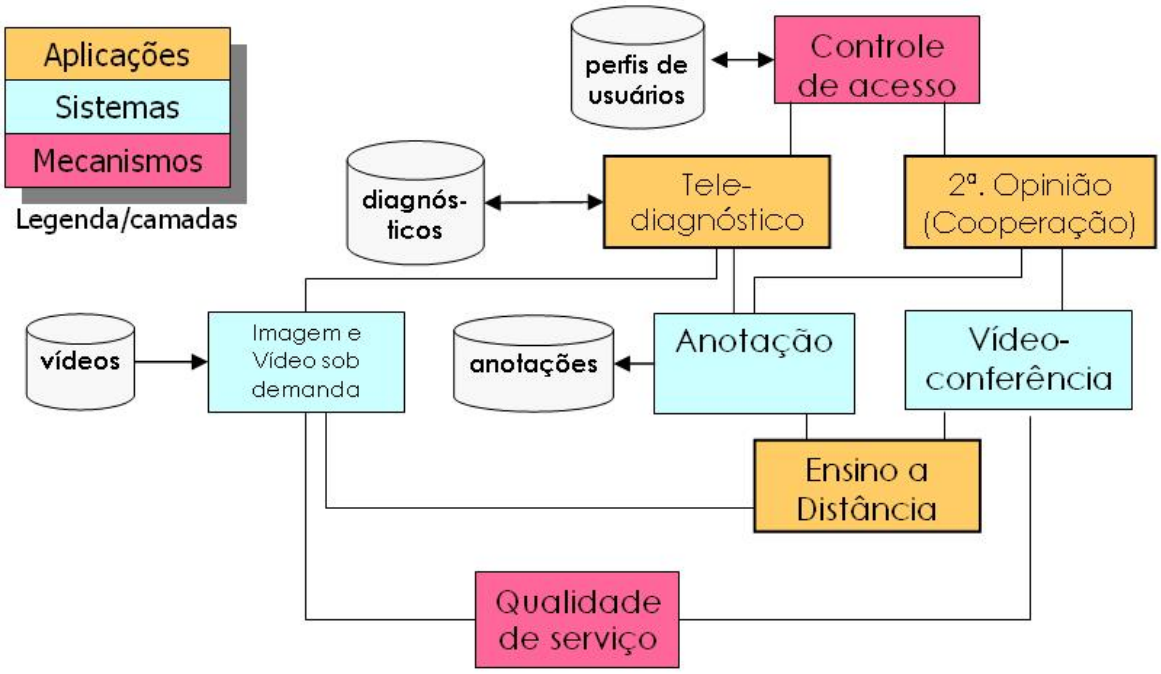

Figura 24: Arquitetura do projeto Infravida [Lage, 2004]

No projeto Infravida, pretendeu-se integrar a implementação de código aberto do padrão H.323 com suporte para transmissão de vídeo de baixa qualidade (H.261 e H263), o

OpenH323, os sistemas de distribuição de vídeo de alta qualidade DynaVideo (MPEG-2 para exames médicos), o sistema *HealthNet* [Barbosa, 2001] de suporte ao trabalho cooperativo de médicos com QoS em redes IP que suportam a qualidade de serviço, através da aplicação da arquitetura de DiffServ [Blake et al., 1998]. Ainda no contexto do projeto Infravida, o Ambiente Brasileiro de Aprendizagem (ABRA), fruto do projeto REMA-Salvador, complementa as funcionalidades das aplicações de telemedicina e funciona como portal de educação a distância (EAD).

A segunda opinião médica consiste em uma sessão de videoconferência pré-agendada, onde um médico especialista (consultor), o solicitante de uma segunda opinião e possivelmente outros colaboradores convidados analisam juntos o prontuário de um paciente e seus exames, inclusive imagens radiológicas e vídeos de procedimentos cirúrgicos ou exames de ultra-som, por exemplo. As sessões de videoconferência e imagem e vídeo sob semanda ocorrem simultaneamente, concorrendo por recursos da rede entre si e com outras aplicações [Lage, 2004].

O sistema *HealthNet* visa implantar um processo de cooperação entre grandes centros médicos especialistas no Recife e locais menos favorecidos da região metropolitana e interior de Pernambuco para a prestação de serviços de diagnóstico a distância. O *HealthNet* prevê três perfis para seus usuários: O solicitante que se caracteriza por qualquer profissional de saúde que precisa de um telediagnóstico, o consultor que é o especialista responsável pelo parecer médico num segundo diagnóstico e o colaborador, que pode ser outro médico convidado a participar do segundo diagnóstico. Solicitante, consultor e colaboradores podem manter uma comunicação assíncrona via e-mail com fins de telediagnóstico, mas o projeto prevê o agendamento de sessões de cooperação síncrona, que incluem a ativação de sistemas de videoconferência e de distribuição de imagens ou vídeo sob demanda associadas a uma

sessão específica de segunda opinião, haja ou não anotação. As sessões de videoconferência e imagem/vídeo sob demanda ocorrem simultâneamente [Lage, 2004].

O DynaVideo [Leite et al., 2001], desenvolvido no contexto do projeto NatalNet, é um serviço de distribuição de vídeo em tempo real utilizado na transmissão do vídeo de alta qualidade (MPEG-2 ou H.262) sendo altamente requerente de qualidade de serviço nas redes que o suportam por ser sensível ao atraso.

Segue uma breve descrição dos componentes da arquitetura do projeto Infravida [Lage, 2004].

- O serviço de videoconferência será utilizado para visualização de imagens ou vídeo onde existe uma necessidade de sincronização dos elementos utilizados entre os agentes de saúde, ou seja, na modalidade cooperação síncrona.

- O serviço de imagem/vídeo sob demanda será utilizado para visualização de imagens ou vídeo onde não haja necessidade de uma sincronização entre agentes de saúde, seja para telediagnóstico ou cooperação assíncrona. O serviço imagem/vídeo sob demanda pode ser também utilizado para visualização de imagens e vídeos durante sessões de cooperação síncrona.

- O serviço de anotações será utilizado para inserir comentários em vídeos ou imagens médicas utilizadas em uma segunda opinião ou telediagnóstico.

- Os mecanismos de controle de acesso serão utilizados na validação de usuários, na disponibilização de funcionalidades adequadas para os mesmos e

na restrição de visualização por parte dos agentes de saúde de informações consideradas eticamente protegidas (ex.: identificação de pacientes, informações de diagnóstico em determinados casos clínicos, etc.).

- Os mecanismos de qualidade de serviço (QoS) serão aplicados às sessões de videoconferência e em todo tipo de visualização de imagem ou vídeo sob demanda, de forma a minimizar perdas ou atrasos na sua exibição, para que as mesmas não sofram pausas indevidas nem percam a qualidade de resolução exigida.

Os mecanismos de qualidade de serviço (QoS) propostos na arquitetura do projeto Infravida tem como objetivo principal priorizar o trafégo das aplicações multimídias do mesmo, como por exemplo a aplicação de segundo diagnóstico que se utiliza de serviço de videoconferência e imagem/vídeo sobre demanda em modalidade de cooperação síncrona. As aplicações que utilizam os mecanismos de QoS no projeto Infravida são:

- Segunda opinião médica
- HeathNet
- DynaVideo

Os serviços que necessitam QoS da segunda opinião médica e *HeathNet* são:

- Sistemas de videoconferência;
- Distribuição de imagens;
- Distribuição de vídeo sob demanda

As aplicações explicadas acima que se utilizam de serviço de videoconferência com áudio, vídeo sob demanda, imagem sob demanda e distribuição de imagens como documentos médicos, são serviços que possuem requisitos mínimos de qualidade de serviço (QoS) para o funcionamento adequado. Esses serviços são sensíveis ao atraso sendo necessária a identificação dos seus requisitos a fim de garantir que esses requisitos sejam cumpridos nas redes que os suportam.

A seguir são especificadas as necessidades de QoS nas redes dos serviços utilizados pelas aplicações do projeto Infravida e alternativas possíveis dos mecanismos de QoS para garantir esses requisitos nas redes IP.

4.1 IDENTIFICAÇÃO DOS REQUISITOS DE QoS – PROJETO INFRAVIDA

No caso do serviço de videoconferência para segunda opinião médica, é usado o *codec* G.711 para o áudio e o *codec* H.261 para o vídeo, por serem os de melhor resultado na implementação com o uso do *Open* H.323 [Rabelo et al., 2001]. O *codec* G.711 requer 64 kbps de banda unidirecional para cada canal de áudio. O *codec* de vídeo H.261 suporta velocidades múltiplas de 64 kbps, mas a implementação do *Open* H.323 possibilita apenas o uso de 64 kbps ou 128 kbps. O *codec* H.263 ainda não é completamente suportado no Open H.323.

Os parâmetros de QoS para a parte de vídeo da videoconferência H.261 são:

- Banda: 64 kbps ou 128 kbps
- Atraso: ≤ 200 ms fim-a-fim
- Variação de atraso (*jitter*): ≤ 30 ms

- Perda: ≤ 3%

Os parâmetros de QoS para a parte de áudio da videoconferência, *codec* G.711 são:

- Banda: 64 kbps
- Atraso: ≤ 200 ms fim-a-fim
- Variação de atraso (*jitter*): ≤ 30 ms
- Perda: ≤ 10%

O serviço de imagem sob demanda usa os padrões DICOM (*Digital Imaging and Communications in Medicine*) [DICOM, 2003] ou JPEG2000 [Charrier et al., 1999] como formato de imagens.

Os parâmetros para o serviço de imagem sob demanda a partir de estimativas adotadas no contexto do projeto Infravida usando padrões DICOM ou JPEG2000:

- Banda: 200 kbps a 20 Mbps
- Atraso: até 1 segundo
- Tolerância à variação de atraso (*jitter*) já que é uma aplicação de *download*
- Perda: 1% a 3%.

Os parâmetros para o uso de *codec* MPEG-2 para vídeo sob demanda são:

- Banda: 4 ou 6 Mbps
- Atraso: até 2 segundos
- Variação de atraso (*jitter*): 20 ms

- Perda: menor que 1%

A Tabela 3 especifica os requisitos de QoS [Lage et al., 2004B] para as aplicações multimídia no projeto Infravida [Lage, 2004].

Tabela 3: Identificação e especificação dos parâmetros dos serviços de imagem, vídeo e áudio – projeto Infravida [Lage, 2004].

	Banda	**Atraso**	***Jitter***	**Perda**
Imagem				
DICOM	20 Mbps	até 1 seg	---	1%
JPEG2000	200 kbps	até 1 seg	---	3%
Vídeo sob Demanda				
MPEG-2	4 ou 6 Mbps	até 2 seg	20 ms	< 1%
Videoconferência				
Áudio (G.711)	64 kbps	200 ms	30 ms	10%
Vídeo (H.261)	64 ou 128 kbps	200 ms	30 ms	3%

4.2 MAPEAMENTO DOS REQUISITOS DE QoS – PROJETO INFRAVIDA NA ARQUITETURA DIFFSERV

A partir dos requisitos de qualidade de serviço (QoS) dos serviços das aplicações multimídia do projeto Infravida é feito a seguir um mapeamento desses requisitos em classes de serviço do DiffServ juntamente com a definição dos comportamentos (PHBs) e DSCP (*Differentiated Services Code Point*) utilizados na implantação da rede protótipo experimental (*testbed)*.

O mapeamento das aplicações do projeto Infravida em classes de serviço leva em consideração as características dos serviços de áudio e vídeo e a dinâmica da aplicação de segunda opinião médica. O vídeo requer maior largura de banda (4 a 6 Mbps) e baixa taxa de perda (melhor que 1%), a imagem requer largura de banda variável (200 kbps a 20 Mbps tipicamente), enquanto a voz requer atraso mínimo fim-a-fim, menor banda, mas pode

suportar uma perda maior que o vídeo. Por estas características o áudio da aplicação de videoconferência (*codec* G.711) foi mapeado para a classe de serviço encaminhamento expresso (EF) [Lage, 2004].

A aplicação de segunda opinião médica implica em uma sessão de videoconferência entre profissionais de saúde, durante a qual podem ser solicitados vídeos ou imagens sob demanda. Durante a visualização de um vídeo ou imagem médica, o seu conteúdo é mais importante para os médicos do que a visualização das imagens dos outros participantes da videoconferência. Ou seja, um vídeo ou imagem no contexto de uma sessão de segunda opinião médica ganha uma característica conversacional. Então o serviço de vídeo/imagem sob demanda deve ser priorizado no contexto da segunda opinião, mas, ao mesmo tempo, outras imagens e vídeos solicitados, por exemplo, no contexto de EAD (ensino a distância) não devem ganhar prioridade, pois consumiriam muitos recursos, prejudicariam a qualidade da aplicação foco e poderiam inclusive bloquear a rede. Em função disto, é preciso incluir na solução um mecanismo que permita distinguir serviço de recuperação de conteúdo e serviço conversacional. A proposta é usar um identificador de classe de prioridade atribuído pela aplicação que venha a ser mapeada numa classe de serviço DiffServ [Lage, 2004]. A tabela 4 exibe o mapeamento das aplicações do projeto Infravida nas classes de serviço do DiffServ.

Tabela 4: Mapeamento das aplicações do projeto Infravida nas classes de QoS DiffServ.

Classe de serviço DiffServ	**Valor de DSCP**	**Mapeamento das Aplicações**
EF *Expedited Forwarding*	(DSCP 5) '101110'	Videoconferência (audio G.711)
AF4 *Assured Forwarding*	(DSCP 4) '100010'	Videoconferência (video H.261)
AF3 *Assured Forwarding*	(DSCP 3) '011010'	Vídeo sob Demanda (MPEG-2) ref. 2ª opinião Infravida
AF2 *Assured Forwarding*	(DSCP 2) '010010'	Imagem sob Demanda (DICOM,JPEG2) ref. 2ª opinião Infravida
AF1 *Assured Forwarding*	(DSCP 1) '001010'	Documentos médicos ref. 2ª opinião Infravida
BE *Best Effort*	(DSCP 0) '000000'	Documentos, imagens e vídeos de outras aplicações diferentes da 2ª. Opinião

4.3 ESPECIFICAÇÕES DOS CENÁRIOS DE TESTE DOS SERVIÇOS DIFFSERV PARA O PROJETO INFRAVIDA.

Visando a realização de campanhas de medição para a validação da arquitetura de QoS proposta, foram definidos três cenários básicos de testes [Lage et al., 2004B]:

1. Rede local com enlace de 10 Mbps
2. Nó central em rede de longa distância com enlace de 2 Mbps
3. Nó remoto em rede de longa distância com enlace de 512 kbps [Lage, 2004].

Algumas premissas foram assumidas:

- Oferecer uma garantia padrão de até 4 participantes de videoconferência, o que implica no estabelecimento de 3 sessões RTP de áudio (3x64 kbps) e vídeo (3x128 kbps ou 3x64 kbps) em *full-duplex* com o consultor, considerando distribuição centralizada. Considera-se o tamanho médio dos pacotes de áudio e vídeo na videoconferência de 200 *bytes*. O serviço tem prioridade sobre as demais aplicações do projeto Infravida, sendo que o áudio tem prioridade absoluta, para garantia de atraso e variação de atrasos mínimos.

- Otimizar a distribuição de vídeo sob demanda, oferecendo banda equivalente à de distribuição em tempo real (6 Mbps) onde houver banda disponível (cenário 1), banda de 1 Mbps para distribuição em vídeo *streaming* no cenário 2, e bloqueando a distribuição de vídeo onde ele não puder ser entregue com qualidade sem monopolizar recursos e prejudicar as demais aplicações (cenário 3). Considera-se o tamanho médio

dos pacotes de vídeo sob demanda de 1500 *bytes*. O serviço de distribuição de vídeo tem prioridade sobre o serviço de distribuição de imagem.

- Oferecer possibilidade de distribuir imagem típica de 10 Mbytes em aproximadamente 30 segundos em rede local (cenário 1), 2 minutos e 30 segundos no cenário 2 ou 5 minutos no cenário 3. Considera-se o tamanho médio dos pacotes de imagem sob demanda de 1500 *bytes*. O serviço de distribuição de imagens tem prioridade sobre a distribuição de documentos. O cálculo da banda requerida foi calculada da seguinte maneira: **10 Mbytes / 30seg ≅ 2,5 Mbps.**

- Oferecer prioridade a documentos médicos (prontuários, anotações e outros documentos) sobre o tráfego das demais aplicações. Considera-se o tamanho médio dos pacotes de documentos médicos transportados via FTP de 1500 *bytes*.

- Reservar uma parte da banda do enlace para o tráfego de outras aplicações de menor prioridade a fim que essas aplicações não “morram” por inanição, ou seja, deixem de funcionar por falta de banda disponível.

A tabela 5 exibe o resumo das especificações definidas acima dos serviços DiffServ.

Tabela 5: Especificações dos serviços DiffServ para o projeto Infravida.

				Configuração1 (10 Mbps)	Configuração2 (2 Mbps)	Configuração3 (512kbps)
Serviço	**Formato**	**Característica do Serviço**	**Classe de Serviço**	**Banda alocada**	**Banda alocada**	**Banda alocada**
Videoconf. (áudio)	G.711	64 kbps (252 bytes)	EF	3 x 64 kbps 192 kbps	2 x 64 kbps 128 kbps	1 x 64 kbps 64 kbps
Videoconf. (vídeo)	H.261	64 / 128Kbps (1040 bytes)	AF4	3 x 128 kbps 384 kbps	2 x 64 kbps 128 kbps	1 x 64 kbps 64 kbps
Vídeo sob Demanda	MPEG-2	4 a 6 Mbps (1040 bytes)	AF3	6 Mbps 6144 kbps	1 Mbps (*) 1024 kbps	---- (**)
Imagem sob Demanda	DICOM JPEG2K	10 Mbytes (1040 bytes)	AF2	2.5 Mbps 2560 kbps	512 kbps	256 kbps
Documentos Médicos	(vários FTP)	200 kbps (500 bytes)	AF1	256 kbps	128 kbps	64 kbps
Outras aplicações	(vários)		BE	512 kbps	128 kbps	64 kbps

(*) *videostreaming*
(**) distribuição de vídeo bloqueada por não poder ser oferecida com qualidade

Nas seções anteriores foram especificados parâmetros de QoS e o mapeamento desses parâmetros para classes de serviço DiffServ. Com base nas especificações definidas acima, as seções seguintes descrevem a especificação e implantação de uma rede protótipo experimental (*testbed*) para demonstrar a viabilidade deste projeto com respeito à qualidade de serviço requerida pelo mesmo. Nesse *testbed* foi montada uma estrutura de forma a espelhar a rede usada pelo projeto Infravida e são realizadas diversas campanhas de teste.

4.4 REDE PROTÓTIPO EXPERIMENTAL (*TESTBED*)

Esta seção explica os passos adotados com a finalidade de obter uma configuração de QoS para os roteadores a ser utilizada na rede protótipo experimental, como a configuração de *hardware* incluindo interfaces de rede, ferramentas utilizadas para garantir qualidade de serviço (QoS) para as aplicações, especificamente emular as funcionalidades do DiffServ.

Explica a topologia de rede utilizada, incluindo a configuração do comutador (*switch*) como também a montagem das VLANs (*Virtual Lans*) no comutador. São descritos os *scritps* utilizados para a configuração de QoS dos roteadores requeridas pelas aplicações do projeto Infravida. Essa seção mostra também a metodologia utilizada para validar os resultados de QoS obtidos no *testbed*, mostra como foi o processo de geração e medição do tráfego aplicando *scripts* de configuração para obtenção de QoS nos roteadores.

O controle de tráfego do GNU/Linux pode ser usado para implantar uma complexa combinação de algoritmos de escalonamento, classes e filtros a depender das necessidades de QoS requeridas pelas aplicações. Como exemplo, pode-se combinar filtros e algoritmos de escalonamento para construir um serviço de classe de encaminhamento expresso (EF) do DiffServ.

Toda a estrutura implantada para o *testbed*, tem como objetivo validar as especificações de QoS das aplicações do projeto Infravida (especificações das tabelas 5 e 3).

4.5 CONFIGURAÇÃO, IMPLANTAÇÃO DA REDE PROTÓTIPO EXPERIMENTAL – *TESTBED* – NO GNU/LINUX.

4.5.1 Topologia

A montagem da rede *testbed* foi configurada com os seguintes componentes:

- 03 PCs Dell P4 2,26GHz 512MB com GNU/Linux instalado, utilizados como roteadores. Cada PC possui 03 interfaces de rede *ethernet*.
- 01 comutador (s*witch*) 24 portas 10/100 3Com com 4 VLANs configuradas.

- 02 PCs utilizados para geração de tráfego e realização de medições associadas às campanhas de teste.

A figura 27 ilustra a topologia de rede utilizada no *testbed*. A configuração da rede protótipo experimental permite o seguinte conjunto básico de configurações de campanhas de teste:

- Geração de tráfego pela rota R1-R2-R3;
- Geração de tráfego direto pela rota R1-R3;

De maneira geral, na configuração de implantação de *testbed* tem-se:

- O hospedeiro “armação” atua como gerador de tráfego (configurável)
- O hospedeiro “ipitanga” atua na medição de tráfego.

Em cada roteador foi configurada uma tabela de rota de forma estática através de do comando “*route*” do GNU/Linux. Na seção 4.5.3 é explicada a forma de configuração dessas rotas em cada roteador.

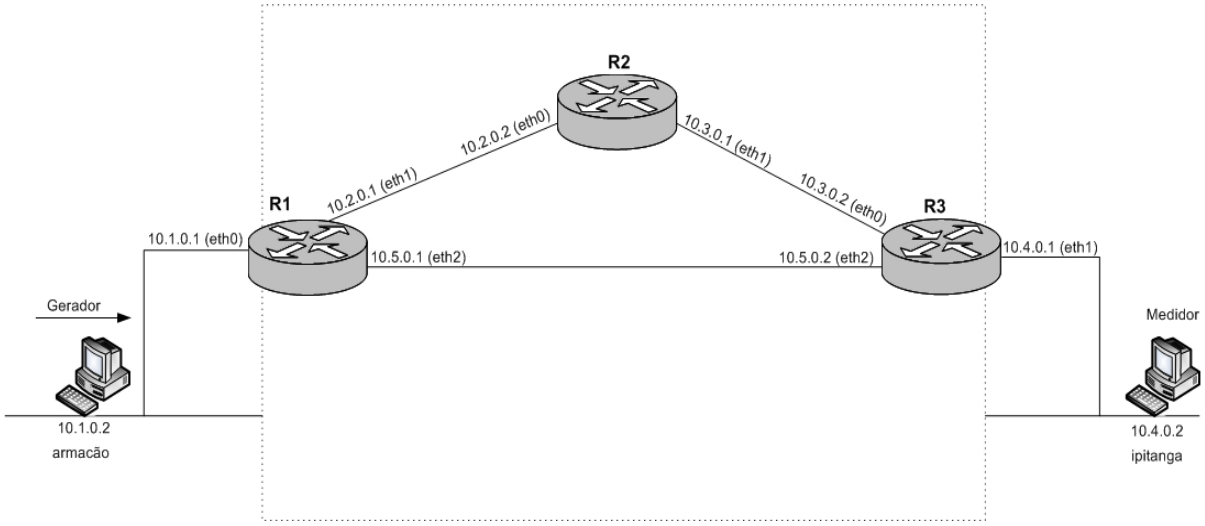

Figura 25: Topologia da rede protótipo experimental – *Testbed.*

4.5.2 Implantação do *testbed* (VLANs)

Na montagem do *testbed* foi utilizado um comutador (*switch)* modelo SUPERSTACK 3 Switch 4228G da 3Com com 24 portas e com suporte a VLANs (rede local virtual). Para o *testbed*, foram configuradas 4 VLANs no comutador. O objetivo da configuração das VLANs para o *testbed* foi criar sub-redes independentes, assim, reduzindo o domínio de *broadcast* na camada MAC e, também, forçar a utilização dos roteadores (R1, R2 e R3) para o encaminhamento dos pacotes entres as redes lógicas criadas.

A figura 28 ilustra as portas do comutador numeradas com suas respectivas VLANs. Da Porta 1 a 4 foi configurada uma VLAN para a rede 10.1.0.0, da porta 5 a 8 foi configurada uma VLAN para a rede 10.2.0.0. Da porta 9 a 12 foi configurada uma VLAN para a rede 10.3.0.0 e por último das portas 13 a 16 a última VLAN para a rede 10.4.0.0.

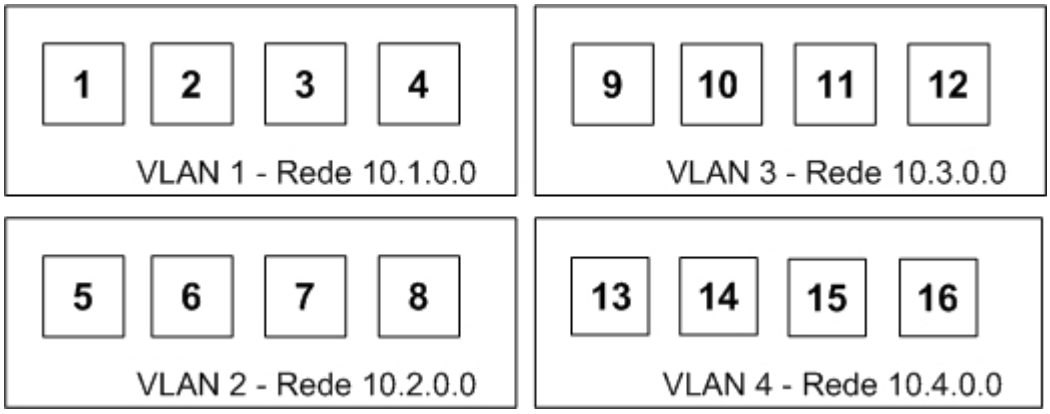

Figura 26: Configuração de VLAN no comutador (*switch*) do *testbed*.

4.5.3 Configuração de rotas – ROTEADORES GNU/LINUX.

Para a configuração de rotas nos roteadores, foi usado o comando "*route*" do GNU/Linux. A seguir são apresentados os *scripts* básicos para cada roteador usados no *testbed* para a seguinte configuração de rotas:

Roteador R1 (10.2.0.1, 10.2.0.1, 10.5.0.1)

```
#deleta possiveis rotas existentes
route del -net 10.1.0.0 netmask 255.255.0.0
route del -net 127.0.0.0 netmask 255.0.0.0
route del -net 0.0.0.0 netmask 0.0.0.0
route del -net 169.254.0.0 netmask 255.255.0.0
#adiciona novas rotas
route add -net 10.1.0.0 netmask 255.255.0.0 eth0
route add -net 10.2.0.0 netmask 255.255.0.0 eth1
route add default gw 10.2.0.2
```

O *script* acima configura as rotas do roteador R1. Primeiro ele apaga todas as possíveis rotas existentes (*route del –net*), depois adiciona uma rota para a rede 10.1.0.0. na interface eth0 e adiciona outra rota para a rede 10.2.0.0 na interface eth1. Como rota padrão, é configurada uma rota para o roteador R2 que possui o IP 10.2.0.2 saindo pela interface eth1. A rota padrão é configurada para o caso em que o endereço de destino do pacote recebido

pelo roteador R1 não pertencer a nenhuma das redes já configuradas anteriormente (10.1.0.0 e 10.2.0.0).

Os *scripts* de configuração de rota para os roteadores R2 e R3 são apresentados em seguida e seguem a mesma lógica de configuração apresentada para o roteador R1.

Roteador R2 (10.2.0.2, 10.3.0.1)

```
#deleta possiveis rotas existentes
route del -net 10.2.0.0 netmask 255.255.0.0
route del -net 10.2.0.0 netmask 255.255.0.0
route del -net 10.3.0.0 netmask 255.255.0.0
route del -net 10.3.0.0 netmask 255.255.0.0
route del -net 169.254.0.0 netmask 255.255.0.0
route del -net 127.0.0.0 netmask 255.0.0.0
#adiciona novas rotas
route add -net 10.3.0.0 netmask 255.255.0.0 eth1
route add -net 10.2.0.0 netmask 255.255.0.0 eth0
route add -net 10.1.0.0 netmask 255.255.0.0 gw 10.2.0.1
route add default gw 10.3.0.2
```

Roteador R3 (10.3.0.2, 10.4.0.1, 10.5.0.2)

```
#deleta possiveis rotas existentes
route del -net 10.4.0.0 netmask 255.255.0.0
route del -net 10.3.0.0 netmask 255.255.0.0
route del -net 169.254.0.0 netmask 255.255.0.0
route del -net 127.0.0.0 netmask 255.0.0.0
route del -net 0.0.0.0 netmask 0.0.0.0
#adiciona novas rotas
route add -net 10.3.0.0 netmask 255.255.0.0 eth0
route add -net 10.4.0.0 netmask 255.255.0.0 eth1
route add -net 10.1.0.0 netmask 255.255.0.0 gw 10.3.0.1
route add -net 10.2.0.0 netmask 255.255.0.0 gw 10.3.0.1
route add -net 10.3.0.0 netmask 255.255.0.0 gw 10.3.0.1
route add default gw 10.4.0.1
```

4.5.4 Configuração dos roteadores DiffServ – cenários de testes.

Para configurar os roteadores do *testbed* de forma a atender aos requisitos de QoS das aplicações do projeto Infravida, especificados na tabela 5, foram definidos *scripts* de

configurações para os roteadores no GNU/Linux, utilizando a ferramenta **tc** do pacote IPRoute2.

Para cada uma das configurações de teste especificados (configuração 01, 02 e 03 tabela 5) foram criados dois tipos de *scripts* com algoritmos de escalonamento diferentes. Um *script* usando o algoritmo de escalonamento HTB e outro *script* utilizando o algoritmo de escalonamento CBQ. Ambos os *scripts* têm o mesmo propósito, que é atender aos requisitos de QoS das aplicações do projeto Infravida, mas usam algoritmos de escalonamento diferentes para a mesma tarefa. O propósito da definição de dois tipos de *scripts* com uma mesma finalidade teve como objetivo fazer na campanha dos testes uma comparação entre os dois de forma a identificar o desempenho de ambos com relação às aplicações. A figura 29 ilustra um exemplo de configuração para o roteador R1 utilizando um *script* que usa algoritmo de escalonamento HTB, representando a situação da configuração 01 da tabela 5.

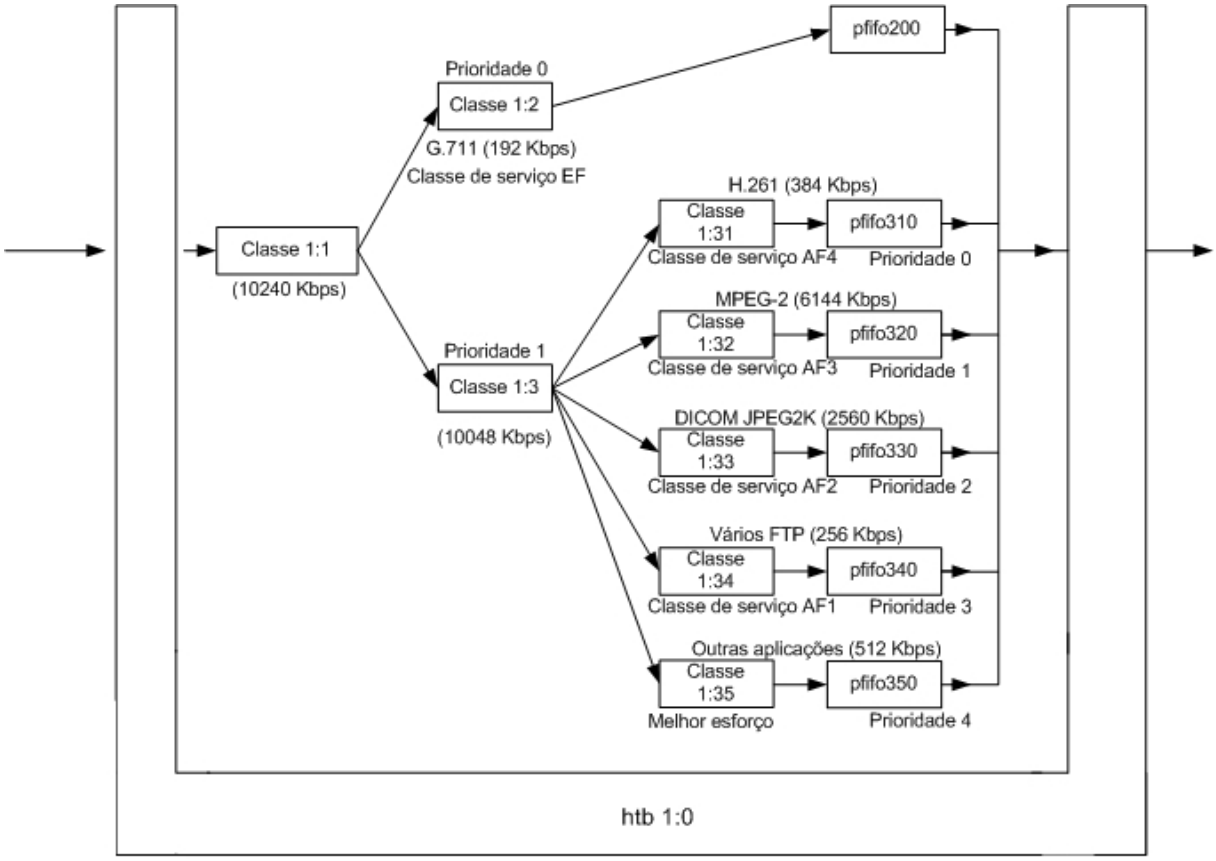

Figura 27: Ilustração de configuração usando HTB para o roteador R1.

Uma classe é criada com 10240 kbps de banda e nomeada de 1:1, duas outras classes 1:2 e 1:3 são criadas tendo como classe pai a 1:1. Classes com números menores têm prioridade maior em relação a classes com números maiores. Ex. A classe com prioridade 0 tem prioridade maior do que a classe com prioridade 1. Essa característica faz com que a classe 1:2 seja ideal para a classe de serviço DiffServ EF mapeada para a aplicação de videoconferência. Para essa classe com prioridade máxima em relação às outras foi definida uma banda mínima de 192 kbps. Ou seja, há uma garantia mínima de largura de banda de 192 kbps. Foram criadas ainda mais 5 classes derivadas da classe pai 1:3. Essas classes nomeadas de 1:31, 1:32, 1:33, 1:34 e 1:35 foram mapeadas para as classes de serviço do DiffServ AF4, AF3, AF2, AF1 e melhor esforço respectivamente. A classe com maior prioridade é a classe

1:31 (AF4) e a de menor prioridade é a classe 1:35 (melhor esforço). No final de cada classe, é associada uma fila pfifo com tamanho de fila de 10 pacotes.

Para que os pacotes que entrarem na interface de rede sejam redirecionado às devidas filas, é considerado que os mesmos já estejam marcados com o código DiffServ DSCP. Assim a ferramenta **tc** com o parâmetro **filter** reconhece a marca DSCP no pacote e direciona-o para sua devida fila. A ferramenta **tc** com a diretiva **filter** abaixo reconhece todos os pacotes que possuam a marca "0xb8" em seu DSCP e direciona o pacote para a classe 1:2 (G.711/ EF).

```
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0xb8
0xff flowid 1:2
```

O *script* completo de configuração de QoS para o Roteador R1, conforme o cenário 01 ilustrado na figura 29 é apresentado a seguir:

```
tc qdisc add dev eth1 root handle 1:0 htb
tc class add dev eth1 parent 1:0 classid 1:1 htb rate 10240kbit
tc class add dev eth1 parent 1:1 classid 1:2 htb rate 192kbit ceil 192kbit
prio 0
tc class add dev eth1 parent 1:1 classid 1:3 htb rate 10048kbit ceil
10048kbit prio 1
#AF4
tc class add dev eth1 parent 1:3 classid 1:31 htb rate 384kbit ceil 384kbit
prio 0
#AF3
tc class add dev eth1 parent 1:3 classid 1:32 htb rate 6144kbit ceil
6144kbit prio 1
#AF2
tc class add dev eth1 parent 1:3 classid 1:33 htb rate 2560kbit ceil
2560kbit prio 2
#AF1
tc class add dev eth1 parent 1:3 classid 1:34 htb rate 256kbit ceil 256kbit
prio 3
#outros
tc class add dev eth1 parent 1:3 classid 1:35 htb rate 512kbit ceil 512kbit
prio 4

#adicionar na ponta dos htb
tc qdisc add dev eth1 parent 1:2 handle 200: pfifo limit 10
tc qdisc add dev eth1 parent 1:31 handle 310: pfifo limit 10
tc qdisc add dev eth1 parent 1:32 handle 320: pfifo limit 10
tc qdisc add dev eth1 parent 1:33 handle 330: pfifo limit 10
tc qdisc add dev eth1 parent 1:34 handle 340: pfifo limit 10
tc qdisc add dev eth1 parent 1:35 handle 350: pfifo limit 10
```

```
#cria os filtros

tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0xb8
0xff flowid 1:2
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x88
0xff flowid 1:31
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x68
0xff flowid 1:32
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x48
0xff flowid 1:33
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x28
0xff flowid 1:34
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x00
0xff flowid 1:35
```

A seguir é apresentado como exemplo o *script* para a configuração do roteador R1, configuração 1 usando o algoritmo de escalonamento CBQ. A lógica de configuração do HTB e do CBQ são semelhantes, o que muda são algumas particularidades de sintaxe. Do mesmo modo do *script* anterior, o *script* a seguir tem a figura 29 como representação visual.

```
tc qdisc add dev eth1 root handle 1: cbq bandwidth 10Mbit avpkt 1000

#EF
tc class add dev eth1 parent 1: classid 1:1 cbq bandwidth 10Mbit rate
192kbit avpkt 252 prio 1 bounded isolated allot 1514
#classe dos AF
tc class add dev eth1 parent 1: classid 1:2 cbq bandwidth 10Mbit rate
10048kbit prio 2 bounded isolated allot 1514
#AF4
tc class add dev eth1 parent 1:2 classid 1:21 cbq bandwidth 10Mbit rate
384kbit avpkt 1040 prio 1 bounded isolated allot 1514
#AF3
tc class add dev eth1 parent 1:2 classid 1:22 cbq bandwidth 10Mbit rate
6144kbit avpkt 1040 prio 2 bounded isolated allot 1514


#AF2
tc class add dev eth1 parent 1:2 classid 1:23 cbq bandwidth 10Mbit rate
2560kbit avpkt 1040 prio 3 bounded isolated allot 1514
#AF1
tc class add dev eth1 parent 1:2 classid 1:24 cbq bandwidth 10Mbit rate
256kbit avpkt 500 prio 4 bounded isolated allot 1514
#BE
tc class add dev eth1 parent 1:2 classid 1:25 cbq bandwidth 10Mbit rate
512kbit prio 5 bounded isolated allot 1514

#adicionar na ponta dos htb
tc qdisc add dev eth1 parent 1:1 handle 200: pfifo limit 10
tc qdisc add dev eth1 parent 1:21 handle 310: pfifo limit 10
tc qdisc add dev eth1 parent 1:22 handle 320: pfifo limit 10
tc qdisc add dev eth1 parent 1:23 handle 330: pfifo limit 10
tc qdisc add dev eth1 parent 1:24 handle 340: pfifo limit 10
```

```
tc qdisc add dev eth1 parent 1:25 handle 350: pfifo limit 10

#cria os filtros

tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0xb8
0xff flowid 1:1
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x88
0xff flowid 1:21
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x68
0xff flowid 1:22
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x48
0xff flowid 1:23
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x28
0xff flowid 1:24
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip tos 0x00
0xff flowid 1:25
```

4.5.5 Gerador de Tráfego

Para simular os fluxos das aplicações especificadas na tabela 5 do projeto Infravida foi utilizado um gerador de tráfego gerando tráfego em uma ponta da rede com as mesmas taxas especificadas na tabela 5 e um coletor no destino para fazer a medição desses fluxos.

Existem dois tipos de medições: as medições passivas e medições ativas. Nas medições passivas, são capturadas todas as informações de pacotes que passam pelo enlace a qual a interface de rede está ligada sem provocar nenhuma interferência no tráfego medido. Como não existe um controle na origem da geração dos pacotes, o mapeamento do tempo de transmissão é prejudicado consequentemente atrapalhando o cálculo do atraso e da variação do atraso dos pacotes. Por essa dificuldade com ferramentas de medição passiva foi usada uma ferramenta de medição ativa, onde pacotes são gerados em uma ponta da rede e são medidos quando chegam ao seu destino, onde é possível identificar em um determinado fluxo, a ordem e o atraso dos pacotes, facilitando a medição.

Para gerar o tráfego de forma a simular os fluxos das aplicações especificados na tabela 5 do projeto Infravida foi usado uma ferramenta de medição ativa denominada RUDE & CRUDE [Laine et al., 2002]. Essa ferramenta é composta de dois executáveis. Um deles, o

RUDE (*Real-time UDP Data Emitter*) que é utilizado para gerar o tráfego de forma totalmente configurável através de um arquivo de configuração. Pode-se montar um cenário de geração com diversos fluxos, configurando o destino desses fluxos, a taxa de geração, o código DSCP marcado no cabeçalho do pacote etc. Outro programa da ferramenta é o CRUDE (*Collector for Real-time UDP Data Emitter*) é executado na máquina de destino e serve para medir todo o tráfego gerado pelo RUDE. O CRUDE coleta e loga todo o tráfego transmitido pelo RUDE. Ele possui diversas opções, tais como medir métricas como vazão, atrasos, variação de atrasos e perdas referentes a apenas um fluxo do tráfego ou todos os fluxos.

O RUDE é instalado na ponta da rede fazendo o papel das aplicações no cenário do *testbed*. Já o CRUDE é instalado no destino para a coleta das informações geradas pelo RUDE. A figura 30 ilustra o posicionamento dessas ferramentas na rede protótipo experimental.

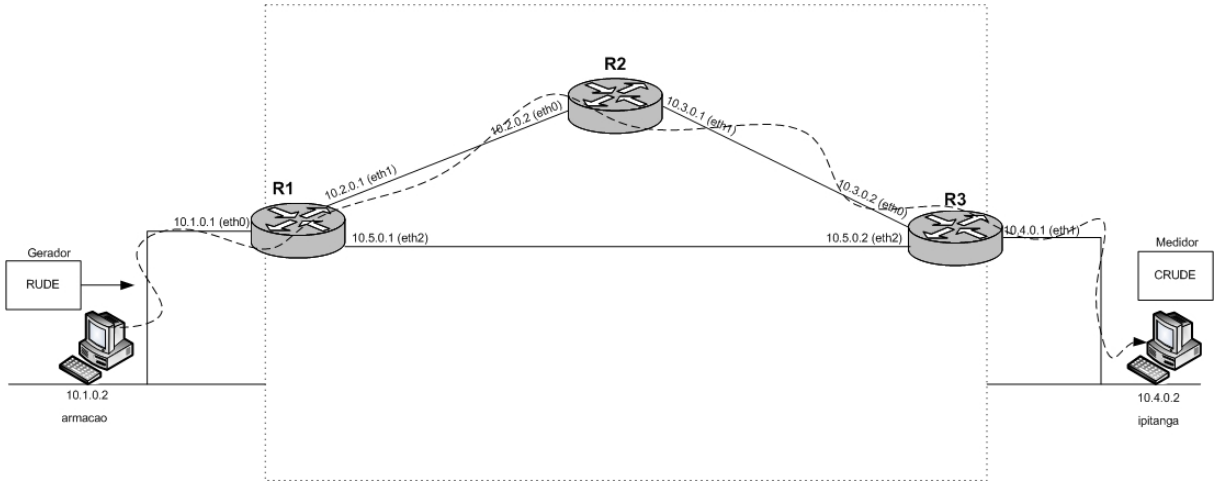

Figura 28: Posicionamento das ferramentas de geração e medição RUDE e CRUDE.

5. CENÁRIOS DE TESTE E CAMPANHAS DE MEDIÇÃO DA IMPLANTAÇÃO DE QOS/DIFFSERV

Para realizar as gerações/medições no *testbed* foram criados diversos cenários de testes que atendessem as especificações de QoS (tabela 5) requeridas pelas aplicações do projeto Infravida. Depois de especificados os cenários de testes de geração, os resultados obtidos são exibidos e analizados validando o modelo de QoS do projeto Infravida especificado na tabela 5.

As medições foram feitas considerando os seguintes parâmetros de QoS.

- Vazão
- Atraso
- Variação de Atraso (*Jitter*)
- Perda

Os parâmetros de QoS foram medidos com a ajuda do coletor CRUDE (seção 4.5.5). Os tamanhos dos pacotes gerados para cada fluxo estão especificados na tabela 5. Todos os tráfegos foram gerados numa taxa constante. Para medir o atraso na transmissão de um pacote de uma máquina para outra, é necessário que o relógio das mesmas estejam sincronizados, caso contrário, a medida do atraso não é calculada corretamente. Para resolver o problema da sincronização as máquinas usadas na rede protótipo experimental foram sicronizadas usando um servidor ligado a um GPS (*Global Position Sattelite*). As máquinas foram sincronizadas através do protocolo NTP [Mills, 1992] em *stratum* 2 na rede local.

O protocolo NTP se baseia em um modelo de sincronização hierárquico. No maior nível da hierarquia encontram-se os servidores de tempo *stratum* 1. Esses servidores estão conectados diretamente aos relógios de referência de altíssima precisão, no caso da rede protótipo experimental o relógio de refêrencia é o GPS. No nosso caso, máquinas que realizaram as medições foram sicronizadas utilizando o servidor de tempo *stratum* 1, logo se tornando stratum 2. Com esta configuração o erro na medição do atraso foi aproximadamente em torno de 30 μs. Uma descrição da arquitetura do projeto Infravida e um breve resumo dos resultados obtidos podem ser encontrados em [Lage et al., 2004A] e em [Lage et al., 2004B].

5.1 CENÁRIO DE TESTE 01

Para o cenário de teste 01, a geração de tráfego simulou o comportamento normal das aplicações, conforme especificado na tabela. O cenário de teste 01 representa a situação ideal, onde a soma da vazão de todos os fluxos gerados não excede a capacidade total do enlace de 10 Mbps e nenhum tráfego paralelo mal comportado é gerado para concorrer com os fluxos das aplicações. Nenhuma configuração de QoS foi aplicada aos roteadores. Os tráfegos foram gerados pelo RUDE durante 60 segundos no hospedeiro “armação” sendo coletados pelo CRUDE no hospedeiro “ipitanga” passando pelos roteadores R1, R2 e R3 como ilustra a figura 31. A tabela 6 especifica os fluxos configurados no RUDE para o cenário de teste 01 com o código DSCP marcado no campo ToS do cabeçalho IP representando as classes de serviço DiffServ.

Este cenário de teste foi definido, pois vai permitir comparar posteriormente os valores medidos (atraso, *jitter*, vazão, perda) com outros valores de outros cenários de teste. Estes outros cenários terão fluxo mal comportado concorrendo com os fluxos das aplicações, não havendo largura de banda suficiente para todos os fluxos e sem nenhuma configuração de

QoS aplicada nos roteadores. Também se espera nos futuros cenários, muita perda de pacotes, alto atraso, alta variação de atraso, prejudicando o funcionamento das aplicações.

Tabela 6: Cenário de teste 01 - fluxos gerados

Fluxo 1	64 Kbps, pacotes marcados com o DSCP EF 0xb8
Fluxo 2	64 Kbps, pacotes marcados com o DSCP EF 0xb8
Fluxo 3	64 Kbps, pacotes marcados com o DSCP EF 0xb8
Fluxo 4	128 Kbps, pacotes marcados com o DSCP AF4 0x88
Fluxo 5	128 Kbps, pacotes marcados com o DSCP AF4 0x88
Fluxo 6	128 Kbps, pacotes marcados com o DSCP AF4 0x88
Fluxo 7	6144 Kbps, pacotes marcados com o DSCP AF3 0x68
Fluxo 8	2560 Kbps, pacotes marcados com o DSCP AF2 0x48
Fluxo 9	256 Kbps, pacotes marcados com o DSCP AF1 0x28
Fluxo 10	512 Kbps sem marcação nos pacotes, melhor esforço.

Como ilustrado na figura 31, o RUDE gera os fluxos simulando o comportamento das aplicações passando pelos roteadores R1, R2 e R3 e em seguida o CRUDE faz as medições dos fluxos no hospedeiro “ipitanga”.

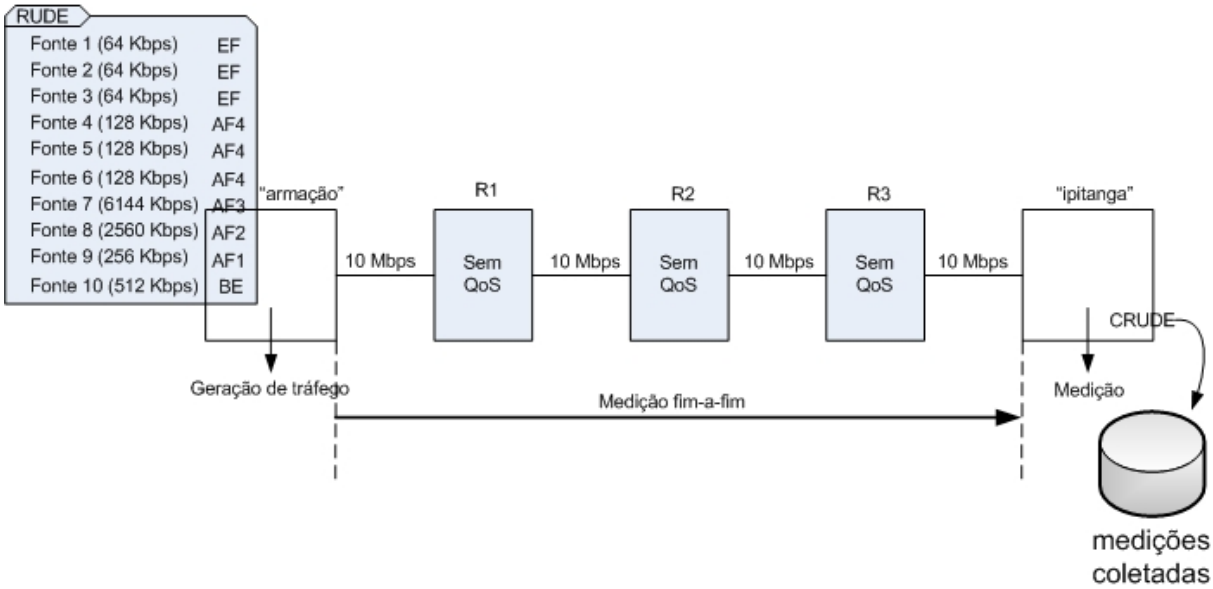

Figura 29: Cenário de teste 01

5.2 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 01

No resultado desta campanha de medição observam-se que os fluxos se comportaram de forma esperada. Não há tráfego concorrente mal comportado para atrapalhar os fluxos das aplicações, há largura de banda sobrando para todos os fluxos, portanto uma situação ideal. A medida de vazão exibida na tabela 7 se aproxima do tráfego gerado pelo RUDE mostrado no cenário de teste 01 (5.1). A medida de atraso é aceitável, correspodendo às espectativas de QoS das aplicações do projeto Infravida, como também, à variação de atraso. Não foi observada a perda de pacotes em nenhum dos fluxos.

Tabela 7: Campanha de medição – Cenário de teste 01.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)
Fluxo 1	1945	63,83 Kbps	17,91 ms	0,147 ms
Fluxo 2	1945	63,83 Kbps	17,93 ms	0,150 ms
Fluxo 3	1945	63,83 Kbps	17,96 ms	0,150 ms
Fluxo 4	936	126,82 Kbps	17,96 ms	0,146 ms
Fluxo 5	936	126,82 Kbps	18,05 ms	0,135 ms
Fluxo 6	935	126,69 Kbps	18,14 ms	0,133 ms
Fluxo 7	45378	6145 Kbps	17,92 ms	0,112 ms
Fluxo 8	18820	2548 Kbps	17,98 ms	0,122 ms
Fluxo 9	3925	255 Kbps	17,92 ms	0,153 ms
Fluxo 10	26200	511 Kbps	17,90 ms	0,134 ms

5.3 CENÁRIO DE TESTE 02

Para este cenário de teste, foi adicionado um fluxo mal comportado para concorrer com o fluxo das aplicações especificados originalmente na configuração 1 da tabela 5. Para este cenário de teste foram gerados fluxos semelhantes aos exibidos na tabela 7 mais um fluxo concorrente com vazão de 81674 Kbps sem marcação nos pacotes simulando o fluxo mal

comportado. Também como no cenário de teste 01, nenhuma configuração de QoS foi aplicada aos roteadores e o tráfego foi gerado pelo RUDE durante 60 segundos no hospedeiro “armação” sendo coletados pelo CRUDE no hospedeiro “ipitanga” tendo como rota os roteadores R1, R2 e R3 como ilustra a figura 32. O objetivo na montagem desse cenário de teste foi criar uma situação de congestionamento nos enlaces, simulando o ambiente real das redes que suportarão as aplicações do projeto Infravida. Espera-se muita perda, alto atraso, baixa vazão nas medições desse cenário de teste.

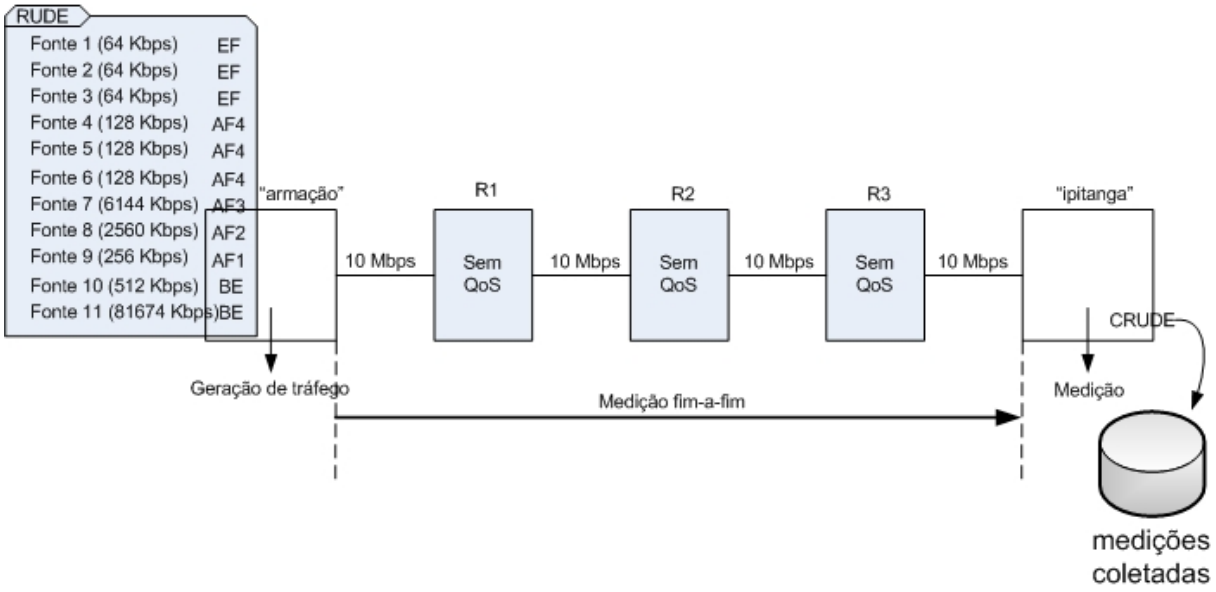

Figura 30: Cenário de teste 02

5.4 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 02

O resultado da campanha de medição do cenário de teste 02 é exibido na tabela 8. Para este cenário os fluxos das aplicações são gerados concorrendo com um fluxo gerado intencionalmente mal comportado. Não há nenhuma configuração de QoS nos roteadores. Observa-se nesses resultados que os requisitos minimos de QoS das aplicações do projeto Infravida não foram satisfeitos. Esperava-se segundo a configuração 1 (enlace 10 Mbps) tabela 5 que os fluxos configurados como serviço EF, fluxos 1, 2 e 3 teriam 64 kbps de vazão cada um, como observado na tabela 8 os fluxos 1, 2 e 3 obtiveram resultados inferiores, vazão de 22,27 kbps, 20,53 kbps e 18,89 kbps respectivamente. Os fluxos configurados como serviço AF4 (fluxos 4, 5 e 6) precisariam de 128 kbps de vazão cada um. Novamente os resultados obtidos ficaram bem abaixo dos requeridos, os fluxos 4, 5 e 6 obtiveram 44,48 kbps, 39,74 kbps e 36,21 kbps de vazão respectivamente. O requisito mínimo de vazão para os fluxos 7, 8 e 9 configurados como serviço AF3, AF2 e AF1 respectivamente também não foram satisfeitos.

As perdas foram muito grandes nos resultados obtidos, todos os fluxos medidos tiveram perdas superiores a 64% tornando-se inviável para as aplicações. Os requisitos de atraso e variação de atraso foram atendidos para todos os fluxos, mas não é considerado ideal, visto que os parâmetros de atraso e variação de atraso são calculados levando em consideração apenas os pacotes que são recebidos pelo CRUDE no hospedeiro “ipitanga”.

Os resultados observados na tabela 8 eram esperados visto que nenhuma configuração de QoS foi aplicada nos roteadores e foi gerado um tráfego superior à capacidade do enlace. Portanto é necessário aplicar uma configuração de QoS visando atender os requisitos mínimos de qualidade de serviço para as aplicações do projeto Infravida.

Tabela 8: Campanha de medição – Cenário de teste 02.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	679	22,27 Kbps	87,49 ms	3,497 ms	1266 (65,08%)
Fluxo 2	626	20,53 Kbps	87,68 ms	3,662 ms	1319 (67,81%)
Fluxo 3	576	18,89 Kbps	87,75 ms	3,615 ms	1369 (70,38%)
Fluxo 4	328	44,48 Kbps	87,27 ms	4,572 ms	608 (64,95%)
Fluxo 5	293	39,74 Kbps	87,08 ms	4,986 ms	643 (68,69%)
Fluxo 6	267	36,21 Kbps	87,47 ms	4,649 ms	668 (71,36%)
Fluxo 7	15953	2158 Kbps	87,35 ms	3,382 ms	29425 (64,84%)
Fluxo 8	6703	906 Kbps	87,43 ms	4,103 ms	12117 (64,38%)
Fluxo 9	1364	88,71 Kbps	87,34 ms	4,421 ms	2561 (65,24%)
Fluxo 10	9202	179,54 Kbps	87,46 ms	3,897 ms	16998 (64,87%)
Fluxo 11	23441	6098 Kbps	81,22 ms	4,731 ms	290160 (92,52%)

5.5 CENÁRIO DE TESTE 03

A geração de tráfego deste cenário de teste é exatamente igual ao do cenário de teste 02, baseado na tabela 05 simulando os fluxos das aplicações do projeto Infravida. É aplicada nos roteadores R1, R2 e R3 uma configuração de QoS com o algoritmo de escalonamento HTB de acordo com as necessidades de QoS das aplicações do projeto Infravida no configuração 1 (tabela 05). A configuração de QoS aplicada nesse cenário de teste prioriza os fluxos EF em relação aos outros fluxos. Para cada fluxo é configurada no roteador uma vazão de acordo com as necessidades de QoS especificadas no cenário 01 da tabela 05. Assim cada fluxo recebe uma largura de banda de acordo com sua necessidade. Os fluxos 10 e 11 não recebem nenhum tratamento nos roteadores, assim, só sendo servido se houver largura de banda sobrando no enlace. Foi gerado tráfego durante 60 segundos com o RUDE no hospedeiro “armação” passando pelos roteadores R1, R2 e R3 e medido pelo CRUDE no hospedeiro “ipitanga”. A figura 33 ilustra a configuração de QoS dos roteadores e a geração de tráfego.

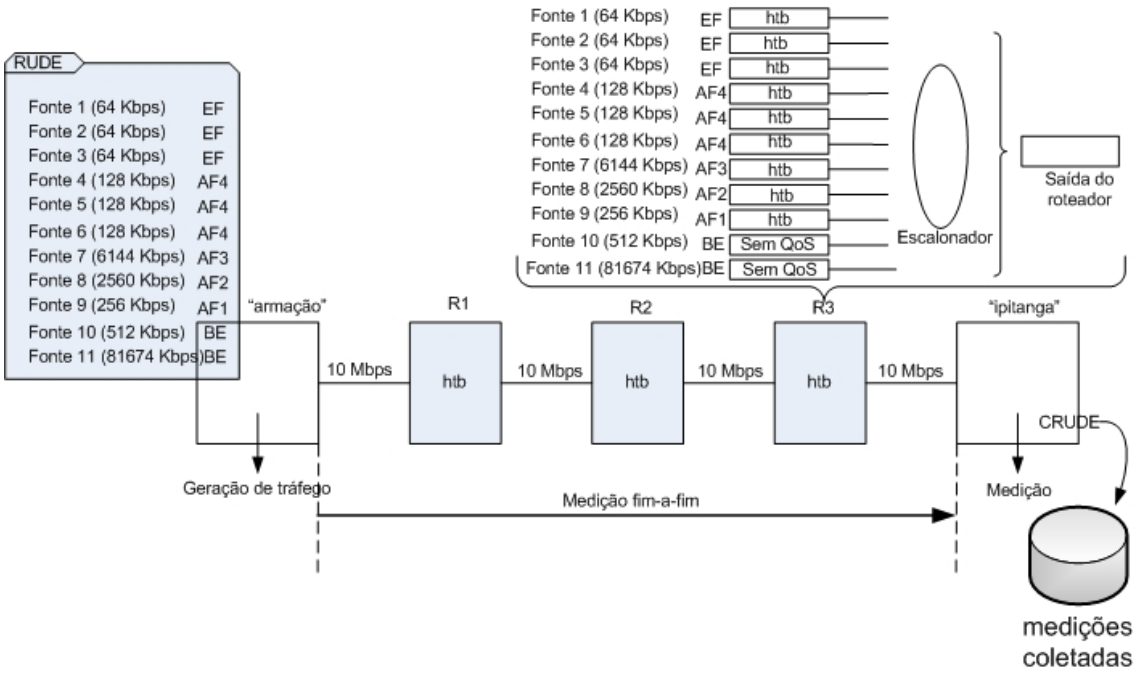

Figura 31: Cenário de teste 03

5.6 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 03

Para este cenário de teste, foi gerado um fluxo mal comportado para concorrer com os fluxos das aplicações. Para os resultados obtidos nesta campanha de teste foram aplicados nos roteadores uma configuração de QoS baseado no algoritmo de escalonamento HTB. À medida obtida do parâmetro de vazão para todos os fluxos corresponde com a configuração aplicada nos roteadores e requerida pelas aplicações. Para o fluxo 10 que corresponde à classe de serviço de melhor esforço, observa-se um resultado para o parâmetro de vazão sofrível (0,185 Kbps) que já era esperado visto que não recebe nenhum tratamento diferenciado pelos roteadores.

As medidas de atraso e variação de atraso obtidas para os fluxos 1, 2 e 3 não atendem às requeridas pelas aplicações com o algoritmo de escalonamento HTB. Essa discrepância se

deve pelo tamanho do pacote configurado (252 bytes) para os fluxos 1, 2 e 3 característico do formato de áudio G.711 que o algoritmo de escalonamento HTB do GNU/Linux não consegue gerenciar de forma satisfatória. Para os demais fluxos que têm os tamanhos dos pacotes maiores que 500 bytes os resultados obtidos para os parâmetros de atraso e variação de atraso são aceitáveis.

Os resultados obtidos para o parâmetro de perda foram muitos diferentes. Para os fluxos 1, 2, 3, 4 e 9 as perdas são aceitáveis e satisfazem os requisitos das aplicações. Para os demais fluxos os valores medidos não são aceitáveis para as aplicações. Não foi observado que o tamanho dos pacotes das aplicações tenha influenciado diretamente nos valores medidos para o parâmetro de perda. O algoritmo de escalonamento HTB do GNU/Linux não satisfaz em parte os valores mínimos de perda requeridos pelas aplicações do projeto Infravida. Na tabela 9 são exibidos os resultados para a campanha de medição do cenário de teste 03.

Tabela 9: Campanha de medição – Cenário de teste 03.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	1945	62,62 kbps	1047 ms	38,78 ms	0 (0%)
Fluxo 2	1757	56,56 kbps	1054 ms	36,71 ms	188 (9,66%)
Fluxo 3	1526	49,13 kbps	1048 ms	32,68 ms	419 (21,54%)
Fluxo 4	936	126,50 kbps	168,64 ms	12,45 ms	0 (0%)
Fluxo 5	936	126,40 kbps	188,92 ms	16,35 ms	0 (0%)
Fluxo 6	870	117,49 kbps	208,54 ms	12,84 ms	66 (7,05%)
Fluxo 7	43686	5916 kbps	16,45 ms	1,997 ms	1692 (3,72%)
Fluxo 8	18820	2466 kbps	36,71 ms	5,414 ms	599 (3,18%)
Fluxo 9	3682	239 kbps	150,84 ms	24,97 ms	243 (6,19%)
Fluxo 10	7	0,185 kbps	157,82 ms	36,074 ms	26193 (96,29%)

5.7 CENÁRIO DE TESTE 04

Nesse cenário de teste, a geração de tráfego é exatamente igual à do cenário de teste 02 e 03. O tráfego é gerado nas mesmas taxas dos cenários de testes 02 e 03 a partir do hospedeiro “armação” durante 60 segundos passando nos roteadores R1, R2 e R3 e sendo medido pelo CRUDE no hospedeiro “ipitanga”. Nos roteadores é aplicada uma configuração de QoS usando o algoritmo de escalonamento CBQ. A configuração de QoS aplicada nos roteadores obedece à mesma lógica do cenário de teste 03, onde serão priorizados os fluxos EF em relação aos outros. Os fluxos 10 e 11 não recebem nenhum tratamento, sendo servidos apenas se houver largura de banda disponível no enlace. O objetivo de usar outro algoritmo de escalonamento para realizar uma mesma tarefa que é garantir as necessidades de QoS das aplicações do projeto Infravida definidas na tabela 05 é comparar os algoritmos de escalonamento observando o algoritmo de melhor desempenho. A figura 34 ilustra o cenário de teste 04.

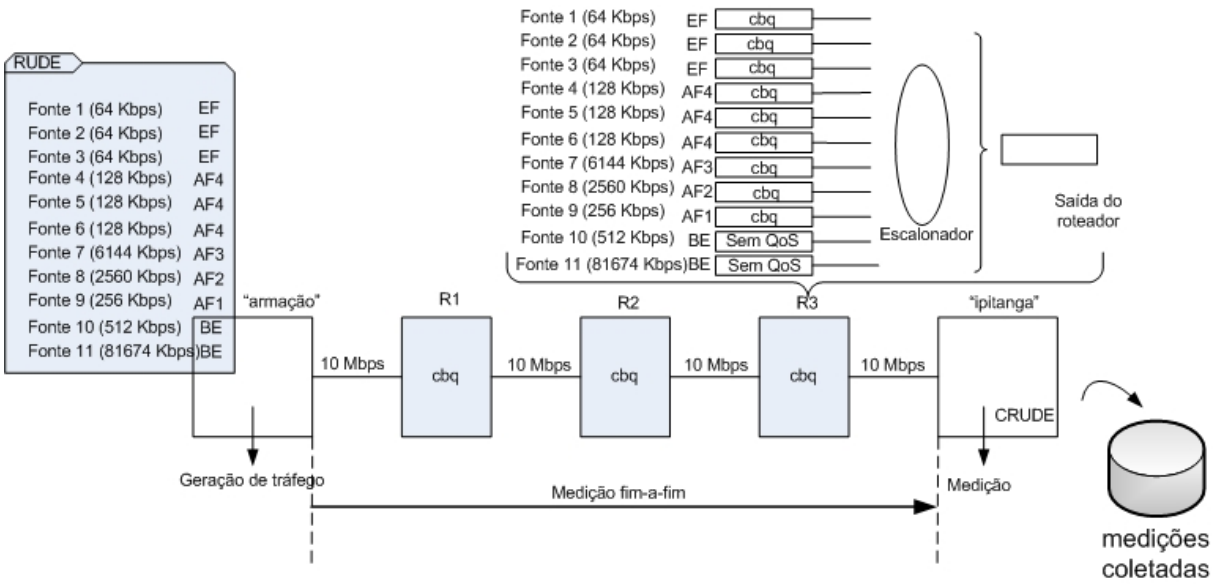

Figura 32: Cenário de teste 04

5.8 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 04

A configuração de QoS aplicada nos roteadores para o cenário de teste 04 foi o algoritmo de escalonamento CBQ. Os resultados obtidos com esse algoritmo de escalonamento foram os que apresentaram os melhores resultados. Todos os requisitos de QoS das aplicações do projeto Infravida foram satisfeitos. Todas as medidas obtidas de vazão corresponderam às configuradas nos roteadores, a medida de atraso e a variação de atraso foram muito boas e não houve perda em nenhum dos fluxos, exceto o fluxo 10. O fluxo 10 que corresponde à classe de serviço de melhor esforço ficou praticamente sem serviço, comportamento já esperado, visto que não recebe nenhum tratamento diferenciado nos roteadores.

Tabela 10: Campanha de medição – Cenário de teste 04.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	1945	63,84 kbps	9,06 ms	0,05 ms	0
Fluxo 2	1945	63,84 kbps	9,05 ms	0,05 ms	0
Fluxo 3	1945	63,84 kbps	12,41 ms	2,13 ms	0
Fluxo 4	936	126,84 kbps	9,20 ms	0,04 ms	0
Fluxo 5	936	126,84 kbps	9,19 ms	0,02 ms	0
Fluxo 6	935	126,83 kbps	19,40 ms	1,156 ms	0
Fluxo 7	45378	6146 kbps	9,41 ms	0,713 ms	0
Fluxo 8	18820	2549 kbps	9,62 ms	0,713 ms	0
Fluxo 9	3925	255 kbps	9,10 ms	0,05 ms	0
Fluxo 10	12	0,24 kbps	75,63 ms	12,05 ms	26188

O gráfico da figura 35 faz uma comparação dos valores medidos das perdas para o cenário 01. O gráfico compara as perdas observadas usando o algoritmo de escalonamento HTB, CBQ e sem configuração de QoS nos roteadores para todos os fluxos.

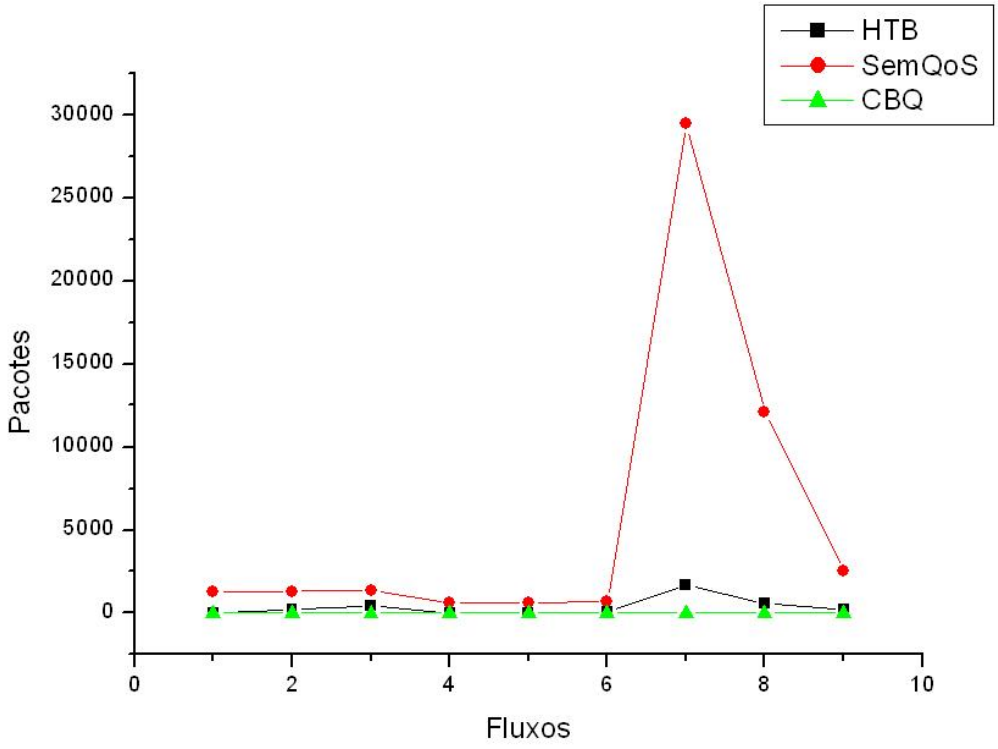

Figura 33: Perda de pacotes cenário 01 (tabela 5).

O gráfico da figura 36 ilustra o atraso obtido apenas do fluxo 1 na configuração 1. A configuração de QoS com o algoritmo de escalonamento CBQ obteve o melhor resultado obtendo medidas de atraso próximas a zero. O algoritmo de escalonamento CBQ atendeu as requisitos de QoS especificados na tabela 3, onde foi especificado um atraso máximo de até 200 ms para o fluxo 1 que representa o serviço DiffServ EF para o fluxo de àudio da videoconferência no formato do *codec* G.711.

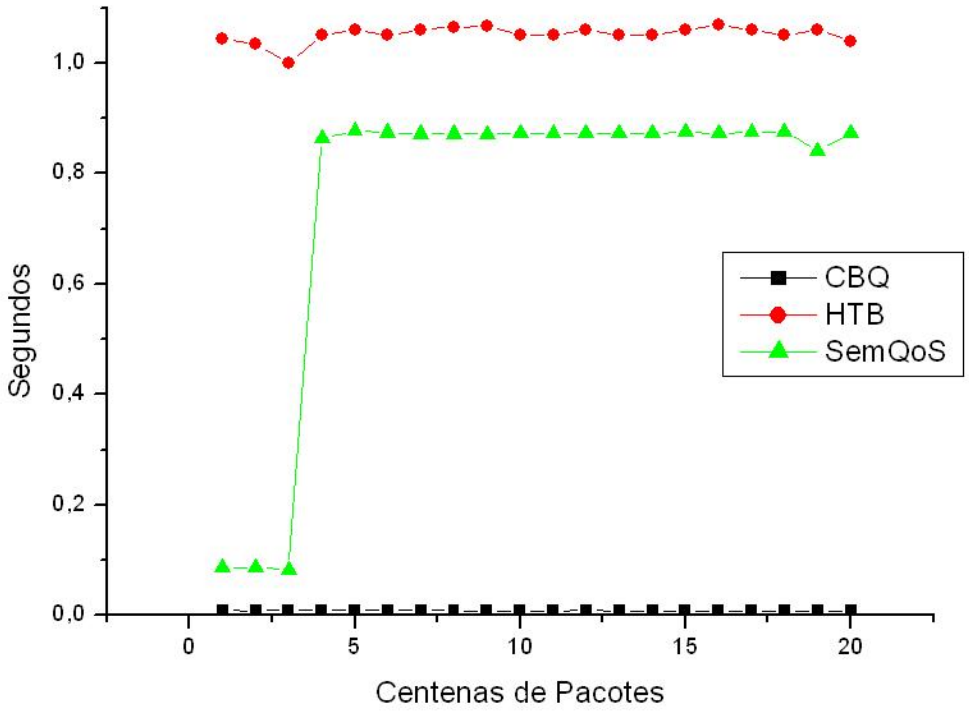

Figura 34: Atraso cenário 01 fluxo 01.

A figura 37 ilustra o resultado obtido da variação de atraso (*jitter*) do fluxo 01 da configuração 1 A configuração do algoritmo de escalonamento HTB não atendeu aos requisitos requeridos de QoS da tabela 3. O algoritmo de escalonamento CBQ se mostrou a melhor alternativa de configuração de QoS, apresentando uma variação de atraso 0,05 ms, próxima a zero.

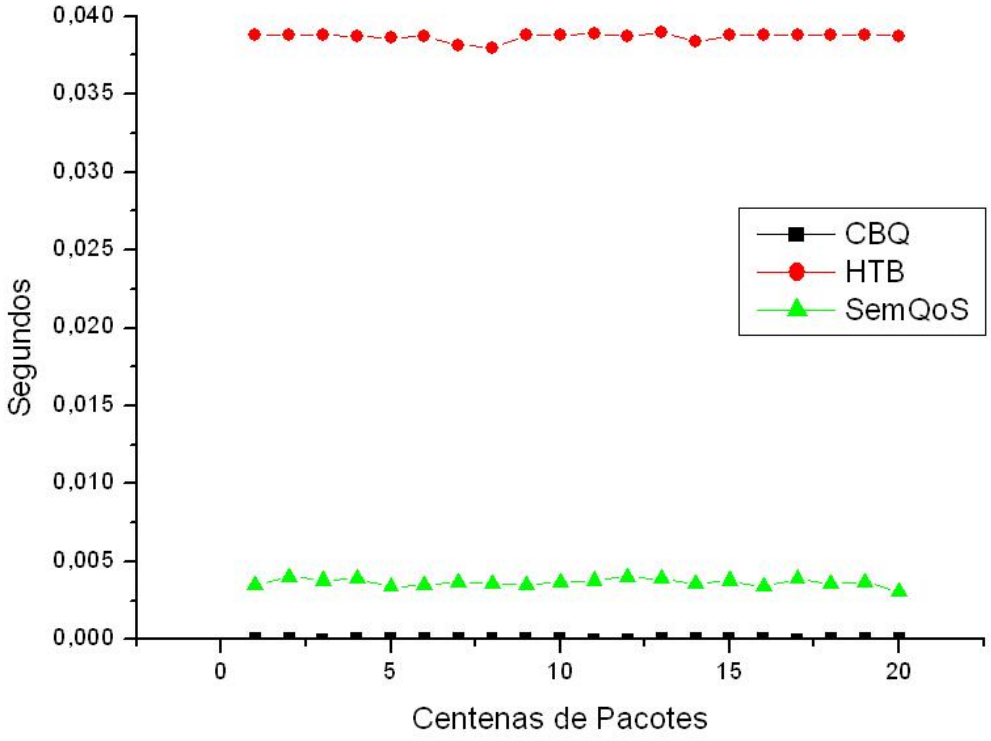

Figura 35: Variação de atraso (*jitter*) cenário 01 fluxo 01.

5.9 CENÁRIO DE TESTE 05

Este cenário de teste simula o comportamento das aplicações definidas na configuração 2 da tabela 05 do projeto Infravida sem nenhum fluxo mal comportado concorrente. Semelhante ao cenário de teste 01, é gerado tráfego durante 60 segundos no hospedeiro “armação” pelo RUDE nas taxas exbibidas na tabela 07, passando pelos roteadores R1, R2 e R3 e sendo medido no hospedeiro “ipitanga” pelo CRUDE. Nenhuma configuração de QoS é aplicada nos roteadores que permitirá comparar os resultados com outros cenários.

Tabela 11: Cenário de teste 05 - fluxos gerados

Fluxo 1	64 kbps, pacotes marcados com o DSCP EF 0xb8
Fluxo 2	64 kbps, pacotes marcados com o DSCP EF 0xb8
Fluxo 3	64 kbps, pacotes marcados com o DSCP AF4 0x88
Fluxo 4	64 kbps, pacotes marcados com o DSCP AF4 0x88
Fluxo 5	1024 kbps, pacotes marcados com o DSCP AF3 0x68
Fluxo 6	512 kbps, pacotes marcados com o DSCP AF2 0x48
Fluxo 7	128 kbps, pacotes marcados com o DSCP AF1 0x28
Fluxo 8	128 kbps, sem marcação nos pacotes, melhor esforço.

A figura 38 ilustra o cenário de teste 05.

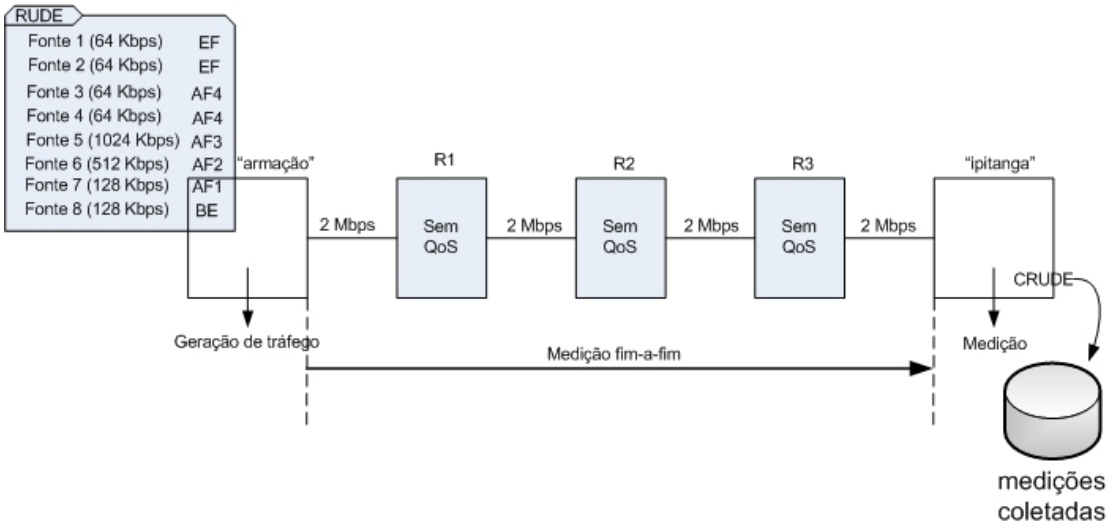

Figura 36: Cenário de teste 05

5.10 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 05

O objetivo dessa campanha de medição é semelhante à campanha de medição 01, medir os fluxos das aplicações em situação ideal, sem tráfego concorrente para atrapalhar os fluxos das aplicações e com largura de banda sobrando no enlace. Para essa campanha de medição não há nenhum tipo de configuração de QoS aplicada nos roteadores. O tráfego gerado é baseado na configuração 2 (tabela 05) com enlace de 2 Mbps.

A medida do parâmetro de vazão exibida na tabela 12 se aproxima do tráfego gerado pelo RUDE mostrado no cenário de teste 05 (5.9). Nessas condições ideais, com largura de banda sobrando no enlace, todos os requisitos mínimos de QoS das aplicações do projeto Infravida foram satisfeitos.

Tabela 12: Campanha de medição – Cenário de teste 05.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)
Fluxo 1	1945	63,83 kbps	5,78 ms	0,125 ms
Fluxo 2	1945	63,83 kbps	5,80 ms	0,131 ms
Fluxo 3	467	63,36 kbps	5,84 ms	0,132 ms
Fluxo 4	467	63,36 kbps	5,91 ms	0,137 ms
Fluxo 5	7569	1025 kbps	5,79 ms	0,129 ms
Fluxo 6	3775	511,25 kbps	5,82 ms	0,115 ms
Fluxo 7	1958	127,51 kbps	5,79 ms	0,128 ms
Fluxo 8	6549	127,92 kbps	5,78 ms	0,128 ms

5.11 CENÁRIO DE TESTE 06

Neste cenário de teste é gerado tráfego de acordo com a tabela 11 baseado na configuração 2 da tabela 05 do projeto Infravida. É gerado um fluxo concorrente de 89826 kbps sem marcação nos pacotes para atrapalhar os fluxos normal das aplicações do projeto Infravida. O tráfego é gerado pelo RUDE no hospedeiro “armação” passando pelos roteadores R1, R2 e R3 e sendo medido no hospedeiro “ipitanga” pelo CRUDE. O tráfego é gerado durante 60 segundos sem configuração de QoS aplicada nos roteadores. A figura 39 ilutra o cenário de teste 06.

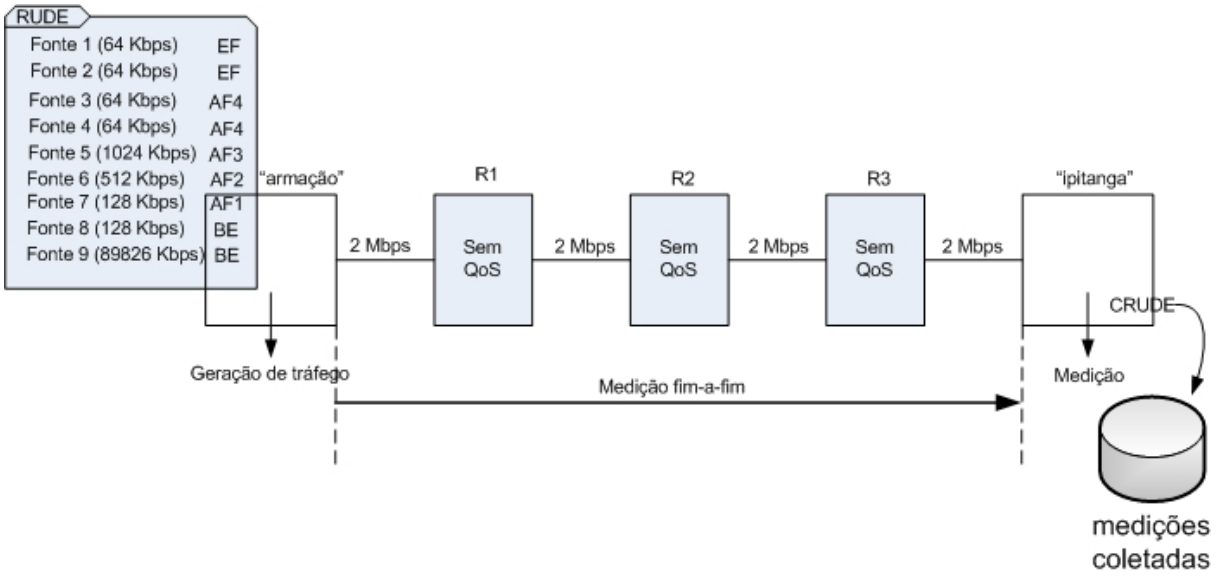

Figura 37: Cenário de teste 06

5.12 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 06

Para esta campanha de medição, nenhuma configuração de QoS foi aplicada nos roteadores e um fluxo mal comportado foi gerado para concorrer com os fluxos das aplicações. O resultado desta campanha de medição se assemelha com o resultado da campanha de medição 02, observa-se que os requisitos mínimos de QoS das aplicações do projeto Infravida não foram satisfeitos, sendo necessária uma configuração de QoS para satisfazer os requisitos mínimos de QoS para as aplicações do projeto Infravida.

Os resultados também já eram esperados nesta campanha de medição devido à ausência de configuração de QoS nos roteadores e à presença de um fluxo mal comportado fazendo com que a soma de todos os fluxos gerados fosse superior à capacidade do enlace. Portanto é necessário aplicar uma configuração de QoS visando atender os requisitos mínimos de qualidade de serviço para as aplicações do projeto Infravida. A tabela 13 exibe os resultados obtidos nesta campanha de medição.

Tabela 13: Campanha de medição – Cenário de teste 06.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	662	21,64 kbps	346,78 ms	10,98 ms	1283 (65,96 %)
Fluxo 2	493	16,09 kbps	356,51 ms	11,12 ms	1452 (74,65 %)
Fluxo 3	149	20,10 kbps	342,96 ms	15,27 ms	318 (68,09 %)
Fluxo 4	114	15,47 kbps	345,48 ms	14,84 ms	353 (75,58 %)
Fluxo 5	2707	364,77 kbps	345,84 ms	9,66 ms	4862 (64,23 %)
Fluxo 6	1312	176,79 kbps	345,60 ms	10,11 ms	2463 (65,24 %)
Fluxo 7	634	41,18 kbps	346,07 ms	10,38 ms	1324 (67,62 %)
Fluxo 8	2283	44,37 kbps	347,20 ms	9,67 ms	4266 (65,13 %)
Fluxo 9	4845	1255 kbps	341,74 ms	7,89 ms	340058 (98,59 %)

5.13 CENÁRIO DE TESTE 07

A geração de tráfego deste cenário é a mesma usada do cenário de teste 06. Nos roteadores é aplicada uma configuração de QoS usando o algoritmo de escalonamento HTB. O tráfego é gerado pelo RUDE no hospedeiro “armação” durante 60 segundos passando pelos roteadores R1, R2 e R3 e sendo medido pelo CRUDE no hospedeiro “ipitanga”. A configuração de QoS aplicada nos roteadores novamente tem o objetivo de atender as necessidades de QoS das aplicaçãoes do projeto Infravida definidas na configuração 2 da tabela 05. Os fluxos EF receberão prioridade sobre outros fluxos. Os fluxos das fontes 8 e 9 não receberam nenhum tratamento nos roteadores. Espera-se que haja muita perda de pacotes e muito atraso para esses fluxos. A figura 40 ilustra o cenário de teste 07.

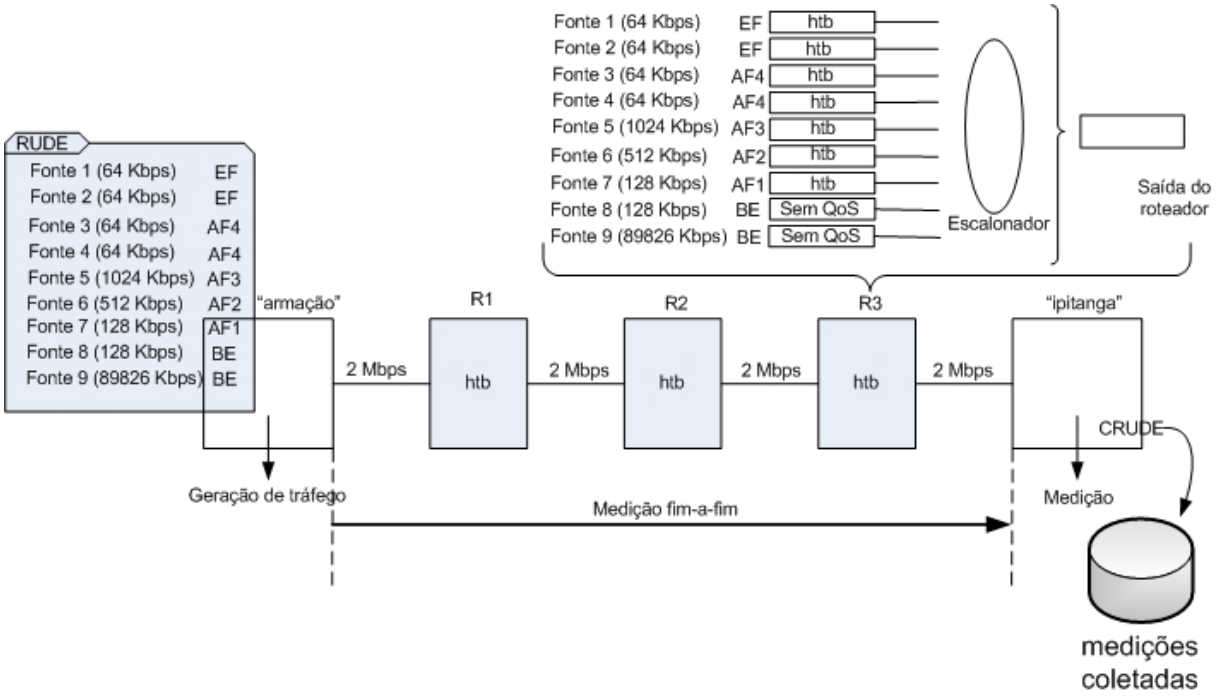

Figura 38: Cenário de teste 07

5.14 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 07

As medidas de vazão obtidas nessa campanha de teste usando como configuração de QoS o algoritmo de escalonamento HTB foram razoáveis. Não é a ideal, mas se aproxima dos requisitos mínimos de QoS das aplicações definidos na tabela 5. O fluxo 8 configurado como melhor esforço ficou sem serviço, devido à concorrência do fluxo 9, gerado a uma vazão maior, inserido como um fluxo mal comportado.

O algoritmo de escalonamento HTB não conseguiu gerenciar de forma correta os fluxos 1 e 2 configurados como EF, melhor serviço, menor atraso, baixa perda. As medidas obtidas de atrasos e variação de atraso para esses fluxos ficaram muito acima das esperadas, maiores que as dos demais fluxos. Apenas as medidas de atraso e variação de atraso dos fluxo 5 e 6 atenderam aos requisitos mínimos de QoS para as aplicações do projeto Infravida. Observa-se aqui que a relação do tamanho de pacote novamente influenciou no resultado final

na medida do atraso e variação de atraso usando o algoritmo de escalonamento HTB. Os fluxos 1 e 2 têm o tamanho de pacote menor (252 bytes) do que os demais fluxos (1040 e 500 bytes) e obtiveram as maiores medidas de atraso e variação de atraso.

Quanto ao parâmetro de perda apenas os fluxos 1, 3, 4 e 7 atenderam aos requisitos mínimos de QoS das aplicações do projeto Infravida, para os demais fluxos os parâmetros de perdas medidos ficaram acima das desejadas.

O algoritmo de escalonamento HTB não atendeu plenamente as necessidades mínimas de qualidade de serviço requerida pelas aplicações do projeto Infravida definidas na tabela 5. A tabela 14 exibe os resultados desta campanha de medição.

Tabela 14: Campanha de medição – Cenário de teste 07.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	1913	61,01 kbps	1515,23 ms	46,33 ms	32 (1,64 %)
Fluxo 2	1608	51,28 kbps	1477,38 ms	55,20 ms	337 (17,32 %)
Fluxo 3	461	61,99 kbps	450,96 ms	12,40 ms	6 (1,28 %)
Fluxo 4	460	61,72 kbps	515,89 ms	11,61 ms	7 (1,49 %)
Fluxo 5	7294	986,92 kbps	81,63 ms	11,70 ms	275 (3,63 %)
Fluxo 6	3654	493,76 kbps	156,98 ms	21,67 ms	121 (3,20 %)
Fluxo 7	1848	119,80 kbps	284,01 ms	47,98 ms	110 (5,61 %)
Fluxo 8	1	-	-	-	6548 (99,98 %)
Fluxo 9	370	95,37 kbps	703,05 ms	76,54 ms	344533 (99,89 %)

5.15 CENÁRIO DE TESTE 08

Neste cenário de teste os roteadores são configurados com o algoritmo de escalonamento CBQ. A geração do tráfego é semelhante a geração do cenário de teste 06. Como nos outros cenários de testes anteriores, o tráfego é gerado pelo RUDE no hospedeiro “armação” passando pelos roteadores R1, R2 e R3 e é medido pelo CRUDE pelo hospedeiro “ipitanga”. A configuração de QoS usando o algoritmo de escalonamento CBQ, tratam os

fluxos de acordo com as necessidades de QoS das aplicações do projeto Infravida definidas na configuração 2 da tabela 05. A figura 41 ilustra o cenário de teste 08.

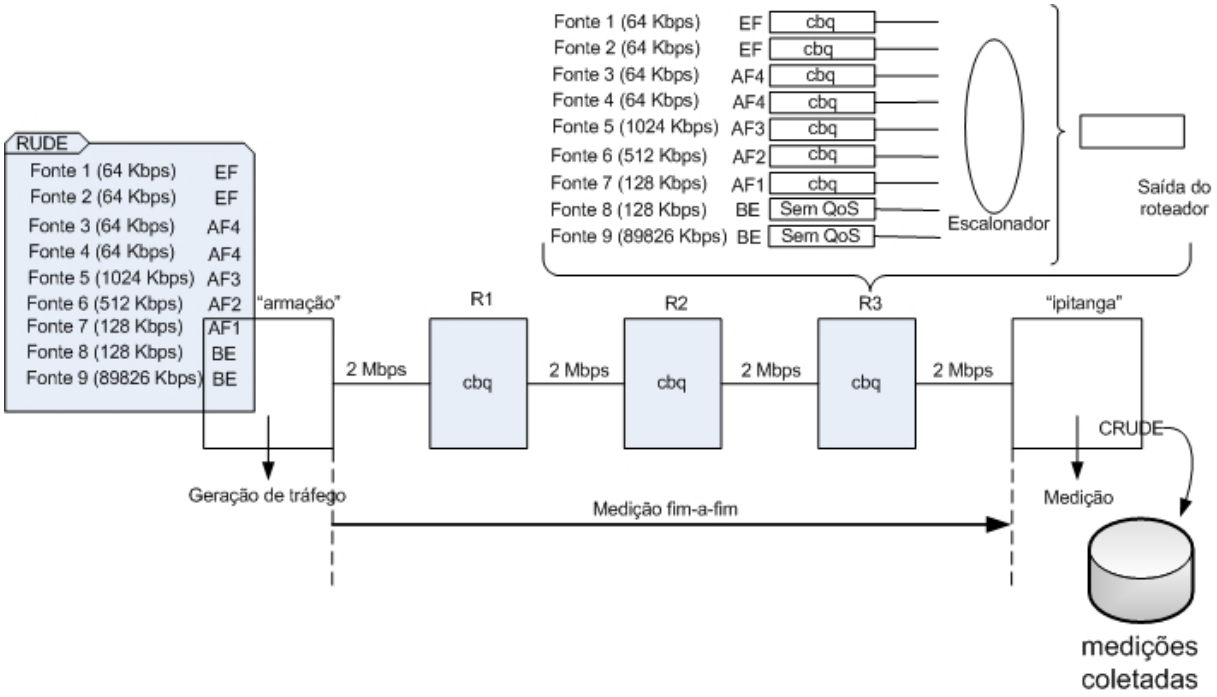

Figura 39: Cenário de teste 08

5.16 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 08

Como na campanha de medição 04, o algoritmo de escalonamento CBQ novamente se mostrou a melhor alternativa de configuração de QoS. Este campanha de medição é baseada no cenário de teste 08, definido no cenário 02 da tabela 05 requerido pelas aplicações do projeto Infravida. Todas as necessidades de QoS requeridas pelas aplicações do projeto Infravida foram satisteifas usando o algoritmo de escalonamento CBQ.

O fluxo 8 que corresponde à classe de serviço de melhor esforço ficou sem serviço, comportamento já esperado, visto que não recebe nenhum tratamento especial nos roteadores. O fluxo 8 concorreu diretamente com o fluxo 9 e como o mesmo foi gerado com uma taxa de vazão expressivamente maior que o fluxo 8, conseguiu supera-lo em termos de vazão. Na tabela 15 é exibido os resultados obtidos nesta campanha de medição.

Tabela 15: Campanha de medição – Cenário de teste 08.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	1945	63,84 kbps	8,83 ms	0,125 ms	0
Fluxo 2	1945	63,84 kbps	8,83 ms	0,131 ms	0
Fluxo 3	467	63,37 kbps	8,97 ms	0,132 ms	0
Fluxo 4	467	63,37 kbps	8,97 ms	0,137 ms	0
Fluxo 5	7569	1025 kbps	8,97 ms	0,129 ms	0
Fluxo 6	3775	511,30 kbps	8,97 ms	0,115 ms	0
Fluxo 7	1958	127,53 kbps	8,88 ms	0,128 ms	0
Fluxo 8	3	0,072 kbps	154,78 ms	0,128 ms	6546 (99,95 %)
Fluxo 9	285	73,62 kbps	348,85 ms	239,51 ms	344618 (99,91 %)

O gráfico da figura 42 ilustra uma comparação dos valores medidos das perdas para a configuração 2. O gráfico compara as perdas obtidas usando como configuração de QoS nos roteadores os algoritmos de escalonamento HTB, CBQ e sem configuração de QoS para todos os fluxos. Os fluxos configurados com o algoritmo de escalonamento CBQ não tiveram perdas, os fluxos configurados com o algoritmo de escalonamento HTB tiveram perdas próximas à zero. Já os fluxos que não receberam tratamento nos roteadores as perdas foram muito altas, já esperadas.

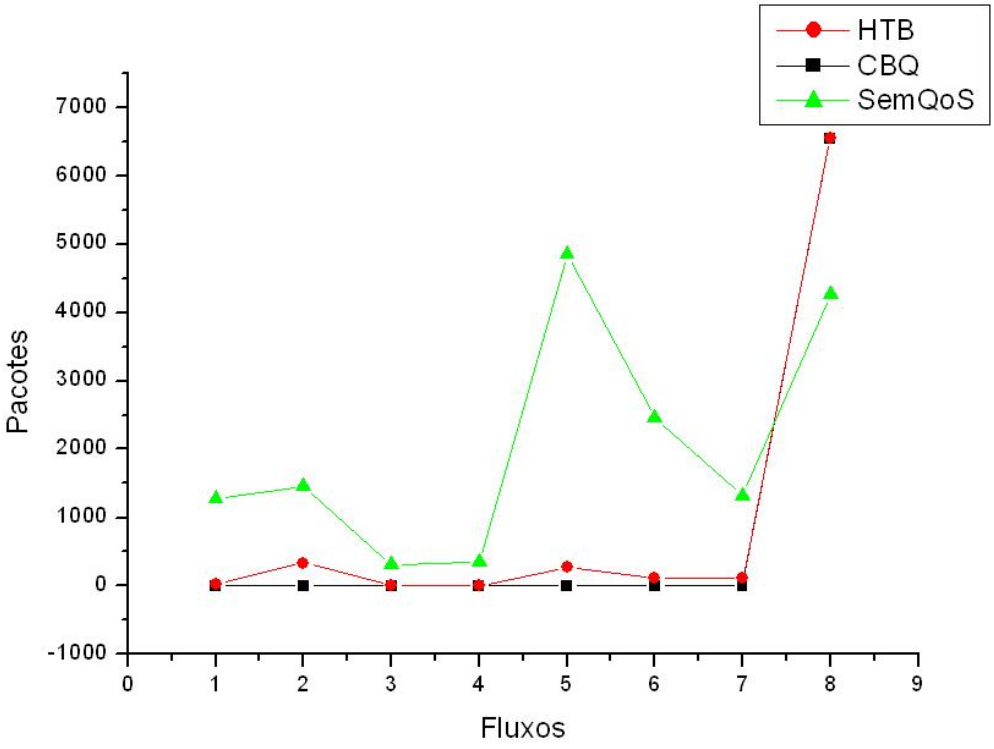

Figura 40: Perda de pacotes configuração 2 (tabela 5).

5.17 CENÁRIO DE TESTE 09

Este cenário de teste simula o comportamento das aplicações do projeto Infravida da configuração 3 definido na tabela 5. Este cenário de teste segue a mesma lógica dos cenários de teste anteriores, é medido primeiro os fluxos das aplicações em situação ideal, sem tráfego mal comportado concorrente. A configuração 3 da tabela 5 do projeto Infravida considera um enlace de 512 kbps de banda alocada. A tabela 16 exibe as taxas dos fluxos gerados para esse cenário de teste.

Tabela 16: Cenário de teste 09 - fluxos gerados

Fluxo 1	64 kbps, pacotes marcados com o DSCP EF 0xb8
Fluxo 2	64 kbps, pacotes marcados com o DSCP AF4 0x88
Fluxo 3	256 kbps, pacotes marcados com o DSCP AF2 0x48
Fluxo 4	64 kbps, pacotes marcados com o DSCP AF1 0x28
Fluxo 5	64 kbps, sem marcação nos pacotes, melhor esforço.

O tráfego foi gerado por 60 segundos pelo RUDE no hospedeiro “armação” passando pelos roteadores R1, R2 e R3 e sendo medido pelo CRUDE no hospedeiro “ipitanga”. Nenhuma configuração de QoS foi aplicada nos roteadores. A figura 43 ilustra o cenário de teste 09.

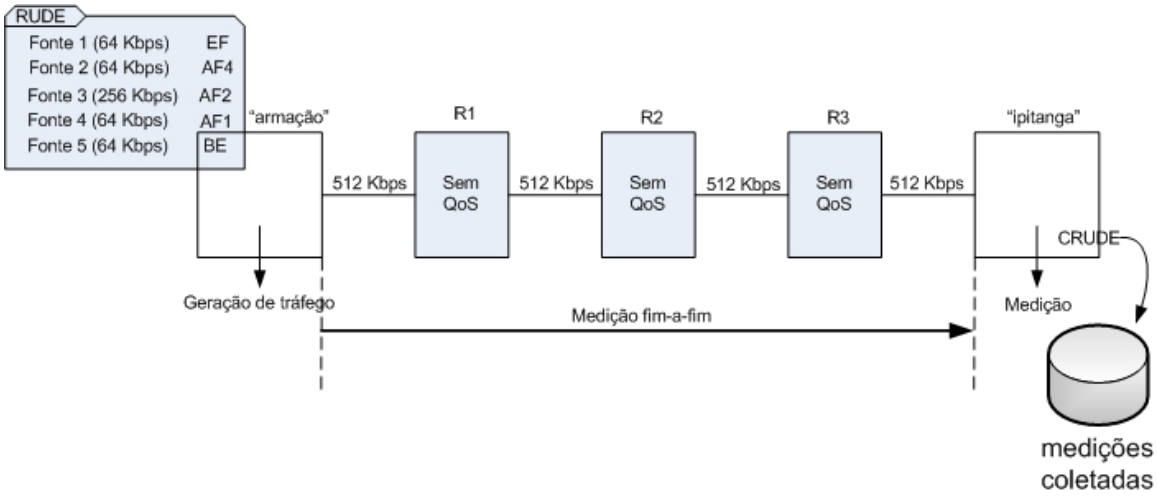

Figura 41: Cenário de teste 09

5.18 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 09

Esta campanha de medição tem a mesma finalidade das campanhas anteriores 1 e 5, medir os fluxos em uma situação ideal, com largura de banda sobrando e sem fluxo mal comportado para concorrer com os fluxos das aplicações. Todo o tráfego foi gerado sem configuração de QoS nos roteadores.

Os valores obtidos medidos nesta campanha atendem plenamente aos requisitos mínimos de QoS requeridos pelas aplicações do projeto Infravida definido na tabela 5. Na tabela 17 são exibidos os resultados obtidos nesta campanha de medição.

Tabela 17: Campanha de medição – Cenário de teste 09.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)
Fluxo 1	1945	63,83 kbps	5,68 ms	0,108 ms
Fluxo 2	466	63,22 kbps	5,73 ms	0,124 ms
Fluxo 3	1883	255,13 kbps	5,73 ms	0,104 ms
Fluxo 4	976	63,55 kbps	5,68 ms	0,116 ms
Fluxo 5	3267	63,81 kbps	5,67 ms	0,112 ms

5.19 CENÁRIO DE TESTE 10

Este cenário de teste é semelhante ao cenário de teste anterior, a diferença é que um fluxo concorrente (fluxo 6) de 91404 Kbps é gerado para atrapalhar o fluxos das aplicações. Este cenário de teste é baseado nas necessidades de QoS das aplicações do projeto Infravida da definidas na tabela 5. Nenhuma configuração de QoS é aplicada nos roteadores. O tráfego é gerado pelo RUDE no hospedeito “armação” passando pelos roteadores R1, R2 e R3 sendo medido no hospedeiro “ipitanga” pelo CRUDE. Espera-se com essa medição altas perdas, grandes atrasos e altas variações de atrado. A figura 44 ilustra o cenário de teste 10.

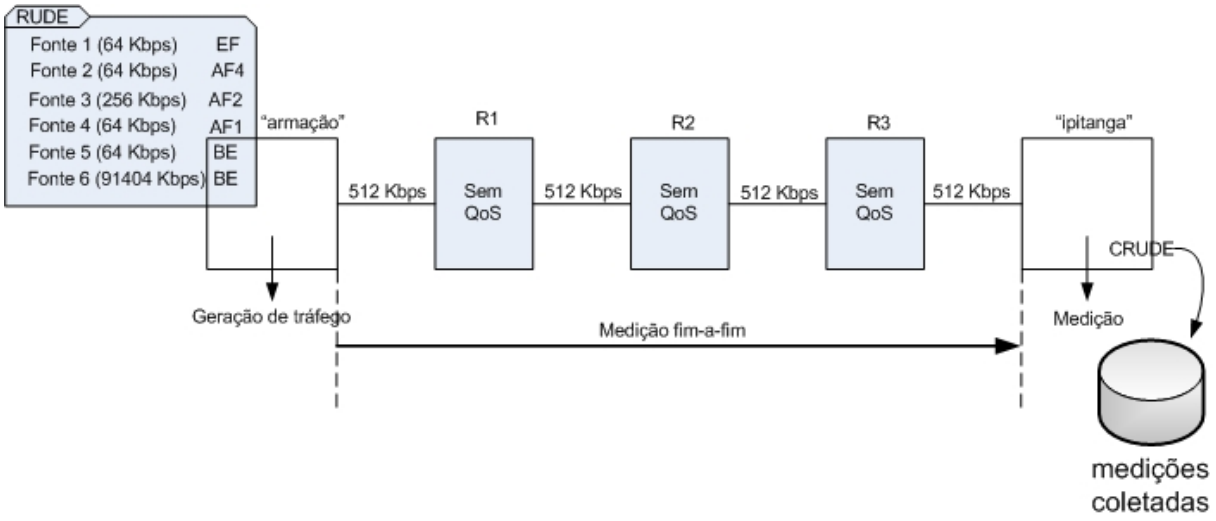

Figura 42: Cenário de teste 10

5.20 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 10

Para esta campanha de medição é usado um enlace de 512 kbps. Esta campanha tem a mesma função das campanhas anteriores 2 e 6, medir os fluxos das aplicações sem configuração de QoS aplicadas nos roteadores e com um fluxo mal comportado (fluxo 6) concorrendo com os fluxos das aplicações. Como exibido na tabela 18, já se esperava resultados ruins. Observa-se alto atraso, alta variação de atraso, alta perda. Os resultados obtidos não atendem os requisitos mínimos de QoS requeridos pelas aplicações do projeto Infravida definido na tabela 05. É necessária uma configuração de QoS nos roteadores para atender aos requisitos requeridos.

Tabela 18: Campanha de medição – Cenário de teste 10.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	556	17,88 kbps	1195,98 ms	25,93 ms	1389 (71,41 %)
Fluxo 2	117	15,66 kbps	1189,86 ms	41,28 ms	349 (74,89 %)
Fluxo 3	520	69,09 kbps	1192,79 ms	26,16 ms	1363 (72,38 %)
Fluxo 4	243	15,53 kbps	1194,40 ms	30,79 ms	733 (75,10 %)
Fluxo 5	946	18,15 kbps	1197,97 ms	22,65 ms	2321 (71,04 %)
Fluxo 6	1331	340,10 kbps	1174,44 ms	15,92 ms	349630 (99,62 %)

5.21 CENÁRIO DE TESTE 11

Neste cenário de teste, a geração de tráfego obedece a mesma lógica do cenário de teste 10. Os roteadores são configurados com o algoritmo de escalonamento HTB. O tráfego é gerado pelo RUDE no hospedeiro “armação” durante 60 segundos passando pelos roteadores R1, R2 e R3 e medido pelo CRUDE no hospedeiro “ipitanga”. O enlace utilizado para este cenário de teste tinha largura de banda de 512 kbps, essa largura de banda foi especificada na configuração 3 da tabela 05 definidas para o projeto Infravida. A figura 45 ilustra o cenário de teste 11.

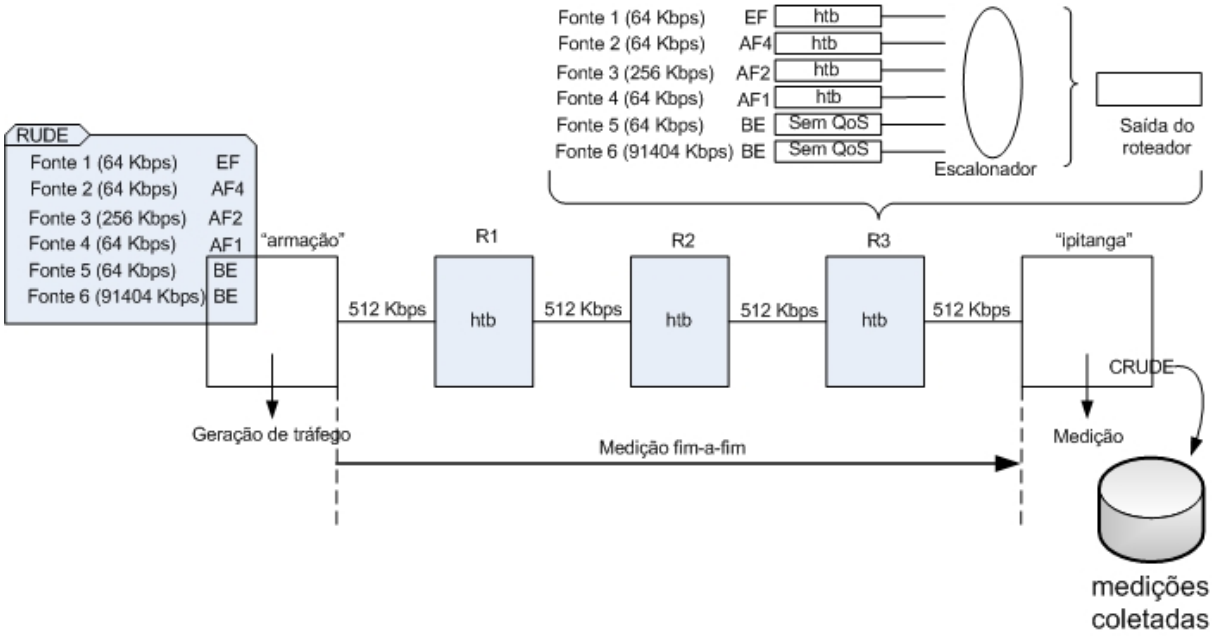

Figura 43: Cenário de teste 11

5.22 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 11

Os resultados obtidos com essa campanha de teste usando o algoritmo de escalonamento HTB atendem plenamente todos os requisitos mínimos de QoS requeridos pelas aplicações do projeto Infravida. O uso do algoritmo de escalonamento HTB nesta campanha de medição, diferentemente das campanhas anteriores 3 e 7, apresentaram resultados muito bons. O tamanho do pacote não influenciou nos resultados, talvez pelo fato do tráfego total gerado pelos fluxos ser bem menor que os cenários anteriores. Observa-se que o algoritmo de escalonamento HTB não consegue ter um bom resultado se o tráfego a ser gerenciado for muito grande, maior que 2 Mbps.

Como observado nas campanhas de medições anteriores, o fluxo pertencente à classe de serviço de melhor esforço quase não foi servido, concorrendo diretamente com o fluxo mal comportado e não tendo nenhum tipo de priorização nos roteadores. A tabela 19 exibe os resultados obtidos para esta campanha de medição.

Tabela 19: Campanha de medição – Cenário de teste 11.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	1945	63,84 kbps	9,40 ms	0,010 ms	0
Fluxo 2	466	63,37 kbps	9,54 ms	0,008 ms	0
Fluxo 3	1883	255,15 kbps	9,54 ms	0,006 ms	0
Fluxo 4	976	63,55 kbps	9,45 ms	0,010 ms	0
Fluxo 5	2	0,04 kbps	84,57 ms	75,05 ms	3265 (99,93 %)
Fluxo 6	1477	383,60 kbps	232,11 ms	123,58 ms	349484 (99,57 %)

5.23 CENÁRIO DE TESTE 12

A geração de tráfego para este cenário de teste é o mesmo do cenário 10 e 11. Um fluxo mal comportado é gerado para concorrer com os fluxos das aplicações. O tráfego é gerado pelo RUDE no hospedeiro “armação” durante 60 segundos passando pelos roteadores R1, R2 e R3 e medido pelo CRUDE no hospedeiro “ipitanga”. algoritmo de escalonamento CBQ é configurado nos roteadores para garantir QoS para as aplicações do projeto Infravida na configuração 3 da tabela 05. A figura 46 ilustra este cenário de teste.

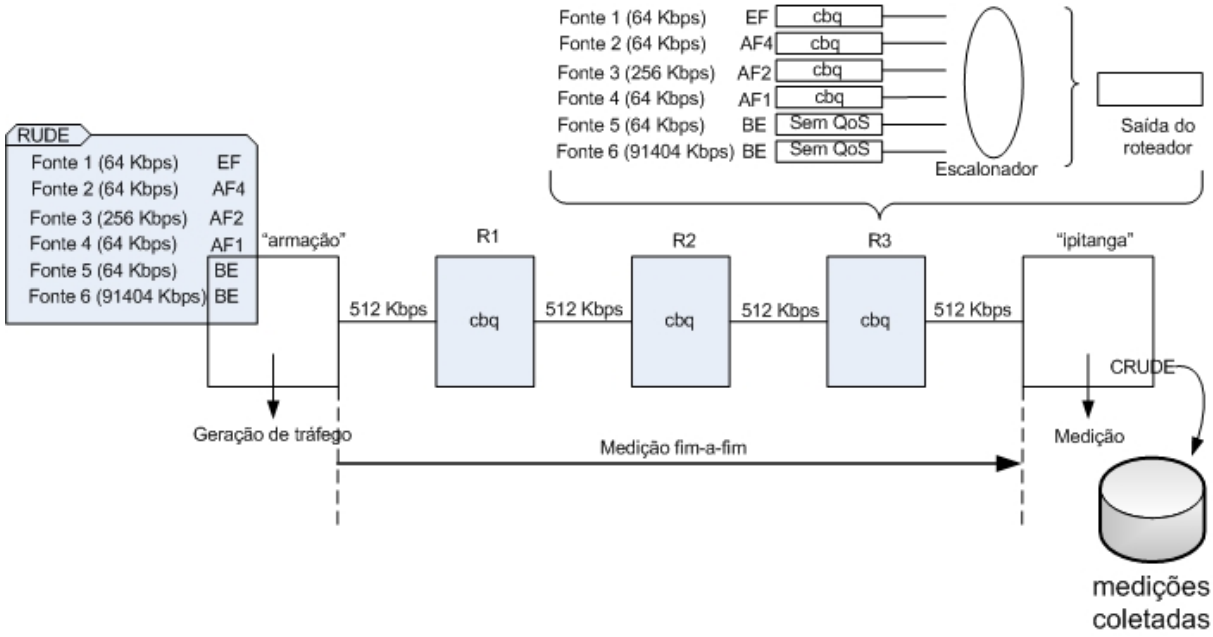

Figura 44: Cenário de teste 12

5.24 CAMPANHA DE MEDIÇÃO – CENÁRIO DE TESTE 12

Os resultados obtidos com essa campanha de medição usando como configuração de QoS o algoritmo de escalonamento CBQ apresentaram os melhores resultados como observado anteriormente nas campanhas de medição 4 e 8. Todos os requisitos de QoS requeridos pelas aplicações foram satisfeitos. Os fluxos (5 e 6) que não receberam tratamento diferenciado pelos roteadores, associados à classe de serviço de melhor esforço novamente tiveram resultados ruins, já esperados. A tabela 20 exibe os resultados medidos para esta campanha de medição.

Tabela 20: Campanha de medição – Cenário de teste 12.

	Pacotes	Vazão	Atraso	Variação de atraso (*jitter*)	Perdas (pacotes)
Fluxo 1	1945	63,84 kbps	8,58 ms	0,011 ms	0
Fluxo 2	466	63,37 kbps	8,72 ms	0,009 ms	0
Fluxo 3	1883	255,15 kbps	8,72 ms	0,006 ms	0
Fluxo 4	976	63,55 kbps	8,62 ms	0,010 ms	0
Fluxo 5	3	0,069 kbps	91,92 ms	56,66 ms	3264 (99,90 %)
Fluxo 6	1063	276,26 kbps	125,42 ms	15,52 ms	349898 (99,69 %)

O gráfico da figura 47 ilustra uma comparação dos valores medidos das perdas para a configuração 3. O gráfico compara as perdas entre o algoritmo de escalonamento HTB, CBQ e sem configuração de QoS nos roteadores para todos os fluxos. Como observado em todos as medições anteriores, os fluxos configurados com o algoritmo de escalonamento CBQ não tiveram perdas, os fluxos configurados com o algoritmo de escalonamento HTB tiveram perdas próximas a zero. Já os fluxos que não receberam tratamento nos roteadores, as perdas foram muito altas.

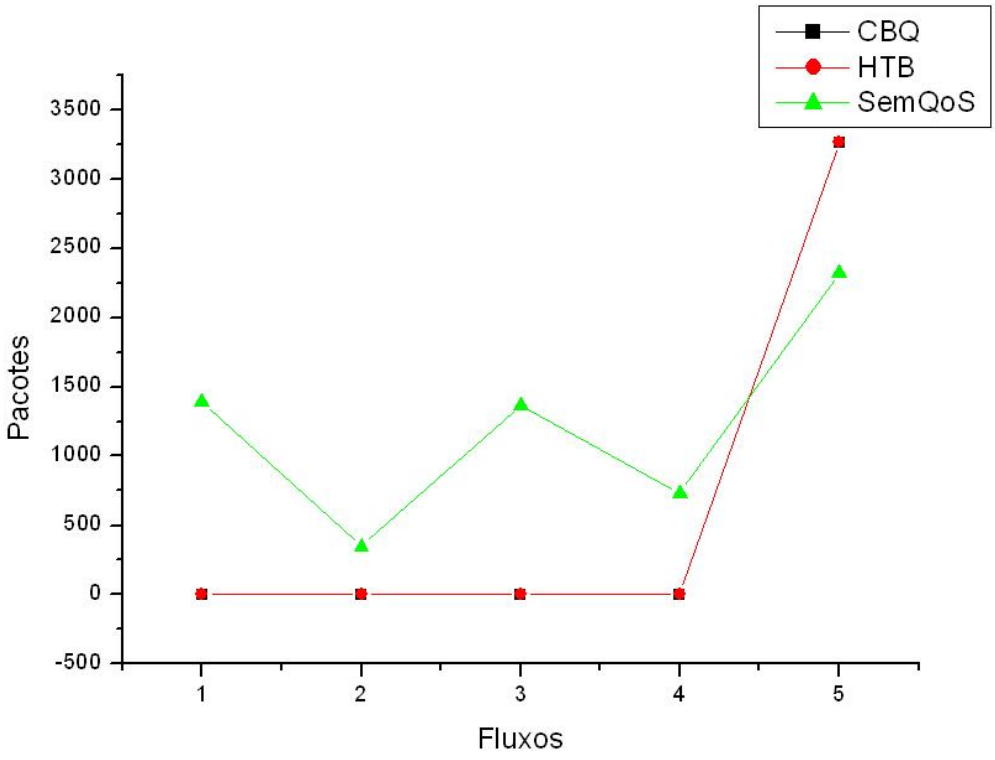

Figura 45: Perda de pacotes configuração 3 (tabela 5).

6. CONCLUSÕES

O DiffServ provê qualidade de serviço para as aplicações possibilitando a implementação de diferenciação de serviços nas redes TCP/IP utilizando agregados de tráfego e permitindo que milhares de fluxos sejam tratados.

Os objetivos deste trabalho foram a especificação, implantação e desenvolvimento e validação de uma rede protótipo experimental (*testbed*) utilizando a solução DiffServ para a abordagem do problema da qualidade de serviço e redes IP. O contexto básico para o qual a rede foi desenvolvida e validada consiste num cenário de redes IP suportando aplicações médicas no projeto Infravida. Para a validação da rede protótipo experimental, foram desenvolvidas campanhas de teste com o objetivo de demonstrar a validade e a viabilidade da especificação e implantação desenvolvidas. Levando em consideração o cenário de trabalho do desenvolvimento proposto se estudou os mecanismos de qualidade de serviço das redes IP, e se avaliou e identificou as aplicações envolvidas no projeto Infravida que necessitavam de QoS, e se identificou os requisitos de QoS destas aplicações. O trabalho ainda resultou em dois artigos publicados em congressos [Lage et al., 2004A] e em [Lage et al., 2004B].

Os resultados através da rede protótipo experimental e campanhas de teste realizadas se mostraram bastantes satisfatórios. A configuração de QoS baseada no algoritmo de escalonamento CBQ aplicados nos roteadores se mostrou o mais adequado com relação aos requisitos de qualidade de serviço requeridas pelas aplicações do projeto Infravida. Todos os parâmetros medidos utilizando essa configuração obtiveram resultados proveitosos, validando adequadamente os parâmentros de QoS especificados pelo projeto.

A configuração de QoS utilizando o algoritmo de escalonamento HTB não atendeu adequadamente os requisitos de QoS das aplicações do projeto Infravida. Alguns fluxos se comportaram de forma inesperada, particulamente fluxos que possuem um pequeno tamanho

do pacote (252 *bytes*), característico do formato do *codec* de áudio G.711. Para fluxos com essa característica foram observadas perdas além do esperado, alto atraso, além de alta variação de atraso.

De maneira geral, a especificação de QoS para as aplicações do Infravida foi mapeada adequadamente na tecnologia DiffServ, fato avaliado pelas campanhas de teste desenvolvidas. A dissertação, em resumo, procurou identificar e validar uma proposta de implantação do DiffServ com roteadores "abertos" (GNU/Linux) fazendo uso de algoritmos de escalonamento igualmente disponíveis como softwares livres, e, nesta perspectiva, teve seus objetivos avaliados e verificados.

Ainda dentro do conjunto de atividades desenvolvidas, porém não descritas nesta dissertação, foram desenvolvidas duas ações complementares:

- A integração mínima das soluções MPLS e Diffserv utilizando o protocolo RSVP-TE (*Resource Reservation Protocol – Traffic Engineering*);
- A implantação de um mecanismo de gerência dinâmica dos roteadores de *backbone.*

Como sugestão de trabalhos futuros, sugere-se a complementação destes desenvolvimentos iniciais visando a integração efetiva e avaliação dos resultados da integração do DiffServ com o MPLS. A sugestão consiste em configurar os roteadores de forma a atender os requisitos de QoS das aplicações do projeto Infravida fazendo um estudo comparativo com o presente trabalho. Além disso, as estratégias de configuração dinâmica do *backbone* podem ser exploradas.

Outra possiblidade é a realização de testes com outros tipos de aplicações fora do escopo do projeto Infravida, testando outras formas de configuração de QoS nos roteadores GNU/Linux.

REFERÊNCIAS BIBLIOGRÁFICAS

[Armitage, 2000] Armitage, G. **Quality of Service in IP Networks.** Sams Publishing, 2000.

[Balliache, 2003] Balliache, L. **Differentiated Service on Linux HOWTO.** Practical QoS, 2003. Disponível em: <http://opalsoft.net/qos/DS-21.htm>. Acesso em: 15 jan. 2006.

[Barbosa, 2001] Barbosa, K.; **HealthNet: um Sistema Integrado de Apoio ao Telediagnóstico e à Segunda Opinião Médica**. Dissertação de Mestrado, Centro de Informática/UFPE, 2001.

[Blake et al., 1998] Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. **An Architecture for Differentiated Services**. RFC 2475, 1998.

[Braden et al., 1994] Braden, R.; Clark, D.; Shenker, S**. Integrated Services in the Internet Architecture: an Overview.** RFC 1633, 1994.

[Braden et al., 1997] Braden, R.; Zhang, L.; Berson, S.; Herzog, S. **Resource Reservation Protocol (RSVP) Version 1: functional specification.** RFC 2205, 1997.

[Brown, 2003] Brown, M. **Introduction to Linux Traffic Control.** Linux On Line, 2003. Disponível em: <http://www.linux.org/docs/ldp/howto/Traffic-Control-HOWTO/index.html>. Acesso em: 24 dez. 2005.

[Charrier et al., 1999] Charrier, M.; Cruz, D.; Larsson, M. **JPEG2000, the Next Millennium Compression Standard for Still Image**. In: Proceedings of the IEEE International Multimedia Computing and Systems, Vol. 1, Florence, 1999, p. 131-132.

[Clark and Fang, 1998] Clark, D.; Fang, W. **Explicit allocation of best-effort packet delivery service** In: IEEE/ACM Transactions on Networking, v. 6, n. 4, 1998, p. 362-373.

[Comer, 1998] Comer, D. **Interligação em Rede com TCP/IP – Volume I.** Editora Campus, 1998.

[Devik, 2002] Devik, M. **HTB Linux queuing discipline manual – user guide**. 2002. Disponível em: <http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm>. Acesso em: 26 nov. 2005.

[Dibona et al., 2005] Dibona, C.; Cooper, D.;Stone, M. **Open Sources 2.0.** O'Reilly, 2005.

[DICOM, 2003] **DICOM – Digital Imaging and Communications in Medicine**. 2003. Disponível em: < http://medical.nema.org/ >. Acesso em: 14 fev. 2005.

[Durand et al., 2001] Durand, B.; Sommerville, J.; Buchmann, M.; Fuller, R. **Administering QoS in IP Networks.** Syngress, 2001.

[Floyd and Jacobson, 1993] Floyd, S.; Jacobson, V. **Random early detection gateways for congestion avoidance.** In: IEEE/ACM Transactions on Networking, Vol. 1 No. 4, 1993, p. 397-413.

[Floyd and Jacobson, 1995] Floyd, S.; Jacobson, V. **Link-sharing and Resource Management Models for Packet Networks**. In: IEEE/ACM Transactions on Networking, Vol. 3 No. 4, 1995, p. 365-386.

[Grossman, 2002] Grossman, D. **New Terminology and Clarifications for Diffserv.** RFC 3260, 2002.

[Huston and Ferguson, 1998] Huston, G.; Ferguson, P. **Quality of Service: Delivering QoS on the Internet and in Corporate Networks.** Wiley, 1998.

[IETF, 2002] **The Internet Engineering Task Force**. 2002. Disponível em: <http://www.ietf.org/>. Acesso em: 14 mar. 2006.

[Jha and Hassan, 2002] Jha, S.; Hassan, M. **Engineering Internet QoS.** Artech House, 2002.

[Lage et al., 2004A] Lage, A. L.; Martins, J. S. B.; Oliveira, J.; Cunha, W. **A Quality of Service Framework for Tele-Medicine Applications.** In: Proceedings of the WebMedia/LA-Web Joint Conference - 2nd Latin American Web Congress and 10th Brazilian Symposium on Multimedia and the Web, Ribeirão Preto, 2004.

[Lage et al., 2004B] Lage, A. L.; Martins, J. S. B.; Oliveira, J.; Cunha, W. **A Quality of Service Approach for Managing Tele-Medicine Multimedia Applications Requirements.** In: Proceedings of the IEEE Workshop on IP Operations and Management - IPOM, Vol. 1, Beijing, 2004, p. 186-190.

[Lage, 2004] Lage, A. **Qualidade de Serviço em Redes IP para Aplicações de Tele-Saúde.** Relatório Técnico, 2004.

[Laine et al., 2002] Laine, J.; Saaristo, S.; Prior, R. **Introduction to RUDE & CRUDE**. 2002. Disponível em: < http://rude.sourceforge.net/#intro >. Acesso em: 10 fev. 2006.

[Leite et al., 2001] Leite, L.; Souza, G.; Batista, T. **DynaVideo - A Dynamic Video Distribution Service**. In: Proceedings of the 6th Eurographics Workshop in Multimedia, Manchester, 2001, p. 95-106.

[Martins et al., 2003] Martins, J.; Levine, P.; Stiller, B.; Sherif, H.M.; Fumagalli, A.; Aracil, J.; Valcarenghi, L. **Managing IP Networks: Challenges and Opportunities.** Wiley-IEEE Computer Society, 2003.

[Martins, 1999] Martins, J. **Qualidade de Serviço (QoS) em Redes IP Princípios Básicos, Parâmetros e Mecanismos.** JSMNet Networking Reviews, Vol. 1, Nº 1, 1999.

[McLagan, 2003] McLagan, M. **What is Linux.** Linux On Line, 2003. Disponível em: <http://www.linux.org/info/>. Acesso em: 10 abr. 2004.

[Mills, 1992] Mills, D. L. **Network Time Protocol (Version 3) --- Specification, Implementation and Analysis.** RFC 1305, 1992.

[Mortensen, 2000] Mortensen, C. **Traffic Control features of the Linux 2.2 kernels.** 2000. Disponível em: <http://wipl-wrr.sourceforge.net/doc-wrr/tc.txt>. Acesso em: 29 nov. 2005.

[Osborne and Simha, 2002] Osborne, E.; Simha, A. **Engenharia de Tráfego com MPLS.** Editora Campus, 2002.

[Rabelo et al., 2001] Rabelo, H.; Nascimento, F.; Costa Jr, A.; Paula, V.; Souza, G. **Utilizando ACME para Descrever o Código Aberto do Projeto Open H.323.** In: Anais do VII Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, Florianópolis, 2001.

[Radhakrishnan, 1999] Radhakrishnan, S. **Linux - Advanced Networking Overview Version 1.** Department of Electrical Engineering & Computer Science, The University of Kansas, Lawrence, 1999. Disponível em: <http://qos.ittc.ku.edu/howto/node7.html>. Acesso em: 05 jan. 2006.

[Rosen et al., 2001] Rosen, E.; Viswanathan, A.; Callon, R. **Multiprotocol Label Switching Architecture.** RFC 3031, 2001.

[Kurose and Ross, 2002] Ross, W.; Kurose, J. **Computer Networking: A Top-Down Approach Featuring the Internet.** Pearson Addison Wesley, 2002.

[Semeria, 2001] Semeria, C. **Supporting Differentiated Service Classes Queue Scheduling Disciplines.** Júpiter Networks, 2001.

[Tanenbaum, 2002] Tanenbaum, A. **Redes de computadores.** Editora Campus, 2002.

[TEQUILA, 2000] **Traffic Engineering for Quality of Service in the Internet, at Large Scale.** Information Society Technologies Programme (IST), 2002. Disponível em: < http://www.ist-tequila.org/>. Acesso em: 14 set. 2004.

[Vegesna, 2001] Vegesna, S. **Ip Quality of Service.** Cisco Press, 2001.

[Werner, 1999] Werner, A**. Linux Traffic Control – Implementation Overview.** In: EPFL ICA, Lausanne, 1999.

[Westerinen et al., 2001] Westerinen, A.; Schnizlein, J.; Strassner, J.; Scherling, M.; Scherling, M.; Quinn, B.; Herzog, S.; Huynh, A.; Carlson, M.; Perry, J.; Waldbusser, S. **Terminology for Policy-Based Management.** RFC 3198, 2001.