INSTITUTO DE PESQUISAS TECNOLÓGICAS DO ESTADO DE SÃO PAULO

MARCELO GUIDO

Propostas para Estender as Funcionalidades do RSVP-TE

São Paulo

2004

MARCELO GUIDO

Propostas para Estender as Funcionalidades do RSVP-TE

Dissertação apresentada ao Instituto de Pesquisas Tecnológicas de São Paulo – IPT, para obtenção do título de Mestre em Engenharia de Computação.
Área de Concentração: Redes de Computadores

Orientador: Prof. Dr. José Roberto de Almeida Amazonas

São Paulo

2004

Guido, Marcelo

Propostas para estender as funcionalidades do RSVP-TE. / Marcelo Guido. São Paulo, 2004. 89p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Redes de Computadores

Orientador: Prof. Dr. José Roberto de Almeida Amazonas

1. Protocolo RSVP-TE 2. Controle de admissão de chamadas 3. Engenharia de tráfego (Redes) 4. Circuito virtual 5. Qualidade de serviços 6. Internet 7. Tese I. Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Centro de Aperfeiçoamento Tecnológico II. Título

CDU 004.738.5.057.4(043)
G948p

Dedicatória

Eu dedico este trabalho aos meus pais, pelos sacrifícios feitos para que eu pudesse chegar até aqui ...

À minha esposa Daniela, pelo apoio incondicional ...

Ao meu sábio avô Delvo ....

Ao meu querido, e sempre presente, avô Alexandre ...

À minha irmã, pela admiração ...

À toda minha família.

Agradecimentos

À minha esposa Daniela, pela compreensão.

Ao amigo Osório, pela ajuda na montagem deste trabalho.

Ao Professor José Roberto Amazonas, pelo apoio e orientação.

A Andrés Fernandez de Prado e Juan Carlos Luna pelas conversas esclarecedoras.

A Matias Heinrich pelo fornecimento de material bibliográfico.

À Impsat Comunicações Ltda pela oportunidade de desenvolvimento.

Resumo

Cada vez mais, as redes baseadas no protocolo IP vêm sendo largamente utilizadas, inclusive por novas aplicações que não suportam o modelo tradicional *Best Effort*. Essas novas aplicações demandam um novo modelo de rede baseado na oferta de vários níveis de qualidade de serviço. Dessa forma, aplicações mais críticas receberiam um nível de QoS mais alto, com prioridade no tratamento desse tráfego sobre os demais.

Portanto, o desafio que se apresenta nas atuais redes IP é como manter o nível de qualidade de serviço solicitado pela aplicação. Este trabalho descreve o método utilizado pelas principais arquiteturas e protocolos de rede disponíveis no mercado, como por exemplo: ATM, MPLS, DiffServ, RSVP-TE, etc, para tentar garantir o nível de QoS requisitado pela aplicação, bem como, aponta suas respectivas fragilidades que, sob determinadas circunstâncias, acabam por comprometer a qualidade fim a fim da aplicação.

Sem dúvida, a arquitetura que melhor garante qualidade de serviço fim a fim para uma aplicação é a ATM. Entretanto, diversos motivos fizeram com que a utilização dessa arquitetura fosse relegada a nichos específicos. Dessa forma, o objetivo deste trabalho consiste em propor novas funcionalidades ao protocolo RSVP-TE para contornar suas fragilidades, tornando-o mais robusto e prático em sua utilização, já que, o RSVP-TE possui várias características em comuns com o ATM, todas elas importantes para se manter o nível de QoS solicitado pela aplicação.

Para poder verificar o funcionamento das novas propostas, foram realizados vários testes em laboratório com o simulador de rede NS-2. Através dessa ferramenta, vários cenários de teste foram montados procurando evidenciar uma determinada condição da rede onde o nível de QoS ofertado a uma aplicação é mantido dentro de níveis aceitáveis, ou é degradado em virtude de um fator específico da rede. Os resultados alcançados são apresentados na forma de gráficos, mostrando o comportamento dos parâmetros que quantificam um certo nível de qualidade de serviço, os quais são: *Throughput*, *Delay*, *Jitter* e Taxa de Perda de Pacotes.

Palavras-chave: CAC - Controle de Admissão de Chamadas; Circuito Virtual; *Delay*; Engenharia de Tráfego; *Jitter*; MPLS; *Policing*; Qualidade de Serviço; Reserva de Recurso; RSVP; *Throughput*.

Abstract

More and more, the based IP protocol network has been largely used, also by new applications that do not support the traditional Best Effort model. These new applications demand a new network model based on the offering of several levels of quality of service. This way, critical applications would receive a high level of quality of service, with priority in the treatment of this traffic over the other.

Therefore, the challenge in the current IP networks is how to maintain the level of quality of service requested by the application. This work describes the method used by the main architectures and network protocols available in the market, for instance: ATM, MPLS, DiffServ, RSVP-TE, etc., to try to guarantee the level of quality of service requested by the application, and it points their respective fragilities that, under certain circumstances, compromise the application's end to end quality of service.

Undoubtedly, the best architecture that guarantees end to end quality of service for an application is ATM. However, many reasons made that the utilization of this architecture was confined to very specific niches. This way, the objective of this work is to propose new functionalities to the RSVP-TE protocol, to outline its fragilities, becoming it more robust and practical in its utilization, since the RSVP-TE has many similar characteristics with ATM, all of them important to maintain the level of quality of service requested by the application.

In order to verify how these new proposals work, some tests were made in laboratory with NS-2 network simulator. Using this tool, several test scenarios were mounted looking for highlighting a specific network condition where the QoS level offered to an application is maintained in acceptable levels, or it is degraded due to a specific characteristic of the network. The reached results are presented in graphics showing the behavior of the parameters that quantify a specific level of quality of service, such as: Throughput, Delay, Jitter and Packets Loss Rate.

Keywords: CAC – Connection Admission Control; Virtual Circuit; Delay; Traffic Engineering; Jitter; MPLS; Policing; Quality of Service; Resource Reservation; RSVP; Throughput.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AAL5 -ATM Adaptation Layer 5

ABR- Available Bit Rate

AF- Assured Forwarding

ATM – Asynchronous Transfer Mode

BA - Behavior Aggregate

BE - Best Effort

BGP-4 – Border Gateway Protocol version 4

B-ISDN - Broadband Integrated Services Network

bps – Bits por Segundo

CAC – Controle de Admissão de Chamadas

CBR – Constant Bit Rate

CBWFQ – Class Based Weighted Fair Queuing

CCITT – Comité Consultatif Internationale de Télégraphie et Téléphonie

CDV – Cell Delay Variation

CDVT – Cell Delay Variation Tolerance

CER – Cell Error Ratio

CIR - Committed Information Rate

CLR – Cell Loss Ratio

CMR – Cell Misinsertion Rate

CoS – Class of Service

CR-LDP – Constraint-based Routing Label Distribution Protocol

CTD – Cell Transfer Delay

CU – Currently Unused

DiffServ – Differentiated Services

DLCI – Data Link Connection Identifier

DSCP – Differentiated Service Code Point

EF - Expedited Forwarding

EF PHB – Expedite Forwarding Per Hop Behavior

ERO – Explicit Route Object

ERP - Enterprise Resource Planning

EXP – Experimental

FF - Fixed Filter

FIFO - First In First Out

FR – Frame Relay

FTP – File Transfer Protocol

IETF – Internet Engineering Task Force

IGP – Interior Gateway Protocol

IntServ – Integrated Services

IP – Internet Protocol

ITU - International Telecommunication Union

ITU-T - International Telecommunication Union – Telecom Standardization

L3PID – Layer 3 Protocol Identification

LBNL - Lawrence Berkeley National Laboratory

LDP – Label Distribution Protocol

LER – Label Edge Router

LIB – Label Information Base

LLQ – Low Latency Queue

LSP – Label Switched Path

LSR - Label Switch Router

Mbps – Megabits por segundo

MBS – Maximum Burst Size

MBZ – Must Be Zero

MCR – Minimum Cell Rate

MF - Multifield Classifier

MPLS – MultiProtocol Label Switching

nrt-VBR – non real Time Variable Bit Rate

NS-2 – Network Simulator version 2

OSPF – Open Shortest Path First

OSPF-TE - Open Shortest Path First with Traffic Engineering

PCR – Peak Cell Rate

PDH – Plesiochronous Digital Hierarchy

PHBs - Per Hop Behaviors

PNNI - Private Network to Network Interface

PQ - Priority Queuing

QoS – Quality of Service

RFC – Request for Comments

RRO – Record Route Object

RSVP – Resource ReSerVation Protocol

RSVP-TE – Resource ReSerVation Protocol with Traffic Engineering

rt-VBR – Real Time Variable Bit Rate

S – Stack

SCR – Sustainable Cell Rate

SDH – Synchronous Digital Hierarchy

SE – Shared Explicit

SECBR – Severely Errored Cell Block Ratio

SLA – Service Level Agreement

TCP – Transmission Control Protocol

ToS – Type of Service

TTL – Time to Live

UBR – Unspecified Bit Rate

UDP – User Datagram Protocol

VCC – Virtual Channel Connection

VoIP – Voice over Internet Protocol

WDM – Wavelength Division Multiplex

WF – Wildcard Filter

Sumário

Capítulo 1 - Introdução

As redes baseadas em protocolo IP – *Internet Protocol* vêm experimentando um rápido crescimento de sua infra-estrutura desde a popularização do uso da Internet. Não foi só a Internet que passou por um grande crescimento, mas as redes coorporativas das empresas, gradativamente vêm trocando sua infra-estrutura de rede com protocolos Frame Relay e X.25 para o protocolo IP. A maioria das *Carriers* começa a oferecer aos clientes novos serviços baseados em redes puramente IP, fazendo com que a infra-estrutura de rede, WDM - *Wavelength Division Multiplex*, SDH - *Synchronous Digital Hierarchy*, PDH - *Plesiochronous Digital Hierarchy*, ATM - *Asynchronous Transfer Mode* e FR - *Frame Relay*, esteja passando a ser suporte da rede IP. Entretanto, essa rápida expansão trouxe também novos desafios: como prover qualidade de serviço nas redes IP, já que nessas redes, tradicionalmente, existe somente um tipo de operação/funcionamento, conhecido como *Best Effort*, onde não há nenhuma garantia, nem mesmo de entrega.

Muitas empresas estão utilizando a Internet para fazer seu gerenciamento administrativo diário, estando dispostas a pagar mais para obter um melhor nível de serviço da Internet. Analogamente, muitos usuários estariam dispostos a pagar mais por um melhor acesso à Internet, para utilizarem aplicações sob demanda como Telefonia IP e videoconferência. Em contrapartida, existem muitos mais usuários que querem utilizar a Internet apenas para serviços elementares, como envio de *e-mail* ou simplesmente navegar pela WEB, pagando menos por isso.

Há poucos anos atrás, as aplicações mais utilizadas na Internet eram *e-mail*, FTP - *File Transfer Protocol* e *newsgroups*. Entretanto, as aplicações evoluíram bastante, devido ao surgimento de novas estruturas de dados e informações que são transferidas pela Internet, como voz, música, *Java Scripts* e *hypertext links*. O resultado dessa mudança, tanto nas expectativas dos usuários como nas aplicações Internet, é que existe uma demanda crescente de usuários interessados em mudar o paradigma de mesmo serviço para todos, para um modelo onde usuários e aplicações são diferenciados segundo suas necessidades de qualidade de serviço. (Dovrolis & Ramanathan, 1999, p.1)

O interesse por aplicações modernas via Internet vem aumentando fortemente entre provedores de serviços e potenciais clientes. Alguns dos serviços atuais e os novos serviços emergentes requerem um alto nível de qualidade, exigindo grande confiabilidade e disponibilidade da rede. Aplicações *real time*, que são muito sensíveis ao tempo de transmissão e *jitter* (variação no tempo de transmissão), e requerem altas capacidades garantidas de banda, são um bom exemplo dessas novas aplicações emergentes.

Aliado às necessidades dessas novas aplicações, e com a escassez de banda passante em alguns enlaces de comunicação que compõem a Internet, apesar de na maioria dos casos ser momentânea e passageira, a necessidade pelo desenvolvimento de mecanismos específicos para assegurar qualidade de serviço para determinados tipos de aplicações aumentou consideravelmente. Muitas pessoas e organizações pelo mundo todo estão dirigindo suas atenções para o tema de qualidade de serviço. Os

efeitos desses esforços podem ser observados facilmente, dado que, atualmente, várias arquiteturas e protocolos de rede, que suportam garantia de entrega de vários níveis de qualidade de serviço, estão disponíveis no mercado, e continuam em desenvolvimento. (Gozdecki, Jajszczyk, & Stankiewicz, 2003)

Com a rápida expansão das redes IP, várias aplicações que até então possuíam seus próprios meios de comunicação, como por exemplo voz e vídeo, migraram para as redes de pacotes para baratear custos operacionais, exigindo assim uma maior qualidade de serviço da rede, pois o modelo tradicional *Best Effort* não as atendia mais. Nessa busca por melhor QoS – *Quality of Service,* para aplicações críticas, foram desenvolvidos vários protocolos e arquiteturas que procuram garantir qualidade de serviço, como por exemplo: ATM, MPLS – *Multiprotocol Label Switching*, RSVP-TE – *Resource Reservation Protocol with Traffic Engineering*, DiffServ – *Differentiated Services* entre outras. (Thomas, 2001) (Mcdysan, 1999)

1.1 - Motivação

Dentre todos os protocolos e arquiteturas existentes atualmente no mercado, nenhum deles trabalha tão bem com a garantia de níveis de qualidade de serviço para diferentes aplicações como a arquitetura ATM.

Essa arquitetura, baseada em circuitos virtuais, possui vários mecanismos para assegurar um determinado nível de qualidade de serviço para uma aplicação, como por exemplo: a) mecanismo de reserva de banda baseado nas características específicas de tráfego de uma aplicação (banda garantida, tamanho de rajada de dados, *delay* fim a fim, *jitter*, etc); b) controle de admissão de novas chamadas; c) policiamento do tráfego de entrada na rede comparando-o ao acordado com o cliente.

Apesar do ATM ser a melhor tecnologia para garantir QoS, sua implementação e operacionalização se mostraram muito onerosas afetando sua utilização em larga escala. Com isto, foram desenvolvidas poucas aplicações que suportam essa arquitetura sendo que sua utilização no mercado atual é restrita a nichos específicos.

A motivação consiste em trazer as fortes características de garantia de qualidade de serviço da arquitetura ATM para o mundo IP, sendo que já existe disponível no mercado, um protocolo de rede IP também baseado em reserva de recursos e em circuitos virtuais. Essa última característica ficou fortemente definida com o advento do MPLS. Esse protocolo é o RSVP-TE, uma evolução do protocolo RSVP. O RSVP-TE é um protocolo de sinalização utilizado nas redes MPLS. (Gallaher, 2001)

Portanto, o protocolo RSVP-TE foi escolhido para ter suas características estendidas para manter qualidade de serviço fim a fim, mesmo em situações de contingência, por possuir características importantes para este fim, como por exemplo:

a) fluxo de dados baseados em circuitos virtuais;

b) reserva de recursos para um determinado fluxo;
c) controle de admissão de novos fluxos;
d) mecanismo de *policing* (controle da entrada de tráfego na rede);

Esses recursos são importantes para manter a qualidade de serviço desejada para os fluxos já estabelecidos na rede, independentemente de problemas que venham a ocorrer, como congestionamentos, ainda que em situações de contingência.

Além das características de reserva de recurso, *policing*, e ser baseado em circuitos virtuais, o protocolo RSVP está disponível no mercado para implementação desde o final de 1997 e, portanto, possui muitas de suas funcionalidades já depuradas.

1.2 - Características Históricas do Protocolo RSVP/RSVP-TE

Na metade da década de 90, o protocolo RSVP (Braden et al., 1997) versão 1 foi desenvolvido para contornar congestionamentos de rede, pela inteligência proporcionada aos roteadores para decidir, antecipadamente, qual o caminho que atenderia às necessidades do fluxo de tráfego de uma aplicação e então, reservar os recursos necessários se eles estivessem disponíveis. Em 1997 o RSVP tornou-se padrão através da RFC 2205.

O protocolo RSVP-TE (Awduche et al., 2001) é uma extensão da versão original do protocolo RSVP para suportar o estabelecimento de LSP – *Label Switched Path,* dentro de uma rede que utilize a arquitetura MPLS como infra-estrutura. Muitas das novas características adicionadas ao RSVP-TE foram motivadas pela necessidade de se realizar engenharia de tráfego em redes MPLS, em particular o suporte a rotas LSP explícitas (definidas manualmente ou não), com ou sem reserva de recursos.

Dentro de uma rede MPLS, é necessário existir um protocolo de roteamento IP, normalmente IS-IS, OSPF ou BGP4 (Black, 2000), para que um caminho possa ser encontrado e, portanto, alcançar o nó de destino. Trabalhando em paralelo com esse protocolo de roteamento, existe um protocolo de sinalização que, além de fazer a distribuição de *labels* entre as diversas interfaces que compõem um caminho, será o responsável por verificar se o caminho escolhido pelo protocolo de roteamento atende às necessidades de QoS de uma determinada aplicação, ou seja, esse protocolo de sinalização é o responsável por fazer engenharia de tráfego da rede, pois nem sempre o caminho mais curto é o melhor. Portanto, o protocolo RSVP-TE trabalha como sendo o protocolo de sinalização de uma rede MPLS. Entretanto, a padronização dessa arquitetura não define a utilização de um protocolo único de distribuição de *label*s, podendo haver outros.

No início de 1997, os primeiros projetistas da arquitetura MPLS tinham várias razões para decidir por estender as funcionalidades do RSVP, para suportar a distribuição de *label*s do MPLS com engenharia de tráfego, em vez de desenvolver outro protocolo de distribuição de *labels*:

a) O RSVP foi originalmente desenvolvido para ser o protocolo de reserva de recurso da Internet e prover mecanismos para criar e manter reservas distribuídas ao longo de caminhos *multicast* e *unicast*. Reserva de recurso é um importante componente da engenharia de tráfego. Assim, fazia mais sentido continuar a se utilizar o RSVP do que iniciar todo um processo de desenvolvimento de um novo protocolo.

b) O RSVP foi especificamente desenvolvido para suportar o tráfego de objetos opacos, ou seja, pedaços de informação que são entregues a módulos de controle específicos dos elementos de rede. Essa característica facilitou o desenvolvimento de outros recursos simples no RSVP, como rotas explícitas e distribuição de *labels*, adicionando-os à reserva de recurso.

As implementações propostas no RSVP-TE não seriam incompatíveis com a versão original do RSVP. Uma implementação com RSVP pode facilmente diferenciar entre uma sinalização LSP e o RSVP padrão pela simples análise dos objetos da mensagem.

Ainda existe um outro protocolo de sinalização utilizado pelo MPLS, que é o CR-LDP – *Constraint-based Routing Label Distribution Protocol* (Jamoussi et al., 2002). Esse protocolo evoluiu de maneira inversa ao RSVP, ou seja, o protocolo LDP, que fazia simplesmente a distribuição de *labels*, foi estendido para suportar engenharia de tráfego. A principal diferença com relação ao RSVP-TE é que o CR-LDP é do tipo *Hard State* (uma vez estabelecido o circuito virtual, ele não será desfeito até que um pedido de desconexão seja enviado), ao passo que o RSVP-TE é do tipo *Soft State* (mensagens de verificação do estado operacional do circuito virtual são trocadas periodicamente).

Entretanto, em virtude da baixa utilização do CR-LDP e da duplicação de esforços, o grupo de trabalho de MPLS do IETF – *Internet Engineering Task Force* decidiu não direcionar novos esforços ao CR-LDP, concentrando seu trabalho no desenvolvimento do RSVP-TE, como o protocolo de sinalização para aplicações que requeiram engenharia de tráfego no MPLS. (Andersson & Swallow, 2003 p.7)

1.3 - Outros Trabalhos Correlatos

Como já foi mencionado, vários organismos, empresas e entidades, ou seja, a comunidade internacional, como um todo, tem se esforçado para alcançar o objetivo de garantir melhores níveis de qualidade de serviço para aplicações mais exigentes em redes IP.

Os trabalhos e propostas que estão sendo desenvolvidos apresentam vários métodos para se alcançar a garantia de qualidade de serviço fim a fim para uma determinada aplicação, não havendo um consenso claro de como alcançá-lo. Alguns desses trabalhos e seus métodos para tentar garantir qualidade de serviço em redes IP para uma determinada aplicação, são:

a) **Projeto AQUILA** (Aquila, 2000). Esse trabalho baseia-se na construção de um *backbone*, com os equipamentos de *core* utilizando a tecnologia DiffServ e no desenvolvimento de um protocolo de sinalização entre os equipamentos de *edge* e os *hosts*. Essa sinalização é necessária para que os *hosts* possam fazer requisições de níveis de qualidade de serviço à rede, sendo que essas requisições trafegam na rede de *host* a *host*, ou seja, fim a fim. Baseia-se também na definição de um mecanismo de controle de recursos de rede hierarquizado, através de um *Admission Control Agent* em cada equipamento de *edge* da rede. (Engel et al., 2003)

b) **Projeto Tequila** (Tequila, 2000). Implementa um modelo de mecanismo de controle de admissão de tráfego na borda da rede, baseado em seu estado de utilização, através do recebimento de *feedbacks* da rede. Esse *feedback* pode ser feito de duas formas: das funções de engenharia de tráfego trabalhando *off-line* ou através de medidas feitas no estado atual da rede. Dessa forma, o controle do tráfego a ser admitido na rede pode ser feito baseado em modelos matemáticos e estatísticos, ou no próprio *feedback* da rede. (Mykoniati et al., 2003)

c) **Rede GÉANT**. Trata-se de uma rede européia de comunicação de dados com velocidade de 10 Gbps no *backbone* entre os principais países, interligando vários centros educacionais e de pesquisa da comunidade européia. Trata-se de uma rede DiffServ tradicional com os níveis de qualidade de serviço baseados no comportamento EF PHB – *Expedit Forwarding Per-Hop Behavior*, ou seja, são baseados nas prioridades das filas de um algoritmo de enfileiramento de pacotes. Basicamente resolve o problema de qualidade de serviço com o simples aumento de velocidade dos enlaces de comunicação que sejam potenciais gargalos na rede. (Roth et al., 2003)

d) ***Draft* OSPF-TE**. *Open Shortest Path First* com *Traffic Engineering* . Existem estudos sendo conduzidos para possibilitar ao protocolo de roteamento IP dinâmico OSPF fazer engenharia de tráfego dentro do MPLS. Dessa forma seriam criadas algumas extensões do OSPF para que ele possa agregar funcionalidades de garantias de qualidade de serviço em redes IP, com a tecnologia DiffServ configurada. (Katz, Yeung & Kompella, 2002)

Portanto, existem várias contribuições e pesquisas sendo feitas pela comunidade internacional sobre o mesmo tema: garantia de qualidade de serviço em redes IP, sendo que todas elas apresentam uma abordagem diferente para se resolver o mesmo problema. Todas as propostas apresentam pontos positivos, que é basicamente a boa escalabilidade de tráfego na rede, e pontos negativos, como: desenvolvimento de sinalização entre *host* e *edge* gerando maior complexidade no *host*, demora da rede em avisar os mecanismos que controlam a entrada de tráfego na

rede podendo congestioná-la, desperdício de recursos financeiros por sempre estar aumentando a banda passante dos enlaces de comunicação, entre outros.

1.4 - Objetivo

Este trabalho tem como objetivo utilizar o protocolo RSVP-TE, em conjunto com a arquitetura MPLS, para, de uma maneira prática e com mecanismos disponíveis no mercado, manter a qualidade de serviço desejada fim a fim para uma aplicação, mesmo em situações de congestionamento e contingência, quando normalmente se forma um gargalo em algum ponto da rede, onde os recursos são inferiores aos necessários para acomodar todo o tráfego re-roteado. Nessas situações, dependendo da arquitetura da rede, pode acontecer uma das duas alternativas descritas a seguir:

a) deixar a rede descartar pacotes aleatoriamente, segundo o tamanho de cada *buffer*, de todas as filas de prioridade do equipamento onde está ocorrendo o gargalo. Nessa situação todas as aplicações sofrerão degradação da qualidade de serviço, inclusive as mais críticas. Entretanto, as aplicações poderão continuar operando dependendo de sua tolerância a erros e retardos na rede.

b) garantir que o fluxo das aplicações mais críticas, já estabelecidas na rede antes da ocorrência de gargalo, continuem a operar sem perda de qualidade de serviço. Nesse caso, os recursos ainda disponíveis dessa interface serão alocados para os fluxos de maior prioridade. Quando os recursos se esgotarem, nenhum outro fluxo de dados será estabelecido. Portanto, haverá aplicações inoperantes, mas as que continuarem manterão seu nível de qualidade de serviço. Essa característica é extremamente importante para aplicações de VoIP – *Voice over Internet Protocol* e videoconferência. No caso de uma aplicação de voz, por exemplo, é preferível que não se estabeleça novas chamadas, do que possuir mais chamadas estabelecidas e todas apresentando degradação do nível de qualidade de serviço requisitado.

Neste trabalho é tratada a situação caracterizada no item b.

Em condições normais de operação de uma rede, todas as aplicações devem estar com seus níveis de qualidade de serviço requisitados sendo atendidos pela rede. Entretanto, em situações de contingência as aplicações de mais baixa prioridade poderão experimentar degradação do nível de qualidade de serviço contratado junto ao provedor ou operadora.

O protocolo RSVP/RSVP-TE apresenta duas fragilidades que impediram que sua utilização também fosse realizada em larga escala. São elas:

a) Para manter a qualidade de serviço fim a fim entre dois *hosts*, é necessário que os mesmos possuam um *software client* RSVP rodando, para poderem conversar com os demais equipamentos da

rede, que também possuem o protocolo RSVP carregado, estabelecendo assim o circuito virtual entre eles.

Bernet em (Bernet, 2000, p. 155) afirma que a necessidade de se utilizar um *software* RSVP rodando no cliente tornou a utilização desse protocolo pouco atrativa. Na verdade, somente algumas versões experimentais tornaram-se disponíveis para algumas plataformas UNIX.

b) O conceito de qualidade de serviço da arquitetura IntServ, utilizada pelo protocolo RSVP-TE, está definido de maneira qualitativa, como o comportamento que a rede apresenta em situações de baixa ou alta carga.

Wroclawski em (Wroclawski, 1997a, p. 2) afirma que: "O comportamento fim a fim de uma aplicação *controlled load service* fornecido pelos vários elementos de rede, se aproxima do comportamento visto pelas aplicações *best effort* quando a rede não está em condições de carga pesada...".

Portanto, as características inovadoras deste trabalho consistem em contornar as fragilidades descritas do RSVP-TE, através da proposta de duas novas funcionalidades ao protocolo, sendo que a primeira atuará no equipamento de borda da rede, e a segunda no equipamento de *core* da rede. Essas novas funcionalidades trazem as seguintes vantagens para o protocolo:

a) Não é necessário haver sinalização entre o *host* e o equipamento de *edge*, ou um *software client* RSVP rodando no *host*, garantindo simplicidade ao equipamento do cliente;
b) Independe de *feedbacks* da rede e, portanto, do risco da rede demorar para enviá-los em situações de congestionamento;
c) Garante uma utilização racional dos recursos da rede;
d) Todos os demais recursos e características existentes no RSVP-TE continuam existindo;

Essas novas funcionalidades, mostram uma integração entre duas diferentes tecnologias de rede IP: a tecnologia IntServ, baseada em reserva de recursos e circuitos virtuais, e a tecnologia DiffServ baseada na configuração de valores pré-determinados do campo DSCP – *Differentiated Service Code Point* de cada pacote IP trafegado na rede.

Existem muitas opiniões divergentes sobre onde essas arquiteturas devem ser utilizadas. Várias correntes pregam o uso da arquitetura DiffServ no *backbone* da rede e IntServ na rede do cliente, ou na borda da rede, como em (Bernet, 2000). Já em (Fineberg, 2002), é proposto um esquema onde se utilizaria RSVP-TE no *backbone* e na rede do cliente, onde os equipamentos de LAN também suportariam os esquemas de reserva de recursos e controle de admissão de conexões. Outros

ainda, apregoam que se houver problemas com qualidade de serviço na rede, a solução é aumentar a banda passante disponível, já que com a grande utilização das fibras ópticas nas redes de telecomunicações, e o avanço no desenvolvimento tecnológico dos equipamentos de transmissão, o custo dos canais de comunicação oferecidos pelas *carriers* será cada vez mais barato. (Roth et al., 2003)

Essa última afirmação não deixa de possuir um certo grau de verdade pois, especialmente no Brasil, com a quebra do monopólio das telecomunicações, o preço de um canal de comunicação vem caindo bastante, motivado, em parte, pela forte concorrência. Entretanto, toda empresa economicamente saudável visa sempre redução dos custos, buscando otimização dos recursos, motivo pelo qual não houve uma explosão de consumo de banda larga por parte das empresas.

Entretanto, esta proposta também apresenta pontos negativos que é a escalabilidade de tráfego no *backbone* da rede, já que nessa arquitetura haverá um circuito virtual para cada nível de qualidade de serviço exigido pelos clientes.

1.5 - Organização da Dissertação

Esta dissertação está estruturada em cinco capítulos, sendo que o primeiro, como já foi descrito, faz uma introdução à necessidade de manter a qualidade de serviço em redes IP. Descreve a motivação do trabalho, comparando-o a outros realizados pela comunidade científica internacional. Este capítulo também apresenta os objetivos do trabalho, destacando as características inovadoras das propostas.

O Capítulo 2 descreve as características fortes de QoS nas redes baseadas em circuitos (*connection oriented*) e fracas nas redes baseadas em pacotes (*connectionless*). Define os conceitos básicos de funcionamento das principais tecnologias e protocolos que procuram garantir níveis de qualidade de serviço, de acordo com o requisitado pela aplicação, são elas: ATM, DiffServ, MPLS e RSVP-TE.

Também é descrito porque estas tecnologias e arquiteturas, sozinhas, não conseguem manter a qualidade de serviço fim a fim para uma aplicação, dependendo do comportamento e do estado de utilização da rede. O capítulo termina com a definição do problema enfrentado pelas redes IP, referente à manutenção da qualidade de serviço para aplicações já estabelecidas na rede.

O Capítulo 3 define as novas propostas para resolver os problemas descritos estendendo as funcionalidades do protocolo RSVP-TE, tornando-o mais robusto e simples de se utilizar. Essas novas funcionalidades atuam nos equipamentos de borda e *backbone* da rede. Após a descrição detalhada das novas funcionalidades, existe um exemplo de funcionamento do RSVP-TE com as novas propostas já implementadas. É definido também o ambiente onde essas propostas são aplicáveis.

O Capítulo 4 trata dos benefícios adicionados ao protocolo RSVP-TE, com a adoção das novas funcionalidades propostas aos equipamentos de *edge* e *core*. Esse capítulo ainda trata da descrição e comentários dos resultados dos testes realizados

por simulação do ambiente de rede proposto, através do NS-2 – *Network Simulator version* 2, bem como as limitações encontradas durante a execução dos testes.

O Capítulo 5 apresenta as conclusões finais do trabalho, mostrando algumas sugestões para a continuidade da pesquisa.

Após o Capítulo 5, são indicadas as Referências Bibliográficas consultadas e citadas neste trabalho.

O trabalho é encerrado com os Anexos, trazendo os *scripts* de programação utilizados nos testes em ambiente simulado.

Capítulo 2 - Qualidade de Serviço nas Redes IP

Qualidade de serviço pode ser definida como sendo a habilidade da rede em garantir e manter determinados níveis de desempenho para cada tipo de aplicação, de acordo com suas necessidades.

Sua implementação é feita utilizando-se vários mecanismos em conjunto, otimizando a utilização dos recursos da rede e direcionando-os para aplicações mais prioritárias. Portanto, a simples adição de banda passante nos circuitos de comunicação de dados não constitui implementação de qualidade de serviço em uma rede. Mesmo porque, segundo o Corolário de Moore (Reed, 2001), ao incrementar a capacidade de qualquer sistema para acomodar a demanda do usuário, essa demanda aumentará para consumir sua capacidade.

De maneira geral, as aplicações geram tráfego em várias velocidades e, idealmente, necessitam que sejam transportados pela rede na mesma velocidade com que são gerados. Como isso não é possível, as aplicações podem ser mais ou menos tolerantes ao tempo de entrega desse tráfego no destino, bem como a variações desse tempo de entrega. Algumas aplicações podem, ainda, tolerar um certo grau de perda de tráfego, enquanto outras necessitam de uma reduzida taxa de perda de pacotes.

Em uma rede ideal, e imaginária, com recursos infinitos, o tráfego seria transportado na velocidade desejada, com taxas zero de latência e de perda de pacotes. Já que essas redes não existem, haverá circuitos de comunicação onde os recursos estarão indisponíveis para garantir a qualidade de serviço requisitada pelas aplicações (Bernet, 2000). Como conseqüência, haverá o surgimento de congestionamento na rede, podendo ser caracterizado pela incapacidade de um determinado equipamento transmitir adiante todo o tráfego recebido.

Dentro do conceito de qualidade de serviço, existem vários níveis de serviço que a rede pode oferecer a uma aplicação, sendo conhecidos como: classes de serviço, ou CoS – *Class of Service* (Crawley, et al., 1998). Essas classes de serviço variam desde as que fornecem rígidas garantias de envio de tráfego (maior prioridade) até as que não fornecem nenhuma garantia (menor prioridade). O número de classes de serviço que uma rede pode oferecer, está relacionado ao tipo de arquitetura e protocolo utilizado por ela. Por exemplo, o ATM define cinco classes de serviço; as redes IP originais definiam apenas uma classe de serviço; o RSVP-TE define três classes e a arquitetura DiffServ padroniza, inicialmente, também três classes.

As classes de serviço oferecidas pela rede são definidas por um conjunto de parâmetros cujos valores indicam o quanto o tráfego foi penalizado ao passar pela rede, até alcançar o destino. Quanto mais o tráfego for penalizado, menos prioritária será sua classe de serviço. Esses parâmetros são:

a) ***Throughput***: vazão líquida com que os dados são transferidos na rede através dos vários elementos de rede.

b) ***Delay***: tempo que um pacote demora para trafegar entre dois pontos da rede. O *delay* fim a fim de uma aplicação é dado pela somatória dos tempos de transmissão dos pacotes nos circuitos de comunicação e do tempo gasto com roteamento e enfileiramento dos pacotes nas filas de saída em cada elemento, entre os pontos de entrada e saída da rede.

c) ***Jitter***: variação nos valores de *delay*.

d) **Taxa de Perda de Pacotes**: porcentagem do número de pacotes enviados pela fonte, que são recebidos no destino.

e) **Taxa de Erro**: porcentagem do número de pacotes enviados pela fonte, que são recebidos com erro no destino.

Quando um cliente contrata um determinado serviço de uma operadora, ou *carrier*, um acordo entre as duas partes é firmado, sendo esse acordo conhecido como: SLA – *Service Level Agreement*. De acordo com o ITU – *International Telecommunication Union*, o SLA deve conter as seguintes informações (ITU-T Y.1241, 2001):

a) Valores dos parâmetros que definem a classe de serviço contratada: *throughput*, *delay*, *jitter* e taxa de erro/perda de pacotes.

b) Disponibilidade e confiabilidade do serviço.

c) Método de monitoração dos parâmetros de classe de serviço e da disponibilidade do serviço.

d) Métodos de autenticação.

e) Compensações financeiras em caso de não cumprimento do SLA.

2.1 - Redes Orientadas a Conexão e Não Orientadas a Conexão

Basicamente, as redes de comunicação de dados são classificadas em dois tipos: orientadas a conexão (*connection oriented*) e não orientadas a conexão (*connectionless*).

As redes orientadas a conexão se caracterizam pela troca de sinalização entre os *hosts* antes de haver transferência de dados entre eles, ocorrendo três etapas bem definidas: estabelecimento da conexão, troca de dados propriamente dita e encerramento da conexão.

Usualmente, redes orientadas a conexão estão associadas a circuitos virtuais, ou seja, caminhos virtuais fim a fim entre os *hosts*, os quais podem ser estabelecidos manualmente pelo administrador da rede ou, automaticamente, através de um protocolo de sinalização. Também existe a garantia de que os dados serão entregues no destino na mesma seqüência com que foram inseridos na rede. Nos *hosts*, as aplicações utilizam esse caminho para trafegar seus dados, sendo que não há interferência entre os vários circuitos virtuais que compõem uma rede. Cada circuito virtual possui características próprias de *throughput*, *delay*, *jitter* e taxa de erro/perda de pacotes.

Entretanto, existem alguns casos em que uma rede orientada a conexão pode tratar o tráfego por pacote em seu *backbone*, ainda assim, mantendo o conceito de circuito virtual para o cliente que está conectado na borda da rede. Dessa forma, a rede irá armazenar os pacotes no destino por meio de *buffers*, para que a ordem de entrega dos mesmos seja mantida.

Dependendo do protocolo de sinalização utilizado, além de estabelecer os circuitos virtuais, ele também realiza reserva de recurso ao longo do caminho, segundo o requisitado por uma aplicação. Em conjunto com este processo, existe um outro mecanismo, conhecido como CAC – Controle de Admissão de Chamadas, que verifica a disponibilidade de recursos nos elementos de rede que compõem o caminho. Se não houver recursos disponíveis o circuito não é estabelecido. Uma vez estabelecido o circuito, não haverá outras consultas de disponibilidade de recursos, ficando ativo até que uma das pontas desative esse circuito. Outro importante mecanismo, que atua após o estabelecimento da conexão, é o de *policing*, que verifica se o tráfego entrante na rede está de acordo com o SLA firmado entre cliente e provedor.

As redes IP que utilizam os protocolos da arquitetura IntServ (RSVP, RSVP-TE) são exemplos de redes orientadas a conexão. As arquiteturas ATM e *Frame Relay* e as redes de telefonia também são exemplos desse tipo de rede.

Nas redes não orientadas a conexão, não existe sinalização sendo trocada entre os elementos de rede, e portanto, também não existe o conceito de circuito virtual utilizado por uma aplicação, ou *host* específico. Nessas redes, o encaminhamento dos pacotes é feito através da análise de seu cabeçalho, o qual possui informações do destino, sempre que o pacote chega em um novo elemento de rede.

Nas redes IP tradicionais, que eram essencialmente *connectionless*, não existiam mecanismos de controle de admissão de tráfego ou de *policing*. Entretanto, com o desenvolvimento da arquitetura DiffServ, mecanismos similares foram criados para dar às redes IP a possibilidade de controlar a qualidade de serviço. Nessas redes, o tráfego é classificado somente na entrada da rede, onde também é controlada sua admissão, e direcionado para as filas de transmissão correspondentes à classe de serviço requisitada pelo pacote IP. Desta forma, os vários fluxos de dados de clientes entrantes na rede são agregados em fluxos maiores, de acordo com o número de classes de serviço existentes na rede.

O protocolo UDP – *User Datagrama Protocol*, as redes IP tradicionais e as redes baseadas na arquitetura DiffServ são exemplos de redes não orientadas a conexão.

A principal diferença entre as duas arquiteturas é que nas redes orientadas a conexão é possível se estabelecer um controle bastante apurado do tráfego que entra na rede para cada fluxo de dados de cada cliente, já que cada fluxo está associado a um circuito virtual. Essa característica é importante quando houver na rede situações

de congestionamento, onde as aplicações já estabelecidas na rede não deverão sofrer degradação do nível de qualidade de serviço contratado. Por outro lado, a necessidade de se manter muitas conexões na rede, pode gerar um problema de escalabilidade.

Nas redes não orientadas a conexão, o controle da qualidade de serviço é feito por pacote e por grandes fluxos de dados que agregam o tráfego de vários clientes. Portanto, na ocorrência de um congestionamento provocado por uma situação de contingência na rede, todas as aplicações pertencentes a uma determinada classe de serviço sofrerão degradação do nível de qualidade de serviço contratado, mesmo as aplicações já estabelecidas na rede. Entretanto, essas redes não apresentam problemas de escalabilidade.

A tabela 1 resume as principais características das redes Orientadas a Conexão e Não orientadas a Conexão.

Tabela 1: Características das redes Orientadas a Conexão e Não orientadas a Conexão.

	Connection Oriented	***Connectionless***
Atuação	Fim a Fim	Local (*per Hop*)
Nível de Atuação	Fluxo Individual	Agregação de Fluxos
Utilização de Recursos (*Scheduling*, *Buffering*, etc)	Fluxo Individual	Agregação de Fluxos
Protocolo de Sinalização	Existente	Não Existente
Reserva de Recursos	Existente	Não Existente
Controle de Admissão de Chamadas	Existente	Não Existente
Policing	Baseado no Solicitado pela Conexão	Baseado em valores Fixos
Escalabilidade	Limitado pelo Número de Fluxos	Limitado pelo Número de Classes de Serviço

Nos próximos itens, serão apresentados os diversos protocolos e arquiteturas que fornecem diferentes níveis de qualidade de serviço para diferentes necessidades das aplicações, entretanto, todos eles possuem determinadas características que acabam por comprometer a qualidades de serviço fim a fim de uma aplicação.

2.2 - ATM – *Asynchronous Transfer Mode*

O modo de transmissão assíncrono, ou ATM, é uma arquitetura baseada na transmissão de pequenos pacotes de tamanho fixo e estrutura definida, denominados

células. Essas células são transmitidas através de circuitos virtuais estabelecidos manualmente ou através de um protocolo de roteamento dinâmico, o PNNI – *Private Network to Network Interface* (ATM Forum, 2002), sendo que sua entrega e comutação são feitas pela rede baseado na informação de seu cabeçalho. O método de chavear pequenos pacotes de tamanho fixo em uma conexão, permite alcançar altas taxas de transmissão, baixa latência de comutação e priorização efetiva de tráfego.

A arquitetura ATM foi escolhida para dar suporte à B-ISDN - *Broadband Integrated Services Digital Network*, permitindo o transporte de diferentes tipos de aplicações (voz, dados, imagem e vídeo) e acesso integrado à rede.

Não faz parte do escopo deste trabalho descrever as características da arquitetura ATM com relação à célula, modelo de referência ou camadas de adaptação. Entretanto, nos próximos itens, serão descritos os conceitos dessa arquitetura pertinentes à garantia de qualidade de serviço solicitada pela aplicação.

2.2.1 - Categorias de Serviço nas Redes ATM

Uma categoria de serviço ATM tem como objetivo traduzir um modelo de serviço, isto é, um conjunto de procedimentos de caracterização de tráfego, requisitos de qualidade de serviço e gerenciamento de recursos que sejam adequados para um tipo de aplicação, permitindo desta forma a alocação eficiente de recursos pela rede.

O ATM Forum define cinco categorias de serviço, ou capacidades de transferência, de acordo com a nomenclatura ITU-T. Cada circuito virtual existente na rede possui sua categoria de serviço, de acordo com as necessidades de qualidade de serviço da aplicação. As categorias de serviço são:

a) **CBR – *Constant Bit Rate***: apropriado para suportar aplicações em tempo real, como voz e video. Provê conexões com banda dedicada, baixa probabilidade de perda de célula e atraso pequeno e previsível. O intervalo entre duas células é constante e pode ser caracterizado como o intervalo mínimo entre células, correspondendo à taxa de pico de emissão de células da aplicação.

b) **rt-VBR – *Real Time Variable Bit Rate***: adequado para um suporte mais eficiente a aplicações *real time* como voz e vídeo comprimidos, permitindo utilização mais eficiente da banda e mantendo os compromissos de *delay* máximo e taxa de perda de células.

c) **nrt-VBR – Non *Real Time Variable Bit Rate***: voltado para aplicações *non real time* com características de rajadas e pouco sensíveis a variações do *delay*. Utilizado para aplicações puras de dados IP ou FR sobre ATM e aplicações de vídeo sob demanda.

d) **ABR** – ***Available Bit Rate***: nessa categoria de serviço, a rede garante apenas uma banda mínima disponível para transmissão. Esse tipo de

serviço é adequado para aplicações que podem variar sua taxa de transmissão, acompanhando a disponibilidade de recurso da rede.

e) **UBR** – ***Unspecified Bit Rate***: corresponde a um serviço do tipo melhor esforço, onde as conexões compartilham a banda restante sem qualquer mecanismo de realimentação, sendo que a rede não garante valores limites de atraso e perda de células.

2.2.2 - Gerência de Tráfego

A integração de vários tipos de serviços na mesma rede requer um sistema que gerencie seu tráfego de maneira inteligente, garantindo que o nível de qualidade de serviço requisitado pela aplicação seja mantido, esse sistema é conhecido como Gerência de Tráfego ou *Traffic Management*. Sua função básica é proteger a rede e os usuários finais de congestionamentos que possam comprometer o SLA acordado entre cliente e provedor.

Para se quantificar os requisitos de desempenho da rede que atendem as categorias de qualidade de serviço definidas, foram criados os seguintes parâmetros:

a) **CLR – *Cell Loss Ratio***: taxa que indica o número de células perdidas com relação ao total transmitido.

b) **CTD – *Cell Transfer Delay***: tempo que a célula leva da saída do emissor até alcançar a entrada no receptor, contabilizando os tempos de: propagação do meio, transmissão, comutação e *buffering*.

d) **CDV – *Cell Delay Variation***: variação ocorrida no CTD, também conhecido como *jitter*.

e) **CER – *Cell Error Ratio***: taxa que indica o número de células com erro em relação ao número total de células transmitidas, ou seja, com e sem erros.

f) **CMR – *Cell Misinsertion Rate***: representa a taxa de células roteadas indevidamente, geralmente devido a problemas no *header* da célula.

g) **SECBR – *Severely Errored Cell Block Ratio***: Representa uma seqüência de blocos de células erradas, perdidas ou inseridas indevidamente em relação ao total transmitido.

Os parâmetros CLR, CTD e CDV são negociáveis no estabelecimento do SLA entre cliente e provedor. Os demais parâmetros não são negociáveis.

Os parâmetros de tráfego são utilizados pela rede para fazer a reserva de recursos e para caracterizar o tráfego em uma determinada direção. São eles:

a) **PCR – *Peak Cell Rate***: velocidade máxima alcançada.

b) **SCR – *Sustainable Cell Rate***: velocidade garantida pela rede.

c) **MBS – *Maximum Burst Size***: quantidade de dados do usuário final que pode trafegar em PCR.

d) **MCR – *Minimum Cell Rate***: define a banda mínima garantida pela rede em conexões ABR.

e) **CDVT – *Cell Delay Variation Tolerance***: máximo valor de CDV que uma aplicação pode suportar.

Os descritores de tráfego são utilizados para decidir quais parâmetros de tráfego devem ser monitorados em uma conexão, sendo acordado entre o provedor da rede e o cliente. Cada descritor de tráfego possui regras próprias para manejar tráfegos não conformes ao acordado.

A tabela 2 mostra a relação entre as categorias de serviço e os parâmetros de qualidade de serviço e de tráfego.

Tabela 2: Relação entre parâmetros de tráfego/QoS e categorias de serviço.

<table>
<tr><th rowspan="2">Parâmetros de QoS/Tráfego</th><th colspan="5">Categorias de Serviço</th></tr>
<tr><th>CBR</th><th>rt-VBR</th><th>nrt-VBr</th><th>ABR</th><th>UBR</th></tr>
<tr><td>CLR</td><td colspan="3">Especificado</td><td>Rede</td><td>Não</td></tr>
<tr><td>CTD/CDV</td><td colspan="2">Especificado</td><td>Não</td><td colspan="2">Não</td></tr>
<tr><td>PCR</td><td colspan="3">Especificado</td><td colspan="2">Especificado (Pode não Cumprir)</td></tr>
<tr><td>SCR/MBS</td><td>Não se Aplica</td><td colspan="2">Especificado</td><td colspan="2">Não se Aplica</td></tr>
<tr><td>MCR</td><td colspan="3">Não se Aplica</td><td>Especificado</td><td>Não se Aplica</td></tr>
<tr><td>CDVT</td><td colspan="3">Especificado</td><td colspan="2">Não</td></tr>
</table>

2.2.3 - Principais Mecanismos

Para que a rede consiga entregar os níveis de qualidade de serviço requisitados pelo cliente, o *traffic management* possui mecanismos que possibilitam a rede gerenciar e controlar o tráfego que está sendo encaminhado por ela, principalmente em situações de contingência. Esses mecanismos estão definidos em (ATM Forum, 1999) e os principais são:

a) **UPC – *Usage Parameter Control***: a função do UPC é assegurar que as células que entram na rede pela interface do usuário, obedeçam o contrato de tráfego definido no momento do estabelecimento da conexão, prevenindo a rede de congestionamentos causados por excesso de tráfego entrante. Possui um rápido tempo de resposta em atuar no tráfego dos usuários que violam seus contratos, por trabalhar na entrada da rede.

b) **CAC – Controle de Admissão de Chamadas**: conjunto de ações tomadas pela rede, durante a fase de estabelecimento de uma conexão, com o objetivo de aceitar ou não a chamada. Essas ações verificam a existência de recursos suficientes na rede para permitir o estabelecimento da conexão, sem afetar as já existentes. Dessa forma, o CAC é considerado como o principal mecanismo para que se evite congestionamentos na rede.

c) ***Traffic Shapping* –** Cadencia uma rajada de tráfego dentro de um VCC – *Virtual Channel Connection* na saída da rede, limitando-o ao valor de PCR e assegurando que o tráfego transmitido esteja conforme com o contrato de tráfego acordado. O uso de *traffic shapping* é recomendado quando existem conexões acessando uma rede ATM externa.

d) ***Congestion Management*** – existem mecanismos que atuam na rede na ocorrência de congestionamentos, com o objetivo de otimizar os recursos que ainda estão disponíveis, utilizando-os de maneira mais eficiente. Esses mecanismos, que atuam somente em conexões AAL5 - ATM *Adaptation Layer* 5, são:

- *Early Packet Discard*: descarta o pacote inteiro se não houver espaço suficiente no *buffer* para acomodá-lo completamente.
- *Partial Packet Discard*: descarta todas as células quando há estouro de *buffer*, menos células de controle para indicar fim e inicio da próxima célula.
- *Late Packet Discard*: descarta todas as células, inclusive as de controle.

2.3 - DiffServ – *Differentiated Services*

A arquitetura DiffServ não é orientada a conexão (*connectionless*). É uma arquitetura, definida pelo IETF, baseada no protocolo IP, para permitir a construção de redes altamente escaláveis, suportando aplicações com demanda de tráfego crescente, sem a necessidade de utilização de protocolos complexos de sinalização e tratamento por fluxo de dados em cada trecho da rede. Seu padrão foi definido pela RFC 2475. (Black et al., 1998)

A Figura 1 mostra onde os principais componentes de uma rede DiffServ são empregados. A arquitetura Diffserv define o conceito de domínio DS – *Differentiated Service*, que é um conjunto contíguo de nós DS que aplicam um conjunto comum de políticas sobre o tráfego que atravessa o domínio. Um domínio DS possui nós de borda e nós de *backbone*. Os nós DS de borda são responsáveis pela classificação e condicionamento do tráfego que entra no domínio DS.

Para cada fluxo de tráfego entrando no domínio pelos nós de borda, a política de QoS define qual receberá um serviço diferenciado, como deverá ser marcado e como será tratado pelos nós de *backbone*. Esses, por sua vez, examinam a marcação dos pacotes atuando de acordo com políticas pré-definidas.

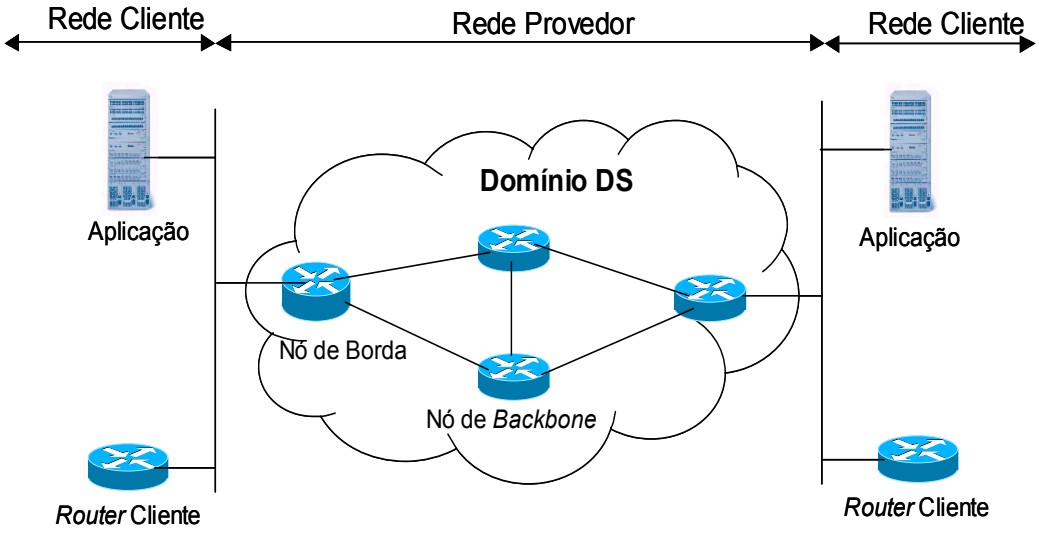


Figura 1: Componentes de um domínio DS.

2.3.1 - Classificação e Condicionamento de Tráfego

A arquitetura DiffServ define importantes componentes dentro dos nós DS: classificação, medição, marcação, formatação, descarte e escalonamento de tráfego, conforme indica a Figura 2. Essas funções são mais complexas em nós de borda, pois devem tratar o tráfego de entrada segundo políticas definidas, enquanto que os nós de *backbone* são mais especializados em rotear o tráfego dentro da rede, segundo certas prioridades.

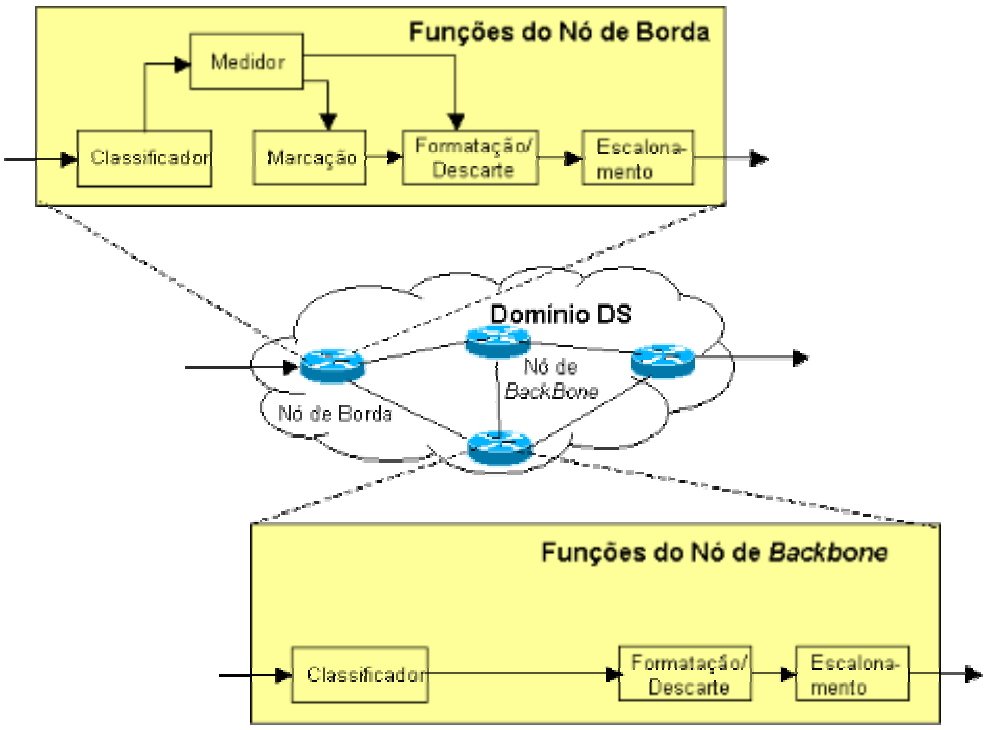


Figura 2: Funções dos nós de Borda e *Backbone*.

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

Existem dois tipos de classificadores: um que classifica o fluxo baseado apenas no valor do campo DSCP e outro que verifica múltiplos campos no cabeçalho IP. Esses classificadores são conhecidos como classificador de comportamento agregado, ou BA - *Behavior Aggregate*, e classificador multi-campo, ou MF - *Multifield Classifier*, respectivamente. A marca de classificação DS é conhecida como *DS codepoint* ou DSCP (Melo, 2001).

De acordo com a chegada do pacote, o **medidor** verifica se o mesmo está em conformidade com o definido no SLA. Dependendo do resultado dessa verificação, três tipos de ações podem ser tomadas pelo nó: marcação, formatação ou descarte.

A ação de **marcação** altera o valor do campo DSCP do pacote para forçar a aplicação de uma determinada política.

A ação de **formatação** atua sobre o tráfego não conforme, armazenando-o em *buffer*. Entretanto, dependendo da quantidade do tráfego não conforme, poderá haver descarte do mesmo. Esse tipo de ação modifica o tráfego de entrada para forçar um determinado perfil de saída.

A ação de **descarte** também atua sobre o tráfego não conforme, descartando-o automaticamente.

Escalonamento é o processo de decidir, no momento da transmissão, qual fila deve ser servida, dependendo da propriedade solicitada pelo pacote. Esse processo está associado a um algoritmo de enfileiramento de pacotes, sendo empregado no caso de otimização de recursos na saída do nó.

2.3.2 - **Encaminhamento de Tráfego**

Como foi citado, o fluxo de tráfego é classificado e marcado nos nós de borda. Os campos DSCP dos pacotes são mapeados para os PHBs - *Per Hop Behavior*s definidos na arquitetura DiffServ. Os PHBs definem o comportamento de encaminhamento de um pacote em um nó DiffServ.

Os PHBs são identificados através de um *label* de 6 *bits* do campo DSCP do cabeçalho IP do pacote, conforme mostra a Figura 3. A arquitetura DiffServ fornece nova interpretação do campo de ToS – *Type of Service*, renomeado para DSCP.

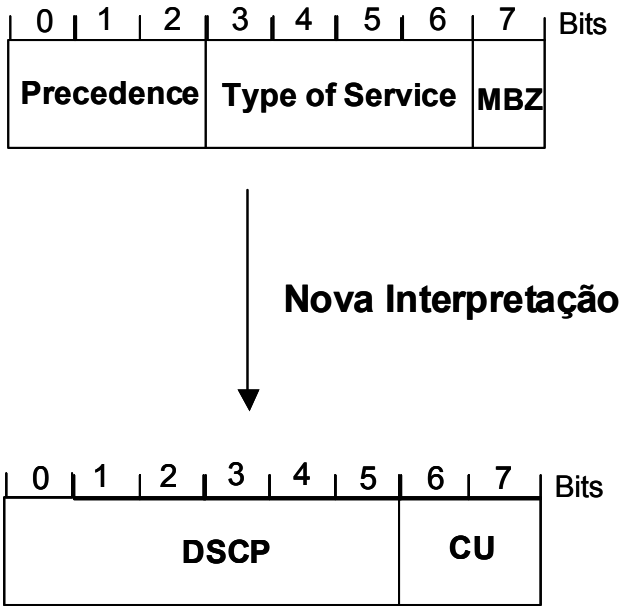


Figura 3: Campo DSCP e ToS do cabeçalho IP.

Os seis *bits* do campo DSCP são usados para selecionar o PHB que o pacote possuirá em cada nó. Esse campo é tratado como um índice em uma tabela usada para selecionar o mecanismo de manipulação de pacotes implementado em cada elemento de rede. Esse campo é definido como sendo não estruturado, para facilitar a definição de futuros PHBs.

Os PHBs podem especificar que, para determinados pacotes, serão dadas certas prioridades relativas a outros, como por exemplo: prioridade de envio, vazão média *(throughput)* ou de preferência para descarte. Os PHBs são implementados

utilizando mecanismos de enfileiramento de pacotes na interface de saída dos elementos de rede.

PHBs são comportamentos individuais aplicados em cada elemento de rede, por isso isoladamente não garantem qualidade de serviço fim a fim. Entretanto, a interligação de equipamentos com os mesmos PHBs podem oferecer, quando não houver congestionamentos na rede, a possibilidade de manter QoS fim a fim.

2.3.3 - PHBs padrões

O PHB *default*, cujo valor é 000000, é utilizado para o tráfego do tipo BE - *Best Effor*t ou de melhor esforço, assegurando compatibilidade com o encaminhamento do tipo melhor esforço padrão em todos os elementos de rede (Baker, 1995). PHBs de tráfego com DSCP mais alto devem possuir encaminhamento preferencial sobre aqueles com valor mais baixo. Por exemplo, o PHB 111000 deve ter prioridade mais alta que o 000000.

Além do *default*, outros dois importantes PHBs foram padronizados: o PHB EF - *Expedited Forwarding* (Jacobson; Nichols & Poduri, 1999) e o PHB AF- *Assured Forwarding* (Heinanen et al., 1999).

O PHB EF, também referido como serviço *premium* ou de canal dedicado, pode ser usado para tráfego com requisitos de baixa perda, baixo atraso, baixo *jitter* e garantia de banda passante. Estes requisitos são alcançados assegurando-se que os agregados de tráfego não encontrem nenhum, ou pouco, enfileiramento.

As implementações deste tipo de PHB provêm meios de limitar o dano que o tráfego EF pode causar sobre outros tráfegos. Por exemplo, o algoritmo de enfileiramento de pacotes LLQ – *Low Latence Queue* (Cisco Systems, 2001) assegura baixa latência e *jitter* reduzido para aplicações *real time*, garantindo também que o tráfego de menor prioridade seja enviado ao longo do tempo, evitando assim o chamado "*Protocol Starvation*" (possibilidade do tráfego de menor prioridade nunca ser enviado). Os dispositivos de borda do domínio DS devem policiar o tráfego EF para assegurar a taxa de *bits* definida, sendo que pacotes em excesso devem ser descartados.

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

Os primeiros três *bits* do campo DSCP identificam a classe de transmissão: 001, 010, 011 e 100, e os três últimos *bits* definem a precedência de descarte: 010 para a precedência mais baixa, ou seja, o último a ser descartado; 100 para precedência média e 110 para a mais alta precedência de descarte, ou seja, o primeiro

a ser descartado. Nas tabelas 3, 4 e 5 são apresentados os valores do campo DSCP recomendados para as classes de tráfego EF, AF e BE.

Tabela 3: Valores do campo DSCP para PHB EF.

Precedência de Descarte	**Classe EF**	
	Binário	**Decimal**
Não se Aplica	101110	46

Tabela 4: Valores do campo DSCP para PHB AF.

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

Tabela 5: Valores do campo DSCP para PHB BE.

Precedência de Descarte	**Classe BE**	
	Binário	**Decimal**
Não se Aplica	000000	0

A marcação da precedência de descartes para tráfegos AF possui dois métodos: marcador com taxa simples e marcador com taxa dupla. Ambos usam um regulador do tipo duplo balde furado, ou *Dual Leaky Bucket*, sendo que no marcador simples, os dois baldes são preenchidos na mesma taxa e no marcador com taxa dupla, os baldes são preenchidos com duas taxas diferentes. Ambos os métodos medem o tráfego marcando o pacote se o fluxo exceder a taxa acordada ou contratada. Um pacote é marcado de acordo com a quantidade de tráfego que excedeu o valor de banda passante contratada. Dependendo da marcação, o pacote possuirá prioridade maior ou menor de descarte.

2.3.4 - Serviços em um Domínio DS

Na arquitetura DiffServ, o serviço está associado ao nível de QoS solicitado pela aplicação, permitindo especificar as necessidades destas em termos de largura de banda, atraso, *jitter* e taxa de perdas, podendo ser quantitativo ou qualitativo. No primeiro caso, o serviço é especificado através de um conjunto de métricas e valores

de referência correspondentes, enquanto que no segundo caso, somente uma definição subjetiva é fornecida. Em ambos os casos a implementação de QoS está baseada em uma determinada combinação dos componentes da tecnologia DiffServ já mencionados.

Os serviços oferecidos por um domínio DS são utilizados para tráfego unidirecional e tráfego agregado apenas, não sendo possível sua aplicação para fluxos individuais. Em (Black et al., 1998), um serviço é definido como o tratamento global de um subconjunto do tráfego do cliente dentro de um domínio DS ou fim a fim.

O campo DSCP do cabeçalho IP pode ser marcado pelos clientes para indicar o serviço desejado, ou pelo equipamento de borda que liga o cliente à rede, baseado na classificação MF. No ponto de ingresso à rede, os pacotes são classificados, policiados e, possivelmente, atrasados para torná-los aderentes a algum perfil de tráfego pré-definido. As regras de classificação, policiamento e atrasos usadas nos roteadores de ingresso à rede são definidas a partir de um acordo de nível de serviço, ou SLA, definido entre cliente e provedor.

2.4 - MPLS – *Multiprotocol Label Switching*

A arquitetura MPLS foi padronizada segundo a RFC 3031 (Rosen et al., 2001) com a promessa de utilizar o melhor de dois mundos totalmente diferentes: as características dos circuitos virtuais do ATM e a escalabilidade e flexibilidade das redes IP.

Em uma rede MPLS, deve existir um protocolo de roteamento que determine a topologia nível três da rede, ou seja, o caminho que deve ser seguido para se alcançar o destino. Um protocolo de distribuição de rótulos, LDP – *Label Distribuition Protocol*, estabelece valores de rótulos, ou *labels*, para cada interface de todo roteador do caminho escolhido pelo protocolo de roteamento.

Nos próximos tópicos serão abordados os principais componentes de uma rede MPLS. Em seguida, será apresentado o encaminhamento dos pacotes pelos diversos componentes da rede.

2.4.1 - *Label*

O *label* é um pequeno identificador de tamanho fixo e significado local. Todo pacote, ao entrar numa rede MPLS, recebe um *label*, podendo ser assumido como uma forma abreviada do cabeçalho IP do pacote. Dessa forma, os elementos de rede só analisam os *labels* para poderem encaminhar o pacote. O cabeçalho MPLS é posicionado depois do cabeçalho da camada 2 e antes do cabeçalho da camada 3, sendo conhecido como *Shim Header*.

O cabeçalho do MPLS está apresentado na figura 4.

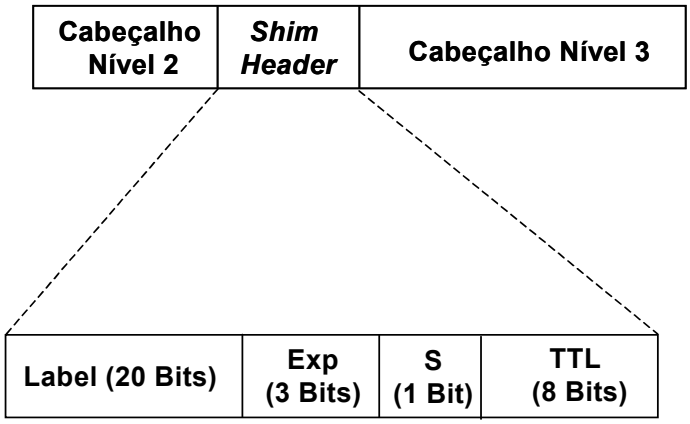


Figura 4: Cabeçalho MPLS.

Os campos que formam o cabeçalho do MPLS estão descritos a seguir:

a) ***Label***: contém um valor numérico que identifica um pacote IP.

b) **Exp – *Experimental***: o valor deste campo indica como os algoritmos de enfileiramento e descarte vão tratar o pacote, enquanto o mesmo é transmitido pela rede.

c) **S – *Stack***: indica o enfileiramento de *labels*, caso o pacote passe por diferentes domínios MPLS

d) **TTL – *Time to Live***: possui o mesmo significado que no pacote IP, ou seja, indica por quantos elementos de rede o pacote passou, até um total de 255. No caso do pacote viajar por mais de 255 elementos de rede, ele é descartado para evitar possíveis *loop*s.

O mecanismo de enfileiramento de *labels* permite realizar uma operação hierárquica de vários domínios MPLS, através da inclusão de mais *labels* em um pacote. Esse empilhamento permite que os equipamentos de *backbone* da rede troquem informações entre si e ajam como equipamentos de borda, definindo um novo domínio MPLS. Portanto, para cada domínio haverá um *label* associado. Essa técnica permite diminuir o tamanho das tabelas de roteamento dos roteadores de entrada da rede MPLS.

A figura 5 mostra o empilhamento de vários *labels* MPLS, dentro do pacote IP.

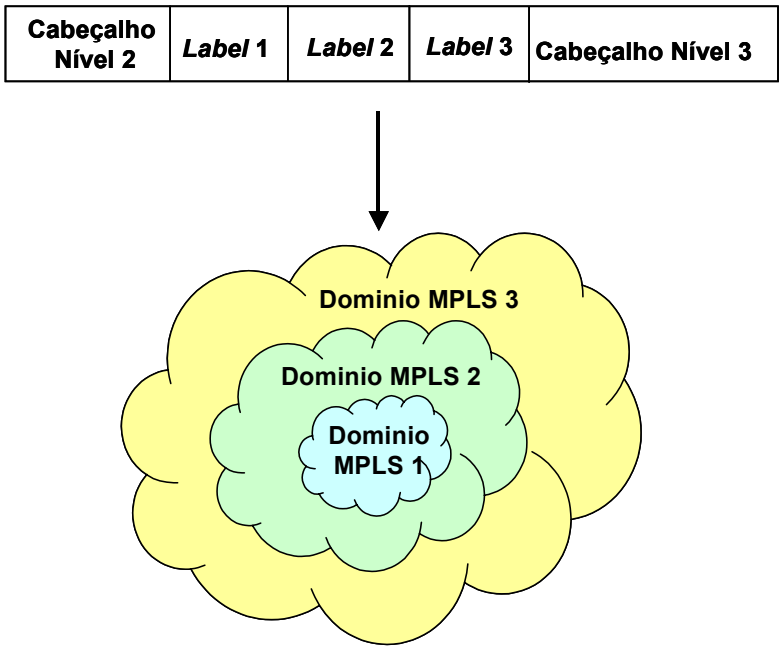


Figura 5: Empilhamento de *labels*.

2.4.2 - LER – *Label Edge Router* e LSR - *Label Switch Router*

O roteador LER está situado na borda da rede MPLS e tem por finalidade incluir um *label* no cabeçalho do pacote IP que entra na rede, associando-o a uma classe de equivalência. Depois de rotulado o pacote, o LER envia-o através de um circuito lógico definido na rede, denominado LSP – *Label Switch Path*.

Quando o pacote alcançar o roteador LER de saída, este retira o *label* MPLS e o envia para seu destino, de acordo com a tabela de roteamento local.

O roteador LSR está localizado no *backbone* da rede MPLS e possui a função de chavear os pacotes baseados somente nos seus *labels*, através de consulta a uma tabela que indica o *label* e a interface de saída para cada pacote, havendo a troca de *labels*. Este processo se repete em todos os LSRs do *backbone* da rede, até alcançar o LER de saída.

2.4.3 - LDP – *Label Distribution Protocol*

O LDP é um protocolo que permite a distribuição de *labels* entre os roteadores que compõem a rede MPLS, possibilitando, desta forma, a criação dos LSPs. Para isto acontecer, o LDP oferece um mecanismo de descobrimento de roteadores LSR e LER, permitindo que encontrem-se uns aos outros e estabeleçam comunicação. (Andersson et al., 2001)

O LDP roda sobre o protocolo TCP para garantir a entrega de mensagens nos destinos, sendo que outros protocolos também podem ser utilizados para esta

função, como por exemplo o RSVP-TE ou o CR-LDP. O padrão MPLS, definido pelo IETF, não define um único protocolo de distribuição de *labels*.

2.4.4 - LSP – *Label Switch Path*

Dentro de um domínio MPLS, um caminho lógico é criado para um determinado pacote baseado em sua classe de equivalência e seu destino. Este caminho lógico é formado através de uma seqüência ordenada de LERs e LSRs, estabelecido entre a origem (LER de entrada da rede) e o destino (LER de saída da rede), dentro de um mesmo domínio. Este caminho lógico é denominado: LSP.

O LSP é unidirecional, sendo estabelecido na rede antes da transmissão de dados.

A Figura 6 mostra um exemplo de uma rede MPLS com o LSP estabelecido entre os LERs e LSRs, havendo diversos caminhos possíveis entre os LERs. Entretanto, o caminho escolhido pelo LDP, segundo um critério pré-definido, foi: LER A – LSR 2 – LSR 3 – LER B.

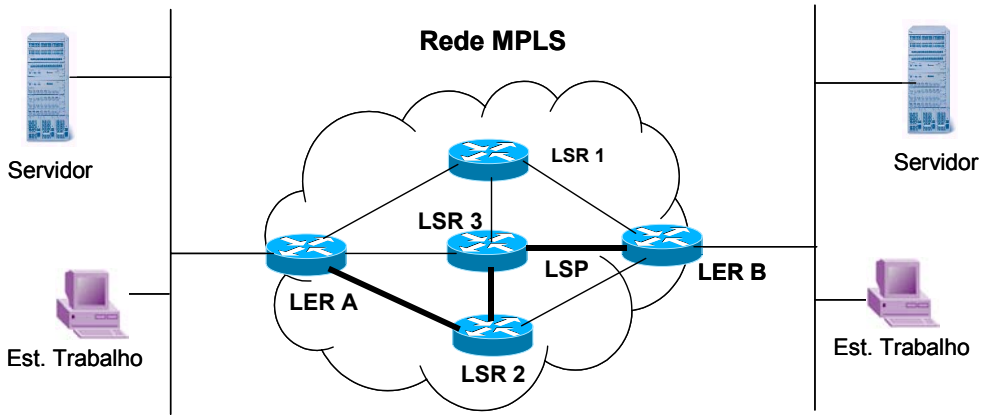


Figura 6: Componentes de uma rede MPLS.

2.4.5 - FEC – *Forwarding Equivalence Class*

A FEC consiste numa classe de equivalência, ou seja, um conjunto de parâmetros que irão determinar um caminho lógico para os pacotes. Os pacotes que estiverem associados à mesma FEC serão encaminhados pelo mesmo caminho lógico. (Harnedy, 2001)

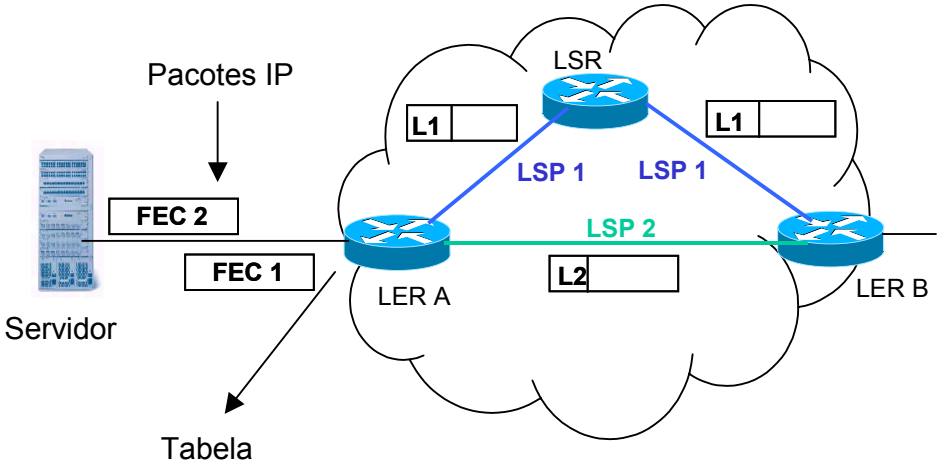
Cada LSP da rede está associado a uma FEC que, por sua vez, é representada por um *label*. Ao receber um pacote, o LER de entrada da rede MPLS verifica a qual FEC ele pertence, encaminhando-o através do LSP correspondente. Portanto, existe a seguinte associação: pacote-*label*-FEC-LSP.

A associação pacote-FEC acontece apenas uma vez, ou seja, quando o pacote entra na rede MPLS, proporcionando flexibilidade e escalabilidade para essa arquitetura de rede.

A FEC pode ser determinada por um ou mais parâmetros, especificados no SLA firmado entre cliente e provedor, podendo ser:

a) Endereço IP da fonte ou destino.
b) Número da porta TCP da fonte ou destino.
c) Campo *Identification* do cabeçalho IP.

A Figura 7 mostra a associação entre o pacote, *label*, FEC e o LSP.



FEC	LSP	*Label*
FEC 1	LSP 1	L1
FEC 2	LSP 2	L2

Figura 7: Associação entre *label*, FEC e o LSP.

2.4.6 - LIB – *Label Information Base*

A LIB contém uma tabela de encaminhamento, ou seja, uma tabela que apresenta informações correlacionando os *labels* às interfaces do roteador. Uma vez criado um LSP, a relação do *label* com a interface será armazenada na LIB.

Quando o pacote entra no LSR, é verificado, através da LIB, para qual interface esse pacote deve ser encaminhado. Depois de definida a interface de saída do equipamento, o *label* de entrada é trocado por um *label* de saída, para que o pacote possa alcançar o próximo elemento de rede.

Dessa forma, a LIB contém uma tabela que é usada para adicionar ou remover um *label* de um pacote, determinando a interface de saída pela qual o pacote deverá ser enviado.

A Figura 8 mostra um exemplo de uma LIB de um LSR.

Label de Entrada	Interface de Entrada	*Label* de Saída	Interface de Saída
3	1	7	3
1	2	4	0
8	0	6	1

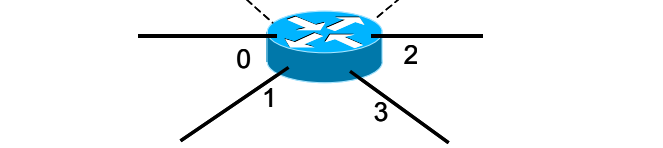


Figura 8: Exemplo de uma LIB.

2.4.7 - Funcionamento Básico

Quando um pacote IP entra numa rede MPLS, o LER irá associá-lo a uma FEC, se a mesma já existir para este pacote, caso contrário, o LER irá criá-la. Dessa forma, o pacote receberá um *label* e, como a FEC está relacionada a um LSP, o LER encaminhará o pacote através do LSP associado.

Nos saltos subseqüentes não há nenhuma análise do cabeçalho da camada de rede do pacote. A cada LSR pelo qual o pacote passa, os *labels* são trocados, pois cada *label* representa um índice na tabela de encaminhamento do próximo roteador. Desta forma, quando um pacote rotulado chega no LSR, o roteador procura em sua LIB pelo índice representado pelo *label*. Ao encontrar esse índice, o roteador substitui o *label* de entrada por um *label* de saída associado à FEC a que pertence o pacote. Após completada a operação de troca de *labels*, o pacote é encaminhado à interface de saída especificada na LIB.

Quando o pacote alcançar o LER de saída da rede, o *label* é removido e o pacote é encaminhado pela interface associada à FEC a qual pertence o pacote. A partir deste ponto, o pacote deixa de ser chaveado pela arquitetura MPLS, sendo tratado como um pacote IP normal pelos protocolos de roteamento existentes na rede.

Portanto, dentro da rede MPLS, o pacote não é roteado segundo seu endereço IP de destino e sim chaveado segundo seu *label*. Isso explica a alta escalabilidade desta arquitetura, já que as decisões de roteamento nível três são tomadas apenas uma vez na borda da rede. Depois desse processo inicial, os pacotes seguem um determinado fluxo, onde apenas os *labels* são processados rapidamente no *backbone*, já que não é necessário analisar o endereço nível três em todos os equipamentos da rede.

A melhor escalabilidade dessa arquitetura também está associada ao fato de na arquitetura MPLS haver o chaveamento de pequenas informações, ou seja, os *labels*, que podem ser armazenadas no *hardware* da própria interface. O roteamento de informações na camada de rede, ou seja, os endereços IP, é feito através de tabelas de roteamento, as quais podem ser armazenadas, dependendo do seu tamanho, em disco havendo *swap* de informações entre disco e memória e, conseqüentemente, perda em performance.

Na Figura 9 é mostrado o funcionamento básico da arquitetura MPLS, através do caminho percorrido pelo pacote IP, desde o recebimento do *label* pelo LER de entrada na rede, até sua retirada ao sair da rede MPLS.

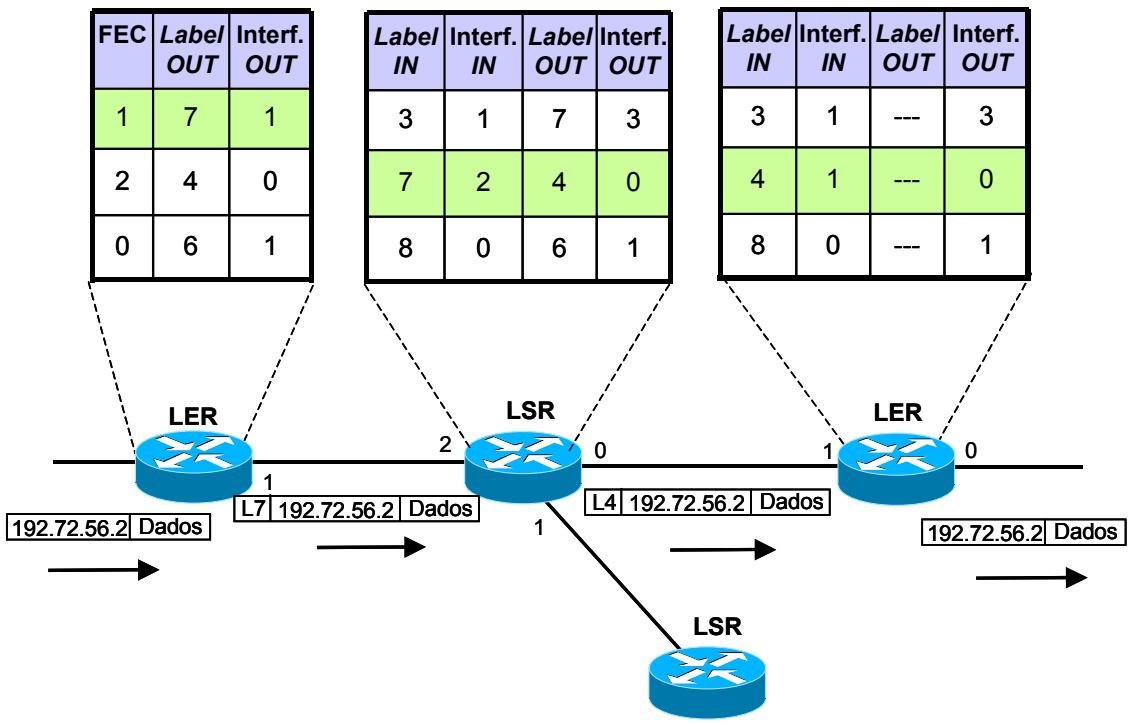


Figura 9: Funcionamento da arquitetura MPLS.

2.5 - IntServ – *Integrated Services*

A arquitetura IntServ (Clark et al., 1994) foi a primeira proposta para se padronizar a utilização de classes de serviço em redes IP, principalmente na Internet, onde existia apenas uma classe de serviço, conhecida como *Best Effort*. Esta arquitetura visava a utilização de aplicações *real time* emergentes na época como:

voz, vídeo e videoconferência. O objetivo era estender a arquitetura tradicional da Internet, fazendo com que os novos mecanismos e componentes desenvolvidos fossem acrescentados à arquitetura atual, e não tornar obsoleto os serviços IP básicos correntemente em uso.

O modelo IntServ trata os diversos fluxos de pacotes IP entre *hosts* e aplicações, os quais possuem os mesmos endereços fonte e destino, as mesmas portas TCP/UDP e o mesmo protocolo. Neste sentido, cada fluxo pode requisitar níveis de serviço específicos da rede. Os níveis de serviço são tipicamente quantificados como sendo uma taxa mínima de transmissão, um *delay* fim a fim máximo tolerável ou uma taxa de perda de pacotes. Portanto, a rede pode aceitar ou rejeitar as requisições dos fluxos, baseando-se na disponibilidade de recursos e na garantia de nível de serviço já ofertada a outros fluxos.

Existem três importantes mecanismos dentro da arquitetura IntServ, os quais são vitais para oferecer e garantir qualidade de serviço aos fluxos de dados dentro da rede. São eles:

a) **Controle de Admissão**: responsável por verificar se a rede possui recursos suficientes para prover a qualidade de serviço desejada.

b) **Envio de Pacotes**: responsável pela classificação dos pacotes e, posteriormente, o envio desses pacotes para determinadas filas de saída.

c) **Protocolo de Reserva de Recursos**: responsável por criar e manter o estado dos fluxos de dados nos roteadores ao longo do caminho seguido pelo fluxo.

O IETF considerou a utilização de várias classes de serviços dentro do modelo IntServ, entretanto, apenas duas delas foram formalmente especificadas: *Guaranteed Services* (Wroclawski, 1997b), e *Controlled Load Service* (Wroclawski, 1997a).

O protocolo RSVP versão 1 foi desenvolvido para atender o padrão proposto pela arquitetura IntServ.

2.6 - RSVP-TE – *Resource Reservation Protocol with Traffic Engineering*

A padronização do protocolo RSVP-TE (Awduche et al., 2001) adiciona algumas novas características ao protocolo RSVP versão 1 (Braden et al., 1997), para permitir o estabelecimento de túneis LSP em redes MPLS, fazendo engenharia de tráfego para manter o nível de qualidade de serviço solicitada por uma aplicação.

Dessa forma, o RSVP-TE, entre outros, trabalha como sendo o protocolo de sinalização de uma rede MPLS. Entretanto, a utilização deste protocolo não exclui a necessidade de haver um protocolo de roteamento nível 3, ou IGP – *Interior Gateway Protocol*. O IGP é necessário para descobrir os caminhos existentes entre

fonte e destino. O protocolo de sinalização irá definir qual é o melhor caminho para atender às necessidades de qualidade de serviço requisitadas.

O protocolo RSVP-TE é orientado a conexão, pois estabelece um caminho para o fluxo de dados, entre o equipamento de origem e o equipamento de destino do cliente. Sua principal característica é de poder fazer reserva de recurso para cada fluxo de dados, de acordo com a qualidade de serviço desejada. O conceito de fluxo de dados ficou mais fortemente caracterizado com a utilização da arquitetura MPLS e o estabelecimento de túneis LSP.

Assim como o RSVP, o RSVP-TE também utiliza os conceitos definidos na arquitetura IntServ (Wroclawski, 1997c), para prover os níveis de qualidade de serviço para cada túnel LSP, os quais são: *Controlled Load* (Wroclawski, 1997a), *Guaranteed Service* (Wroclawski, 1997b) e *Best Effort*.

2.6.1 - PATH *Message* e RESV *Message*

O protocolo RSVP possui duas importantes mensagens que são trocadas entre origem e destino para o estabelecimento do túnel LSP: PATH *Message* e RESV *Message*.

A PATH *Message* é enviada do *host* de origem para o *host* de destino, contendo as informações do tipo de tráfego e da qualidade de serviço que devem ser trocadas entre eles. No caminho entre a origem e o destino são armazenados todos elementos de rede pelo qual esta mensagem passou.

A RESV *Message* é enviada do *host* de destino para o *host* de origem, pelo mesmo caminho seguido pela PATH *Message*, fazendo as reservas de recursos necessárias para atender às necessidades desejadas de QoS. É por este motivo, que o protocolo RSVP é conhecido como *Receiver Oriented*, ou orientado pelo destinatário. Caso não exista a banda necessária para ser alocada, ou não puder ser determinado um *label* para o fluxo em uma interface, o túnel não será estabelecido.

A Figura 10 mostra a troca de mensagens PATH e RESV entre as estações fonte e destino. (Gallaher, 2001)

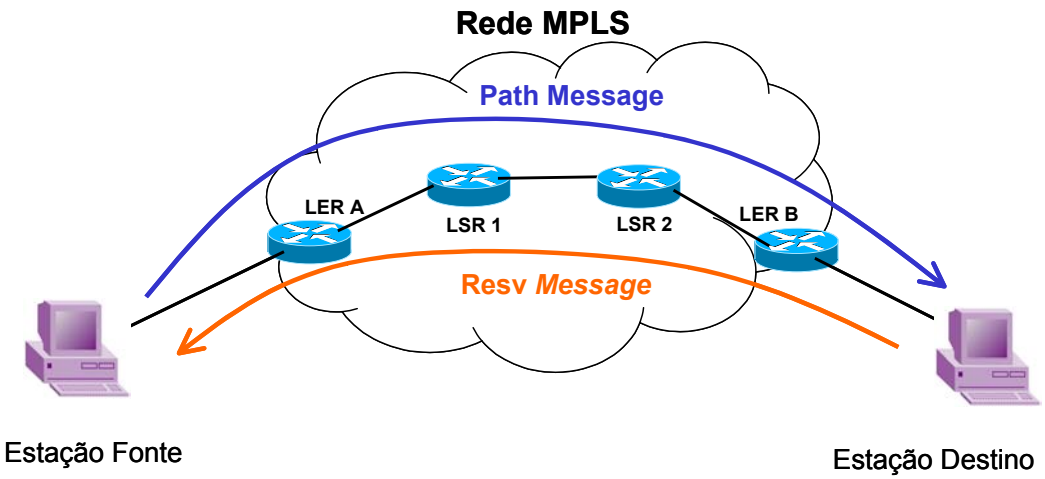


Figura 10: PATH e RESV *Messages* sendo trocadas.

A Figura 10 mostra uma rede MPLS com o protocolo RSVP-TE configurado nos roteadores de borda (LER) e nos de *backbone* (LSR). A estação fonte deseja transmitir dados para a estação destino com um determinado nível de qualidade de serviço. Para tanto, a estação fonte monta uma PATH *Message* com informações sobre as características de tráfego que serão enviadas para a estação destino.

O protocolo IGP da rede MPLS define um possível caminho para as duas estações se comunicarem. Quando o LER A receber a PATH *Message* vinda da estação fonte, ele utilizará o caminho escolhido pelo IGP para verificar se o mesmo atende às características de qualidade de serviço solicitadas pela estação fonte. Antes do LER A enviar a PATH *Message* para o LSR 1, ele verificará se possui os recursos necessários para atender o nível de QoS solicitado e a disponibilidade de alocação de um *label* para este LSP. Se essas duas tarefas forem executadas, o LER A enviará a PATH *Message* para o LSR 1. (Juniper Networks Inc, 2002)

O LSR 1, ao receber a PATH Message, irá executar as duas tarefas obrigatórias para enviar a PATH Message para o LSR 2: verificação de disponibilidade de recursos e alocação de um *label* para o LSP. Se essas duas tarefas forem executadas, o LSR 1 enviará a PATH *Message* para o LSR 2.

Ambos os procedimentos de verificação são executados em todos os elementos de rede que compõem o túnel LSP dentro da rede MPLS, até a PATH *Message* alcançar a estação destino.

Se em qualquer elemento da rede não houver recursos suficientes para serem reservados ou indisponibilidade de alocação de *labels*, o LSP não é estabelecido na rede, sendo retornado uma mensagem de erro para a origem.

Quando a estação destino receber a PATH *Message*, será gerada uma RESV *Message* em direção à estação fonte, passando pelos mesmos elementos de rede que a PATH *Message* passou. Ao receber a RESV Message, o LER B fará a alocação da

banda solicitada e estabelecerá um *label* para o LSP que está sendo ativado. Esse processo ocorre em todos os equipamentos que compõem o caminho percorrido pela PATH *Message*, até a RESV *Message* chegar na estação fonte.

A partir deste ponto, o túnel LSP é estabelecido na rede MPLS estando pronto para trafegar os dados entre as estações, sendo sua operação similar à de uma rede MPLS sem engenharia de tráfego. No *backbone* da rede, os pacotes IP são transmitidos através da comutação de *labels* e nas extremidades, bordas da rede, os pacotes são roteados segundo seu endereço IP.

Esse processo de sinalização, ou estabelecimento do LSP, é conhecido como *soft state*, pois as mensagens de PATH e RESV são trocadas periodicamente entre os elementos que compõem a rede. Caso algum dos elementos não receba essas mensagens dentro de limites pré-definidos, o caminho será desfeito. Essa característica apresenta pontos positivos, como o rápido re-roteamento de LSPs em caso de falha no circuito, e pontos negativos, como um possível problema de escalabilidade.

Na figura 11 é mostrada a contínua troca de mensagens PATH e RESV até que uma das extremidades desfaça o caminho.

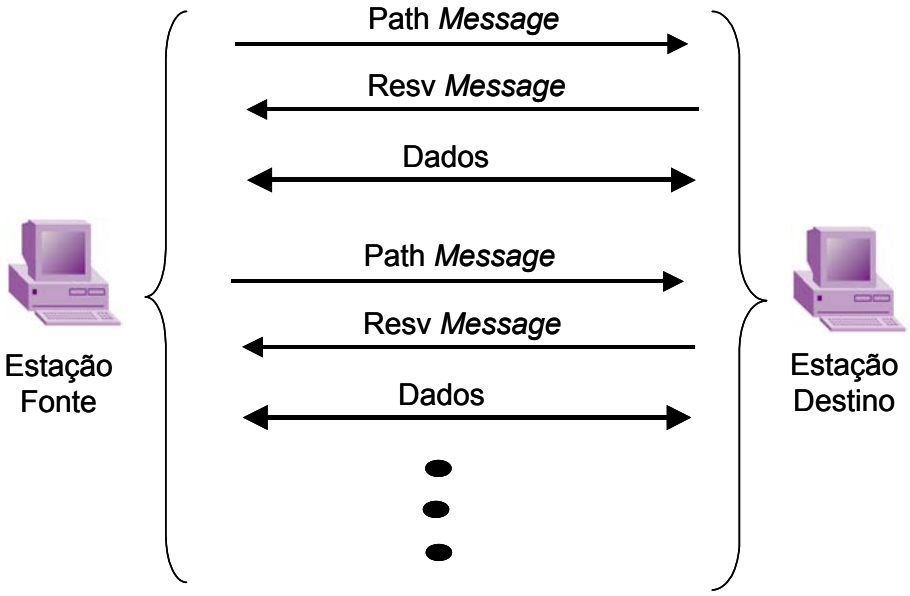


Figura 11: Troca de PATH e RESV *Messages* indicando o conceito de *soft state*.

2.6.2 - Objetos utilizados pelas PATH *Message* e RESV *Message*

As mensagens de PATH e RESV são trocadas entre os elementos de rede para estabelecerem um caminho, ou LSP. Entretanto, são os objetos transportados por essas mensagens que informam as características e recursos de cada LSP, como por exemplo: *label* solicitado, características de qualidade de serviço, rotas explícitas, prevenção de *loops*, etc. Os objetos utilizados pelas mensagens de RESV e PATH estão descritos a seguir:

a) ***Label Request Object***: utilizado pela PATH *Message* para solicitar ao roteador MPLS que um *label* seja reservado para um LSP. Esse objeto também identifica qual o protocolo de nível 3, L3PID – *Layer* 3 *Protocol Identification*, que será utilizado no LSP. Essa informação é útil pois existem *labels* reservados para alguns protocolos específicos. Um equipamento que aceita o *Label Request Object*, mas não pode alocar um label para o PATH Message, enviará uma mensagem de erro, PathErr, com o código: *Routing Problem* e com o valor: MPLS *label allocation failure*. Esse problema acontece quando um *label* é solicitado fora da faixa disponível.

   Existem 3 tipos de *Label Request Object*:

   - solicitação de *label* sem especificação de uma determinada faixa.
   - solicitação de *label* especificando os valores máximos e mínimos de VPI e VCI, para redes ATM.
   - solicitação de *label* especificando os valores máximos e mínimos de DLCI – *Data Link Connection Identifier*, para redes *Frame Relay*.

b) ***Label Object***: utilizado pela RESV *Message* para confirmar a alocação do *label* solicitado pelo *Label Request Object*, atualizando a LIB do roteador. Esse objeto pode conter um único valor ou uma pilha de *labels*. Os valores de *labels* possuem significado local, ou seja, apenas na interface pela qual passa o LSP. Desta forma, os valores de *labels* podem ser repetidos em interfaces diferentes.

c) ***Flowspec Object***: esse objeto não sofreu alterações na padronização do RSVP-TE, sendo igual ao definido no RSVP versão 1. É utilizado pela PATH *Message* para especificar as características de qualidade de serviço requisitada pela aplicação, como:

   - nível de QoS (*controlled load* ou *guaranteed service*).
   - banda a ser alocada.
   - máximo *delay*.
   - taxa de perda de pacotes.

   Em muitos casos, não é necessário reservar recursos ou manter níveis de QoS, no estabelecimento do LSP. Neste caso, o *Flowspec Object* não possui informação alguma e o tráfego será associado ao tipo *best effort*.

d) **ERO – *Explicit Route Object***: utilizado pela PATH *Message* para indicar uma rota específica para o estabelecimento do LSP, independente dos caminhos disponibilizados pelo IGP. Esse objeto deve ser utilizado somente quando o *host* conhecer todo o caminho, sabendo que o mesmo atenderá o nível de QoS solicitado e todos os roteadores,

suportam o protocolo RSVP-TE, pois se algum roteador não o suportar, ou não houver recursos suficientes para serem alocados, o LSP não será estabelecido na rede.

O ERO é composto de vários sub-objetos, sendo que cada sub-objeto representa um elemento de rede através do qual o LSP será estabelecido. Atualmente, existem quatro tipos de sub-objetos definidos:

- IPv4: identifica um elemento de rede com suporte a IP versão 4.
- IPv6: identifica um elemento de rede com suporte a IP versão 6.
- *Autonomous System Number*: identifica o sistema autônomo ao qual o elemento de rede pertence.
- Terminação MPLS LSP: indica que o elemento de rede deve remover o primeiro *label* do *label stack* de todos os pacotes que trafegarem nesse LSP.

e) **RRO – *Record Route Object***: utilizado pelas PATH e RESV *Messages* para guardarem a identificação de cada elemento de rede por onde as mensagens passaram. O RRO também é composto de vários sub-objetos, sendo que cada um deles informa o elemento de rede pelo qual a mensagem passou. Atualmente, estão definidos dois tipos de sub-objetos: IPv4 e IPv6.

O RRO possui um número máximo de sub-objetos que podem ser armazenados. Se a somatória de todos os sub-objetos for maior que o tamanho da PATH (ou RESV) *Message*, o RRO será retirado da mensagem e um PathErr (ou ResvErr) será enviado para o originador da mensagem.

Existem três possíveis aplicações para o RRO:

- descobrimento de *loops* de roteamento nível três ou *loops* provenientes de EROs mal configurados.
- Coleta, em tempo real, de informações *hop-by-hop* de todos os elementos de rede que compõem o LSP.
- Com pequenas alterações, as informações do RRO podem ser utilizadas pelo ERO, podendo ser utilizada para definir novos caminhos fixos na rede, segundo algum critério pré-definido.

f) ***Session Object***: utilizado pela PATH *Message* para identificar um LSP. Contém o endereço IP do LER de entrada na rede e um identificador do LSP que permanece constante enquanto o LSP estiver ativo, mesmo se ele for re-roteado.

g) ***Sender_Template Object***: utilizado pela PATH *Message* para informar o formato dos dados de um originador em específico.

h) ***Filter_Spec Object***: define, em conjunto com o *Session Object*, o fluxo de dados que possuirá as características definidas pelo *Flowspec Object*.

i) ***Session_Atribute Object***: utilizado pela PATH *Message* para controlar a prioridade do LSP em casos de preempção.

2.6.3 - Estilos de Reserva

Cada sessão RSVP estabelecida na rede, dependendo de sua necessidade de qualidade de serviço, possui um estilo de reserva de recursos apropriado. Existem três estilos definidos:

a) **FF - *Fixed Filter***: neste estilo de reserva os recursos são alocados para um único par origem – destino, não sendo compartilhados com nenhuma outra estação da rede. Portanto, a quantidade total de banda reservada no estilo FF em um circuito de comunicação será igual à soma das reservas de cada sessão RSVP.

b) **SE - *Shared Explicit***: esse estilo de reserva permite a um originador especificar os elementos de rede que compartilham um determinado recurso reservado. Como os diferentes elementos de rede podem possuir requisitos de quantidade de recursos a serem alocados diferentes, a reserva é feita pelo de maior necessidade. Esse estilo de reserva é utilizado em aplicações *multicast* ou em re-roteamento de LSPs, utilizando o conceito *make-before-break*.

c) **WF – *Wildcard Filter***: neste estilo, um único recurso reservado pode ser compartilhado por vários originadores de sessão, não havendo portanto, manutenção de QoS para uma determinada aplicação. Portanto, pela falta de aplicabilidade para engenharia de tráfego, o WF não é utilizado pelo RSVP-TE.

2.6.4 - Re-roteamento de LSPs

Um dos requisitos para se fazer engenharia de tráfego, é a capacidade de re-rotear LSPs já estabelecidos na rede, sob certas condições, ou políticas administrativas. Por exemplo, uma determinada política pode estabelecer que um LSP deverá ser re-roteado quando um caminho melhor se tornar disponível. Outra característica importante é quando um LSP é re-roteado devido a uma falha em um circuito de comunicação, e quando essa falha estiver resolvida, o LSP deverá ser re-estabelecido pelo caminho original.

É altamente desejável que quando houver necessidade de re-roteamento de LSPs não haja interrupção de tráfego, apesar de em alguns casos isso não ser possível. Para se realizar esse processo sem quedas de tráfego ou queda de performance, é necessário estabelecer um novo LSP para que o tráfego seja

transferido antes de desativar o primeiro. Esse conceito é chamado de: *make-before-break*.

Para suportar o conceito de *make-before-break* é necessário que os recursos alocados para o antigo LSP não sejam liberados antes que todo o tráfego seja transferido para o novo caminho, e as reservas feitas para o estabelecimento do novo LSP não deverão ser duplicadas, pois, dessa forma, pode haver problemas de conexão rejeitada por falta de recursos.

Por exemplo, esse conceito é utilizado em situações onde a banda passante de um LSP deva ser aumentada. Nesse caso, é utilizada uma combinação do estilo de reserva SE em conjunto com o *Session Object*.

2.7 - Por que, Isoladamente, as Tecnologias Descritas não Garantem QoS Fim a Fim?

Como foi descrito nos itens anteriores, foram desenvolvidos vários protocolos e arquiteturas que fornecem diferentes níveis de qualidade de serviço para diferentes necessidades das aplicações, entretanto, todos eles possuem características que acabam por comprometer a qualidade de serviço fim a fim de uma aplicação em determinadas situações.

Dentre todas as tecnologias descritas, a arquitetuta ATM é a que possui melhores condições de garantir qualidade de serviço para uma aplicação, mesmo em situações de congestionamento. Entretanto, a tecnologia ATM não se tornou padrão de mercado para prover QoS como era esperado. Devido ao seu alto custo financeiro de implementação e à relativa complexidade de operação, provedores e usuários procuraram alternativas mais baratas e mais fáceis. Hoje, o ATM é utilizado em aplicações específicas, como aglutinante e para transporte de tráfego IP em *backbones* de rede.

A arquitetura DiffServ possui um grande problema, que se manifesta quando ocorre congestionamento devido a mudança de rotas. Grande quantidade de tráfego pode ser direcionada para um caminho disponível, cuja capacidade pode não ser suficiente para acomodar todos os tipos de tráfego. Desta forma, aplicações de maior criticalidade ou sensíveis a variação de *delay* e perda de pacotes, sofrerão degradação da sua qualidade de serviço requisitada. Dependendo do grau de congestionamento ocorrido, todas as aplicações, independentemente de sua prioridade, serão afetadas indiscriminadamente.

Nas redes puramente MPLS, quem define o melhor caminho dentro da rede é um protocolo de roteamento IP. Entretanto, esse caminho muitas vezes pode não atender às necessidades específicas de qualidade de serviço, já que o caminho mais curto pode não ser o melhor caminho. O protocolo de roteamento IP em questão pode direcionar mais tráfego por um caminho, sem se preocupar se haverá recursos suficientes para atender toda a demanda de tráfego. Portanto, para haver manutenção

de QoS, o protocolo de distribuição de *labels* deverá se preocupar com engenharia de tráfego na rede.

Diferentemente da arquitetura DiffServ, a arquitetura IntServ não é configurada nos equipamentos que compõem uma rede. Para tanto, foi desenvolvido o protocolo RSVP, que implementa as especificações contidas na arquitetura IntServ.

O protocolo RSVP/RSVP-TE, através do estabelecimento de sessões entre *hosts*, é o que mais se assemelha à arquitetura ATM, possuindo, portanto, bons recursos para manter qualidade de serviço a uma aplicação. Entretanto, a necessidade de se utilizar um *software* RSVP rodando no cliente tornou a utilização desse protocolo pouco atrativa. Na verdade, somente algumas versões experimentais tornaram-se disponíveis para algumas plataformas UNIX (Bernet,Y 2000, p. 155).

Também, o conceito de qualidade de serviço da arquitetura Intserv, utilizada pelo protocolo RSVP-TE, está definido de maneira qualitativa, como o comportamento que a rede apresenta em situações de baixa ou alta carga, como exemplificado a seguir: "O comportamento fim a fim de uma aplicação *controlled load service* fornecido pelos vários elementos de rede, se aproxima do comportamento visto pelas aplicações *best effort* quando a rede não está em condições de carga pesada..." (Wroclawski, 1997a, p. 2).

2.8 - Definição do Problema

O problema que se apresenta nas redes IP, com relação à manutenção do nível de QoS, é: como manter qualidade de serviço fim a fim para aplicações críticas, cujo tráfego já tenha sido estabelecido na rede, em situações de congestionamento?

Normalmente, esse problema tem sido resolvido pela simples adição de banda passante em redes com problemas de gargalo ou congestionamento. Entretanto, essa solução é por demais onerosa, além de não otimizar recursos.

Dos protocolos e arquiteturas utilizados atualmente, o protocolo RSVP/RSVP-TE é o que mais possui condições de resolver o problema apresentado. Entretanto, as duas fragilidades desse protocolo, descritas no item 1.4, fazem com que o mesmo não seja utilizado em larga escala. Para facilitar a compreensão do leitor, as referidas fragilidades são aqui repetidas:

a) Para manter a qualidade de serviço fim a fim entre dois *hosts*, é necessário que os mesmos possuam um *software client* RSVP rodando, para poderem conversar com os demais equipamentos da rede, que também possuem o protocolo RSVP carregado, estabelecendo assim o circuito virtual entre eles.

b) O conceito de qualidade de serviço da arquitetura IntServ, utilizada pelo protocolo RSVP-TE, está definido de maneira qualitativa, como o

comportamento que a rede apresenta em situações de baixa ou alta carga.

Portanto, as novas propostas apresentadas neste trabalho procuram resolver essas fragilidades, fazendo com que o protocolo ganhe mais robustez e versatilidade.

Capítulo 3 - Definição das Propostas para Estender as Funcionalidades do RSVP-TE.

Esse trabalho apresenta duas propostas para estender as funcionalidades do protocolo RSVP-TE visando manter a qualidade de serviço fim a fim de uma aplicação baseada no protocolo IP. Essas propostas consistem na definição de dois mecanismos trabalhando como um *gateway* entre a tecnologia DiffServ e o protocolo RSVP-TE. Um mecanismo atuará na borda da rede e o outro no *backbone*, ou seja, esses mecanismos possuirão a capacidade de extrair as informações relevantes de cada arquitetura, aproveitando suas melhores características em um único ambiente de rede.

Portanto, esses mecanismos atuarão em um ambiente de rede formado pela utilização da arquitetura DiffServ na rede local do cliente, a qual estará conectada a um roteador de borda da rede do provedor, e pela utilização do protocolo RSVP-TE e da arquitetura MPLS no *backbone* da rede do provedor.

A primeira proposta consiste em definir um mecanismo que atue no roteador de borda da rede. Esse mecanismo deverá possuir a habilidade de interpretar o campo DSCP do cabeçalho IP dos pacotes vindos da rede do cliente e estabelecer um túnel LSP, através do RSVP-TE, até a localidade remota com o nível de qualidade de serviço solicitada.

A segunda proposta consiste em definir um mecanismo que atue nos roteadores de *backbone* da rede, possuindo a habilidade de encaminhar o tráfego de cada LSP estabelecido na rede para determinadas filas de um algoritmo de enfileiramento de pacotes, de acordo com a qualidade de serviço do LSP. Por exemplo, o tráfego de LSPs com aplicações *real time* será direcionado para uma fila de alta prioridade de envio.

Com a implementação desses mecanismos, as aplicações e estruturas de rede utilizadas pelo cliente se mantêm inalteradas, minimizando as alterações necessárias no protocolo RSVP-TE. A utilização do protocolo RSVP-TE no *backbone* da rede será transparente para as aplicações do cliente, que implementam a tecnologia DiffServ.

3.1 - Funcionalidade Proposta aos Roteadores de Borda da Rede.

O roteador de borda da rede é aquele que fará a conexão entre a rede do cliente e a rede do provedor, sendo conhecido como LER.

O mecanismo que atuará nesse roteador analisará o campo DSCP de todos os pacotes IP que entram na interface diretamente conectada à rede do cliente. Uma vez identificado o valor do campo DSCP, o mecanismo disparará um processo de consulta a uma tabela que indique o nível de qualidade de serviço associado ao valor do campo DSCP analisado.

Dentro do *backbone*, o protocolo RSVP-TE utilizará os três níveis de qualidade de serviço definidos pelo *Integratted Services Group* (Wroclawski, 1997c, p. 2): *Guaranteed Service* (Wroclawski, 1997b) , *Controlled Load Service* (Wroclawski, 1997a) e *Best Effort* e, portanto, esses três níveis de QoS serão mapeados para os vários LSPs a serem estabelecidos na rede, sempre de acordo com o valor do campo DSCP do pacote IP.

O nível de qualidade de serviço *Guaranteed Service* é o mais prioritário e será utilizado para aplicações *real time* como voz e vídeo. O nível de qualidade de serviço *Controlled Load* possui prioridade intermediária e será destinado a aplicações que necessitem de tempo de entrega de pacotes relativamente constante, porém com tolerância a variações desse tempo, como aplicações ERP - *Enterprise Resource Planning*. O nível de qualidade de serviço *Best Effort* é o de mais baixa prioridade e será destinado a tráfego de *e-mails* e transferência de arquivos.

A tabela 6 mostra a relação entre os valores do campo DSCP e os níveis de qualidade de serviço do RSVP-TE.

Tabela 6: Relação entre o valor do campo DSCP e os níveis de qualidade de serviço do RSVP-TE.

***Bits* DSCP**	**QoS do RSVP-TE**
00 XXXX	*Guaranteed Service*
01 XXXX	*Controlled Load Service*
10 XXXX	*Best Effort*
11 XXXX	Não Utilizado

Os dois primeiros *bits* do campo DSCP, da esquerda para a direita, indicam o nível de qualidade de serviço do LSP a ser estabelecido na rede. Portanto, se a aplicação necessitar que seu tráfego possua nível de QoS igual a *Guaranteed Service*, ela deverá configurar os dois primeiros *bits* do campo DSCP igual a zero.

Este trabalho define apenas três níveis de qualidade de serviço, apesar de se poder definir outros níveis mais. O motivo para se definir apenas três níveis de QoS é que, na prática, os clientes possuem três grandes grupos de aplicações, sendo normalmente: voz e vídeo, aplicações ERP e transferências de arquivo e *e-mail*. Conforme houver mais níveis de QoS, mais complicado será o processo de venda e *billing* do provedor para com seus usuários. (Cisco Systems, 2001)

Conhecendo o nível de qualidade de serviço requisitada pela aplicação, o mecanismo irá verificar a existência de um túnel LSP, com esse nível de qualidade de serviço, já estabelecido na rede entre a origem e o destino do pacote.

Se não houver nenhum túnel LSP estabelecido na rede com o nível de qualidade de serviço indicado pelo pacote, o mecanismo enviará uma solicitação ao RSVP-TE do LER de entrada na rede, para que um LSP seja estabelecido entre o LER local e o LER remoto, com o nível de qualidade de serviço solicitado.

Para manter a qualidade de serviço solicitada pela aplicação, todas as reservas de recurso feitas na rede deverão ser do tipo FF, ou seja, cada túnel ou originador possuirá seus próprios recursos, não sendo compartilhados por outros túneis. Portanto, esse trabalho não aborda a utilização de aplicações *multicast* ou *broadcast*, podendo ser uma sugestão de continuidade dessa pesquisa.

Se o RSVP-TE conseguir estabelecer na rede o LSP desejado, o mecanismo irá consultar a LIB do LER de entrada para saber o número do *label* MPLS a ser colocado no pacote. De posse desse número, o mecanismo agregará ao pacote IP o *label* correspondente e o entregará ao MPLS para que o mesmo seja encaminhado dentro do *backbone*. Se o LSP não puder ser estabelecido, o mecanismo registrará a mensagem de erro enviada pelo RSVP-TE e gerará um novo pedido de estabelecimento do LSP.

Se o LSP já estiver estabelecido na rede, o mecanismo verificará se existe banda passante disponível para trafegar mais pacotes nesse LSP. Se houver banda disponível, o mecanismo irá consultar a LIB do LER de entrada para saber o número do *label* MPLS a ser colocado no pacote. Caso contrário, o pacote será descartado e o mecanismo analisará e identificará o campo DSCP do próximo pacote IP.

No LER de saída da rede, o *label* é retirado e o pacote é entregue à rede local do cliente para ser capturado pela aplicação.

A figura 12 mostra um diagrama de blocos com os processos e etapas do mecanismo proposto aos roteadores de borda da rede.

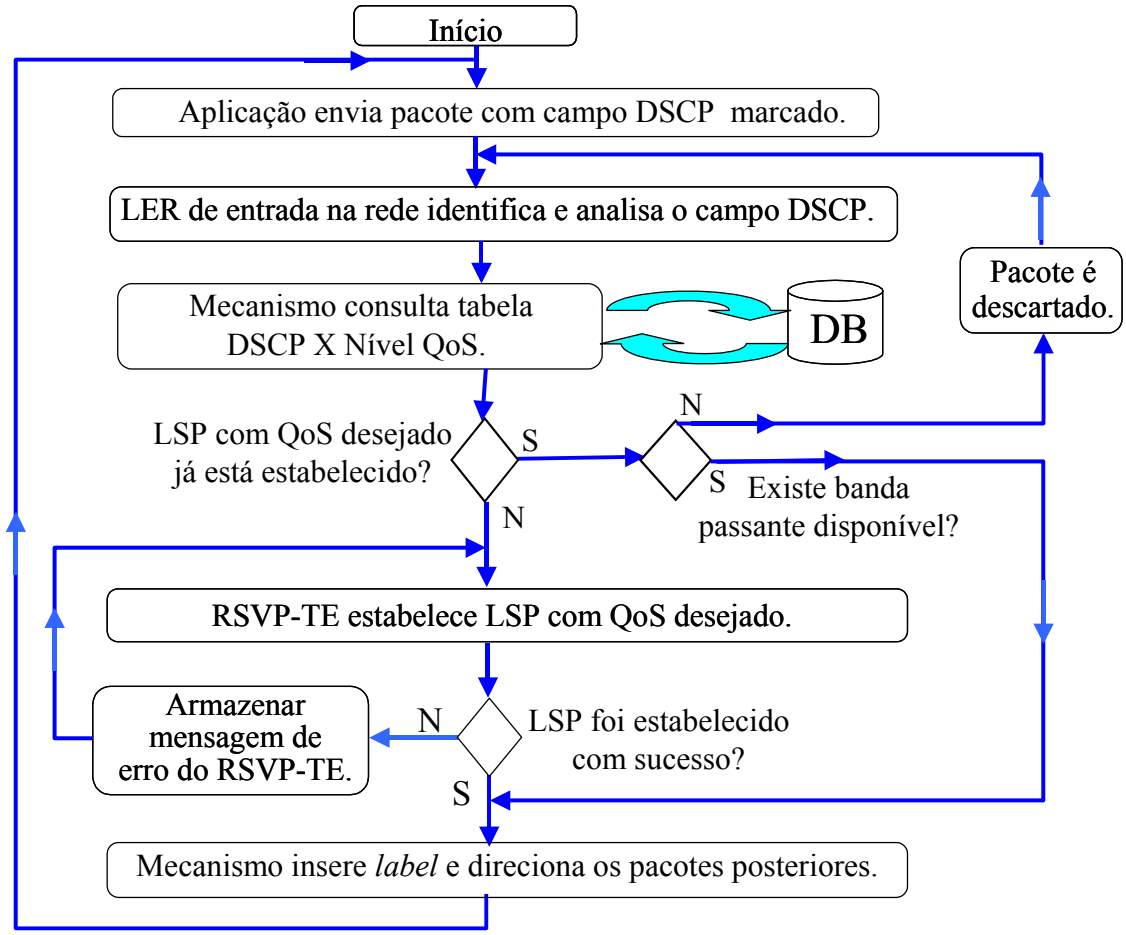


Figura 12: Diagrama de blocos do mecanismo proposto ao LER.

Para que o RSVP-TE possa estabelecer os túneis com as características apropriadas na rede, deverá ser acordado um SLA entre cliente e provedor, para que os seguintes parâmetros possam ser definidos:

a) Banda passante do LSP a ser estabelecido na rede (bps).

b) Velocidade de pico (bps).

c) Tamanho de *Burst* (*bits*).

d) Nível de qualidade de serviço para cada valor do campo DSCP.

Essa proposta admite também a possibilidade do cliente querer que o tráfego de diferentes aplicações, com mesmo nível de qualidade de serviço, sejam tratadas de maneira diferente entre si, ou seja, um LSP para cada aplicação, ainda que, em termos de reserva de recurso e prioridade de envio, elas compitam entre si, pois possuem o mesmo nível de qualidade de serviço.

A tabela 7 indica a relação entre os valores do campo DSCP e os níveis de qualidade de serviço do RSVP-TE para esse caso.

Tabela 7: Relação entre os valores do campo DSCP e os níveis de QoS para o caso onde exista mais de um LSP para o mesmo nível de QoS.

***Bits* DSCP**	**QoS do RSVP-TE**
00 001 X	*Guaranteed Service* – Voz Coorporativa
00 010 X	*Guaranteed Service* – Video Conferência
00 100 X	*Guaranteed Service* - Telemedicina
01 001 X	*Controlled Load* – ERP (Financeiro)
01 010 X	*Controlled Load* – ERP (Jurídico)
10 000 X	*Best Effort* – Transferência de Arquivo

Os dois primeiros *bits* do campo DSCP, da esquerda para a direita, indicam a prioridade do LSP a ser estabelecido na rede. Os três *bits* seguintes indicam o tipo de aplicação. Eventualmente, poderá ser utilizado o sexto bit do campo DSCP ou, até mesmo, os dois *bits* do campo CU do pacote IP, para definir mais aplicações dentro de um mesmo nível de QoS.

A vantagem de se utilizar esse tipo de solução é que, em situações de congestionamento, onde não há recursos disponíveis para todas as aplicações, as chances de haver algumas aplicações operantes é maior do que se todas as aplicações, de mesma prioridade, estivessem concentradas em um único LSP. Provavelmente, esse LSP não seria estabelecido na rede por falta de recursos disponíveis, tornando todas as aplicações inoperantes.

Quando o nível de qualidade de serviço requisitado for do tipo *Best Effort*, será montado um LSP na rede entre a origem e o destino, mas não haverá reserva de recursos nos demais elementos de rede que compõem o caminho. Esse fluxo de dados será trafegado na rede tendo como base apenas no protocolo de roteamento sobre o qual o MPLS se apóia, ou seja, utilizará os recursos remanescentes nos diversos roteadores que compõem o LSP. Entretanto, não é esperado que esse tipo de tráfego nunca possa ser enviado quando houver um elemento de rede congestionado.

3.2 - Funcionalidade Proposta aos Roteadores de *Backbone.*

Os roteadores de *backbone* LSR serão encarregados de fazer a leitura e a troca de *labels* dos pacotes de entrada em cada interface, encaminhando-os para as diversas interfaces de saída segundo uma tabela de *labels*, conhecida como LIB, montada e atualizada pelo RSVP-TE.

O mecanismo que atuará nos roteadores LSR agregará as seguintes informações a mais em sua LIB: nível de qualidade de serviço do LSP que será estabelecido na rede e prioridade de fila para o algoritmo de enfileiramento de pacotes.

A tabela 8 mostra um exemplo de LIB com as informações de nível de qualidade de serviço e prioridade de fila de cada LSP.

Tabela 8: Relação entre *Labels*, Interfaces, QoS e Prioridades.

Interf. Entrada	**Label IN**	**Interf. Saída**	**Label OUT**	**Qualidade de Serviço**	**Prioridade para o Algor. de Enfileir.**
2	10	3	15	*Guaranteed Service*	Máxima
3	13	2	54	*Controlled Load*	Média
1	10	1	12	*Best Effort*	Mínima
2	27	2	17	*Guaranteed Service*	Máxima

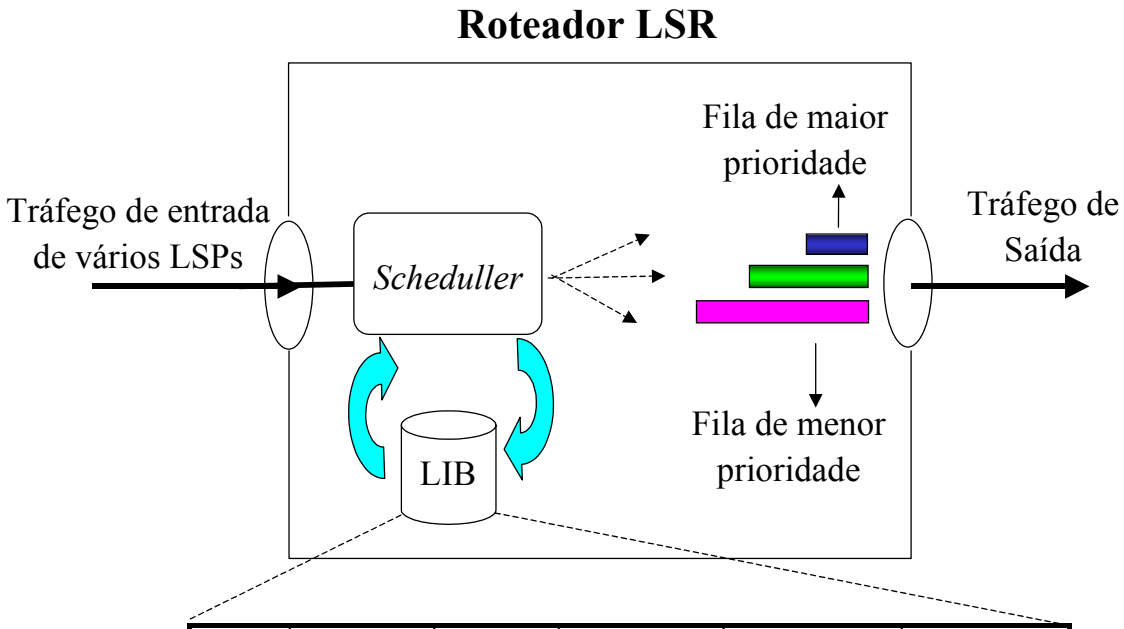
Uma vez determinada a qualidade de serviço do LSP na LIB, o mecanismo proposto deverá monitorar continuamente o tráfego de cada LSP direcionando-o para uma determinada fila de um algoritmo de enfileiramento de pacotes na interface de saída do roteador LSR. Portanto, o tráfego com nível de qualidade de serviço mais alto será direcionado para uma fila de prioridade máxima, a qual possuirá prioridade de envio de tráfego sobre as demais filas menos prioritárias, provendo assim uma clara distinção entre os níveis de QoS.

Dessa forma, as três qualidades de serviço do protocolo RSVP-TE (*Guaranteed Service*, *Controlled Load* e *Best Effort*) ficam, compativelmente, mapeadas e definidas em três filas com prioridades relativas de envio entre sí, da tecnologia DiffServ.

Existem diferentes algoritmos de enfileiramento propostos na literatura, como por exemplo: PQ - *Priority Queuing*, LLQ - *Low Latence Queuing*, etc. Alguns deles possuem características importantes para a obtenção de qualidade de serviço, como: baixa latência e *jitter* reduzido para aplicações *real time* ou são sensíveis a esses parâmetros. Podem garantir, também, que o tráfego de menor prioridade seja enviado ao longo do tempo, evitando assim o chamado "*Protocol Starvation*" (possibilidade do tráfego de baixa prioridade não ser enviado nunca, dependendo das condições da rede).

De maneira geral, os algoritmos de enfileiramento de pacotes permitem a utilização de várias filas com outras prioridades. Entretanto, como neste trabalho estão sendo propostos apenas três níveis de qualidade de serviço, somente as três primeiras filas do algoritmo serão utilizadas.

A figura 13 mostra um esquema de como o mecanismo proposto funcionará no roteador LSR.



Interf. *In*	***Label In***	**Intef. *Out***	***Label Out***	**Nível QoS**	**Prioridade**
2	10	3	15	*Guaranteed Service*	Máxima
1	7	2	9	*Best Effort*	Mínima

Figura 13: Funcionamento do mecanismo proposto ao LSR.

3.3 - Exemplo do Funcionamento do RSVP-TE com os Novos Mecanismos Propostos

Através de um exemplo de conexão entre as redes do cliente e do provedor, será descrito o funcionamento fim a fim do protocolo RSVP-TE com os novos mecanismos descritos nos itens 3.1 e 3.2.

Para tanto, supõe-se que o SLA descrito na tabela 9 tenha sido definido previamente entre cliente e provedor.

Tabela 9: Exemplo de SLA entre Cliente e Provedor.

	Aplicação 1	**Aplicação 2**	**Aplicação 3**	**Aplicação 4**
Descrição	Voz sobre IP	Transf. Arquivo + e-mail	ERP	Internet
Banda Passante (kbps)	256	512	512	256
Veloc. de Pico (kbps)	256	512	512	256
***Burst* (*bits*)**	5.000	10.000	10.000	10.000
QoS	*Guaranteed Service*	*Best Effort*	*Controlled Load*	*Best Effort*
Config. do Campo DSCP	00XXXX	10XXXX	01XXXX	10XXXX

3.3.1 - Etapa 1 – Definição do Nível de Qualidade de Serviço do LSP e Envio da PATH *Message*.

O roteador LER do provedor, que está conectado à rede do cliente, analisa o campo DSCP dos pacotes em sua interface de entrada. Uma vez identificado o valor desse campo, o nível de qualidade de serviço do LSP, a ser estabelecido na rede, é definido através de consulta à tabela de QoS X valor do campo DSCP do pacote.

Nesse exemplo, é suposto que não existe nenhum LSP com o nível de QoS solicitado estabelecido na rede, sendo, portanto, necessário estabelecê-lo antes de poder trafegar os dados.

O roteador LER de origem monta a PATH_*Message* do RSVP-TE encaminhando-a para o LSR diretamente conectado. Nesse roteador, serão reservados os recursos para garantir o nível de qualidade de serviço solicitado pela aplicação e requisitada a reserva de um *label*. Se essas duas ações forem completadas com sucesso, a PATH *Message* será enviada ao próximo LSR do caminho, caso contrário, será gerada uma mensagem de erro e o LSP não será estabelecido na rede.

Esse processo é repetido em todos os LSRs que compõem o caminho definido pelo IGP, até alcançar o LER remoto.

3.3.2 - Etapa 2 – Alocação dos Recursos nos Roteadores de *Backbone.*

Quando o roteador LER remoto receber a PATH_*Massage*, será gerada uma mensagem de retorno RESV_*Message*. Esta mensagem de retorno passará pelos mesmos roteadores LSR pelos quais a PATH_*Message* passou, alocando definitivamente os recursos previamente reservados pela PATH_*Massage* e definindo um *label*.

No exemplo proposto, supondo que a aplicação de voz deseje transmitir dados para a localidade remota, será montado um LSP com nível de QoS *Guaranteed Service* com a banda passante de 256 Kbps.

3.3.3 - Etapa 3 – Estabelecimento do LSP.

Quando a RESV_*Message* chegar no roteador LER que originou o pedido de estabelecimento do LSP, o caminho é estabelecido na rede, estando pronto para trafegar os pacotes gerados pelas aplicações.

De maneira semelhante, outros LSPs com outros níveis de qualidade de serviço são estabelecidos na rede. A partir deste ponto, o mecanismo atuante no LER começará a direcionar os pacotes para os diversos LSPs estabelecidos na rede baseando-se no valor do campo DSCP de cada pacote.

Quando o tráfego dos vários LSPs estiver passando dentro dos roteadores LSRs, além da troca de *Label* (*IN – OUT*) que acontece nos pacotes, o LSR consultará o nível de qualidade de serviço desse LSP e encaminhará seus pacotes para uma determinada fila de seu algoritmo de enfileiramento, com prioridade de envio compatível com o nível de QoS do LSP.

Dessa forma, os níveis de qualidade de serviço de cada LSP são rigidamente preservados e mantidos.

3.3.4 - Etapa 4 – Recebimento dos Pacotes pelo Roteador do Cliente no Destino.

Quando o pacote de um determinado LSP alcançar o roteador LER remoto, o campo relativo ao *label* do pacote é retirado, entregando-o na rede local do cliente exatamente como o LER de origem o recebeu: um pacote IP com o campo DSCP configurado para uma determinada prioridade.

A figura 14 mostra um exemplo de conexão entre a rede do cliente e a rede do provedor. Nesse exemplo, o cliente pode conter pontos remotos de sua rede sendo interligados de várias maneiras, como por exemplo: rede de um provedor de serviços de telecomunicações, a sua própria rede coorporativa ou até mesmo a Internet.

Nesse último caso, para manter fim a fim o nível de qualidade de serviço solicitado pela aplicação, todos os roteadores que compõem o caminho utilizado para interligar os pontos origem e destino deverão suportar os protocolos MPLS e o RSVP-TE estendido com as novas funcionalidades propostas neste trabalho.

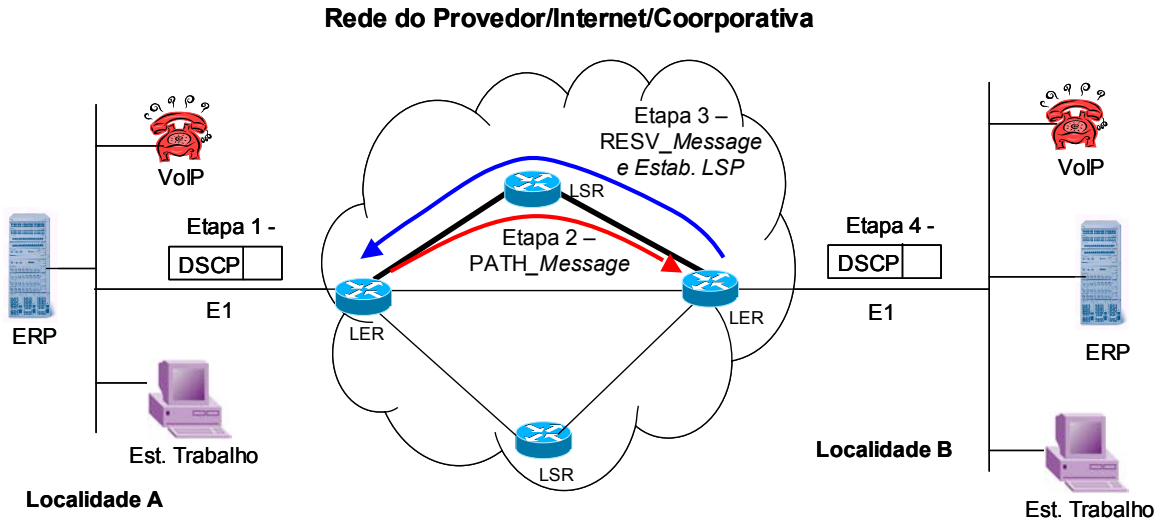


Figura 14: Esquema de rede entre Cliente e Provedor mostrando o funcionamento das novas propostas.

3.4 - Definição do Ambiente Onde as Novas Propostas Podem ser Aplicadas.

Basicamente, as propostas apresentadas neste trabalho, para estender as funcionalidades do RSVP-TE, são aplicáveis em redes cuja infra-estrutura básica de transmissão de dados seja baseada em MPLS, ATM ou *Frame Relay*, tendo o RSVP-TE como o protocolo de sinalização dessas redes, ou seja, realizando a alocação e troca de *labels* entre as interfaces dos diversos equipamentos. Entretanto, apesar da possibilidade de utilizar RSVP-TE em redes ATM e *Frame Relay*, ele é mais comumente utilizado em redes IP/MPLS.

Dessa forma, essas novas propostas podem ser aplicadas em redes coorporativas das empresas em geral, redes de provedores de serviços de telecomunicações ou até mesmo na Internet. Em redes coorporativas e de provedores, por se tratarem de ambientes controlados e com uma tendência maior à utilização de equipamentos de mesmas características, essas propostas seriam mais eficientemente aplicáveis. Normalmente, essas redes se caracterizam por utilizarem equipamentos de um, ou no máximo dois, fabricantes, com um único protocolo de sinalização configurado.

Com relação à Internet, o problema que se apresenta é o grande número de protocolos e arquiteturas de rede que a compõe, sendo que muitas vezes os mesmos se completam. Portanto, é altamente improvável que exista, ou venha a existir, somente um único tipo de protocolo ou arquitetura de rede para toda a Internet, visto

a influência e pressão que grandes coorporações fazem para impor suas soluções de rede como padrão.

Entretanto, o fato de haver vários protocolos e arquiteturas de rede na Internet, não significa afirmar que as novas propostas apresentadas neste trabalho, e portanto, o próprio RSVP-TE, não sejam aplicáveis na Internet. Esse protocolo foi desenvolvido para suportar o estabelecimento de conexões mesmo por caminhos onde os equipamentos não suportem RSVP-TE. Nesse caso, o equipamento que suporta esse protocolo, ao saber que o equipamento diretamente conectado a ele não suporta RSVP-TE, não enviará adiante o *Label Request Object* (Awduche et al., 2001 p. 23). Nesse caso, o nível de qualidade de serviço fim a fim não poderá ser mantido para uma aplicação.

Entretanto, uma rede que suporte o protocolo RSVP-TE poderá ter problemas de escalabilidade, já que se trata de uma arquitetura orientada a conexão e, portanto, no estabelecimento de circuitos virtuais. Além disso, o RSVP-TE é baseado no conceito de *Soft State*, pois depende da troca de mensagens de controle entre origem e destino, para que o circuito virtual continue ativo.

Portanto, em uma rede com muitos clientes e, conseqüentemente com muitos circuitos, poderá haver uma carga considerável de processamento nos elementos de rede para controlar e manter todos os caminhos virtuais estabelecidos na rede. Esse possível problema é uma das sugestões de continuidade dessa pesquisa.

Capítulo 4 - Benefícios Adicionados ao Protocolo RSVP-TE e Resultados Experimentais.

O objetivo desse capítulo é mostrar os benefícios obtidos pelo administrador de uma rede IP com a utilização das novas propostas para estender as funcionalidades do protocolo RSVP-TE, bem como, comentar os vários cenários de testes utilizados em ambiente simulado. Também são descritos os resultados alcançados nos vários cenários propostos.

4.1 - Benefícios Adicionados ao RSVP-TE.

Os benefícios adicionados ao protocolo RSVP-TE com a utilização das novas propostas são:

4.1.1 - Maior Versatilidade

Com a capacidade de interpretar o campo DSCP do cabeçalho IP dos pacotes oriundos das aplicações dos clientes, o protocolo RSVP-TE ganha mais versatilidade com relação à sua utilização, pois parece ser uma tendência as redes IP adotarem a tecnologia DiffServ para prover qualidade de serviço aos clientes, mesmo a despeito de não manterem fim a fim o nível de QoS solicitado pela aplicação, quando houver congestionamento na rede.

Entretanto, em situações normais, não é esperado haver congestionamento na rede do cliente, visto que se trata de um ambiente controlado. Portanto, a tecnologia DiffServ, nessa parte do circuito, não comprometeria a qualidade de serviço fim a fim.

Outra característica importante desse benefício é a total transparência para o cliente ou aplicação, que já utilize o campo DSCP para indicar o nível de qualidade de serviço requisitado, com relação à utilização do RSVP-TE com as novas funcionalidades no *backbone* da rede.

4.1.2 - Não Necessidade de Utilização do *Software* RSVP Cliente.

Para que o cliente se beneficiasse da utilização dos níveis de qualidade de serviço dos protocolos RSVP/RSVP-TE por aplicação, era necessário carregar em suas máquinas um *software* RSVP cliente, para estabelecer uma sessão entre fonte e destino, mantendo fim a fim o nível de QoS, conforme descrito no item 1.4.

Com a nova funcionalidade proposta aos roteadores de borda da rede, não é mais necessária à utilização do *software* RSVP cliente, já que não haverá mais sessões sendo estabelecidas diretamente entre os *hosts*. As sessões, ou túneis LSP, serão estabelecidos somente dentro do *backbone* da rede, com o nível de qualidade de serviço mapeado do campo DSCP do pacote IP configurado pela aplicação.

4.1.3 - Clara Definição dos Níveis de Qualidade de Serviço.

A definição entre os níveis de qualidade de serviço dos protocolos RSVP/RSVP-TE é subjetiva, conforme descrito no item 1.4.

Com a utilização da funcionalidade proposta aos roteadores do *backbone* da rede, conforme descrito no item 3.2, esses protocolos passam a oferecer aos clientes níveis de QoS bem definidos, já que os pacotes pertencentes a aplicações mais importantes serão direcionados a filas de alta prioridade de envio. Os pacotes de aplicações menos importantes serão direcionados a filas de baixa prioridade de envio, de acordo com as regras do algoritmo de enfileiramento de pacotes utilizado.

4.1.4 - Possibilidade de Haver Vários Níveis de Qualidade de Serviço.

De acordo com as novas propostas, o nível de qualidade de serviço solicitado pela aplicação está associado à configuração do campo DSCP do pacote IP, que por sua vez está relacionado a uma fila de envio de um algoritmo de enfileiramento de pacotes.

Portanto, o número de níveis de QoS a serem oferecidos às aplicações dos clientes será igual ao número de filas de envio do algoritmo de enfileiramento utilizado nos equipamentos de rede. Dessa forma, o protocolo RSVP-TE não ficaria limitado aos três níveis de qualidade de serviço definidos originalmente na arquitetura IntServ.

4.2 - Descrição e Comentário dos Resultados dos Testes Realizados no Simulador NS-2.

Para determinar o comportamento dos equipamentos de *backbone* e de borda da rede com as novas propostas, foi utilizado o simulador de rede NS-2 - *Network Simulator* versão 2 (Fall & Varadhan, 2003).

O NS-2 foi desenvolvido pelo LBNL - *Lawrence Berkeley National Laboratory* da Universidade da Califórnia e pode ser obtido da WEB gratuitamente, sendo compatível com vários sistemas operacionais. O NS-2 pode simular diferentes tipos de tráfego em diferentes topologias, possuindo uma arquitetura aberta e permitindo aos seus usuários contribuir com novas funcionalidades (Califórnia, s.d.).

O simulador é orientado a objeto e seu código é escrito em C++, o qual é utilizado para definir as classes dos vários objetos que compõem uma rede, bem como para executar os processos de simulação do ambiente a ser testado como um todo. O simulador também utiliza a linguagem Otcl, que é uma versão orientada a objeto da linguagem tcl - *tool command language*. O Otcl é utilizado como uma interface de programação, mais amigável, dos *scripts* a serem rodados entre o usuário e o simulador (Pieda et al., 2000).

Não foi possível testar a funcionalidade proposta aos roteadores de borda da rede em virtude da não existência de módulos que implementassem os protocolos RSVP ou RSVP-TE no simulador NS-2. Adicionalmente, não foi considerado o desenvolvimento desses módulos por não fazerem parte do escopo deste trabalho.

A funcionalidade proposta aos roteadores de *backbone* da rede, no que diz respeito ao tratamento dado ao tráfego oriundo dos diversos LSPs, de diferentes níveis de QoS do RSVP-TE, foi testada instalando-se o módulo que implementa a tecnologia DiffServ no simulador. Esse módulo utiliza o algoritmo de enfileiramento de pacotes PQ para priorizar o tráfego segundo uma determinada configuração do campo DSCP do pacote IP.

Existem diferentes algoritmos de enfileiramento propostos na literatura, como por exemplo, PQ, LLQ, CBWFQ – *Class Based Weighted Fair Queueing*, etc. Entretanto, foge ao escopo deste trabalho um estudo aprofundado dos mecanismos de enfileiramento, para se determinar o melhor deles. Assim, sem perda de generalidade, o ambiente de simulação utiliza o algoritmo PQ, que é implementado pelo NS- 2.

4.2.1 - Ambiente de Teste Utilizado.

O ambiente montado no simulador para testar o comportamento das três fontes de tráfego, nos vários cenários de teste, está esquematizada na figura 15.

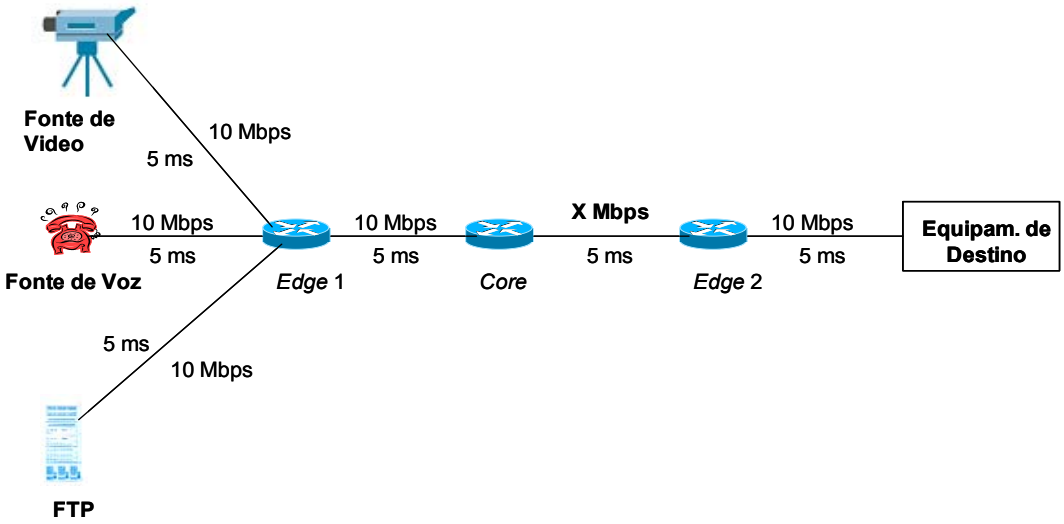


Figura 15: Ambiente de teste utilizado no simulador.

Para simular as aplicações na rede do cliente, foram configuradas três fontes de tráfego: uma fonte de tráfego de vídeo com prioridade máxima, uma fonte de tráfego de voz com prioridade intermediária e uma fonte de transferência de arquivos (FTP) com prioridade mínima, simulando uma aplicação puramente de dados não sendo do tipo *real time*.

As fontes de tráfego estão conectadas ao equipamento *edge* 1, que realiza a agregação dos três tipos de tráfego. Esse equipamento também configura o campo DSCP dos pacotes de acordo com o tipo de tráfego:

a) Pacotes de Vídeo: valor do campo DSCP igual a 001010 (10).

b) Pacotes de Voz: valor do campo DSCP igual a 010100 (20).

c) Pacotes de FTP: valor do campo DSCP igual a 011110 (30).

O equipamento *edge* 1 está conectado ao equipamento *core*, que, efetivamente, faz a priorização de tráfego de acordo com os valores do campo DSCP. Nesse equipamento foi configurado o algoritmo de enfileiramento de pacotes PQ, o qual possui 4 filas principais (da fila 0, que é a mais prioritária, até a fila 3, que é a menos prioritária), sendo que cada uma possui até três sub-filas.

No ambiente de simulação, foi utilizada apenas uma sub-fila por fila principal, sendo que a fila 0, sub-fila 0, foi reservada para o tráfego de vídeo, a fila 1, sub-fila 0, para o tráfego de voz e a fila 2, sub-fila 0, para o tráfego de FTP. O algoritmo de enfileiramento PQ ainda permite que seja configurado um valor máximo de banda passante por fila, limitando assim seu *throughput*.

Portanto, o equipamento *edge* 1 simula a rede local do cliente, configurando o campo DSCP dos pacotes de acordo com o tipo da fonte de tráfego, e entregando-os para o equipamento *core* , em um único fluxo de dados.

O equipamento *edge* 2 recebe o tráfego do equipamento *core*, entregando-o ao equipamento de destino. O equipamento *edge* 2 não realiza nenhuma função especial e foi colocado no ambiente de teste apenas para simular a outra borda da rede.

O equipamento de destino recebe os três tipos de tráfego e é utilizado pelo módulo de monitoração de QoS do simulador, para calcular os quatro parâmetros que quantificam os níveis de qualidade de serviço por aplicação, sendo eles:

a) *throughput* em bps.

b) taxa de pacotes perdidos.

c) *jitter* em segundos.

d) *delay* em segundos.

O tempo de simulação para cada cenário foi de 60 segundos e o módulo de monitoração de QoS foi configurado para fazer as amostragens dos valores de *throughput*, *delay*, *jitter* e taxa de perda de pacotes a cada segundo.

A fonte de tráfego de vídeo é do tipo Pareto (Fall & Varadhan, 2003), a qual procura simular as oscilações inerentes a um sistema de transferência de imagens, ou seja, com períodos de picos de tráfego. A geração de dados dessa fonte foi configurada, na média, em 550 kbps com tamanho de pacote de 480 *bytes*.

A fonte de tráfego de voz foi configurada em 64 kbps, com tamanho de pacote de 90 *bytes*. A fonte de dados FTP foi configurada em 1.200 kbps, com tamanho de pacote de 1.500 *bytes*. As duas fontes apresentam um comportamento semelhante a um tráfego determinístico.

Portanto, a somatória dos tráfegos das três fontes será sempre igual a 1.814 kbps para todos os cenários de simulação descritos no item 4.2.2, salvo para os cenários onde houver tráfego interferente, o qual será definido oportunamente.

Todos os canais de comunicação entre os diversos equipamentos que compõem o ambiente de teste possuem a mesma latência de 5 ms e *throughput* de 10 Mbps. Entretanto, o circuito entre os equipamentos *edge* 2 e *core*, onde está configurado o algoritmo que prioriza o tráfego pelo campo DSCP, foi utilizado para simular situações de congestionamento, diminuindo-se o valor da banda passante desse canal.

4.2.2 - Cenários de Simulação.

Foram desenvolvidos vários cenários de teste para verificar o funcionamento da proposta feita aos roteadores de *backbone*. Cada cenário busca evidenciar uma condição onde o nível de qualidade de serviço solicitado pela aplicação é mantido dentro de níveis aceitáveis, ou é degradado em virtude de um fator específico dentro da rede.

Para poder determinar se o nível de qualidade de serviço das aplicações de vídeo, voz e FTP foi mantido, ou saber o quanto foi afetado ao passar pelos equipamentos de rede, foi utilizada a tabela 10 (Barra, 2003). Essa tabela mostra os valores limítrofes superiores e inferiores dos quatro parâmetros que medem o nível de qualidade de serviço.

Tabela 10: Valores dos parâmetros que medem QoS.

Parâmetro	Serviço de voz	Serviço de dados	Serviço de áudio e imagem
PDT_V1	100 ms	1000 ms	100 ms
PDT_V2	400 ms	4000 ms	400 ms
PDV_V1	1 ms	10 ms	1 ms
PDV_V2	100 ms	100 ms	100 ms
THRU_V2	10 kbps	1 kbps	10 kbps
THRU_V1	64 kbps	10 kbps	128 kbps
PLR_V1	10^{-4}	10^{-6}	10^{-4}
PLR_V2	10^{-3}	10^{-4}	10^{-3}

Os parâmetros descritos na tabela são:

- **PDT**: *Packet Delay Transfer* ou tempo de entrega dos pacotes.
- **PDV**: *Packet Delay Variation* ou variação no tempo de entrega dos pacotes. Também é conhecido como *Jitter*.
- **THRU:** *Throughput* ou banda passante pela qual a aplicação consegue enviar seus dados.
- **PLR**: *Packet Loss Rate* ou taxa de perda de pacotes.
- **V1**: Limite inferior
- **V2**: Limite superior

Portanto, de acordo com o resultado obtido na simulação dos diversos cenários, dir-se-á que a aplicação de voz, por exemplo, estará com seu nível de qualidade de serviço mantido se:

a) o tempo de entrega dos pacotes (PDT) estiver entre 100 e 400 ms.

b) a variação do tempo de entrega dos pacotes (PDV) estiver entre 1 e 100 ms.

c) o *throughput* da aplicação (THRU) estiver entre 10 e 64 Kbps.

d) a taxa de perda de pacotes (PLR) estiver entre 10^{-4} e 10^{-3}, ou seja, um pacote com erro entre 1.000 a 10.000 transmitidos.

4.2.2.1 - Cenário 1: Fila sem Prioridade e com Congestionamento

O objetivo desse cenário é mostrar que em um ambiente onde não existam mecanismos para manter o nível de qualidade de serviço solicitado pelas aplicações,

em situações de congestionamento, todas as aplicações são penalizadas independente de seus requisitos de qualidade de serviço solicitados.

Nesse ambiente, foi configurado o algoritmo de enfileiramento FIFO - *First In First Out* na interface de saída do equipamento *core*. A banda passante do canal entre os equipamentos *core* e *edge* 2 foi configurada em 1.500 kbps, sendo que a somatória dos tráfegos das três fontes é 1.814 kbps. Portanto, haverá congestionamento nesse ambiente.

As figuras 16 a 19 mostram o comportamento dos parâmetros *throughput*, *delay*, *jitter*, e taxa de perda de pacotes para esse cenário.

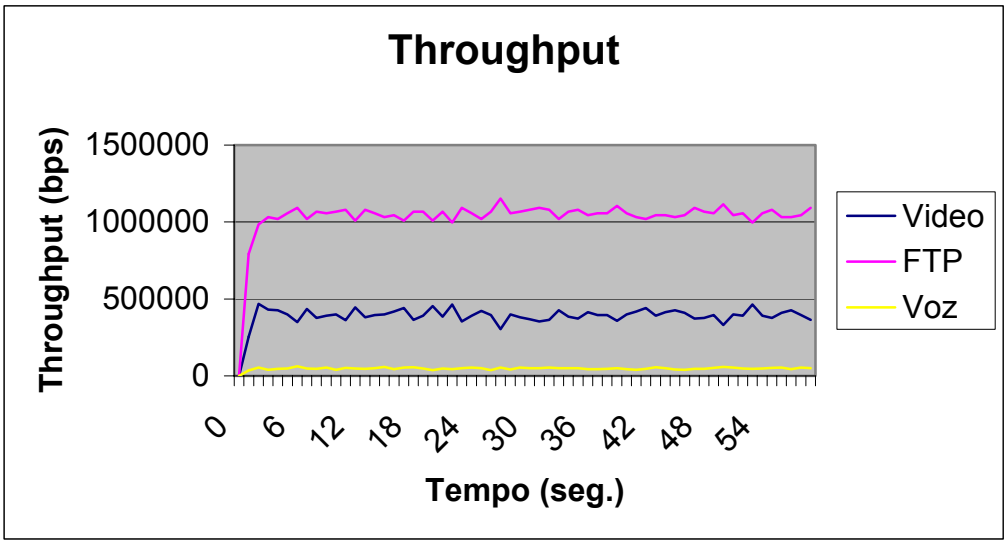


Figura 16: Comportamento do *Throughput* para o Cenário 1 - Fila sem Prioridade e com Congestionamento.

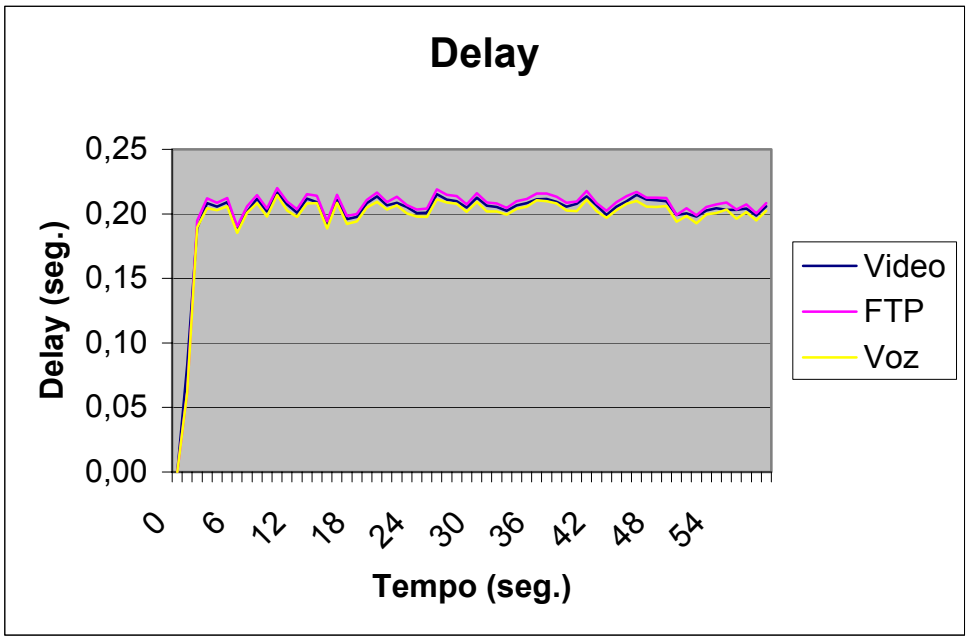


Figura 17: Comportamento do *Delay* para o Cenário 1 - Fila sem Prioridade e com Congestionamento.

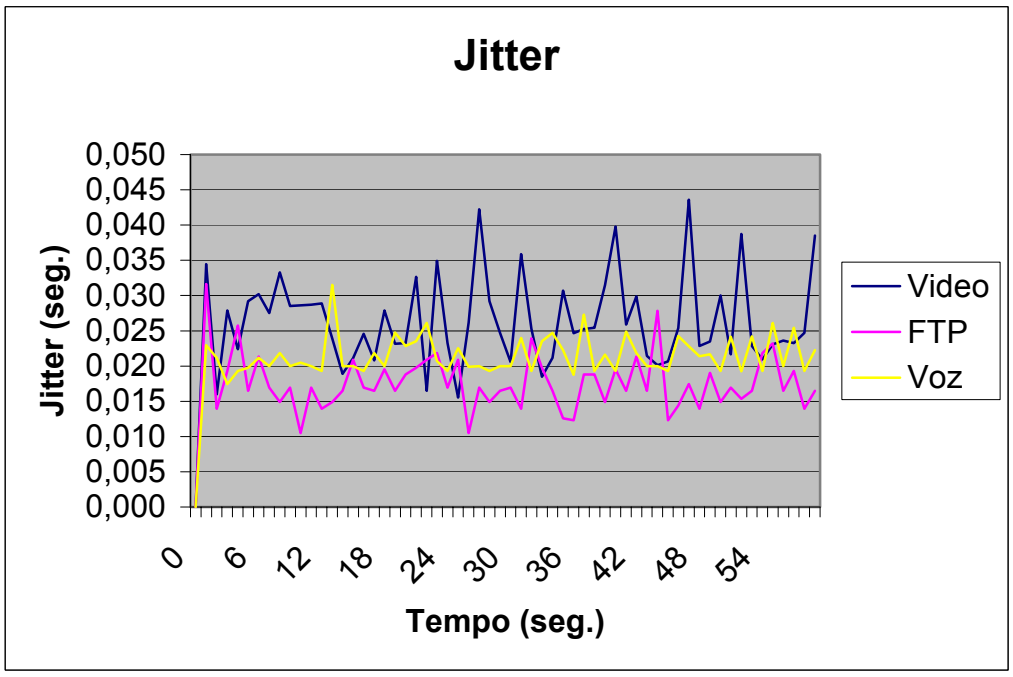


Figura 18: Comportamento do *Jitter* para o Cenário 1 - Fila sem Prioridade e com Congestionamento.

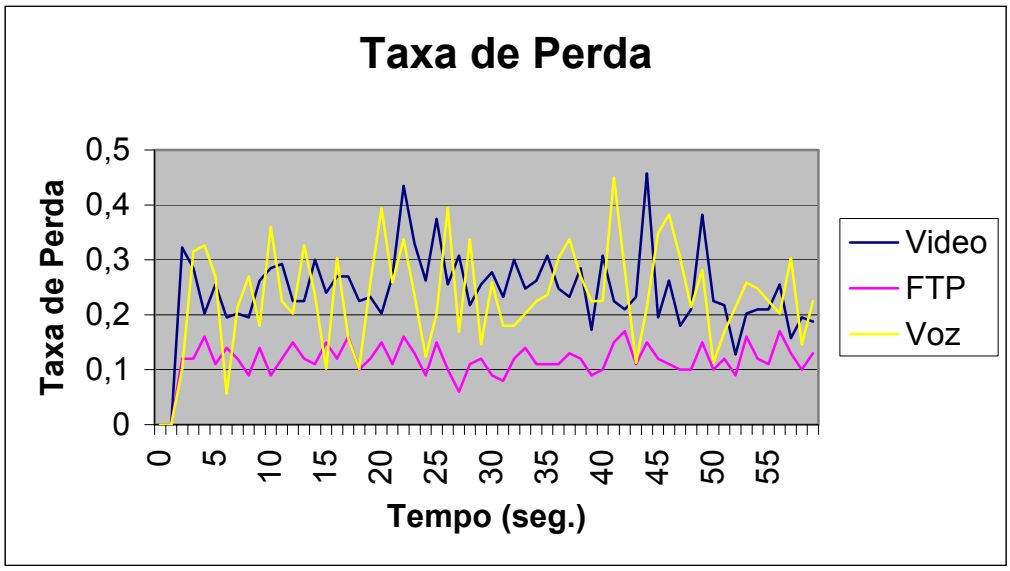


Figura 19: Comportamento da Taxa de Perda para o Cenário 1 - Fila sem Prioridade e com Congestionamento.

Pelos resultados apresentados, pode-se verificar que os valores de *throughput* e taxa de perda de pacotes ficam fora das especificações do nível mínimo de QoS das respectivas aplicações, conforme definidas na tabela 10.

Esse modelo de fila, ou algoritmo de enfileiramento de pacotes, é o mais comumente utilizado nos equipamentos que formam a Internet, provendo uma única qualidade de serviço, conhecida como *Best Effort*. Portanto, não serve para oferecer níveis de qualidade de serviço diferenciados para as aplicações.

4.2.2.2 - Cenário 2: Fila com Prioridade e sem Congestionamento

O objetivo desse cenário é mostrar que em um ambiente onde existam mecanismos para fornecer qualidade de serviço às aplicações, como por exemplo, um algoritmo de enfileiramento de pacotes com priorização de tráfego, e sem congestionamento, o nível de qualidade de serviço é mantido para todas as aplicações.

Para tanto, foi configurado na interface de saída do equipamento *core* o algoritmo de enfileiramentos de pacotes PQ, que prioriza o envio dos dados de determinadas filas.

Como foi descrita no item 4.2.1, a fonte de vídeo possuirá o campo DSCP configurado como 10 e estará associado à fila de mais alta prioridade. A fonte de voz possuirá o campo DSCP configurado como 20 e estará associada à fila de prioridade média e, por último, a fonte de dados FTP possuirá o campo DSCP configurado como 30 e estará associada à fila de menor prioridade.

Nesse cenário, a velocidade do circuito entre os equipamentos *core* e *edge* 2 está configurado em 2.200 kbps, não constituindo, portanto, congestionamento no ambiente.

As figuras 20 a 23 mostram o comportamento dos parâmetros de *throughput*, *delay*, *jitter* e taxa de perda de pacotes para esse novo cenário.

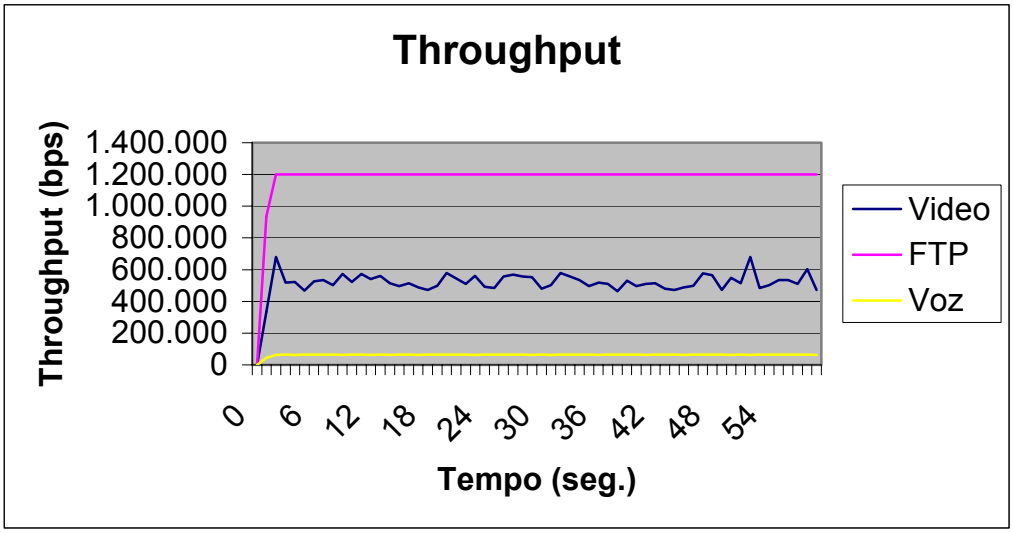


Figura 20: Comportamento do *Throughput* para o Cenário 2 - Fila com Prioridade e sem Congestionamento.

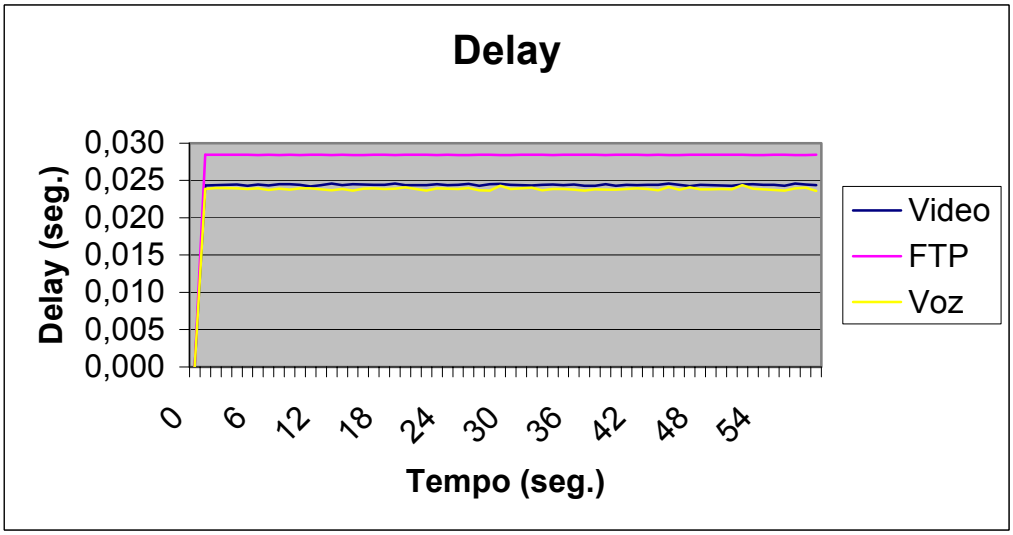


Figura 21: Comportamento do *Delay* para o Cenário 2 - Fila com Prioridade e sem Congestionamento.

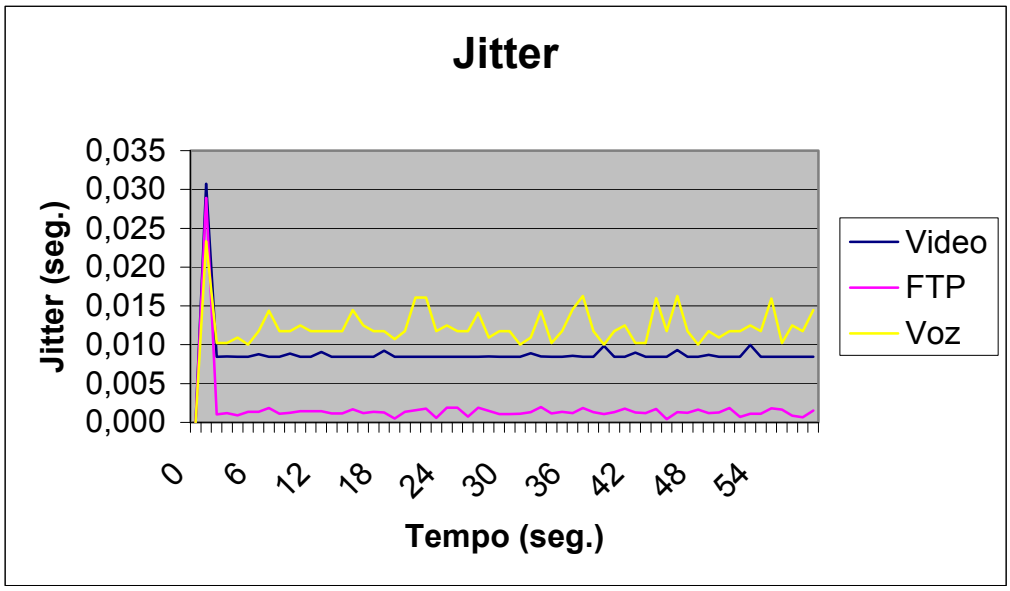


Figura 22: Comportamento do *Jitter* para o Cenário 2 - Fila com Prioridade e sem Congestionamento.

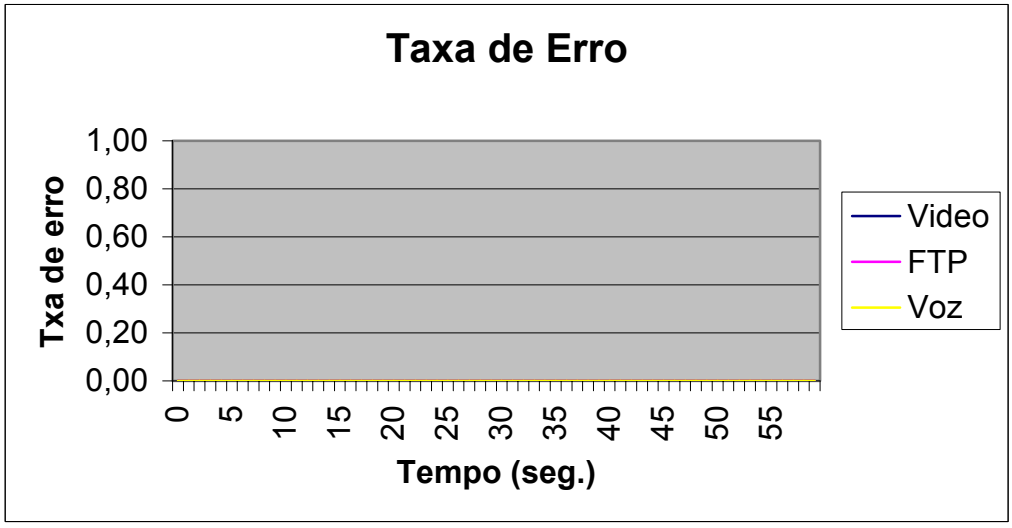


Figura 23: Comportamento da Taxa de Perda de pacotes para o Cenário 2 - Fila com Prioridade e sem Congestionamento.

Como nesse cenário não há congestionamento, todas as aplicações conseguem manter sua taxa máxima de transferência de arquivos, o *delay* das três aplicações são próximos ao valor de 25 a 30 ms, a taxa perda de pacotes é nula e o *jitter* das aplicações de voz e vídeo estão, na média, em 10 ms. O *jitter* da aplicação de FTP está na média em 2 ms, aproximadamente.

Embora a aplicação de FTP esteja associada à fila de menor prioridade o *jitter* dela foi menor que o *jitter* das aplicações de voz e vídeo. Esse fato pode ser

explicado pela grande diferença entre os tamanhos dos pacotes de vídeo (480 *bytes*), voz (90 *bytes*) e FTP (1500 *bytes*).

Portanto, todos os valores apresentados estão dentro dos especificados pela tabela 10, ou seja, o nível de qualidade de serviço das três aplicações foi mantido.

4.2.2.3 - Cenário 3: Fila com Prioridade e com Congestionamento

O objetivo desse cenário é mostrar que em um ambiente onde existam os mecanismos para fornecer qualidade de serviço, as aplicações mais prioritárias continuarão com seu nível de qualidade de serviço mantido, mesmo na ocorrência de congestionamentos na rede.

Para tanto, o cenário de teste utilizado no item 4.2.2.2 manteve-se praticamente inalterado para esse novo cenário, apenas a banda passante do canal de comunicação entre os equipamentos *core* e *edge* 2 foi diminuída para 1.200 kbps, gerando um severo congestionamento no ambiente.

As figuras 24 a 27 mostram o comportamento dos parâmetros de *throughput*, *delay*, *jitter* e taxa de perda de pacotes para esse ambiente.

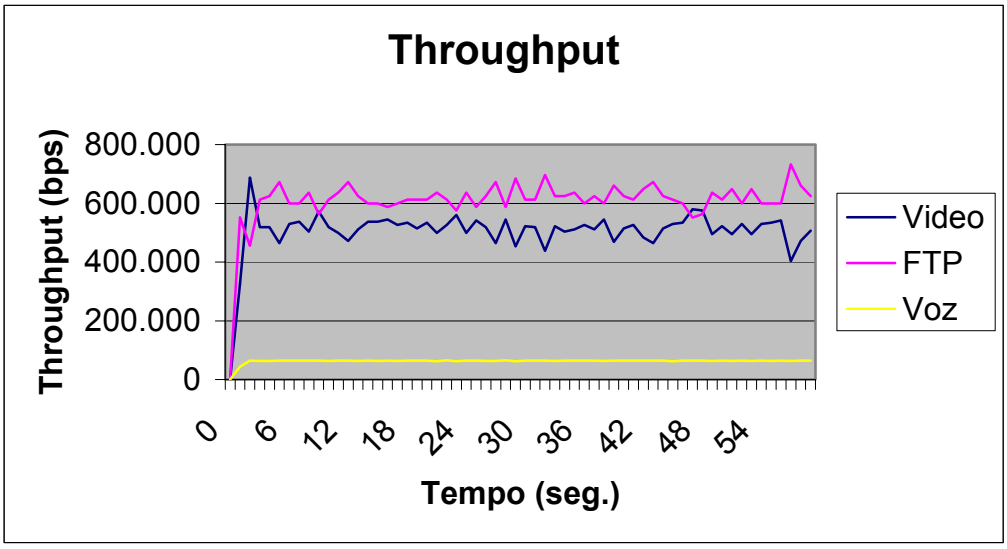


Figura 24: Comportamento do *Throughput* para o Cenário 3 - Fila com Prioridade e com Congestionamento.

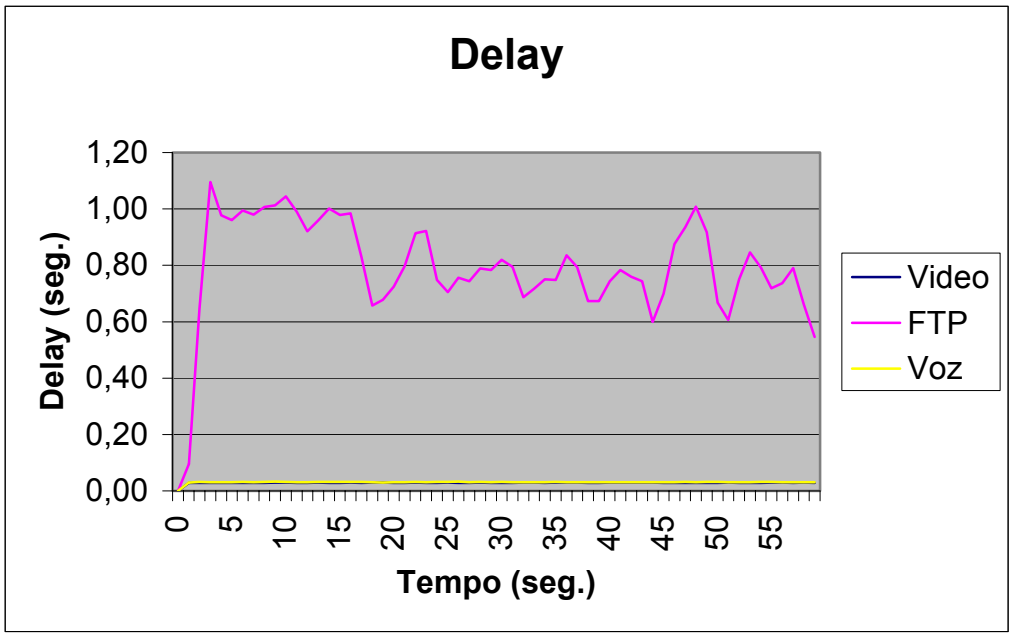


Figura 25: Comportamento do *Delay* para o Cenário 3 - Fila com Prioridade e com Congestionamento.

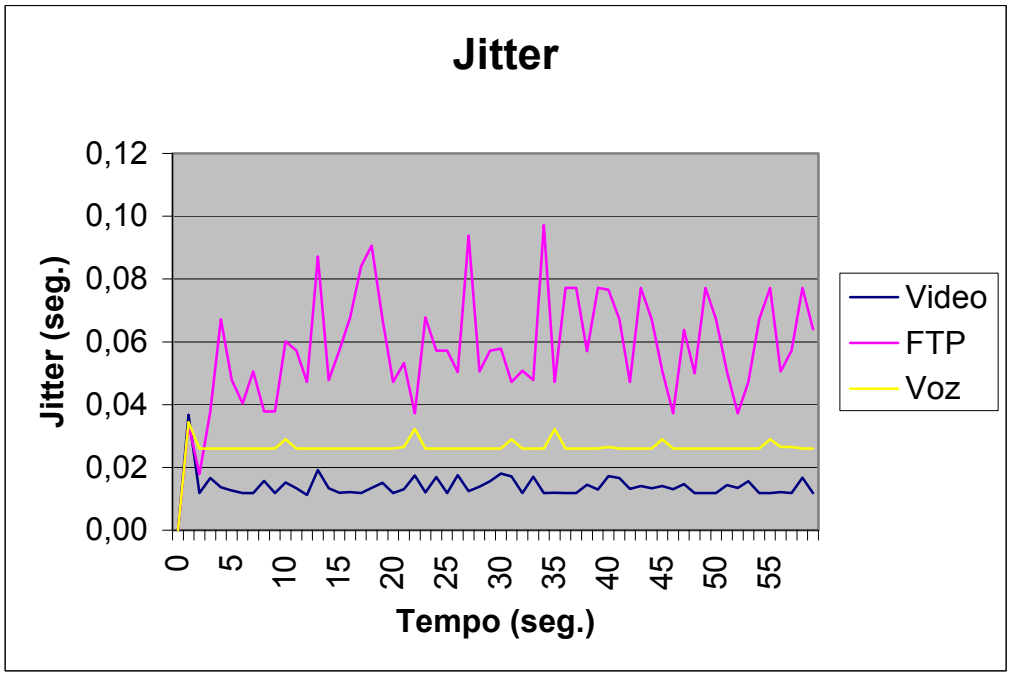


Figura 26: Comportamento do *Jitter* para o Cenário 3 - Fila com Prioridade e com Congestionamento.

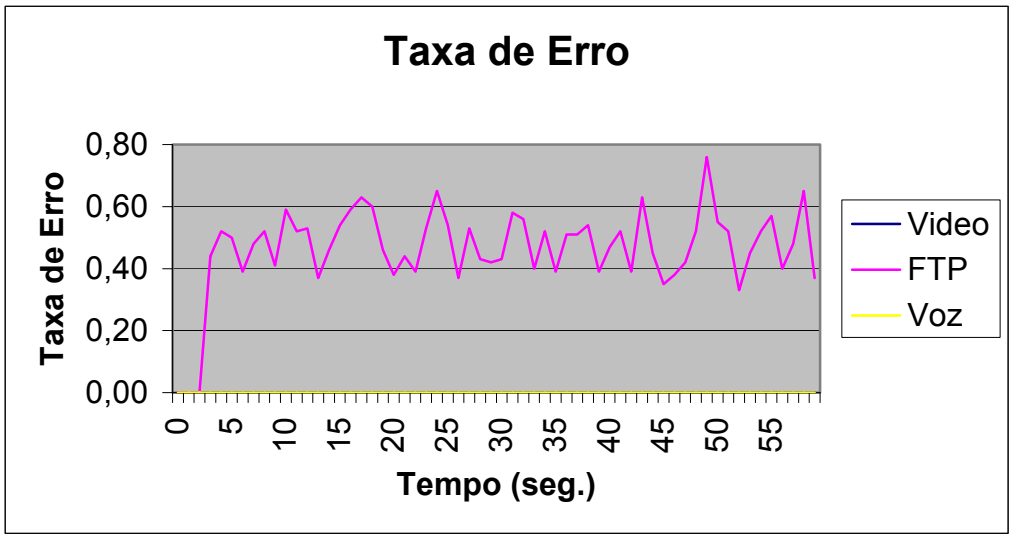


Figura 27: Comportamento da Taxa de Perda de pacotes para o Cenário 3 - Fila com Prioridade e com Congestionamento.

Nessa situação, as figuras 24 a 27 mostram que os níveis de QoS das aplicações de vídeo e voz foram preservados, mantendo os valores de *throughput* , *delay*, *jitter* e taxa de perda de pacotes iguais aos do cenário 4.2.2.2, o qual é um cenário sem congestionamento.

Em virtude da ocorrência de congestionamento, a aplicação de menor prioridade (FTP) foi penalizada em todos os parâmetros de QoS, com sensível aumento dos valores de *delay*, *jitter* e taxa de perda de pacotes e diminuição do valor de *throughput*.

4.2.2.4 - Cenário 4: Fila com Prioridade, sem Congestionamento e com Tráfego Interferente

O objetivo desse cenário é mostrar que no ambiente com a tecnologia DiffServ configurada e em redes onde não há disponibilidade de recursos para acomodar todo o tipo de tráfego, ao ser aceita mais uma fonte de tráfego de maior prioridade, a fonte de tráfego de menor prioridade, será penalizada para que o equipamento desvie mais recursos do ambiente, os quais já estavam alocados para outros fluxos de dados já estabelecidos na rede, para atender as requisições de qualidade de serviço dessa nova fonte de tráfego.

Para tanto, foi considerada a entrada de um tráfego interferente no ambiente proveniente de uma hipotética situação de contingência, em virtude de rearranjo de rotas e desvios de tráfego para outros caminhos.

Para simular a entrada desse tráfego interferente, foi acrescentada, ao ambiente de teste, mais uma fonte de tráfego de vídeo com as mesmas características da fonte de vídeo já existente. O *throughput* da fonte de vídeo interferente foi configurado em 550 kbps. Esse tráfego, por ser de alta prioridade, também possuirá o

valor do campo DSCP configurado como 10 e estará associado à fila de mais alta prioridade. Esse tráfego entrará no ambiente depois de decorridos 30 segundos de simulação.

Nesse cenário, não existe congestionamento na rede até o momento da entrada do tráfego interferente de vídeo. A somatória do tráfego antes da entrada do vídeo interferente é 1.814 kbps, e a somatória do tráfego depois da entrada do vídeo interferente é 2.364 kbps. A banda passante do canal entre os equipamentos *core* e *edge* 2 foi configurada em 2.300 kbps. As configurações das outras fontes de tráfego foram mantidas iguais às configurações do cenário do item 4.2.2.2.

A figura 28 mostra a configuração desse novo cenário.

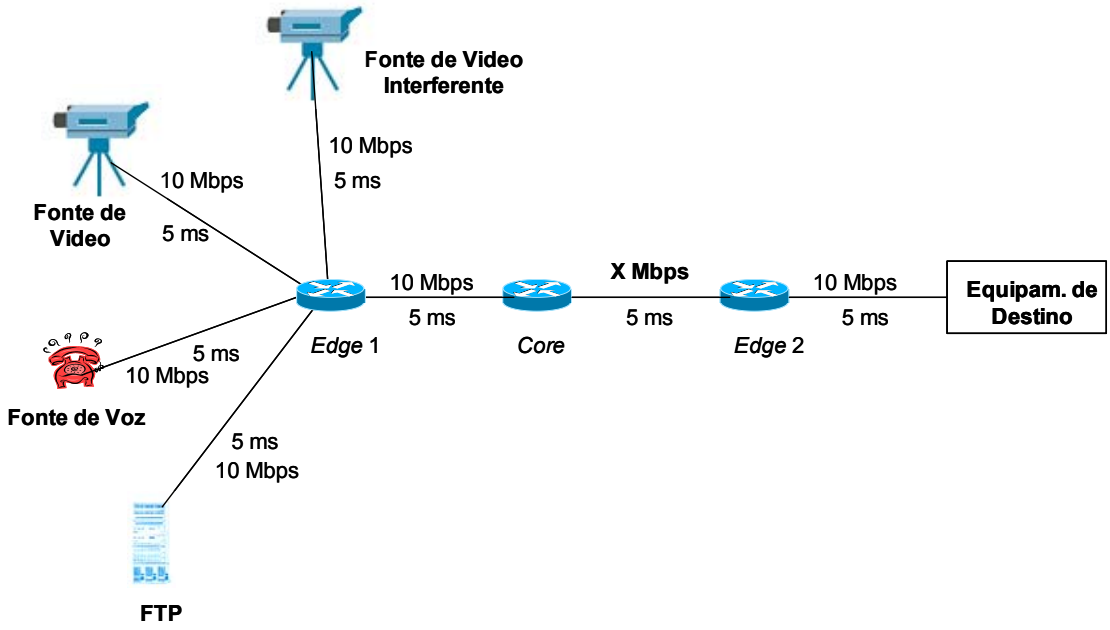


Figura 28: Ambiente de teste com entrada de tráfego interferente.

As figuras 29 a 32 mostram o comportamento dos parâmetros de *throughput*, *delay*, *jitter* e taxa de perda de pacotes para esse novo ambiente.

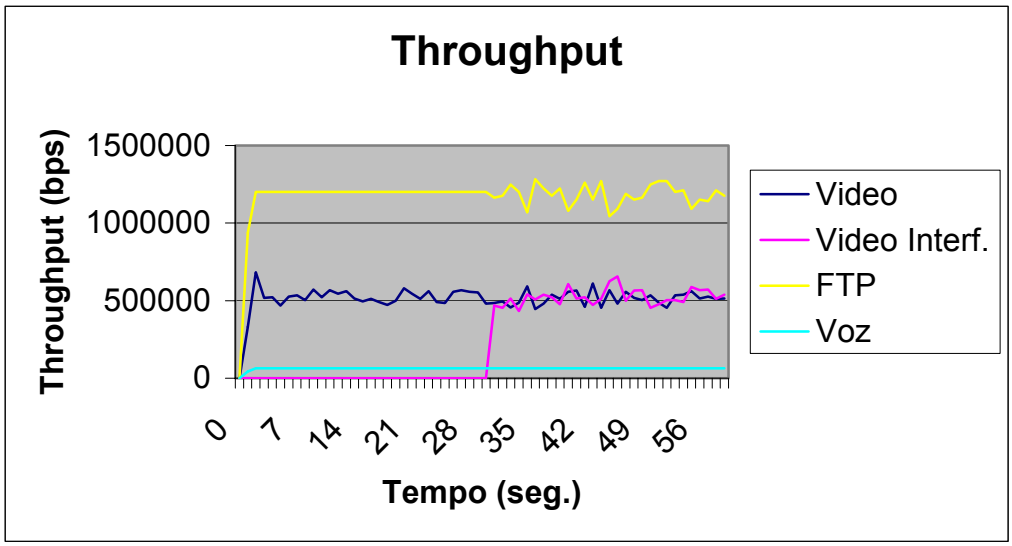


Figura 29: Comportamento do *Throughput* para o Cenário 4 - Fila com Prioridade, sem Congestionamento e com Tráfego Interferente.

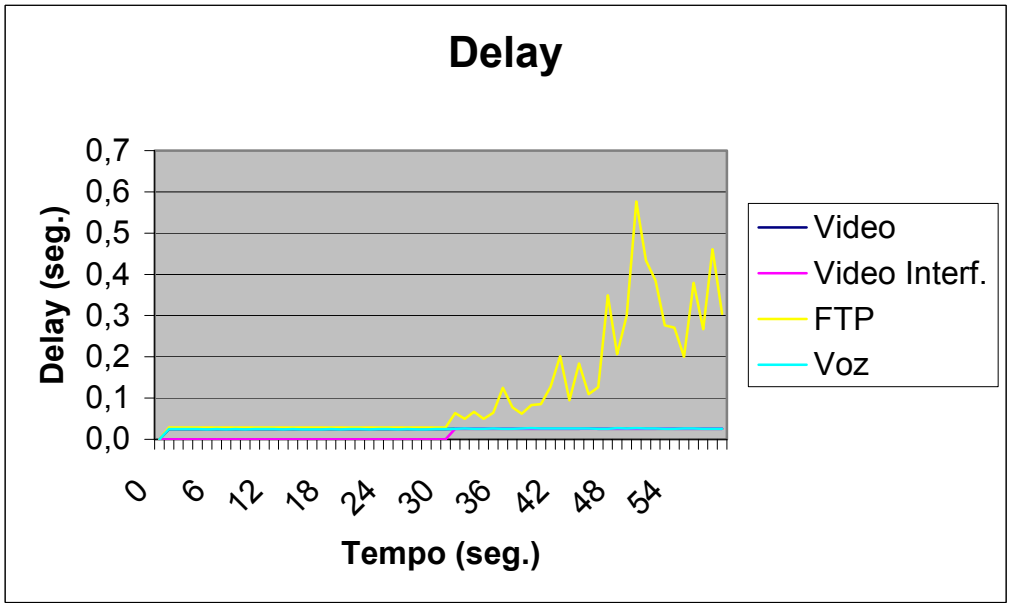


Figura 30: Comportamento do *Delay* para o Cenário 4 - Fila com Prioridade, sem Congestionamento e com Tráfego Interferente.

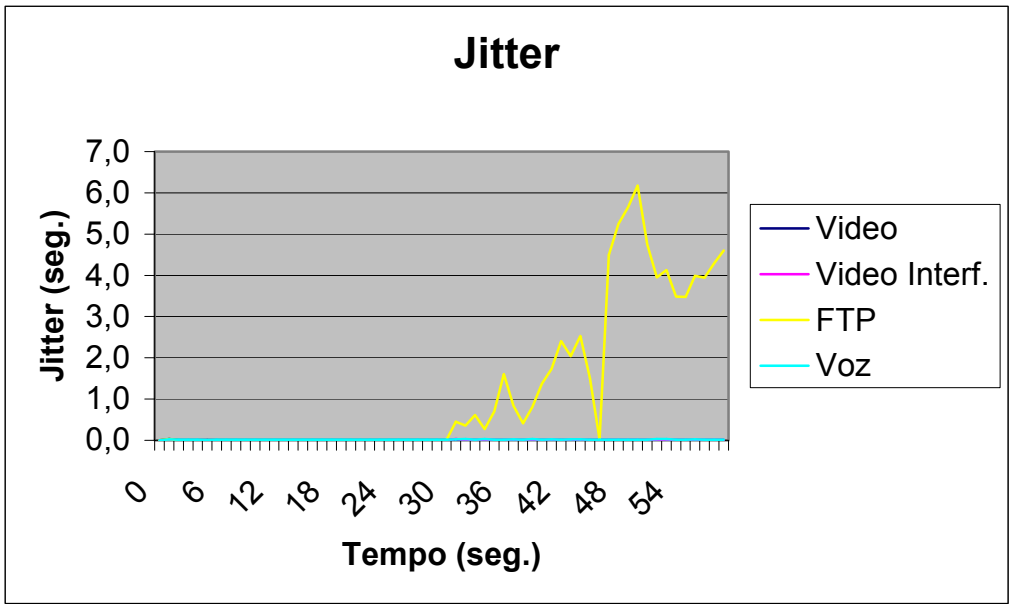


Figura 31: Comportamento do *Jitter* para o Cenário 4 - Fila com Prioridade, sem Congestionamento e com Tráfego Interferente.

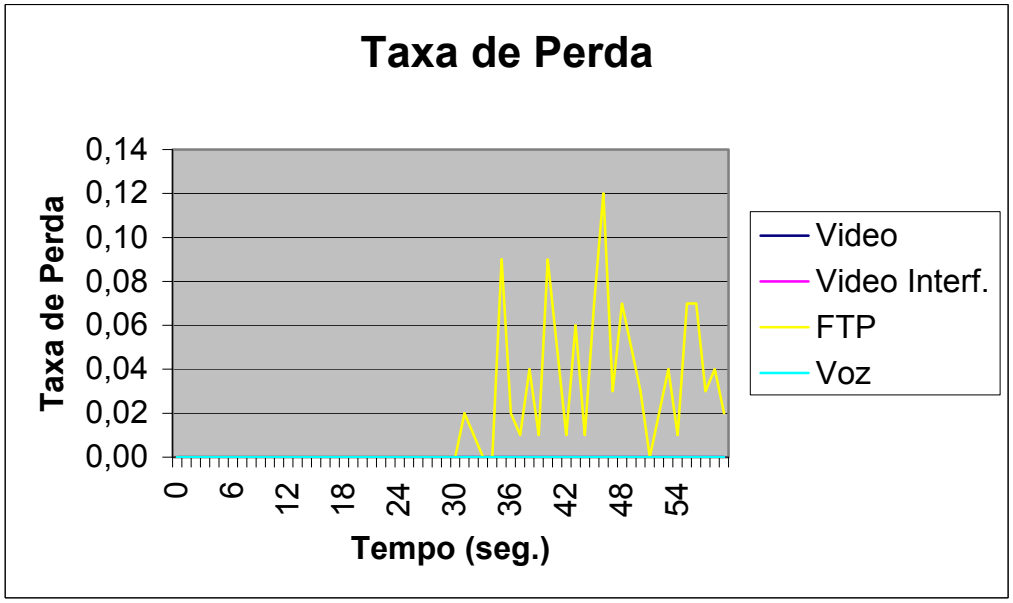


Figura 32: Comportamento da Taxa de Perda de pacotes para o Cenário 4 - Fila com Prioridade, sem Congestionamento e com Tráfego Interferente.

As figuras 29 a 32 mostram que após a entrada do tráfego interferente apenas a aplicação de FTP foi penalizada para acomodar a nova fonte de vídeo. Para essa aplicação, os valores de *delay*, *jitter* e taxa de perda de pacotes, que se mantinham iguais aos de um ambiente sem congestionamento, passaram a sofrer degradação após a entrada do tráfego interferente, bem como o *throughput* passou a sofrer oscilações.

Em contrapartida, as aplicações de voz e vídeo continuaram com seus valores praticamente inalterados, sendo que aplicação de vídeo interferente possui comportamento igual à aplicação de vídeo original.

4.2.2.5 - Cenário 5: Fila com Prioridade, sem Congestionamento, com Tráfego Interferente e com limitação na fila de vídeo.

O objetivo desse cenário é mostrar que em um ambiente com a tecnologia DiffServ configurada, havendo limitação na entrada de novos fluxos de dados na rede, os fluxos de dados já estabelecidos na rede com mesma prioridade do fluxo de dados interferente, serão afetados em seus níveis de qualidade de serviço, bem como o próprio fluxo de dados interferente.

Nesse cenário, além de existir a entrada de um tráfego interferente de vídeo, após 30 segundos de simulação, foi configurada uma limitação na banda passante da fila de mais alta prioridade do algoritmo de enfileiramento no valor de 682 Kbps. Portanto, o tráfego de vídeo original mais o tráfego de vídeo interferente será limitado nesse valor. Os demais parâmetros de configuração do ambiente foram mantidos iguais aos parâmetros do cenário do item 4.2.2.4.

As figuras 33 a 36 mostram o comportamento dos parâmetros de *throughput*, *delay*, *jitter* e taxa de perda de pacotes para este cenário.

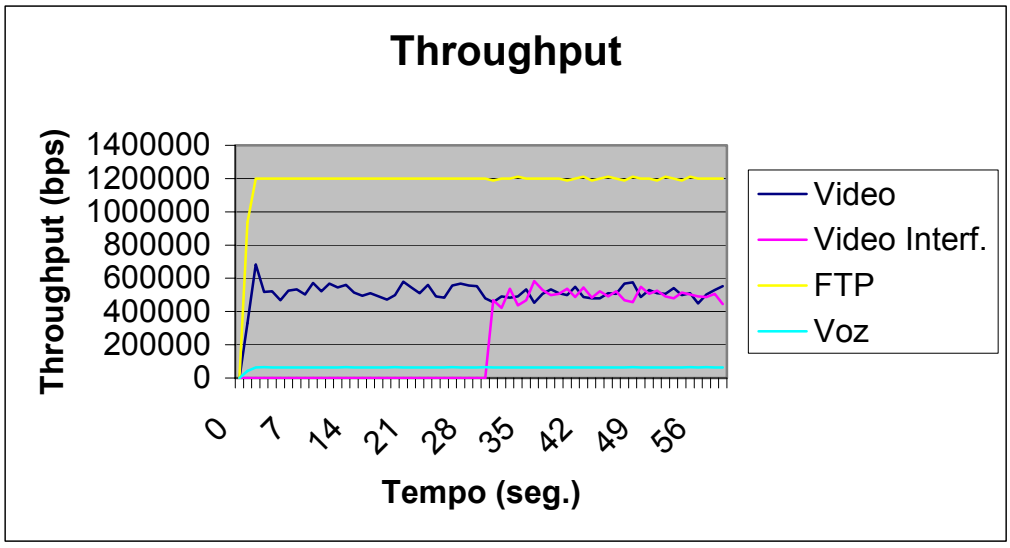


Figura 33: Comportamento do *Throughput* para o Cenário 5 - Fila com Prioridade, sem Congestionamento, com Tráfego Interferente e com Limitação na Fila de Vídeo.

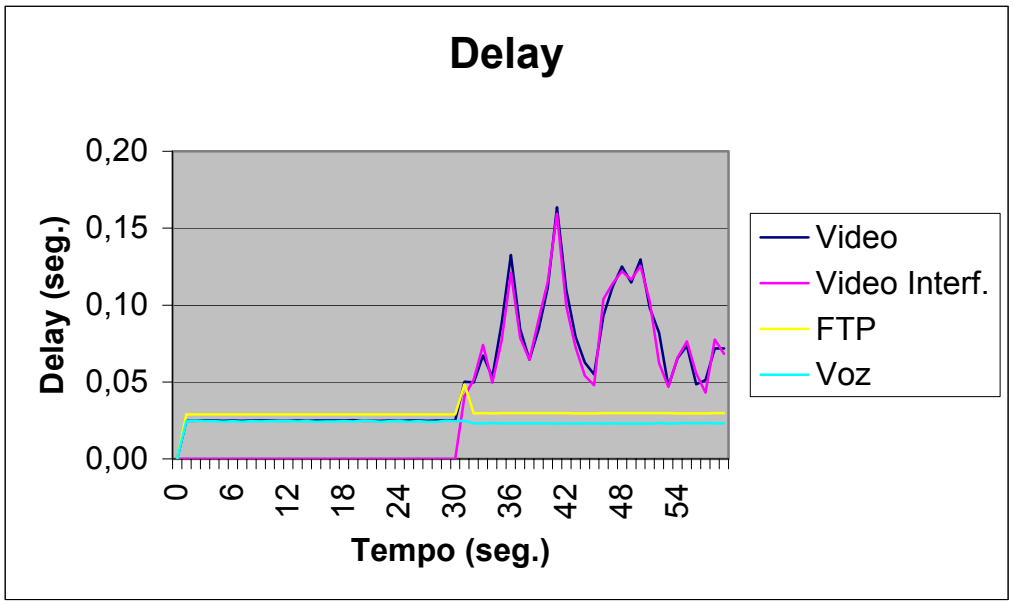


Figura 34: Comportamento do *Delay* para o Cenário 5 - Fila com Prioridade, sem Congestionamento, com Tráfego Interferente e com Limitação na Fila de Vídeo.

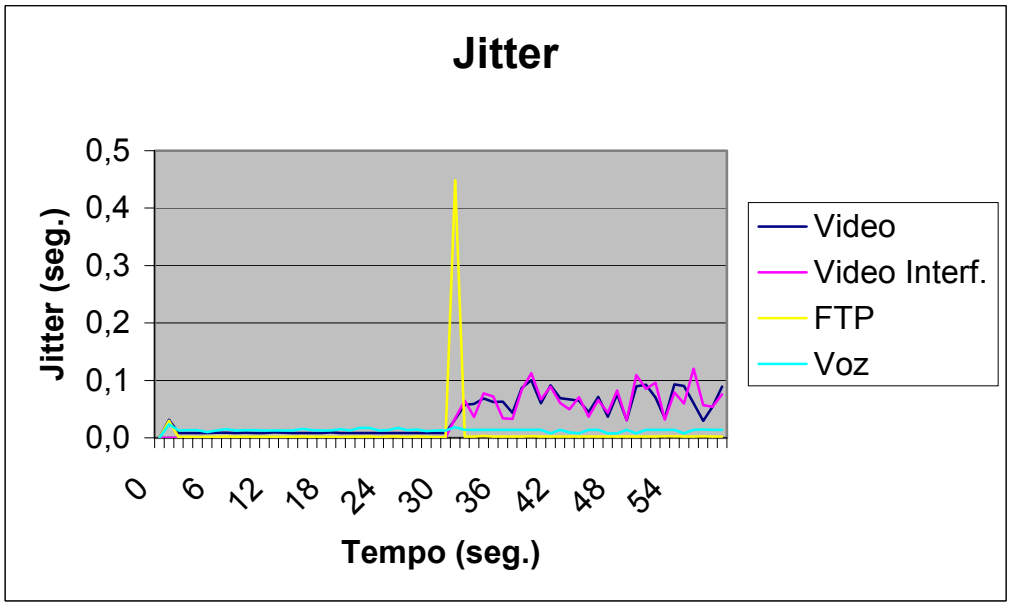


Figura 35: Comportamento do *Jitter* para o Cenário 5 - Fila com Prioridade, sem Congestionamento, com Tráfego Interferente e com Limitação na Fila de Vídeo.

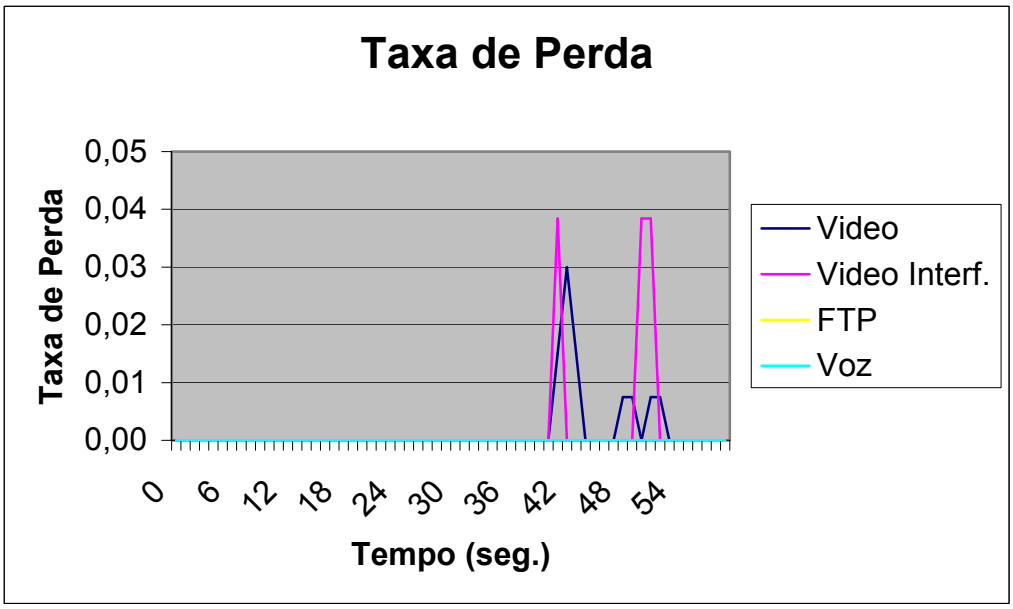


Figura 36: Comportamento da Taxa de Perda de pacotes para o Cenário 5 - Fila com Prioridade, sem Congestionamento, com Tráfego Interferente e com Limitação na Fila de Vídeo.

As figuras 34 a 36 mostram que, apesar de possuírem alta prioridade, apenas as aplicações de vídeo, original e interferente, foram penalizadas quando a aplicação de vídeo interferente foi introduzida no sistema, já que os valores de *jitter*, *delay* e taxa de perda de pacotes aumentaram consideravelmente. As aplicações de voz e FTP não sofreram alterações nos valores dos parâmetros de *throughput*, *delay*, *jitter* e taxa de perda de pacotes.

Apenas o comportamento do valor de *throughput* não ficou dentro do esperado, pois, de acordo com a figura 33, a somatória das duas fontes de tráfego de vídeo está em torno de 1.100 kbps, sendo que a limitação na fila é de 682 kbps. Para uma melhor compreensão sobre o comportamento específico desse parâmetro, foi realizado um teste adicional para esse cenário, sendo configurada uma situação de congestionamento antes da entrada do tráfego interferente. O novo comportamento do *throughput* está descrito na figura 37.

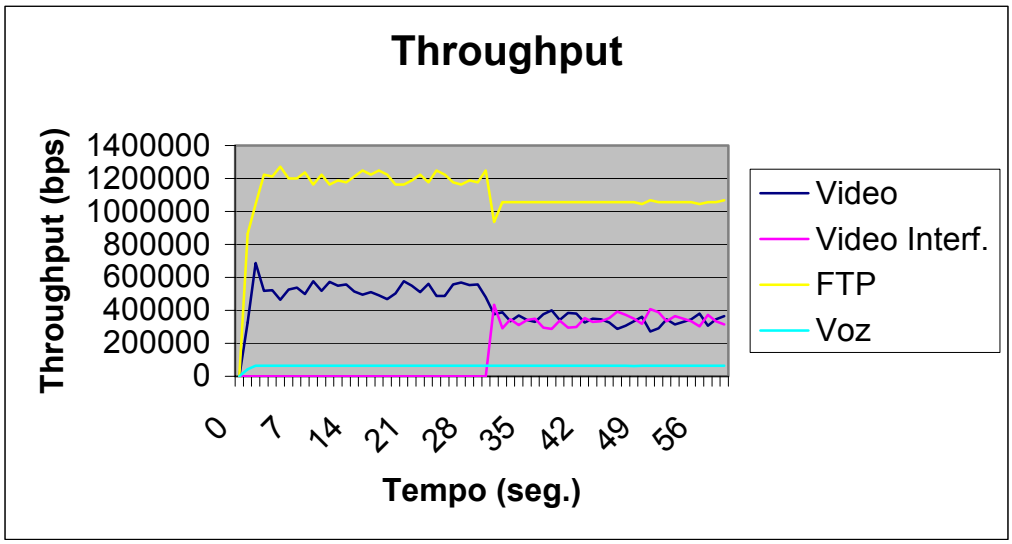


Figura 37: Comportamento do *Throughput* para o cenário do item 4.2.2.5 com congestionamento antes da entrada de tráfego interferente.

Nesse novo ambiente, a somatória do tráfego das quatro fontes é de 2.364 kbps e a banda passante do link entre os equipamentos *core* e *edge* 2 foi configurada em 1.800 kbps. Nessa nova situação, o *throughput* da aplicação de voz permanece inalterado antes e depois de entrar o tráfego interferente.

A aplicação de dados FTP sofreu uma diminuição do seu *throughput*, em virtude da utilização por essa aplicação da diferença de banda entre o limite da fila de vídeo, que é igual a 682 kbps, e o *throughput* médio da aplicação de vídeo original, antes da entrada do vídeo interferente, que é inferior a 682 kbps.

Quando a aplicação de vídeo interferente foi introduzida no ambiente, o *throughput* médio da aplicação de vídeo original baixou, de forma que a somatória dos *throughputs* das duas fontes de vídeo está bem próxima de 682 kbps, que é o limite da banda passante da fila de vídeo.

Pelos resultados práticos extraídos dessa simulação, conclui-se que quando existir banda suficiente para acomodar todas as aplicações ou não existir congestionamento na rede, o valor limite de banda passante em uma fila não é respeitado. Quando não há banda passante suficiente para todas as aplicações, ou na existência de congestionamento na rede, o limite da fila é respeitado.

Entretanto, isso parece ser uma característica, ou defeito, do simulador pois, segundo alguns fabricantes de equipamentos de comunicação de dados (Nortel Networks e Cisco) esse comportamento não existe nos seus equipamentos que provêm qualidade de serviço. Este comportamento também foi descrito na lista de usuários do simulador NS-2. Entretanto, nenhum comentário foi feito sobre essa característica do simulador até o momento da escrita deste trabalho.

4.2.3 - Conclusões Sobre os Resultados do Testes

De acordo com os resultados obtidos no cenário do item 4.2.2.1, algoritmos de enfileiramento de pacotes que não priorizam o envio de determinados tipos de tráfego não servem para entregar diferentes níveis de QoS para as aplicações já que, em situações onde a rede apresenta congestionamento, os valores de *throughput*, *delay*, *jitter* e taxa de perda de pacotes são penalizados igualmente para todas as aplicações.

Pelo cenário do item 4.2.2.2, pôde ser comprovado que quando a rede não passa por situações de congestionamento, todas as aplicações conseguem receber da rede o nível de QoS desejado. Esse resultado tenderia a se repetir mesmo se pudesse ser implementado no simulador o protocolo RSVP-TE. Portanto, quando a rede não está passando por congestionamento, independentemente da arquitetura utilizada (DiffServ ou IntServ), todas as aplicações conseguem receber o nível de QoS desejado.

O problema começa a ocorrer quando a rede passa por congestionamentos ou quando tráfegos oriundos de rotas que se tornaram inoperantes são direcionados para o circuito que permaneceu ativo. Nesse caso, a aplicação de menor prioridade começa a sofrer degradação do nível de qualidade de serviço requisitado, sendo essa degradação tanto maior quanto maior for o congestionamento ou o tráfego reroteado. Essa situação pôde ser verificada no cenário 4.2.2.3, onde a aplicação de menor prioridade (FTP) é fortemente penalizada se comparada ao mesmo tipo de tráfego do cenário 4.2.2.2.

No cenário 4.2.2.4, pôde ser observada a entrada do tráfego interferente, ou reroteado, que na tecnologia DiffServ é de difícil controle. Nesse caso, o tráfego de alta prioridade (vídeo) foi reroteado e a rede, novamente, penalizou a aplicação de menor prioridade (FTP), pois os recursos utilizados por ele foram transferidos para a nova aplicação de vídeo. Portanto, um fluxo de dados que já estava estabelecido na rede e que vinha recebendo o nível de QoS requisitado, passou a ser penalizado com violações do SLA contratado.

Nesse ambiente, uma forma de se controlar a quantidade de tráfego reroteado foi implementar um valor máximo de banda que uma determinada fila pode suportar, como o configurado no cenário 4.2.2.5. Dessa forma, a rede, como um todo, fica protegida contra excessos de tráfegos reroteados. Nesse cenário, a aplicação de menor prioridade (FTP) não foi penalizada quando a de vídeo interferente foi introduzida na rede. Porém, a aplicação de vídeo, originalmente estabelecida, assim como a de vídeo interferente, ficaram com seus níveis de qualidade de serviço, ou SLA contratados, comprometidos.

Supondo que nesse ambiente o protocolo RSVP-TE pudesse ser configurado, antes que o tráfego interferente fosse reroteado, ou o circuito virtual fosse estabelecido na rede, o mecanismo de CAC seria consultado para verificar se existiriam recursos disponíveis para garantir o SLA contratado desse novo fluxo. Se houvesse recursos disponíveis, o circuito seria estabelecido, caso contrário o novo

fluxo de dados não seria estabelecido na rede e, portanto, não penalizaria os fluxos de dados já estabelecidos na rede.

Capítulo 5 – Conclusão e Trabalhos Futuros

5.1 – Conclusão Final

Este trabalho abordou o tema da necessidade de haver qualidade de serviço nas atuais redes IP. Através da pesquisa bibliográfica realizada, concluiu-se que as aplicações IP emergentes necessitam receber níveis de qualidade de serviço diferenciados da rede, pois o tradicional nível de QoS *Best Effort*, igual para todas as aplicações, não as atende mais.

Ainda de acordo com a pesquisa bibliográfica efetuada, mostrou-se que existem várias arquiteturas e protocolos de rede disponíveis no mercado, como por exemplo: ATM, DiffServ, MPLS, RSVP-TE, etc, que procuram garantir qualidade de serviço fim a fim para as aplicações. Entretanto, todas elas possuem determinadas fragilidades que acabam por comprometer o nível de QoS fim a fim solicitado por uma aplicação.

Sem dúvida, a melhor arquitetura para alcançar esse objetivo era o ATM, pois, além de ser baseada em circuitos virtuais, possui diversos mecanismos importantes como: reserva de recursos, controle de admissão de novas chamadas e policiamento de tráfego. Porém, em virtude das dificuldades operacionais e do alto custo financeiro, o ATM acabou sendo relegado a aplicações muito específicas.

Entretanto, este trabalho mostra que o RSVP-TE, cuja primeira versão já disponível no mercado desde 1997, é um protocolo de rede IP também baseado em circuitos virtuais e com mecanismos semelhantes aos do ATM.

Por possuir essas características fundamentais, o RSVP-TE foi escolhido para ter suas características estendidas, contornando duas deficiências que impediram sua utilização em larga escala, as quais são: necessidade de haver um *software* cliente rodando nos *hosts* e subjetividade na definição dos níveis de classe de serviço, em virtude da forma qualitativa que está definida na sua padronização.

Para contornar essas fragilidades foram propostas duas novas funcionalidades, sendo que a primeira atua nos roteadores de borda da rede, fazendo um mapeamento entre a tecnologia Diffserv da rede local do cliente e o RSVP-TE do *backbone*. A segunda proposta atua nos roteadores de *backbone* da rede, direcionando o tráfego dos LSPs para filas de um algoritmo de enfileiramento de pacotes, de acordo com sua prioridade.

Para verificar o comportamento da funcionalidade proposta aos roteadores de *backbone*, foram realizados testes no simulador de rede NS-2. Através dos resultados alcançados, verificou-se que algoritmos de enfileiramento de pacotes do tipo FIFO, que implementam o nível de QoS *Best Effort*, não servem para oferecer qualidade de serviço às aplicações, pois em situações de congestionamento todas as aplicações são penalizadas.

Em situações de congestionamento, os algoritmos de enfileiramento que possuem priorização de tráfego penalizam as aplicações com nível de qualidade de serviço mais baixo. Dependendo do nível de congestionamento que a rede passar, essas aplicações podem não sofrer degradação do SLA contratado, pois, tipicamente, possuem uma boa tolerância a variações dos parâmetros que medem o nível de QoS.

Também se observou que a entrada de tráfegos interferentes no sistema, por exemplo oriundos de reroteamento de circuitos que se tornaram inoperantes, são de difícil controle na tecnologia DiffServ. De acordo com os resultados obtidos no item 4.2.2.5, a aplicação de vídeo que já estava estabelecida, mesmo sendo de mais alta prioridade, sofreu degradação de seu nível de qualidade de serviço quando o tráfego interferente foi adicionado ao ambiente.

Ainda com relação a tecnologia DiffServ, o fato de haver limitação do tráfego por nível de qualidade de serviço na entrada da rede não impede que as aplicações já estabelecidas sofram degradação do seu nível de QoS, conforme mostram os resultados do item 4.2.2.5.

Em um ambiente com o protocolo RSVP-TE configurado esse problema não aconteceria, pois se não houvesse recursos disponíveis na rede o circuito virtual referente ao tráfego interferente não seria estabelecido na rede. Dessa forma, a aplicação de vídeo, que já estava estabelecida na rede, continuaria com seu nível de QoS sendo mantido.

Portanto, este trabalho, através de propostas práticas e com mecanismos disponíveis no mercado, fornece uma outra alternativa para alcançar a manutenção fim a fim do nível de qualidade de serviço em redes IP, mesmo em situações de congestionamento e contingência, quando normalmente se forma um gargalo em algum ponto da rede.

5.2 – Sugestões para Continuidade da Pesquisa

No decorrer deste trabalho, verificou-se a necessidade de aprofundar os estudos e experimentos nas seguintes áreas:

5.2.1 – Suporte do RSVP-TE a Aplicações *Multicast*

A padronização do RSVP-TE (Awduche et al., 2001) não suporta aplicações *Multicast*. Como a utilização do protocolo IP para aplicações de vídeo e imagem vem aumentando, como por exemplo IP TV, será necessário que esse protocolo passe a suportar esse tipo de aplicação, visando sua maior aplicabilidade. Dessa forma, essas aplicações se beneficiariam das vantagens de se manter qualidade de serviço fim a fim na rede, agregando mais confiabilidade à aplicação.

5.2.2 – Escalabilidade dos LSPs

As redes muito grandes, onde o protocolo RSVP-TE esteja instalado, podem apresentar problemas de escalabilidade, já que cada cliente, ou fonte de tráfego do cliente, representa um circuito virtual, ou LSP. Portanto, se faz necessário realizar um estudo mais aprofundado para identificar qual o limite de circuitos virtuais que uma dada rede suportaria. Identificado esse limite, deverão ser desenvolvidos meios de otimizar o número de circuitos virtuais estabelecidos entre os equipamentos, como por exemplo: agregar o tráfego de vários LSPs em apenas um, diminuindo a troca de mensagens. Em relação a esse ponto, um estudo detalhado da arquitetura MPLS é muito importante.

5.2.3 – Medições

Embora os resultados obtidos nos testes práticos, com as ferramentas utilizadas, tenham sido satisfatórios, determinadas situações do comportamento das fontes de tráfego, que simulam as aplicações, não puderam ser avaliadas por limitações do simulador utilizado. Neste trabalho, ficou evidenciado que o algoritmo de enfileiramento de pacotes é um importante componente para se manter o nível de qualidade de serviço solicitado pela aplicação.

Entretanto, apenas priorizar tráfego não significa que o nível de QoS seja mantido. Portanto, deverão ser realizados outros testes, com outros algoritmos de enfileiramento que, além de priorizar o tráfego, faça reserva de *buffer* para uma determinada fila de envio de pacotes.

Adicionalmente, para poder alcançar resultados mais específicos nos testes das novas funcionalidades propostas, o simulador de rede deverá possuir módulos adicionais que suportem o protocolo RSVP-TE com os mecanismos que façam reservas de recurso e policiamento de tráfego.

Referências Bibliográficas

Andersson, L. et al. **LDP Specification**, IETF RFC 3036, January, 2001

Andersson, L.; Swallow, G. **The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols**, IETF RFC 3468, February 2003

AQUILA, **Project Home Page**, January 01, 2000. Disponível em: www.ist-aquila.org/; Acesso: 18.08.2003

ATM Forum, Technical Committee, **Private Network-Network Interface Specification Version 1.1 (PNNI 1.1), af-pnni-0055.002**, April 2002

ATM Forum, Technical Committee, **Traffic Management Specification Version 4.1, AF-TM-0121.000**, March 1999

Awduche, D et al. **RSVP-TE: Extensions to RSVP for LSP Tunnels**, IETF RFC 3209, December 2001

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

Barra, C. R. **Controle Dinâmico de Qualidade de Serviço em Redes IP**, 2003, 45 p. Exame de Qualificação (Doutorado em Engenharia Elétrica – Sistemas Eletrônicos), Escola Politécnica, Universidade de São Paulo

Bernet, Y. **The Complementary Roles of RSVP and Differentiated Services in the Full-Service QoS Network,** IEEE Communications Magazine, February 2000

Black, D et al. **An Architecture for Differentiated Services**, IETF RFC 2475, December 1998

Black, U. **IP Routing Protocols**, 2000

Braden, R. et al. **Resource Reservation Protocol (RSVP) Version 1 Functional Specification**, IETF RFC 2205, September 1997

Califórnia, Universidade da **The Network Simulator – NS-2**, s.d. Disponível em: http://www.isi.edu/nsnam/ns/. Acesso: 15.08.2003

Cisco Systems Inc. **Explorando as Tecnologias VPN-IP para sua Rede Corporativa**, 2001, Seminário

Clark, D. et al. **Integrated Services in the Internet Architecture: an Overview**, IETF RFC 1633, June 1994

Crawley, E. et al. **A Framework for QoS-based Routing in the Internet**, IETF RFC 2386, August 1998

Dovrolis, C.; Ramanathan, P. **A Case for Relative Differentiated Services and the Proportional Differentiation Model,** IEEE Network, September/October 1999

Engel, T. et al. **AQUILA: Adaptive Resource Control for QoS Using an IP-Based Layered Architecture,** IEEE Communications Magazine, January 2003

Fall, K.; Varadhan, K. **The ns Manual**, The VINT Project A collaboration between researchers at UC Berkeley, LBL, USC/ISI and Xerox PARC, November 2003

Fineberg, V. **A Practical Architecture for Implementing End-to-End QoS in an IP Network,** IEEE Communications Magazine, January 2002

Gallaher, R. **Advanced MPLS Signalling,** Converge Network Digest, December 10, 2001. Disponível em: www.convergedigest.com/tutorials/mpls3/page1.html; Acesso: 18.08.2003

Gozdecki, J.; Jajszczyk, A.; Stankiewicz, R. **Quality of Service Terminology in IP Networks,** IEEE Communications Magazine, March 2003

Harnedy, S. **An Introduction to Multiprotocol Label Switching**, Ed. Prentice Hall, 2001

Heinanen, J. et al. **Assured Forwarding PHB Group**, IETF RFC 2597, June 1999

ITU-T (antiga CCITT) Recommendation Y.1241 **Support of IP-Based Services Using IP Transfer Capabilities**, March 2001

Jacobson V.; Nichols K.; Poduri, K. **An Expedited Forwarding PHB**, IETF RFC 2598, June 1999

Jamoussi, B et al. **Constraint-Based LSP Setup using LDP**, IETF RFC 3212, January 2002

Juniper Networks Inc. **RSVP Signaling Extensions for MPLS Traffic Engineering**, 2002, White Paper

Katz, D.; Yeung, D.; Kompella, K. **Traffic Engineering Extensions to OSPF,** IETF Internet Draft, August 2003

Melo, E. T. L. **Qualidade de Serviço em Redes IP com DiffServ: Avaliação através de Medições**, 2001 Dissertação (Mestrado em Ciência da Computação), Universidade Federal de Santa Catarina

Mcdysan, D. **ATM Theory and Application**, 1999

Mykoniati, E. et al. **Admission Control for Providing QoS in DiffServ IP Networks: The TEQUILA Approach,** IEEE Communications Magazine, January 2003

Pieda, P et al. **A Network Simulator Differentiated Services Implementation**, Nortel Networks, July 2000

Reed, F. **M&A Deskbook**, Mc Graw-Hill, 2001

Roth, R. et al. **IP QoS Across Multiple Management Domains: Practical Experiences from Pan-European Experiments,** IEEE Communications Magazine, January 2003

Rosen, E. et al**. Multiprotocol Label Switching Architecture**, IETF RFC 3031, January, 2001

TEQUILA **Project Home Page**, January 01, 2000. Disponível em: www.ist-tequila.org/; Acesso: 18.08.2003

Thomas, Stephen A**. IP Switching and Routing Essentials**, 2001

Wroclawski, J. **Specification of the Controlled-Load Network Service**, IETF RFC 2211, September 1997a

Wroclawski, J. **Specification of the Guaranteed Service**, IETF RFC 2212, September 1997b

Wroclawski, J. **The Use of RSVP with IETF Integrated Services**, IETF RFC 2210, September 1997c

Anexo A – *Scripts* de Simulação

Esse anexo mostra o código fonte escrito em linguagem Otcl e que foi utilizado no simulador NS-2 para testar o cenário do item 4.2.2.5. Para os outros cenários foram utilizados variações desse código.

```
#   -------
#   |video|-----------
#   -------   10 Mb   \
#              5 ms    \
#                       \
#   -------   10 Mb      \----           ------          ----
------
#   |Voice|-------------- |e1|-----------|core|----------|e2|-------
----|dest|
#   -------    5 ms     / ----   10 Mb   ------   X Mb   ----   10
Mb   ------
#                       /         5 ms            5 ms           5 ms
#                      /
#    -----            /
#    |FTP|-----------
#    -----   10 Mb
#             5 ms
#
#-------------------------------------------------------------------
---------------

#Create a simulator object
set ns [new Simulator]

# Paramtros de Police
set cirvd  1000000
set cirvdif 1000000
set cirvc  1000000
set cirft  1000000

set packetSize 1000

set interval 1.0

#Open the output files
set flstvd [open vdlst.out w]
set fthrvd [open vdthr.out w]
set fdlyvd [open vddly.out w]
set fjttvd [open vdjtt.out w]
set fallvd [open vdall.out w]

set flstvdif [open vdiflst.out w]
set fthrvdif [open vdifthr.out w]
set fdlyvdif [open vdifdly.out w]
set fjttvdif [open vdifjtt.out w]
set fallvdif [open vdifall.out w]

set flstvc [open vclst.out w]
set fthrvc [open vcthr.out w]
```

```
set fdlyvc [open vcdly.out w]
set fjttvc [open vcjtt.out w]
set fallvc [open vcall.out w]

set flstft [open ftlst.out w]
set fthrft [open ftthr.out w]
set fdlyft [open ftdly.out w]
set fjttft [open ftjtt.out w]
set fallft [open ftall.out w]

#Open the nam trace file
#set fnam [open out.nam w]
#$ns namtrace-all $fnam

#Open the trace file
#set ftr [open out.tr w]
#$ns trace-all $ftr

# Set up the network topology shown at the top of this file:
set Vd [$ns node]
set Vdif [$ns node]
set Vc [$ns node]
set Ft [$ns node]
set e1 [$ns node]
set core [$ns node]
set e2 [$ns node]
set dest [$ns node]

# Set some traffic rates
#set vdinterval .005

# Pareto - Video
set vdpktsize 480
set vdrate 512000
set vdbt .3
set vdit .1

# Pareto - Video Interferente
set vdifpktsize 480
set vdifrate 300000
set vdifbt .3
set vdifit .1

# Voice - RTP
set vcinterval .01125
set vcpktsize 90

# FTP - RTP
set ftinterval .01
set ftpktsize 1500

$ns duplex-link $Vd $e1 10Mb 5ms DropTail
$ns duplex-link $Vdif $e1 10Mb 5ms DropTail
$ns duplex-link $Vc $e1 10Mb 5ms DropTail
$ns duplex-link $Ft $e1 10Mb 5ms DropTail

$ns simplex-link $e1 $core 10Mb 5ms dsRED/edge
$ns simplex-link $core $e1 10Mb 5ms dsRED/core
```

```
$ns simplex-link $core $e2 2.2Mb 5ms dsRED/core
$ns simplex-link $e2 $core 2.2Mb 5ms dsRED/edge

$ns duplex-link $e2 $dest 10Mb 5ms DropTail


$ns duplex-link-op $Vdif $e1 orient down-right
$ns duplex-link-op $Vd $e1 orient down-right
$ns duplex-link-op $Vc $e1 orient right
$ns duplex-link-op $Ft $e1 orient up-right
$ns duplex-link-op $e1 $core orient right
$ns duplex-link-op $core $e2 orient right
$ns duplex-link-op $e2 $dest orient right


set qE1C [[$ns link $e1 $core] queue]
set qCE2 [[$ns link $core $e2] queue]


# Set DS RED parameters from Edge1 to Core:
$qE1C meanPktSize $packetSize
$qE1C set numQueues_ 4
$qE1C setNumPrec 1

$qE1C addPolicyEntry [$Vd id] [$dest id] TSW2CM 10 $cirvd
$qE1C addPolicyEntry [$Vdif id] [$dest id] TSW2CM 10 $cirvdif
$qE1C addPolicyEntry [$Vc id] [$dest id] TSW2CM 20 $cirvc
$qE1C addPolicyEntry [$Ft id] [$dest id] TSW2CM 30 $cirft

$qE1C addPolicerEntry TSW2CM 10 11
$qE1C addPolicerEntry TSW2CM 20 21
$qE1C addPolicerEntry TSW2CM 30 31

$qE1C addPHBEntry 10 0 0
$qE1C addPHBEntry 11 3 0
$qE1C addPHBEntry 20 1 0
$qE1C addPHBEntry 21 3 0
$qE1C addPHBEntry 30 2 0
$qE1C addPHBEntry 31 3 0
$qE1C configQ 0 0 20 40 0.02
$qE1C configQ 1 0 10 20 0.10
$qE1C configQ 2 0 30 40 0.25
$qE1C configQ 3 0 50 60 0.50

# Set DS RED parameters from Core to Edge2:
$qCE2 setSchedularMode PRI
#$qCE2 addQueueRate 0 682000
$qCE2 meanPktSize $packetSize
$qCE2 set numQueues_ 4
$qCE2 setNumPrec 1
$qCE2 addPHBEntry 10 0 0
$qCE2 addPHBEntry 20 1 0
$qCE2 addPHBEntry 30 2 0
$qCE2 addPHBEntry 11 3 0
$qCE2 addPHBEntry 21 3 0
$qCE2 addPHBEntry 31 3 0
$qCE2 configQ 0 0 20 40 0.02
$qCE2 configQ 1 0 10 20 0.10
$qCE2 configQ 2 0 30 40 0.25
```

```
$qCE2 configQ 3 0 50 60 0.50

#Define a 'finish' procedure
proc finish {} {
      global ns ftr fnam flstvd fthrvd fdlyvd fjttvd fallvd flstvc
fthrvc fdlyvc fjttvc fallvc flstft fthrft fdlyft fjttft fallft
flstvdif fthrvdif fdlyvdif fjttvdif fallvdif
      $ns flush-trace
      #Close the trace files
      #close $fnam
      #close $ftr
      close $flstvd
      close $fthrvd
      close $fdlyvd
      close $fjttvd
      close $fallvd
      close $flstvdif
      close $fthrvdif
      close $fdlyvdif
      close $fjttvdif
      close $fallvdif
      close $flstvc
      close $fthrvc
      close $fdlyvc
      close $fjttvc
      close $fallvc
      close $flstft
      close $fthrft
      close $fdlyft
      close $fjttft
      close $fallft
      #Call xgraph to display the results
      exec xgraph vdlst.out vclst.out ftlst.out vdiflst.out -M -nl -
bar -brb 0 -bof 0 -brw .001 -geometry 800x400 &
      #exec xgraph ftdly.out -geometry 800x400 &
      exec xgraph vddly.out vcdly.out ftdly.out vdifdly.out -
geometry 800x400 &
      #exec xgraph ftthr.out -geometry 800x400 &
      exec xgraph vdthr.out vcthr.out ftthr.out vdifthr.out -
geometry 800x400 &
      exec xgraph vdjtt.out vcjtt.out ftjtt.out vdifjtt.out -
geometry 800x400 &
      #Execute nam on the trace file
        #exec nam out.nam &
      exit 0
}

#Define a procedure which periodically records QoS
proc record {} {
      global sinkvd sinkvc sinkft flstvd fthrvd fdlyvd fjttvd fallvd
flstvc fthrvc fdlyvc fjttvc fallvc flstft fthrft fdlyft fjttft
fallft interval vdinterval vdpktsize vdrate vcinterval vcpktsize
ftinterval ftpktsize sinkvdif flstvdif fthrvdif fdlyvdif fjttvdif
fallvdif fallvdif vdifinterval vdifpktsize vdifrate
      #Get an instance of the simulator
      set ns [Simulator instance]
      #Set the time after which the procedure should be called again
      set time $interval
```

```
      #Video QoS parameters
      #How many packets have been lost by the traffic sinks?
      set nlost [$sinkvd set nlost_]
      #set lost [expr [expr $nlost*$vdpktsize*8/$interval]/[expr
$vdpktsize*8/$vdinterval]]
      set lost [expr [expr $nlost*$vdpktsize*8/$interval]/$vdrate]
      #How many bits have been received by traffic sinks?
      set bytes [$sinkvd set bytes_]
      #set interval $qos_rate
      set thru [expr $bytes*8/$interval]
      #What's the end-to-end delay?
      set delay [$sinkvd set delay_]
      #What's the end-to-end jitter?
      set jitter [$sinkvd set jitter_]
      #Get the current time
      set now [$ns now]
      #Write results to the files
      puts $flstvd "$now $lost"
      puts $fthrvd "$now $thru"
      puts $fdlyvd "$now $delay"
      puts $fjttvd "$now $jitter"
      puts $fallvd "$now $lost $thru $delay $jitter"
      #Reset the values on the traffic sinks
      $sinkvd clear

      #Video Interferent QoS parameters
      #How many packets have been lost by the traffic sinks?
      set nlost [$sinkvdif set nlost_]
      set lost [expr [expr
$nlost*$vdifpktsize*8/$interval]/$vdifrate]
      #How many bits have been received by traffic sinks?
      set bytes [$sinkvdif set bytes_]
      set thru [expr $bytes*8/$interval]
      #What's the end-to-end delay?
      set delay [$sinkvdif set delay_]
      #What's the end-to-end jitter?
      set jitter [$sinkvdif set jitter_]
      #Get the current time
      set now [$ns now]
      #Write results to the files
      puts $flstvdif "$now $lost"
      puts $fthrvdif "$now $thru"
      puts $fdlyvdif "$now $delay"
      puts $fjttvdif "$now $jitter"
      puts $fallvdif "$now $lost $thru $delay $jitter"
      #Reset the values on the traffic sinks
      $sinkvdif clear

      #Voice QoS parameters
      #How many packets have been lost by the traffic sink?
        set nlost [$sinkvc set nlost_]
        set lost [expr [expr $nlost*$vcpktsize*8/$interval]/[expr
$vcpktsize*8/$vcinterval]]
      #How many bits have been received by traffic sinks?
      set bytes [$sinkvc set bytes_]
      #set interval $qos_rate
      set thru [expr $bytes*8/$interval]
      #What's the end-to-end delay?
      set delay [$sinkvc set delay_]
```

```
      #What's the end-to-end jitter?
      set jitter [$sinkvc set jitter_]
      #Get the current time
      set now [$ns now]
      #Write results to the files
      puts $flstvc "$now $lost"
      puts $fthrvc "$now $thru"
      puts $fdlyvc "$now $delay"
      puts $fjttvc "$now $jitter"
      puts $fallvc "$now $lost $thru $delay $jitter"
      #Reset the values on the traffic sinks
      $sinkvc clear

      #FTP QoS parameters
      #How many packets have been lost by the traffic sinks?
        set nlost [$sinkft set nlost_]
        set lost [expr [expr $nlost*$ftpktsize*8/$interval]/[expr
$ftpktsize*8/$ftinterval]]
      #How many bits have been received by traffic sinks?
      set bytes [$sinkft set bytes_]
      #set interval $qos_rate
      set thru [expr $bytes*8/$interval]
      #What's the end-to-end delay?
      set delay [$sinkft set delay_]
      #What's the end-to-end jitter?
      set jitter [$sinkft set jitter_]
      #Get the current time
      set now [$ns now]
      #Write results to the files
      puts $flstft "$now $lost"
      puts $fthrft "$now $thru"
      puts $fdlyft "$now $delay"
      puts $fjttft "$now $jitter"
      puts $fallft "$now $lost $thru $delay $jitter"
      #Reset the values on the traffic sinks
      $sinkft clear


      #Re-schedule the procedure
      $ns at [expr $now+$time] "record"
}

#Create one traffic sink and attach them to the node dest
set sinkvd [new Agent/QoSMonitor]
$ns attach-agent $dest $sinkvd

set sinkvdif [new Agent/QoSMonitor]
$ns attach-agent $dest $sinkvdif

set sinkvc [new Agent/QoSMonitor]
$ns attach-agent $dest $sinkvc

set sinkft [new Agent/QoSMonitor]
$ns attach-agent $dest $sinkft

#Create an UDP Agent and attach it to node Vd
set udpvd [new Agent/UDP]
$ns attach-agent $Vd $udpvd
```

```
#Create an UDP Agent and attach it to node Vdif
set udpvdif [new Agent/UDP]
$ns attach-agent $Vdif $udpvdif

#Create Video traffic source - Pareto
set srcVd [new Application/Traffic/Pareto]
$srcVd set packetSize_ $vdpktsize
$srcVd set burst_time_ $vdbt
$srcVd set idle_time_ $vdit
$srcVd set rate_ [expr $vdrate*($vdbt+$vdit)/$vdbt]
$srcVd set shape_ 4.0
$srcVd attach-agent $udpvd
$ns connect $udpvd $sinkvd

#Create Video Interferent traffic source - Pareto
set srcVdif [new Application/Traffic/Pareto]
$srcVdif set packetSize_ $vdpktsize
$srcVdif set burst_time_ $vdbt
$srcVdif set idle_time_ $vdit
$srcVdif set rate_ [expr $vdrate*($vdbt+$vdit)/$vdbt]
$srcVdif set shape_ 4.0
$srcVdif attach-agent $udpvdif
$ns connect $udpvdif $sinkvdif
#In practical tests the video throughput was 550 kbps

#Create Voice traffic source
set srcVc [new Agent/RTP]
$srcVc set interval_ $vcinterval
$srcVc set packetSize_ $vcpktsize
$ns attach-agent $Vc $srcVc
$ns connect $srcVc $sinkvc

#Create FTP traffic source
set srcFt [new Agent/RTP]
$srcFt set interval_ $ftinterval
$srcFt set packetSize_ $ftpktsize
$ns attach-agent $Ft $srcFt
$ns connect $srcFt $sinkft

#Start
$ns at 0.0 "record"
#Start the trafoutic sources
$ns at 0.2 "$srcFt start"
$ns at 0.3 "$srcVc start"
$ns at 0.4 "$srcVd start"
$ns at 30.0 "$srcVdif start"

#Stop the traffic sources
#$ns at 2.0 "$srcVc stop"
#$ns at 3.0 "$srcFt stop"
#$ns at 4.1 "$srcVc start"
#$ns at 5.0 "$srcFt start"

#Call the finish procedure after 1 second simulation time
$ns at 60.0 "finish"

#Run the simulation
        $ns run
```