LILIAN BARBARA BENDER PORTUGAL

MODELOS E MECANISMOS DE QUALIDADE DE SERVIÇO EM REDES IP

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Curso de Pós-Graduação em Telecomunicações, Setor de Tecnologia, Departamento de Engenharia Elétrica, Universidade Federal do Paraná.

Orientador: Prof. Eduardo Parente Ribeiro

Co-orientador: Prof. Carlos Marcelo Pedroso

CURITIBA

2002

UFPR

UNIVERSIDADE FEDERAL DO PARANÁ
Programa de Pós-Graduação em Engenharia Elétrica
Área de Concentração Telecomunicações
Setor de Tecnologia

RELATÓRIO DE DEFESA DE MESTRADO

Aos dois dias do mês de julho de 2002, no Laboratório didático do Departamento de Engenharia Elétrica - DELT, foi instalada pelo Prof. José Ricardo Descardeci, coordenador do Programa de Pós-Graduação em Engenharia Elétrica, a Banca Examinadora para a quinta Dissertação de Mestrado área de concentração: TELECOMUNICAÇÕES . Estiveram presentes no Ato, além do coordenador do Curso de Pós-Graduação, professores, alunos e visitantes.

A Banca Examinadora, atendendo determinação do Colegiado do Programa de Pós-Graduação em Engenharia Elétrica, ficou constituída pelos professores doutores **Eduardo Parente Ribeiro** (UFPR), **José Ricardo Descardeci** (UFPR), **Keiko Verônica Ono Fonseca**(CEFET-PR).

Às 14:00 horas, a banca iniciou os trabalhos, convidando o(a) candidato(a) **Lilian B. B. Portugal** a fazer a apresentação da dissertação intitulada " Modelos e Mecanismos de qualidade de serviço em Redes IP ". Encerrada a apresentação, iniciou-se a fase de argüição pelos membros participantes.

Tendo em vista a dissertação e a argüição, a banca atribuiu as seguintes notas: Prof. Dr. José Ricardo Descardeci Nota: 100, Profa. Dra. Keiko Verônica Ono Fonseca , Nota: 80. Prof. Dr. Eduardo Parente Ribeiro Nota: 70. A média obtida: B, resulta na APROVAÇÃO do candidato, (de acordo com a determinação dos Artigos 61,62,63,64 da Resolução 38/96 de 14.06.96), e corresponde ao conceito A/B/C/D.

Curitiba, 02 de julho de 2002.

Keiko V. O. Fonseca
Profa.Dra.Keiko Verônica Ono Fonseca

Prof. Dr. José Ricardo Descardeci

Prof. Dr. Eduardo Parente Ribeiro

UNIVERSIDADE FEDERAL DO PARANÁ SETOR DE TECNOLOGIA PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

AGRADECIMENTOS

Ao Beto, Bruna e Paula, por suportarem as intermináveis horas de abondono do convívio do lar e do cumprimento das obrigações de esposa e mãe.

Ao professor Descardeci pela oportunidade de realizar este mestrado e ao colega Márcio Protzek pelo apoio nos momentos difíceis.

Ao professor Marcelo Pedroso pelo grande suporte na discussão de idéias, pelas informações prestadas e que, sem dúvida, ajudaram em muito o enriquecimento desta dissertação.

Finalmente, ao Professor Eduardo Parente Ribeiro pela orientação e dedicação para a realização deste trabalho.

SUMÁRIO

1. INTRODUÇÃO *1*

1.1. Motivação **1**

1.2. Objetivo **2**

1.3. Organização da Dissertação **2**

2. CONCEITO E MÉTRICAS DE QoS *4*

2.1. Conceitos Preliminares **4**

2.1.1. Modos de Conexão 4

2.1.2. Elementos de Rede 5

2.1.3. Formato do Datagrama IP 6

2.2. Conceitos sobre QoS **7**

2.2.1. Grau de Serviço (GoS) 8

2.2.1.1. Perda do Grau de Serviço 8

2.2.1.2. Atraso no Grau de Serviço 9

2.2.2. Classe de Serviço 9

2.2.3. Tipo de Serviço 9

2.2.4. Contrato 11

2.3. Pontos de Vista sobre QoS **11**

2.3.1. Sob o ponto de vista do Fluxo de Tráfego 12

2.3.2. Sob o Ponto de Vista de Sub-redes 13

2.3.3. Sob o ponto de Vista de Negociações de Parâmetros 13

2.3.4. Sob o Ponto de Vista de Camadas 14

2.3.4.1. Camada de Transporte 14

2.3.4.2. Camada de Rede 15

2.4. Métricas **16**

2.4.1. Métricas IP 16

2.4.2. Causa de Atrasos 18

2.4.3. Causa das Perdas 20

2.5. Aplicações **21**

3. Modelos de QoS *22*

3.1. Serviços Integrados - IntServ **23**

3.1.1. Classificador (*Classifier*) 24

3.1.2. Caracterização do tráfego 24

3.1.3. Escalonamento de Pacotes (*Packet Scheduler*) 27

3.1.3.1. Escalonamento Priorizado (*Priority Queuing*) 28

3.1.3.2. Escalonamento Customizado (*Custom Queuing*) 30

3.1.3.3. Escalonamento Justo (*Fair Queuing*) 32

3.1.3.4. Escalonamento Diferenciado (*Class Based Queuing*) 34

3.1.4. Controle de Admissão (*Admission Control*) 34

3.1.5. Protocolo de Sinalização (*Setup Protocol*) 36

3.1.6. Classes de Serviço da Arquitetura IntServ 39

3.1.6.1. Serviço Garantido 39

3.1.6.2. Serviço de Carga Controlada 40

3.2. Serviços Diferenciados - DiffServ **40**

3.2.1. Comportamento por Salto (*Per Hop Behavior*) 43

3.2.2. Classificador de Tráfego 44
3.2.2.1. Condicionador de Tráfego 45
3.2.2.1.1. Medição 45
3.2.2.1.2. Marcação 45
3.2.2.1.3. Moldagem 45
3.2.2.1.4. Policiamento 46
3.2.2.1.5. Descarte 46
3.2.3. Serviço *Default* (DE) 49
3.2.4. Encaminhamento Expedito (EF) 49
3.2.5. Encaminhamento Assegurado (AF) 49

3.3. Comutação Multiprotocolo por Rótulos - MPLS 50
3.3.1. Componentes do Modelo MPLS 51
3.3.2. Pilha de Rótulos (*Label Stack*) 53
3.3.2.1. Entrada de Rótulo do Próximo Salto - NHLFE 54
3.3.2.2. Mapeamento do Rótulo de Entrada - ILM 54
3.3.2.3. Mapeamento FEC para NHLFE - FTN 55
3.3.3. Encaminhamento dos Pacotes 55
3.3.4. Métodos de retenção de Rótulos 56
3.3.5. Encapsulamento de Rótulos 56
3.3.5.1. Encapsulamento ATM 56
3.3.5.2. Encapsulamento Frame Relay 60
3.3.5.3. Encapsulamento - PPP & LAN 61
3.3.6. Distribuição do Rótulo 62
3.3.6.1. Através do Protocolo de Distribuição de Rótulos - LDP 62
3.3.6.2. Através do Protocolo de Reserva de Recursos RSVP 63
3.3.6.3. Através do Protocolo entre Sistemas Autônomos BGP 64
3.3.7. Seleção da Rota 65
3.3.8. Fusão (*Merging*) 65

4. IMPLEMENTAÇÃO DE AMBIENTE EXPERIMENTAL 67

4.1. Simulador NS 67

4.2. Simulação 70
4.2.1. Topologia da Rede IP 70
4.2.2. Geração de Tráfego 72
4.2.3. Rede IP com Escalonador CBQ/WRR 74
4.2.4. Topologia da Rrede IP sobre MPLS 74
4.2.5. Resultados da Ssimulação 75
4.2.6. Conclusão da Simulação 77

5. Conclusão 79

5.1. Temas para Trabalhos Futuros 80

REFERÊNCIAS BIBLIOGRÁFICAS 81

Anexo A – Scripts gerados para Simulação 86

LISTA DE FIGURAS

Figura 1. Tipo de Serviço [19] 10
Figura 2. Mapeamento QoS para Internet Unidirecional [40] 20
Figura 3. Arquitetura do modelo de serviços integrados [30] 23
Figura 4. Formato de especificação de tráfego [46] 25
Figura 5. Algoritmo do Balde de Fichas 27
Figura 6. Algoritmo de Escalonamento Priorizado (adaptado de [48]) 29
Figura 7. Escalonamento Priorizado adaptado de [48] 30
Figura 8. Escalonamento Customizado [47] 32
Figura 9. Algoritmo de escalonamento Customizado [48] 32
Figura 10. Escalonamento Justo Ponderado [48] 34
Figura 11. Protocolo de Reserva de Recursos 38
Figura 12. Pontos de Código do Serviço Diferenciado [57] 42
Figura 13. Octeto Tipo de Serviço 43
Figura 14. Arquitetura do Serviço Diferenciado 44
Figura 15. Componentes da Arquitetura MPLS 52
Figura 16. Formato do Cabeçalho MPLS 53
Figura 17. Pilha de Rótulos 54
Figura 18. Encapsulamento ATM [71] 57
Figura 19. Malha Completa do Modelo Overlay 59
Figura 20. Resolução do Próximo Salto NHRP 60
Figura 21. Encapsulamento Frame Relay [71] 61
Figura 22. Encapsulamento PPP e LAN [71] 61
Figura 23. Distribuição de Rótulos 63
Figura 24. Atualização do Protocolo RSVP [71] 64
Figura 25. Fusão VP 66
Figura 26. Visão simplificada do simulador NS [82] 68
Figura 27. Árvore Hierárquica [82] 69
Figura 28. Topologia da Rede Simulada 71
Figura 29. Percentual perda 76
Figura 30. Atraso médio 77

LISTA DE TABELAS

Tabela 1. Formato de um Datagrama IP [9] 6
Tabela 2. Componentes Humanos e Técnicos [13] 8
Tabela 3. Diferenças entre Termos IETF e ITU-T 17
Tabela 4. Parâmetros de Desempenho IP [38] 18
Tabela 5. Atrasos Unidirecionais [40] 19
Tabela 6. Níveis de geração de tráfego 73
Tabela 7. Percentual de perda 73

LISTA DE ABREVIATURAS E SIGLAS

AAL5	ATM Adaptantion Layer 5
ACK	Acknowledge
AF	Assured Forwarding
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
ATMARP	ATM Address Resoltuion Protocol
BGP	Border Gateway Protocol
BLS	Burst Loss Sensitivity
CBQ	Class Based Queuing
CBR	Constant Bit Rate
CHQ	Controlled Highest Quality
CL	Connectionless
CLIP	Classical IP over ATM
CO	Connection Oriented
COS	Class of Service
CQ	Custom Queuing
DE	Default
DiffServ	Differentiated Services
DLCI	Data Link Connection Identifier
DS	Differentiated Service
DSCP	Differentiated Service Code Point
DWFQ	Distributed Weight Fair Queuing
EF	Expedited Forwarding
FCFS	First Come First Serve
FEC	Forwarding Equivalence Class
FF	Fixed Filter
FIFO	First In First Out

Fspec	Filter Specification
FTN	FEC to NHLFE
FTP	File Transfer Protocol
GOS	Grade of Service
HLEN	Header Length
HQA	Highest Quality Attainable
HS	Hard State
ICMP	Internet Control Message Protocol
ID	Identification
IETF	Internet Engineering Task Force
ILM	Incoming Label Map
IntServ	Integrated Services
IP	Internet Protocol
IPPM	Internet Protocol Performance metrics
IPX	Internet Packet Exchange
ISDN	Integrated Services Digital network
ISSL	IntServ Specification Link Layer Working Group
ITU-T	International Telecommunication Union
LAN	Local Area Network
LDP	Label Distribution Protocol
LER	Label Edge Router
LI	Loss Interval
LIFO	Last In First Out
LIS	Logical IP Subnets
LQA	Lowest Quality Acceptable
LS	Loss Sensitivity
LSP	Label Switch Path
LSR	Label Switch Router
MAC	Media Access Control

MBZ	Must be Zero
MDN	Minimum Delay Noticed
MDV	Minimum Delay Variation
MNS	MPLS Network Simulator
MPLS	Multiprotocol Label Switching
MTR	Maximum Transmission Rate
MTU	Maximum Transmission Unit
NAM	Network Animator
NHLFE	Next Hop Label Forwarding Entry
NHRP	Next Hop Resolution Protocol
NHS	Next Hop Server
NLRI	Network Layer Reachability Information
NS	Network Simulator
OSI	Open Systems Interconnection
PHB	Per Hop Behavior
PPP	Point-to-Point Protocol
PQ	Priority Queuing
QoG	Quality of Guarantee
QoS	Quality of Service
RED	Random Early Detection
RESV	Reservation
RFC	Request for Comments
Rspec	Request Specification
RSVP	Resource Reservation Protocol
RTI	Real Time Intolerant Application
SAFI	Subsequent Address Family Identifier
SE	Shared Explicit
SLA	Service Level Agreement
SS	Soft State

ST-II	Revised Internet Stream Protocol
SVP	Switched Virtual Path
TBR	Token Bucket Rate
TBS	Token Bucket Size
TCP	Transfer Control Protocol
TOS	Type of Service
Tspec	Traffic Specification
TTL	Time to Live
UDP	User Datagram Protocol
VC	Virtual Channel
VCI	Virtual Circuit Identifier
VER	Version
VPI	Virtual Path Identifier
WF	Wilcard Filter
WFQ	Weight Fair Queuing
WRED	Weighted Random Early Dropping
WRR	Weighted Round Robin
XTP	Extended Transport Protocol

ABSTRACT

Due to the multimidia applications demand and IP users growth causing frequently overload situations in the network, some Quality of Service (QoS) mechanisms have been created. These mechanisms were developed to decrease packet delay and loss, by controlling the overload state in the network and granting the users with the needs these new applications require.

This work surveys the main QoS Mechanisms used by the IP network routers to provide quality of service as: Integrated Services (IntServ), Differentiated Services (DiffServ) and Multiprotocol Label Switching (MPLS). The dissertation makes a quantitative analysis of performance by means of a network simulator.

RESUMO

Devido a grande demanda de aplicações multimídia e do crescimento de usuários nas redes IP causando freqüentemente momentos de sobrecarga na rede, surgiram mecanismos de qualidade de serviço QoS (*Quality of Service*). Estes mecanismos objetivam diminuir atrasos e perda de informações, controlando o estado de sobrecarga na rede e atendendo às novas necessidades que estas novas aplicações exigem.

Esta dissertação reúne os principais mecanismos de QoS utilizados atualmente pelos roteadores das redes IP para prover qualidade de serviço: serviços integrados (IntServ), serviços diferenciados (DiffServ) e a comutação multiprotocolo por rótulos (MPLS). O trabalho realiza uma análise quantitativa de desempenho através de um simulador de rede.

1. INTRODUÇÃO

A atual arquitetura de protocolos da Internet é baseada em serviço de melhor esforço (*best effort*), não-orientado à conexão, que usa o protocolo IP (*Internet Protocol*) [1]. Neste tipo de serviço não há quaisquer garantias de vazão, atraso ou outro requisito de Qualidade de Serviço QoS (*Quality of Service*). Durante a fase de expansão da rede (meados de 1980), ocorreu um colapso devido ao congestionamento que resultou em uma degradação do serviço, resultado da falta de atenção dos mecanismos dinâmicos de encaminhamento de datagramas. Surgiram algumas recomendações pertinentes ao gerenciamento de filas e mecanismos para evitar congestionamento para os serviços de melhor esforço [2]. Na verdade, eles foram os primeiros vestígios de mecanismos de qualidade de serviço para redes IP.

1.1. MOTIVAÇÃO

Existem duas razões para dissertar sobre qualidade de serviço em redes IP. A primeira razão é que com o aumento da capacidade de processamento dos computadores pessoais e o surgimento de novas aplicações multimídia (imagens, voz e vídeo), a Internet tende a receber uma quantidade cada vez maior deste tipo de tráfego. O modelo de melhor esforço adotado pela Internet para o envio de informações, não atende às novas necessidades que estas novas aplicações exigem [3].

A segunda razão é que os problemas de congestionamento na Internet se devem ao tipo de tráfego transportado pela rede e como, normalmente, o tráfego é em rajadas, acarreta momentos de sobrecarga na rede quando diversas fontes enviam dados simultaneamente. Daí a necessidade de utilização de algoritmos que são usados para determinar a taxa de saída do tráfego e com isso controlar melhor os recursos disponíveis na rede.

1.2. OBJETIVO

O objetivo desta dissertação de mestrado é realizar uma análise qualitativa e quantitativa dos mecanismos de QoS. Para a análise qualitativa serão usadas referências bibliográficas sobre o tema.

A análise quantitativa é realizada através de um simulador de rede. Pretende-se criar uma rede hipotética com vários roteadores e aumentar gradativamente a geração de tráfego até que se alcance um nível de congestionamento onde a perda de pacotes degrade a qualidade de serviço. Depois disto, a rede é configurada de tal modo a suportar um mecanismo diferenciado de escalonamento de pacotes e finalmente simular uma rede IP sobre MPLS. Será realizada uma comparação para determinar o desempenho de três configurações propostas.

1.3. ORGANIZAÇÃO DA DISSERTAÇÃO

O restante da dissertação está organizado da seguinte maneira: o capítulo 2 introduz o conceito de qualidade de serviço e as principais métricas usadas para medir seu desempenho, focando principalmente nos parâmetros perda e atraso de pacotes.

O capítulo 3 descreve os principais modelos e mecanismos de qualidade de serviços. O modelo de serviços integrados IntServ (*Integrated Services*) estabelece e mantém reservas de recursos para uma determinada aplicação, provendo garantias de QoS fim-a-fim em nível de fluxo. Descreve também mecanismos de prevenção de congestionamento onde são usados algoritmos de escalonamento, conforme critérios pré-estabelecidos de classificação de datagramas. O modelo de serviços diferenciados DiffServ (*Differentiated Services for the Internet*), fornece tratamento diferenciado, redefinindo o campo reservado para ToS (*Type of Service*) no cabeçalho IP e a partir da análise deste campo provê diferentes tratamentos aos pacotes de cada classe transportados pela rede. Em caso de congestionamento, o capítulo descreve, dentre outros, mecanismos de descarte inteligentes de pacotes

que atuam para diminuir atrasos na rede e perda de informações, controlando conseqüentemente, o estado de sobrecarga na rede. Encerra-se o capítulo com a apresentação da comutação multiprotocolo por rótulos MPLS (*Multiprotocol Label Switching*), que insere um rótulo ao pacote e este governará sua transmissão pelo domínio.

O capítulo 4 descreve os procedimentos usados para simular três configurações distintas com o simulador NS (*Network Simulator*), comparando o desempenho entre as mesmas.

O capítulo 5 conclui a dissertação apontando temas para trabalhos futuros e finalmente o anexo A mostra os scripts usados para a realização da simulação.

2.CONCEITO E MÉTRICAS DE QOS

Este capítulo destina-se a conceituar o termo "qualidade de serviço" definido pela união internacional de telecomunicações ITU-T (*International Telecommunication Union*), principal órgão de padronização em telecomunicações, e pela força tarefa de engenharia da Internet, órgão denominado de IETF (*Internet Engineering Task Force*).

A subjetividade do conceito reforça ainda mais a idéia de que é através dos parâmetros que o nível de QoS pode ser objetivamente determinado e diferenciado para os diversos tipos de aplicações, cuja classificação encerra o capítulo. A idéia básica da qualidade de serviço é que o tráfego pode ser diferenciado através de diferentes níveis de serviço, cuja granularidade da diferenciação pode ser um conjunto de classes ou um fluxo individual. Com isso pode-se diferenciar os aplicativos que requerem características distintas quanto à perda e atraso dos pacotes.

2.1. CONCEITOS PRELIMINARES

Torna-se importante definir alguns conceitos para o melhor entendimento dos capítulos subseqüentes da dissertação.

2.1.1. Modos de Conexão

Existem dois modos de conexão disponíveis em telecomunicações: o serviço que usa o modo orientado à conexão (CO *Connection Oriented*) e o modo não-orientado à conexão (CL *Connectionless*).

O modo orientado à conexão, também conhecido como serviço de circuito virtual [4], fornece um alto nível de garantia de entrega, pois o caminho lógico é

estabelecido entre a fonte e o destino e permanece até que uma das partes libere a conexão [5]. Neste tipo de serviço, é possível alocar previamente todos os recursos necessários, de tal forma a otimizar o controle de congestionamento entre redes. As redes de comutação por circuitos (telefonia), comutação por quadros (*frame relay*) e comutação por células ATM (*Asynchronous Transfer Mode*), são alguns exemplos de protocolos que operam neste modo.

O modo não-orientado à conexão CL, também conhecido como serviço de datagrama [4], não prevê o estabelecimento prévio de uma conexão [5]. Neste modo nenhum acordo dinâmico é efetuado no momento da utilização do serviço. Não há garantias de chegada em seqüência, nem de entrega do datagrama. A mensagem é tratada de forma individual e entregue ao destino através do caminho mais conveniente definido através de algoritmos de roteamento, que utilizam o caminho mais curto para alcançar o destino, onde "mais curto" não significa necessariamente a menor distância física, e sim o menor número de saltos ao destino ou outro parâmetro [6].

Os dois modos requerem características distintas: o modo orientado à conexão requer protocolos de sinalização, protocolos de roteamento e protocolos para transferência de dados. Já o modo não-orientado à conexão requer apenas os protocolos de roteamento e de transferência de dados do usuário.

2.1.2. Elementos de Rede

As redes IP foram concebidas inicialmente através da utilização de roteadores. O aumento de novas redes à nuvem IP causa freqüentemente pontos de congestionamento que degradam o desempenho de processamento dos roteadores. Para minimizar estes efeitos, introduziram-se comutadores, elementos caracterizados pela alta velocidade de comutação.

Tanto o roteador como o comutador objetivam encaminhar os pacotes recebidos pela interface de entrada para a interface de saída. A principal diferença citada em

[7] é que o roteador executa as funções de encaminhamento da informação através de software e o comutador através de hardware [9]. Para não haver dúvidas quanto aos termos usados nesta dissertação, o termo roteador será usado para nomear os dispositivos que executam funções da camada de rede (camada 3) e comutador para aqueles que executam as funções da camada de enlace, ou seja, a camada 2 do modelo de interconexão de sistemas abertos OSI (*Open Systems Interconnection*).

2.1.3. Formato do Datagrama IP

A unidade básica de transferência de dados em redes é denominada de datagrama IP [8]. Um datagrama divide-se em duas partes: cabeçalho e área de dados. O cabeçalho contém, dentre outras informações, os endereços de origem e destino e o campo de dados contém a carga útil do datagrama. A tabela 1 ilustra a organização dos campos em um datagrama [9]:

TABELA 1. FORMATO DE UM DATAGRAMA IP [9]

0	8	16	31
VER	HLEN	TOS	TOTAL LENGTH
ID		FLAGS	FRAGMENT OFFSET
TTL	PROTOCOL	HEADER CHECKSUM	
SOURCE IP ADDRESS			
DESTINATION IP ADRESS			
OPTIONS			PADDING
DATA			
...			

O primeiro campo VER (*version*) contém a versão do protocolo IP. O campo de comprimento do cabeçalho HLEN (*Header Length*) fornece o comprimento do cabeçalho do datagrama medido em palavras de 32 bits. O campo tipo de serviço TOS (*type of service*) especifica como o datagrama deve ser tratado. Este campo será visto com mais detalhes no próximo capítulo. O campo TOTAL LENGTH especifica o conteúdo total do datagrama, incluindo cabeçalho e carga útil. O campo ID (*identification*) permite que a máquina de destino determine a qual datagrama pertence um fragmento recém-chegado. Os bits pertencentes ao campo FLAGS

controlam a fragmentação e o campo FRAGMENT OFFSET especifica o deslocamento dos dados que estão sendo transportados no fragmento. O campo TTL (*time-to-live*) é um contador usado para limitar a o número de saltos dados por um pacote. Quando o datagrama é remontado totalmente a camada de rede precisa saber o que fazer, ou seja, qual o processo de transporte que deve ser aplicado ao datagrama. Isto é determinado pelo campo PROTOCOL. O campo HEADER CHECKSUM realiza a verificação da soma do cabeçalho para assegurar a integridade de seus valores. Os campos SOURCE IP ADDRESS e DESTINATION IP ADDRESS determinam os endereços IP de origem e destino, respectivamente. O campo OPTIONS foi projetado para permitir que versões posteriores do protocolo incluam informações inexistentes do projeto original, possibilitando a implementação de novas idéias. O campo PADDING representa bits contendo zero, usado para garantir que um cabeçalho se estenda até o múltiplo exato de 32 bits. Finalmente o campo DATA contém a carga útil do datagrama.

2.2. CONCEITOS SOBRE QOS

ITU-T define qualidade de serviço como [10] : "Efeito coletivo do desempenho do serviço, que determina o grau de satisfação do usuário".

Pela IETF [11]: "conjunto de requerimentos que devem ser satisfeitos pela rede quando do transporte de um fluxo ". Fluxo é definido como sendo uma seqüência de pacotes da fonte ao destino (*unicast* ou *multicast*), com uma QoS a ela associada.

Existem outras definições, como por exemplo, a conceituada por [12], que define QoS como sendo um conjunto de tecnologias que habilitam os administradores de rede a gerenciar os efeitos de congestionamento no tráfego proveniente das aplicações, através do uso otimizado dos recursos da rede, e não apenas adicionando continuamente capacidade.

Além destas definições, existe ainda outra [13] que demonstra que QoS possui não apenas componentes técnicos mensuráveis, mas também componentes

humanos, que são subjetivos. A relação entre estes componentes estão ilustrados na tabela 2.

TABELA 2. COMPONENTES HUMANOS E TÉCNICOS [13]

Fatores Humanos	Fatores Técnicos
Estabilidade da qualidade de serviço	Confiabilidade
Disponibilidade de linhas de assinantes	Expansão
Tempo de espera	Eficiência
Tempo de recuperação de falhas	Manutenção
Informações de assinantes	Tempo de congestionamento
Estabilidade de operação do sistema	Qualidade de transmissão

Como as definições acima não esclarecem completamente o conceito de QoS adotou-se nesta dissertação, como será apresentado em capítulos subseqüentes, o conceito de que uma característica de QoS pode ser identificada, quantificada e usada para modelar o comportamento do sistema. As exigências de QoS são expressas em termos de parâmetros, que pode ser um vetor ou um valor escalar e estar relacionado a uma grande variedade de características [14].

2.2.1. Grau de Serviço (GoS)

O grau de serviço GoS (*Grade of Service*) está sendo usado na indústria de telecomunicações para indicar componentes que contribuem para a qualidade de serviço baseada nas experiências do usuário.

Quando o volume de tráfego excede a capacidade da rede, o resultado é um congestionamento [15]. As limitações da rede afetam os serviços e podem ser expressas em valores de GoS, como por exemplo: probabilidade de perda de informações, média de atrasos, taxa de insucessos devido ao congestionamento, etc. Na rede comutada por circuitos, o GoS foi dividido em dois padrões [16]:

2.2.1.1. Perda do Grau de Serviço

Representa a probabilidade que uma conexão não possa ser estabelecida, entre

um circuito de entrada e um circuito de saída, no momento da tentativa de estabelecimento de uma conexão. Para centrais telefônicas digitais internacionais a probabilidade não pode exceder a 0,2 % em carga normal e 1% em momentos de maior movimento.

2.2.1.2. Atraso no Grau de Serviço

Dependendo da tecnologia usada a informação pode conter muitas componentes. Para a rede digital de serviços integrados, ISDN (*Integrated Services Digital Network*), por exemplo, os atrasos são definidos para as seguintes mensagens de sinalização: pré-seleção, pós-seleção, sinal de resposta e de liberação da conexão [17] .

2.2.2. Classe de Serviço

O conceito de classe de serviço COS (*Class of Service*) divide o tráfego em classes diferentes e a rede disponibiliza serviços dependendo da classe atribuída. A CoS possui medidas relativas: uma classe pode ter uma probabilidade de perda de pacote de 10^{-6}, enquanto outra 10^{-3}.

Deve-se usar algum fator para se poder classificar o tráfego da rede. Estes fatores podem ser [18]: identificador de protocolo, número da porta TCP (*Transport Control Protocol*) ou UDP (*User Datagram Protocol*) de origem.

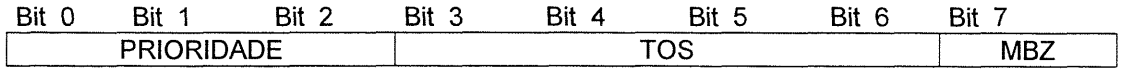
2.2.3. Tipo de Serviço

O tipo de serviço TOS (*Type of Service*), também denominado de prioridade ou precedência (*priority / precedence*), é um mecanismo de advertência e não um mecanismo adequado para garantir o serviço requisitado, por duas razões [19]: nem todas as redes levam em consideração o valor do campo TOS, no momento de decisão sobre como tratar e rotear pacotes, e a segunda razão é que o mecanismo

TOS não é poderoso o suficiente para uma aplicação quantificar o nível de serviço desejado. Por exemplo, não seria apropriado para uma rede decidir se descarta um pacote com o *throughput* solicitado, se não existe nenhuma rota disponível com este valor de *throughput*.

A facilidade TOS é apenas uma das características do campo ToS dentro do cabeçalho do datagrama IP [19]. A figura 1 mostra que o tipo de serviço é composto de três campos: prioridade, TOS e MBZ (*must be zero*).

FIGURA 1. TIPO DE SERVIÇO [19]



As redes tratam o tráfego prioritário como sendo o tráfego mais importante nos casos de maior movimento, geralmente aceitam apenas tráfegos cujo valor estejam dentro de um determinado limiar. O campo MBZ é configurado para zero e ignorado pelos roteadores. O campo TOS é subdividido em: atraso, *throughput* , confiabilidade e custo.

Normalmente a escolha está na combinação de três indicações: baixo atraso, alta confiabilidade e alto *throughput*:

1000 – minimizar atraso

0100 -- maximizar *throughput*

0010 -- maximizar confiabilidade

0001 – minimizar custo

0000 – serviço normal

Por exemplo, se o campo TOS estiver configurado para "1000" , quer dizer que a rede tentará escolher a rota disponível com o menor atraso, baseado nas informações sobre retardos nas rotas. Embora este campo estivesse presente desde o início no cabeçalho do datagrama IP, a maior parte dos roteadores simplesmente o ignoravam [8]. Recentemente este campo foi redefinido e é utilizado por vários roteadores no modelo DiffServ, como será visto mais adiante.

2.2.4. Contrato

O contrato entre a provedora de serviço e o usuário, SLA (*Service Level of Agreement*) [14] é usado para descrever as ações que a provedora de serviços deve tomar baseado nos acordos realizados com o usuário do serviço. As ações podem ser:

- Nenhuma ação
- Monitoramento da QoS alcançada
- Controle do fluxo de informação
- Restrição da QoS alcançada
- Reserva ou reposição de recursos
- Advertência sobre os limiares ultrapassados
- Suspensão ou aborto da atividade

Pode-se definir muitos outros níveis contratuais. A recomendação da ITU-T X641 [14] define 3 níveis de acordo que podem ser negociados: melhor esforço, compulsório (o serviço deve ser abortado se os limiares forem ultrapassados, contudo este nível não é garantido) ou garantido (o serviço não será iniciado se os parâmetros não estiverem nos limites acordados). Normalmente, a QoS é uma combinação de várias características, dependendo da aplicação:

Dados brutos : alto *throuput,* com baixa taxa de erros;

Dados interativos : baixo retardo, com baixa taxa de erros;

Dados isócronos : alto *throuput*, com atraso constante;

Dados sensitivos ao atraso : atraso constante, com *throughput* fixo .

2.3. PONTOS DE VISTA SOBRE QOS

Huston [20] não acredita que se possa obter QoS em um ambiente heterogêneo de multiprovedores, como é o caso da rede pública IP. Porém muitas idéias surgiram e esta seção descreve alguns pontos de vista quanto ao provisionamento de QoS.

2.3.1. Sob o ponto de vista do Fluxo de Tráfego

No caso da concatenação de domínios de (sub-) redes [11], as facilidades de sinalização são transferidas entre as redes, de tal forma, que as exigências de QoS sejam realizadas fim-a-fim. A solução de Qos pode ser dividida em três partes sob o ponto de vista do fluxo de tráfego [21]:

Controle de QoS por salto : seu objetivo é habilitar comutadores e roteadores, que se encontram em pontos de congestionamento, a fornecer características previsíveis sobre a perda, latência, variações nos atrasos (*jitter*) para determinadas classes de tráfego de interesse. Este modelo separa o tráfego das diversas filas, em cada ponto de congestionamento, atribuindo aos mesmos classes diferenciadas e enviando-os às filas dimensionadas com a perda, latência e *jitter* desejados. O acesso à linha de saída é mediado pelo escalonador (*scheduler*), que esvazia cada fila em proporção ao compartilhamento da linha ou conforme sua prioridade. Para tanto, roteadores e comutadores devem classificar os pacotes, diferenciá-los por classe e finalmente fornecer um escalonamento controlável e previsível para transmissão dos pacotes para cada classe (fila) para a linha de saída. Esta arquitetura é denominada de CQS (*Classify, Queue , Schedule*), classificação e escalonamento.

Engenharia de Tráfego : conhecida também como roteamento explícito (*Explicit Routing*) e roteamento baseado em restrições (*Constraint-based Routing*) [22]. Existem duas razões pelas quais quase todos os esquemas de QoS tentam prover serviços diferenciados em condições de maior movimento: primeiro porque em condições de congestionamento a demanda excede os recursos disponíveis e segundo a distribuição de tráfego é desigual. No primeiro caso, pode-se aumentar a capacidade da rede ou limitar o uso de recursos através de mecanismos de QoS. No segundo caso, pode-se distribuir a carga de forma mais balanceada. A engenharia de tráfego combina o fluxo de tráfego de tal forma que o congestionamento causado por utilização desigual pode ser evitado e este é uma das principais motivações [23].

Sinalização ou Provisionamento : estes dois termos diferenciam-se pelo tempo de configuração de uma ação. Sinalização é usada quando a (re)configuração pode ser solicitada pelo usuário a qualquer instante e fornecido em segundos, já o provisionamento é medido em minutos ou horas. De qualquer forma, a ação de (re)configuração envolve o estabelecimento ou modificação da informação usada por comutadores ou roteadores para controlar os mecanismos de roteamento. Alguns protocolos foram desenvolvidos única e exclusivamente com a finalidade de realizar o processo de sinalização, como o protocolo de reserva de recursos RSVP (*Resource Reservation Protocol*) e o protocolo de distribuição de rótulos LDP (*Label Distribution Protocol*).Sem o processo de sinalização, os roteadores e comutadores regridem para o mecanismo de roteamento pelo melhor esforço e retornam às regras de CQS.

2.3.2. Sob o Ponto de Vista de Sub-redes

Segundo Katsupshi Iida existem três tipos de redes na Internet [24]: redes de domínio, redes de acesso e *stub* e portanto duas formas de realizar QoS fim-a-fim:

Desempenho do Retardo em Redes de Acesso : os pacotes podem vivenciar grandes atrasos em enlaces de baixa velocidade, e este problema está sendo discutido pelo grupo de trabalho ISSL (*Intserv Specific Link Layer Working Group*) da IETF. Com o objetivo de reduzir o atraso, duas tecnologias estão sendo propostas IETF : compressão de cabeçalho e segmentação [25].

Gerência de Fluxo em Redes de alta Velocidade : os modelos usados são os serviços integrados IntServ e os serviços diferenciados DiffServ, onde se estudam maneiras distintas de realizar o provisionamento de QoS fim-a fim.

2.3.3. Sob o ponto de Vista de Negociações de Parâmetros

Para a ITU-T X642 [26], os métodos e mecanismos de QoS possuem três fases distintas : fase de predição, de estabelecimento e fase operacional. Os parâmetros

ou características de QoS são negociadas na fase de estabelecimento da conexão. As definições, porém não abrangem condições de sobrecarga repentina. Na fase de predição são coletadas as informações de histórico das medições de QoS, que refletem os níveis de QoS anteriores. A fase operacional é a fase de comunicação propriamente dita. A fase de estabelecimento será tratada com mais detalhes, pois é a fase que compreende métodos para estabelecer acordos, alocação de recursos e métodos de inicialização.

As topologias para o estabelecimento de negociações de parâmetros de QoS podem ser: ponto-a-ponto, 1xN multidestinos ou então NxM multidestinos. A topologia ponto-a-ponto prevê a negociação de parâmetros de QoS envolvendo três partes, os dois usuários envolvidos na comunicação e o provedor de serviço. O usuário inicial propõe um valor, o provedor pode aceitar, recusar ou negociar outro valor. As partes também podem especificar intervalos que desejam operar, através do valor operacional de mais alto nível de qualidade HQA (*Highest Quality Attainable*), limite de qualidade controlada CHQ (*Controlled highest quality*), ou então o limite da menor qualidade aceitável LQA (*Lowest Quality Acceptable*).

2.3.4. Sob o Ponto de Vista de Camadas

O artigo [27] revisa a pesquisa na área de QoS de organizações não *standards*, porém discute a importância de duas camadas para as organizações de padronização: camadas de transporte e rede.

2.3.4.1. Camada de Transporte

Um grande número de pesquisadores tem investigado o provisionamento de QoS para camada de transporte. Trabalhos recentes apontam o provisionamento de protocolos baseados em taxas sobre redes de alta velocidade, como o XTP (*Extended Transport Protocol*) por exemplo [28]. XTP é um protocolo estendido da camada de transporte que fornece suporte a serviços orientados à conexão com

entrega em seqüência, QoS configurável, renegociável e com notificação de erro. O serviço orientado à conexão leva em consideração os seguintes parâmetros de QoS: *throughput*, retardo, *jitter*, estratégia de seleção de erro e prioridade relativa. O protocolo apresenta ainda três semânticas de QoS de transporte, além do melhor esforço: QoS compulsório, QoS limiar e QoS máximo.

2.3.4.2. Camada de Rede

A literatura possui uma vasta cobertura sobre o tema relacionado para garantir qualidade de serviço em redes de serviços integrados [29] [30]. Tem-se pesquisado muito as áreas de especificação de fluxos, controle de admissão de fluxos, reserva de recursos [31], modelagem de tráfego e esquemas de gerência de filas. A tese de doutorado [28] encontrou na literatura diferentes modos de fornecer garantias à qualidade de serviço :

Modo controlado, baseado em disciplinas de filas (serviço multiplexado), que preserva a forma do tráfego garantindo que as características dos fluxos sejam as mesmas que a da fonte;

Modo aproximado, que utiliza disciplinas simples como FIFO (*first in first out*) e que aproveita as vantagens da multiplexação estatística;

Modo limiar (*bounding approach*), que leva em consideração qualquer distorção do fluxo, como compartilhamento do processador e enfileiramento justo ponderado WFQ (*Weighted Fair Queueing*), resultando em limiares de desempenho matematicamente determináveis para fornecer garantias estatística e determinística dos serviços;

Modo baseado na observação, que utiliza o comportamento medido do tráfego agregado, juntamente com a especificação de fluxo fornecida pelo usuário para tomar decisões de admissão.

2.4. MÉTRICAS

Apesar da largura de banda ser o primeiro passo na disponibilização de aplicações em tempo-real, não é suficiente para evitar atrasos durante a transmissão de tráfego em rajadas, pois durante a rajada a largura de banda demandada excede a reservada. Mesmo em uma rede IP sem muito tráfego, os atrasos na transmissão podem variar consideravelmente, afetando as aplicações em tempo real [32]. Os atrasos na entrega causam problemas para estes tipos de aplicações, porém os maiores desafios são as aplicações em full duplex, como transmissão de voz por exemplo.

Um dos problemas é quando as aplicações de dados requerem características completamente inversas: para a transmissão de voz existe a necessidade de assegurar baixos atrasos, para a transmissão de dados, em que o tráfego é normalmente em rajadas, a necessidade é garantir largura de banda para descarregar uma página Web, por exemplo, seguido de longos períodos de inatividade. Estas diferenças frustaram todas as tentativas para a criação de uma única rede para disponibilizar todos os tipos de serviços. Por isso, as operadoras basearam-se seu negócio em voz e criaram redes de dados separadas, como *Frame Relay* ou ATM (*Asynchronous Transfer Mode*).

2.4.1. Métricas IP

O grupo de trabalho da IETF, IPPM (*Internet Protocol Performance Metrics*) se reúne para definir métricas para o desempenho da Internet. Segundo [33], o objetivo das métricas IP é alcançar uma situação, na qual usuários e provedores de serviço de Internet possuam uma compreensão comum e precisa do desempenho e confiabilidade das nuvens IP.

O documento [33] define critérios para estas métricas, terminologias, as métricas propriamente ditas, a metodologia e algumas considerações práticas, incluindo fontes de incerteza e erros. O mesmo documento inclui argumentos que

recomendam que as propriedades da Internet não deveriam ser consideradas em termos probabilísticos, pois estes termos freqüentemente incorrem em interpretações sobre o comportamento da rede, ou seja, não existe um estado estático na Internet.

Em [34] e [35] são descritas algumas diferenças na terminologia, visualizadas na tabela 3, no que tange tempo, entre as definições da ITU-T e as do grupo de trabalho IPPM. Normalmente, estas diferenças provêm do fato dos dois órgãos possuir experiências distintas: os documentos da ITU-T possuem uma origem de telefonia, enquanto a IETF possui uma base na área de computação. Portanto, os termos podem gerar certa confusão.

TABELA 3. DIFERENÇAS ENTRE TERMOS IETF E ITU-T

Termo do IPPM	Termo da ITU-T	Significado
Sincronização	Erros na temporização	Diferença entre dois relógios
Precisão	Erro de temporização da UTC	Diferença para a resolução de tempo real
Resolução	Período de amostragem	Medidas de precisão de um determinado relógio
Skew	*time drift*	Medidas de diferença de precisão ou do sincronismo

Em [34], [35] e [36], as métricas do grupo IPPM são baseadas na amostragem randômica de tráfego e para cada métrica ou grupo de métricas existem diferentes recomendações RFCs (*Request For Comments*). Como conectividade é o principal motivo de existência da Internet [37], as métricas definidas pela IETF são baseadas na conectividade tanto em apenas uma direção quanto nas duas direções.

Para as redes IP, a ITU está desenvolvendo uma recomendação que define métricas IP semelhantes aos parâmetros de desempenho de transferência de pacotes [38].

TABELA 4. PARÂMETROS DE DESEMPENHO IP [38]

Métricas IP		Descrição
IPTD	*IP packet transfer delay.* Atraso na transferência do pacote IP	Retardo do datagrama (ou do último fragmento) entre dois pontos de referência. Retardo fim-a-fim ou latência dentro de uma rede.
MIPTD	*Mean* IP *packet transfer delay.* Retardo médio na transferência do Pacote IP	Média aritmética dos atrasos de transmissão dos datagramas de interesse.
IPDV	IP *packet delay variation.* Variação do atraso do pacote IP	Variação de atraso admissível na rede para evitar congestionamentos.
IPER	IP *packet error ratio.* Taxa de erros do pacote IP	Taxa de datagramas errados de todos os pacotes IP recebidos.
IPLR	IP *packet loss ratio.* Taxa de Perda de pacotes IP	Taxa de datagramas perdidos de todos os pacotes IP de interesse transmitidos

Quanto maior for o valor do atraso ou da perda de datagramas, mais difícil será para as camadas de transporte sustentar grandes larguras de banda. O valor mínimo da métrica fornece uma indicação do atraso devido à latência da rede, retardos de propagação e transmissão [36]. Os valores da métrica (atraso) acima do valor mínimo representa uma indicação de congestionamento presente ao longo do caminho.

Como os fluxos de QoS devem ser mapeados em termos de métricas, pois são elas que definem os tipos de QoS que a rede pode suportar [11], será descrito com mais detalhes as métricas que considero as mais importantes: atraso (*delay*) e perda (*loss*), ou seja, com as mesmas pode-se mapear o maior número de serviços com QoS.

2.4.2. Causa de Atrasos

A latência da rede é causada principalmente pelo atraso na propagação, atraso no processamento, atraso na transmissão e atraso nas filas, causado por congestionamento [39]. O atraso na propagação é constante para fibras óticas e fios de cobre. O atraso na transmissão é linear ao tamanho do pacote. Juntos eles formam a latência da rede, parte do atraso fim-a-fim. Além disso, existem os atrasos do sistema operacional, atrasos nas interfaces de entrada e saída, atrasos nas

aplicações e atrasos de compressão e descompressão, dentre outros.

Estudos [40] indicam que a tolerância a atrasos pode variar significativamente de usuário para usuário. Para aplicativos de voz, usuários tolerantes que solicitaram atrasos de 200 ms ou menos, os valores de 300–800 ms foram aceitáveis. Segundo [41], humanos podem tolerar aproximadamente 250 ms de latência antes de notar os efeitos, sendo que latência está relacionada com três fatores: tipos de roteadores (desempenho), disponibilização de roteadores em circuitos privados, organizações e provedores de serviço (para controlar a utilização de largura de banda e com isso a latência) e a Internet não foi originalmente projetada para comunicações em tempo real.

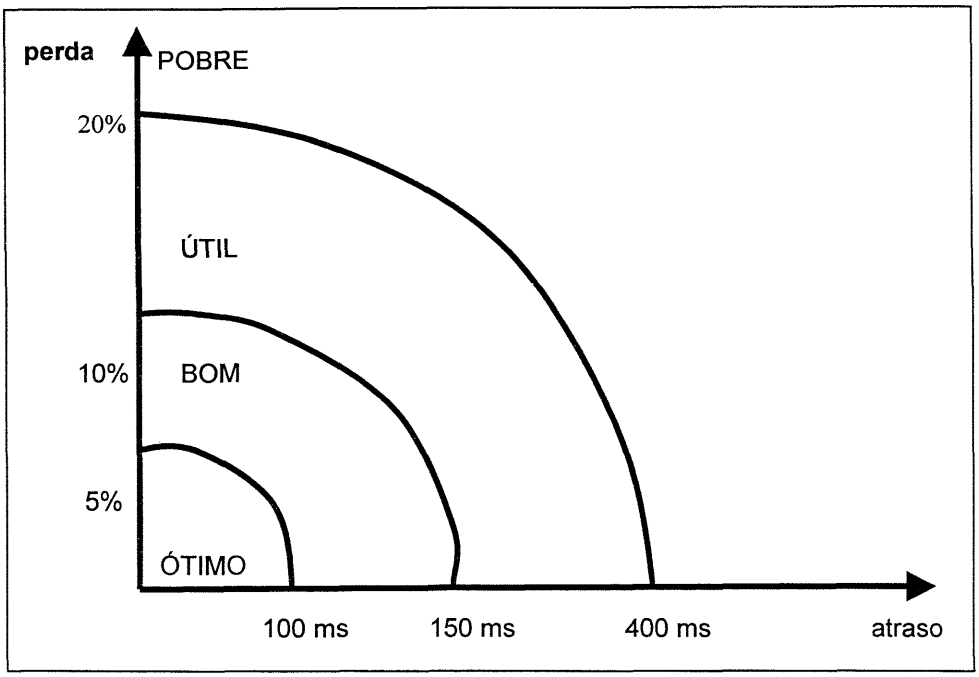
A tabela 5 ilustra fontes de atraso em conexões de roteador-roteador e PC-PC, para as interfaces G.723 e G.729, respectivamente, onde os maiores valores referem-se ao tempo de propagação e enfileiramento e os atrasos presentes em enlaces de acesso de baixa velocidade. A tabela não leva em consideração atrasos de sistema operacional, que pode aumentar significativamente os atrasos , assim como perdas ocasionais de pacotes devido a congestionamentos.

TABELA 5. ATRASOS UNIDIRECIONAIS [40]

Fonte do Atraso	Roteador-Roteador		PC-PC	
	G.723.1	G.729A	G.723.1	G.729A
Codificação/Decodificação	82 ms	30 ms	82 ms	30 ms
Acesso	40-80 ms	40-80 ms	100-340 ms	100-340 ms
Transmissão	< 1 ms	< 1 ms	20-40 ms	10-30 ms
Internet (Propagação e Enfileiramento)	30-100 ms	30-100 ms	30-100 ms	30-100 ms
Total	152-162 ms	100-210 ms	232-562 ms	170-500 ms

A figura 2 ilustra a qualidade percebida em função do atraso e perda para aplicações de voz. Para que haja uma QoS satisfatória, a perda de pacotes não deve ultrapassar 20% e o valor total dos atrasos não deve ultrapassar 400 ms.

FIGURA 2. MAPEAMENTO QOS PARA INTERNET UNIDIRECIONAL [40]



2.4.3. Causa das Perdas

As perdas são causadas pelo congestionamento e instabilidade de roteamento, como mudança de rotas, falhas e perdas temporárias dos enlaces. O congestionamento é a causa mais comum da perda. A instabilidade no roteamento e perda temporária dos enlaces são menos comuns e as perdas devido a falhas nos enlaces são ainda mais raras nas redes de *backbone*.

Para detectar perdas, todo pacote deve possuir um número de seqüência único. Porém, é difícil saber se o pacote ainda não chegou ou se está perdido. Na prática, usa-se um *timeout* de 5 segundos para o tempo de viagem de ida-volta RTT (*Round Trip Time*). Para os atrasos apenas de ida, torna-se um pouco mais difícil de usar um *timeout* , se os relógios não estão sincronizados, mas se alguns pacotes de número de seqüência maior chegaram, é quase certo que o pacote anterior foi perdido [40]. Normalmente, a medição do tempo de viagem de ida-volta é realizada através de um pacote UDP ou ICMP *echo* (*Internet Control Message Protocol*), este último usado para lidar com mensagens de erro e de controle, e espera-se pelo seu retorno.

2.5. APLICAÇÕES

As aplicações podem reagir a atrasos e a perda de informações de formas distintas.

Quanto aos atrasos, as aplicações podem ser definidas como [42]: fluxos em tempo real (*real-time streaming*); transferência de blocos em tempo real (*real-time block transfer*) e aplicações que não exigem tempo real (*non-real time applications*).

Tempo real refere-se à sensibilidade em relação ao tempo da informação entregue pela aplicação. A informação, por sua vez, pode ser classificada em "baseada no tempo" (*time-based*) e "não–baseado no tempo" (*non-time based*). Por exemplo, vídeo, áudio e animação são informações baseadas no tempo, pois geram uma seqüência contínua de blocos de dados que deve ser executado consecutivamente em instantes de tempo pré-determinados (exemplo: vídeo pode ser mostrado a uma taxa de 15 ou 30 quadros de vídeo por segundo).

Quanto à perda de pacotes, as aplicações podem reagir de maneira [38]:

Frágil : quando a perda de pacote excede certos limites, a qualidade é degradada.

Tolerante: a aplicação pode tolerar perda de pacotes, mas quanto maior for a perda de pacotes, menor será a qualidade do serviço.

Elástica: a aplicação pode tolerar até uma grande taxa de perda de pacotes, mas seu desempenho pode ser prejudicado quando a taxa de transmissão é muito elevada.

3. MODELOS DE QOS

O protocolo IP além de ser um protocolo não-orientado à conexão, também é "sem estado" (*stateless*), pois os roteadores ao longo do caminho não mantêm informações específicas sobre o estado de cada fluxo. Os pacotes, pertencentes a um fluxo, são roteados apenas conforme suas tabelas de roteamento. Devido à simplicidade e escalabilidade deste esquema, é que a Internet foi tão bem sucedida. Porém não é o suficiente para fornecer QoS, ou seja, o conjunto original de protocolos TCP/IP não fornece gerenciamento nem controle dos recursos da rede necessários para se obter a qualidade desejada [43].

As razões pelas quais o IP não ter utilizado as noções de QoS são [44]: Primeiramente, devido ao fato do conjunto de protocolos TCP/IP ter sido idealizado para que todos recebessem o mesmo tratamento no acesso, ou seja, igualdade e justiça para todos. Segundo, porque os roteadores utilizam a estratégia de fila FIFO. Terceiro, o colapso na Internet trouxe mudanças ao TCP que permitiram que o mesmo adaptasse sua taxa de transmissão conforme os recursos disponíveis na rede. Essa modificação é denominada de *slow start* [45]. Como nenhuma aplicação necessitava de QoS nunca houve antes a necessidade de reprojetar o TCP/IP.

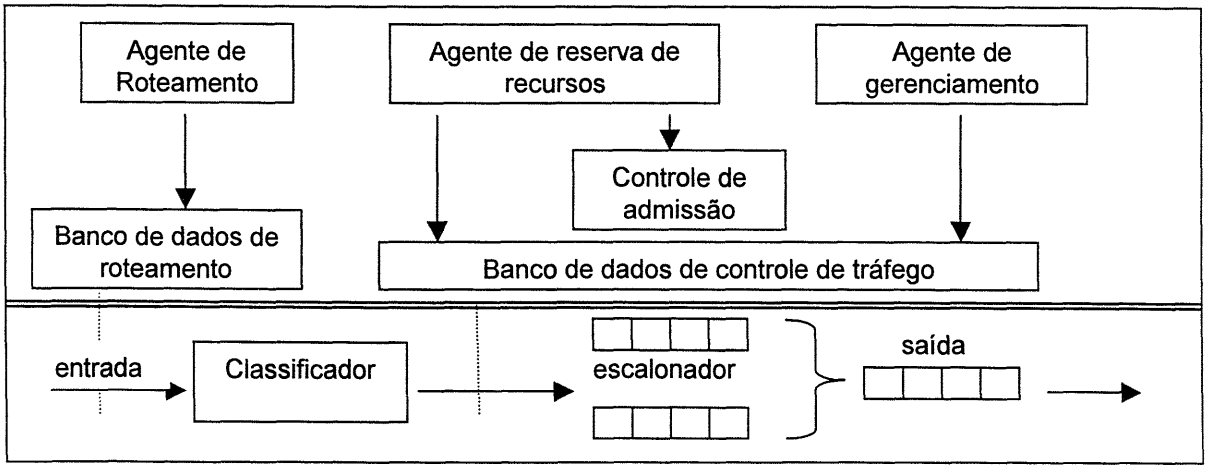
Quando QoS significa prover certa prioridade a determinados tipos de tráfego, o problema é "aonde" atribuir a prioridade. O IETF possui duas propostas para atribuir prioridade para o tráfego na Internet: através da reserva de recursos salto-a-salto (*hop-by-hop*) pelo serviço integrado IntServ ou marcando o pacote nas bordas do domínio pelo serviço diferenciado DiffServ. Outra proposta que objetiva melhorar o desempenho dos roteadores no núcleo da rede é rotular o datagrama IP na entrada do domínio, através da comutação de rótulos MPLS.

3.1. SERVIÇOS INTEGRADOS - INTSERV

Os serviços de "melhor esforço" ou "sem QoS" são implementados através de filas denominadas como "primeiro que entra , primeiro que sai" , ou seja, a fila tipo FIFO, que nada mais é que a capacidade de armazenar e em seguida enviar. Também conhecido como "primeiro que chega é o primeiro a ser servido", FCFS (*first come first served*). De forma bem simples, os pacotes são armazenados em uma fila quando a rede está congestionada e são enviados quando não há congestionamento. Se a fila estiver cheia, os pacotes são descartados. Não existe nenhum tipo de diferenciação de fluxo ou classe, nenhuma garantia de taxa ou de atraso. FIFO é o algoritmo de fila *default*, por isso não exige nenhum tipo especial de configuração.

A falta de garantias ou níveis de QoS na Internet é considerada uma de suas maiores limitações. Para resolver este problema o IETF formou um grupo para pesquisar uma forma de reservar recursos na rede e fornecer garantias de desempenho aos usuários, de forma a fornecer garantias de QoS baseado em fluxos. Também denominado de IntServ [30], definida pela RFC 1633, ou "*hard-QoS*", o modelo é composto de quatro elementos, conforme mostra a figura 3: escalonamento de pacotes, classificação dos pacotes, controle de admissão e protocolo de sinalização.

FIGURA 3. ARQUITETURA DO MODELO DE SERVIÇOS INTEGRADOS [30]



3.1.1. Classificador (*Classifier*)

O classificador mapeia cada fluxo entrando na rede para uma classe e todos os pacotes, pertencentes à mesma classe, recebem o mesmo tratamento do escalonador. Uma classe pode corresponder a muitos fluxos, como por exemplo, todos os fluxos de vídeo, ou uma classe pode conter apenas um único fluxo. O endereço de destino não é o suficiente para selecionar a classe de serviço que o pacote deve receber, ou seja, necessita-se de mais informações. Uma maneira, seria permitir que o classificador analise mais campos do pacote, como o endereço da fonte, o número do protocolo e as portas. Dessa maneira, os fluxos de vídeo poderiam ser reconhecidos pela porta no cabeçalho do protocolo.

3.1.2. Caracterização do tráfego

Depois do tráfego ter sido classificado, ele deve ser medido de acordo com níveis pré-definidos, em termos de largura de banda e rajada. O principal mecanismo de implementação utilizado é o balde de fichas (*token bucket*), mostrado na figura 5. Ele é definido por uma taxa de dados r e uma rajada b. A analogia é imaginar um balde com uma determinada capacidade máxima que contém fichas que são inseridas regularmente. Uma ficha corresponde à permissão para transmitir uma quantidade de bits. Quando chega um pacote, o seu tamanho é comparado com a quantidade de fichas no balde. Se existir uma quantidade suficiente de fichas, o pacote é enviado. Senão, geralmente é inserido em uma fila para que aguarde até haver fichas suficientes no balde. Isso é chamado de suavização ou moldagem de tráfego. Caso o tamanho da fila seja zero, todos os pacotes fora do perfil (que não encontram fichas suficientes) são descartados.

A RFC 1363 [46] descreve a especificação que o fluxo deve ter para qualquer tipo de solicitação, seja para fluxos garantidos ou aplicações que desejam informar à rede sobre suas necessidades.

A figura 4 mostra o formato da especificação para o tráfego de fluxos com seus respectivos campos [46]:

FIGURA 4. FORMATO DE ESPECIFICAÇÃO DE TRÁFEGO [46]

0	16	31
VER		MTU
TBR		TBS
MTR		MDN
MDV		LS
BLS		LI
QoG		

A especificação do fluxo é unidirecional, para fluxos multidirecionais deve-se solicitar a especificação nos dois sentidos. O campo VER (*version*) identifica o número da versão da especificação do fluxo. O campo MTU (*Maximum Transmission Unit*) corresponde ao número máximo de bytes no maior pacote possível de ser transmitido sobre o fluxo. O campo TBR (*Token Bucket Rate*) é um dos três campos usados para definir como o tráfego é injetado na rede pela aplicação. Ele determina qual a taxa de inserção de fichas no balde, onde para cada fluxo mantém-se um balde separado. Para enviar um pacote sobre o fluxo, deve-se remover o número de fichas representando o tamanho do pacote. Se não houver fichas suficientes, deve-se esperar a inserção de novas fichas. Caso este campo esteja setado para zero, significa que a aplicação não realizou nenhuma solicitação para garantir a largura de banda. Os outros dois campos são TBS (*Token Bucket Size*) e MTR (*Maximum Transmission Rate*). O TBR controla a quantidade de dados que o fluxo pode enviar na taxa de pico. Se o valor do campo TBR for B e a taxa de inserção de fichas no balde R, sobre qualquer intervalo de tempo T, a quantidade de dados que o fluxo transmite não pode ultrapassar o valor B + (R * T) em bytes. O campo MTR limita a velocidade de transmissão consecutiva dos pacotes, pode-se dizer em outras palavras, que corresponde à taxa de esvaziamento do balde em bytes. O campo MDN (*Minimum Delay Noticed*) informa à rede o valor mínimo de atraso fim-a-fim que a aplicação pode tolerar em microsegundos. O campo MDV

(*Maximum Delay Variation*) informa à rede a variação de atraso tolerável pela aplicação e corresponde à diferença de atraso (em microsegundos) entre o atraso máximo e mínimo permitido. O campo LS (*Loss Sensitivity*) indica a sensibilidade do tráfego de fluxos quanto a perdas de pacotes. Este campo pode ser representado de duas maneiras: como um número de perda de pacotes de tamanho MTU em um determinado intervalo ou um valor de indicação de sensibilidade, que pode ser 0 (fluxo insensível a perdas) ou 1 (fluxo sensível a perdas), neste caso, a rede deve escolher a rota com a menor taxa de perda possível . O campo BLS (*Burst Loss Sensitivity*) expressa a sensibilidade a perdas consecutivas de pacotes. Este campo enumera o número máximo de pacotes de tamanho MTU que podem ser perdidos. Se o campo contiver um valor 0, significa que o fluxo não é sensível a perdas consecutivas. O campo LI (*Loss Interval*) determina o período pelo qual ocorre o maior número de perdas por intervalo de tempo, ou seja, o número de perdas não deve ultrapassar o número ou valor estipulado no campo LS. Finalmente, o campo QoG (*Quality of Guarantee*) pode conter um dos seis valores abaixo relacionados:

0 – a aplicação não solicita nenhuma garantia, apenas especifica um desempenho para o fluxo.

100 (hex) – a aplicação solicita uma garantia imperfeita.

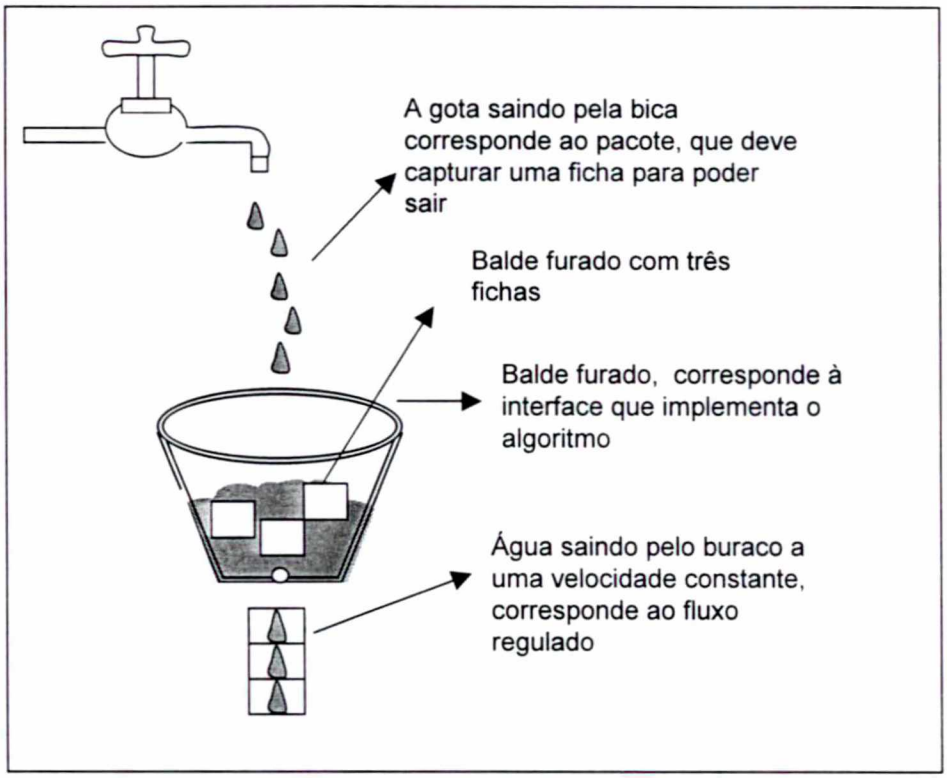
200 (hex) – a aplicação solicita um serviço garantido previsível e se isto não for possível então nenhum fluxo deve ser estabelecido.

201 (hex) – a aplicação solicita um serviço previsível, porém tolera imperfeições.

300 (hex) – a aplicação exige serviço garantido e se isto não for possível então nenhum fluxo deve ser estabelecido.

301 (hex) – a aplicação exige serviço garantido, porém é flexível quanto a imperfeições.

FIGURA 5. ALGORITMO DO BALDE DE FICHAS



3.1.3. Escalonamento de Pacotes (*Packet Scheduler*)

O escalonador gerencia as filas de diferentes fluxos de pacotes utilizando um conjunto de mecanismos de escalonamento ou outros mecanismos como temporizadores. O escalonador de pacotes é implementado no local onde os pacotes estão enfileirados e pode implementar funções de policiamento de tráfego, que assegura que uma máquina não viole as características prometidas.

A função básica do escalonador [2] é reordenar a saída das filas e isso pode ser realizado de diversas maneiras. A forma mais simples, porém, é a estratégia de prioridades, na qual os pacotes são ordenados pela sua prioridade e os que possuem prioridade mais alta deixam a fila antes. Outra forma, é um esquema de escalonamento alternativo, denominado de cíclico (*round-robin*), que atribui acessos distintos aos pacotes com classes diferentes para compartilhar um enlace.

Uma forma que as redes tratam o tráfego em caso de sobrecarga (*overflow*), para

evitar que o congestionamento ocorra, é através do uso de algoritmos de escalonamento usados para classificar os pacotes e então determinar qual a melhor maneira de priorizá-los e enviá-los para a porta de saída. Os métodos mais conhecidos são [47]:

- Escalonamento priorizado PQ (*Priority Queuing*)
- Escalonamento customizado CQ (*Custom Queuing*)
- Escalonamento justo ponderado WFQ (*Weighted Fair Queuing*)
- Escalonamento baseado em classes CBQ (*Class Based Queuing*)

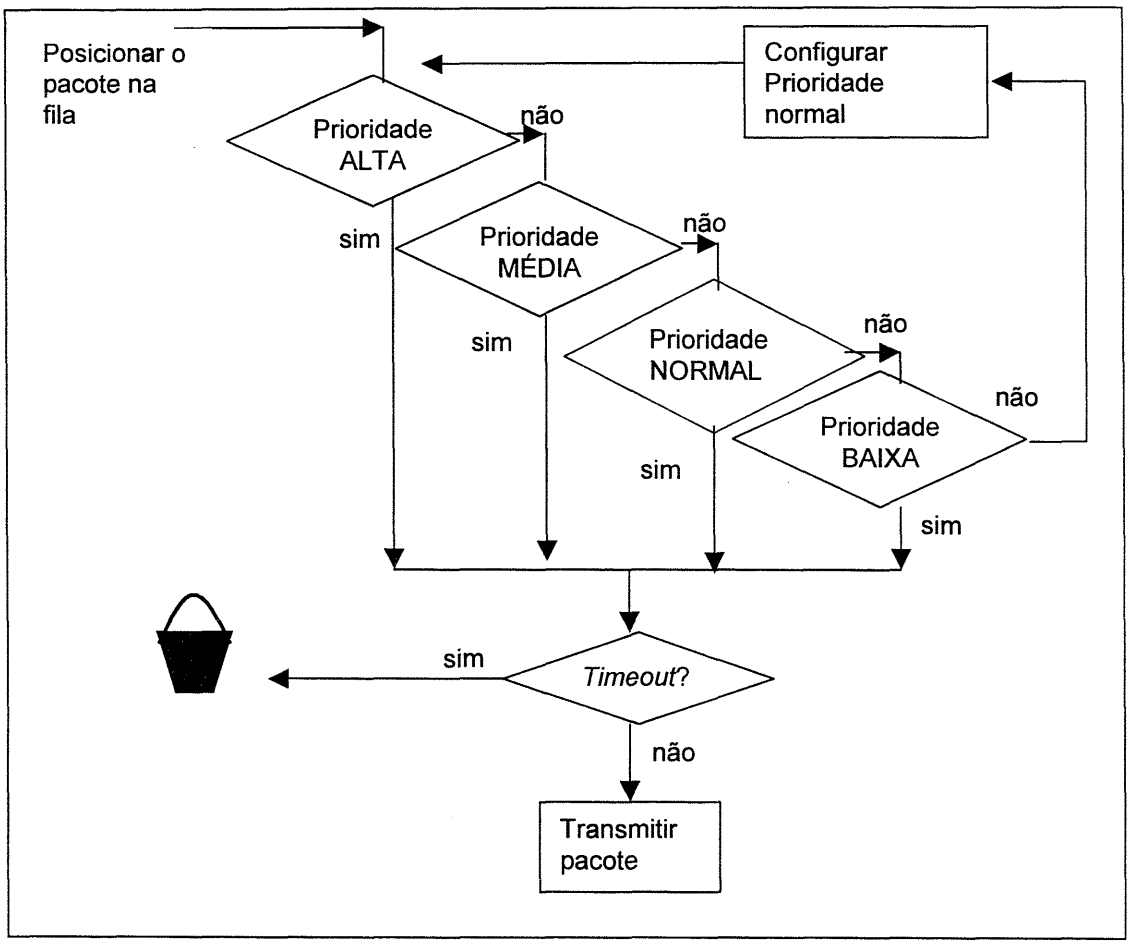
As técnicas de controle de congestionamento são métodos implementados em roteadores no núcleo da rede (*core routers*), para fornecer, de certa forma, uma classe de serviço diferenciado. Os métodos envolvem a criação de diferentes tipos de filas para diferentes classes de tráfego.

3.1.3.1. Escalonamento Priorizado (*Priority Queuing*)

O escalonador priorizado foi projetado para priorizar estritamente o tráfego considerado mais importante [47]. Neste caso, deve-se marcar cuidadosamente o tráfego que atravessa os diversos enlaces pertencentes à rede WAN para que efetivamente recebam tratamento com prioridade.

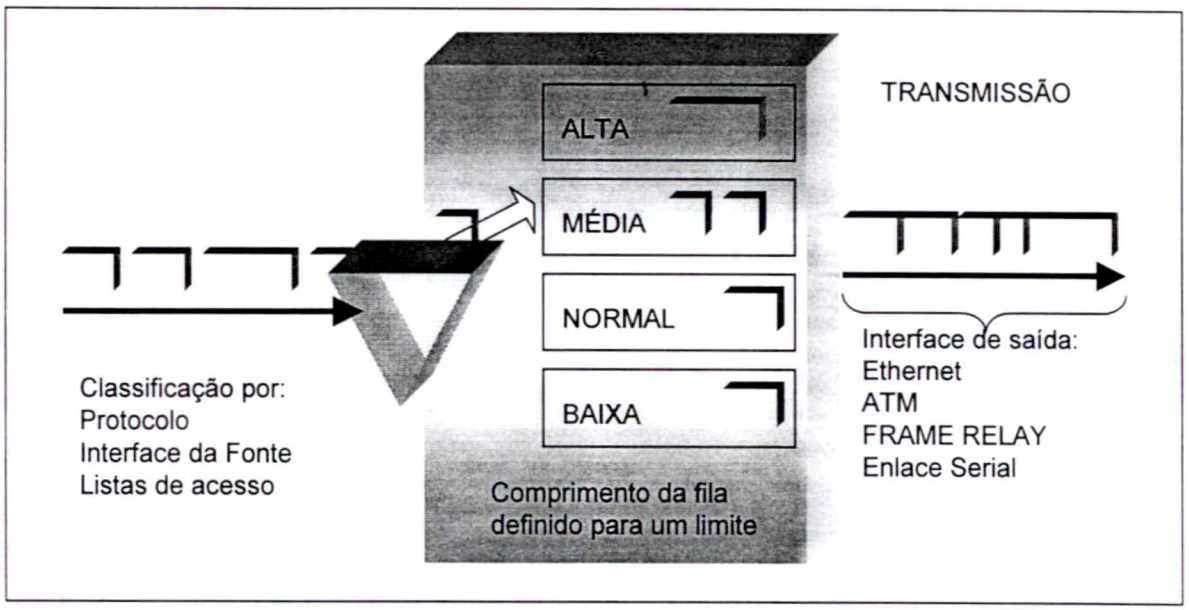
Basicamente, trata-se de uma disciplina de agendamento de múltiplas filas FCFS, ou seja, primeiro que entra é o primeiro que sai, com uma política de escalonamento priorizada, onde o pacote com prioridade mais alta é o primeiro a ser transmitido. Possui uma diferenciação de classes, mas sem garantias quanto aos parâmetros de atraso e taxa de transmissão. A figura 6 ilustra como opera este algoritmo.

FIGURA 6. ALGORITMO DE ESCALONAMENTO PRIORIZADO (ADAPTADO DE [48])



O algoritmo de escalonamento priorizado PQ, mostrado na figura 6, fornece uma fila individual para cada classe de prioridade, que significa, que cada pacote é posicionado em uma das quatro filas: alta, média, normal ou baixa. Os critérios de classificação podem ser o tipo de protocolo, a interface de entrada, tamanho do pacote, segmentos a listas de acesso, conforme ilustrado na figura 7. Pacotes que não tenham sido classificados são tratados pela fila normal. Em caso de congestionamento, as filas são esvaziadas na seguinte ordem: alta, média, normal e baixa e se uma fila ficar maior que o limite pré-estabelecido, os pacotes são descartados. A vantagem deste tipo de escalonador é que fornece um tratamento especial para tráfegos de prioridades mais altas. Sua desvantagem é que sua configuração é estática, ou seja, não se adapta às constantes mudanças da rede.

FIGURA 7. ESCALONAMENTO PRIORIZADO ADAPTADO DE [48]



3.1.3.2. Escalonamento Customizado (*Custom Queuing*)

No algoritmo de escalonamento customizado CQ, as filas são mantidas separadamente conforme as classes de tráfego [47]. CQ foi projetado para permitir que várias aplicações ou organizações pudessem compartilhar os recursos da rede entre as aplicações, com uma largura de banda. Nestes casos, a largura de banda deve ser compartilhada proporcionalmente entre as aplicações e os usuários.

Depois que o tráfego entrando na rede tenha sido adequadamente classificado, através de exigências e solicitações pré-estabelecidas, o escalonador customizado reserva uma certa quantidade de espaço nas filas para cada classe de pacotes. Por exemplo, pode-se reservar metade da largura de banda para uma certa aplicação e permitir que a outra metade seja usada por protocolos como o IP e o IPX (*Internetwork Packet Exchange*).

O algoritmo posiciona as mensagens em uma de suas 17 filas. A fila 0 é reservada para mensagens do sistema (como mensagens de sinalização, por exemplo) e é a primeira fila a ser esvaziada. Além disso, nenhum outro tráfego pode

ser atribuído à esta fila. As filas de 1 a 16 são esvaziadas conforme sua classificação. Os pacotes são retirados das filas ciclicamente (*round-robin)*, e cada fila possui um contador de byte que especifica quantos bytes de dados o sistema deve transmitir antes de ir para a próxima fila. Por isso, a largura de banda, por exemplo, pode ser indiretamente especificada em termos de quantidade de bytes por contador.

Suponha o seguinte exemplo: uma solicitação foi feita para que a largura de banda fosse dividida igualmente entre três protocolos e que os mesmos possuem exigências diferentes: o primeiro protocolo requer pacotes de 1086 bytes por pacote, outro 291 bytes e o terceiro 831 bytes. Para dividir igualmente a largura de banda entre os protocolos, o administrador pode configurar, de forma genérica e resumida, o contador de bytes da seguinte maneira [47]:

Passo 1 Para cada fila, atribuir a percentagem de banda que se deseja alocar à fila, através do tamanho do pacote. Por exemplo: 20% , 60% e 20%.

Passo 2 Normalizar os números , dividindo pelo número de mais baixo valor. O resultado é a razão de número de pacotes que deve ser enviado. No exemplo dado, os resultados seriam 1 e 11,2 e 1,3.

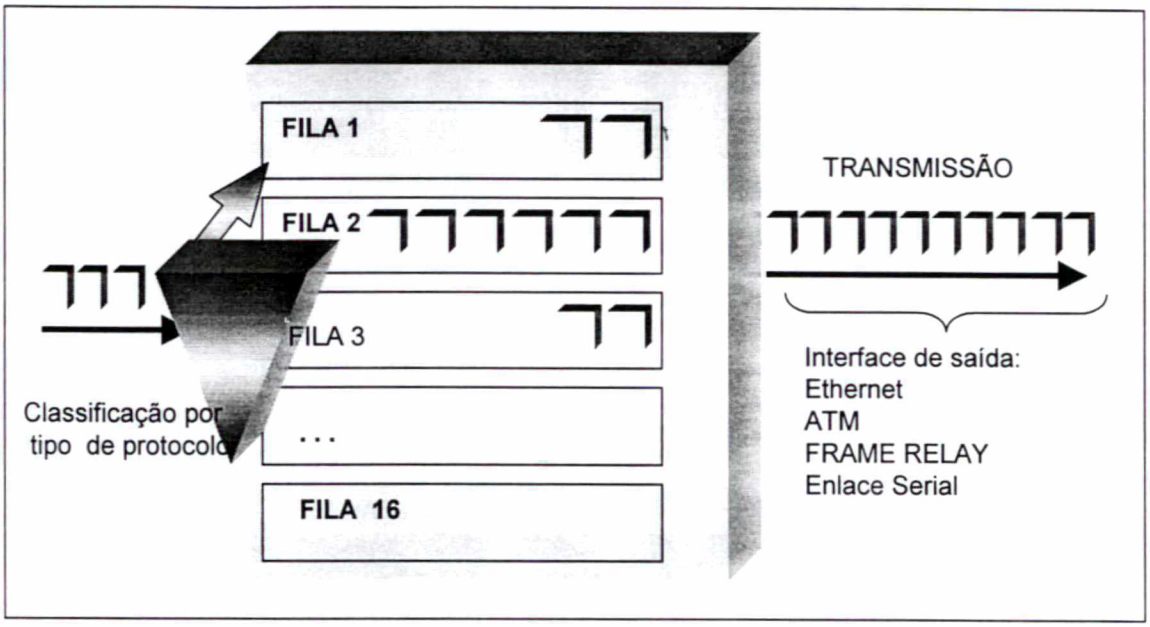
Passo 3 A fração para qualquer um dos valores representa que um pacote adicional deve ser enviado, portanto deve-se arredondar os números para se obter a contagem de pacotes. Portanto: 1, 12 e 2.

Passo 4 Converter a razão do número do pacote em bytes, multiplicando-os pelo tamanho total do pacote. Portanto: (1 * 1086) + (12 * 291) + (2 * 831) = 1086 + 3492 + 1662 = 6240 bytes.

Passo 5 Determinar a percentagem de distribuição de banda real, que será enviada por cada fila, ou seja, no exemplo, dividindo-se cada um pelo valor total de bytes: 1086/6240 = 17,4%; 3492/6240 = 56%; 1662/6240 = 26,6%.

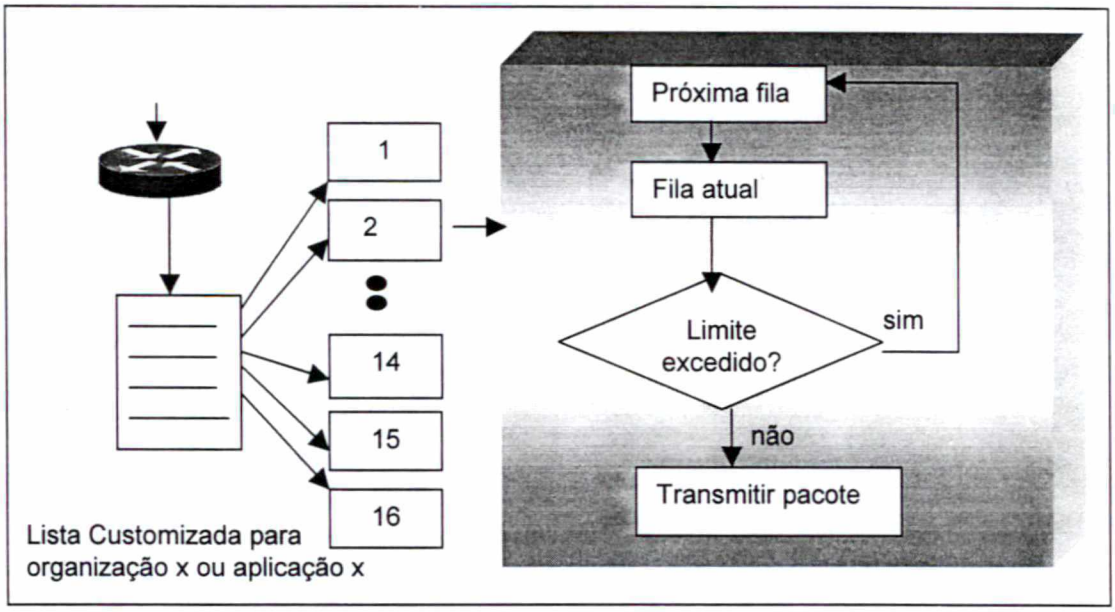
O resultado acima é próximo do valor de distribuição desejado de 20%, 60% e 20%. A figura 8 ilustra de forma esquemática o exemplo.

FIGURA 8. ESCALONAMENTO CUSTOMIZADO [47]



Assim como o escalonamento priorizado, o escalonamento customizado possui uma configuração estática e não se adapta automaticamente às alterações da rede. A figura 9 ilustra esquematicamente o algoritmo de escalonamento customizado.

FIGURA 9. ALGORITMO DE ESCALONAMENTO CUSTOMIZADO [48]



3.1.3.3. Escalonamento Justo (*Fair Queuing*)

O escalonador justo FQ, também denominado de escalonador justo ponderado

distribuído DWFQ (*Flow-based Distributed Weight Fair Queuing*) [47], é um algoritmo dinâmico baseado em fluxo, que realiza duas funções simultaneamente: agenda o tráfego interativo para o começo da fila, para reduzir o tempo de resposta, e compartilha "igualmente" (pesos iguais) a largura de banda remanescente entre fluxos. Em outras palavras, o FQ permite priorizar tráfego de baixo volume (como sessões *Telnet*) sobre tráfego de maior volume (como sessões de transferência de arquivos FTP (*File Transfer Protocol*) [47]. O escalonamento no roteador ou comutador é reorganizado toda vez que um pacote novo entra na rede [44].

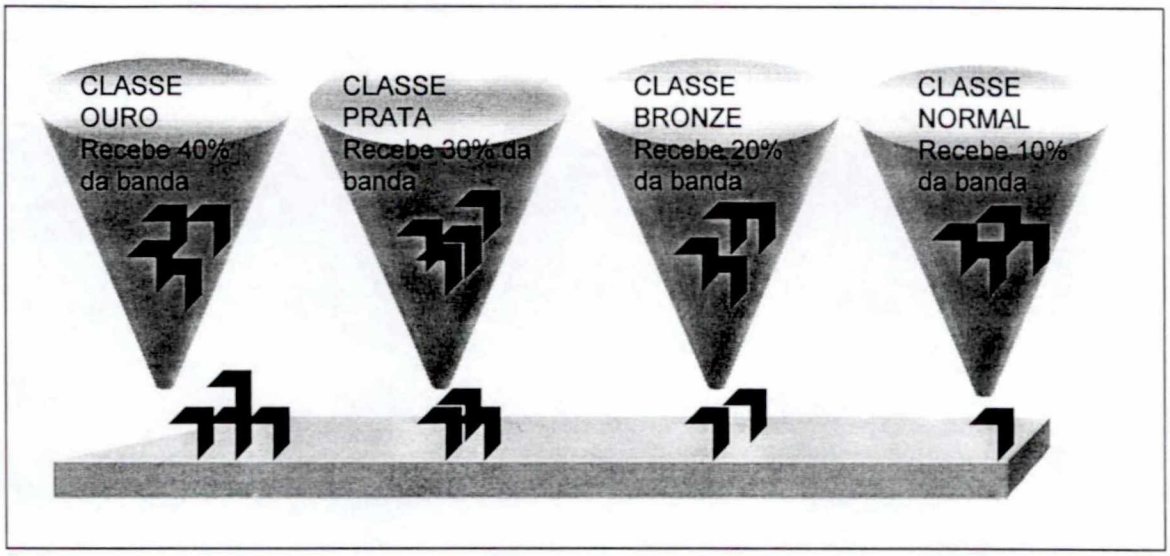
O algoritmo FQ classifica o tráfego em diferentes fluxos, baseado no endereço do cabeçalho (endereço de fonte e destino ou endereço físico MAC – *Media Access Control*), baseado no protocolo, na porta da sessão, nos identificadores da camada de enlace, como no valor do DLCI (*Data Link Connection identifier*) do Frame Relay e no valor do tipo de serviço do campo ToS.

Existem duas categorias de fluxos: as sessões que exigem alta largura de banda e as sessões de baixa largura de banda, sendo que esta última (maioria do tráfego), possui prioridade mais elevada. O tráfego de maior taxa compartilha o serviço de transmissão proporcionalmente, conforme a atribuição de pesos.

Quando a distribuição da largura de banda é realizada atribuindo-se diferentes pesos (*weights*), denomina-se de justo ponderado. Este é o escalonamento padrão para a maioria das interfaces seriais configuradas para transmitir fluxos E1 (velocidades de 2.048 Kbps) ou inferior. Os pesos fornecem informação de prioridade. A figura 10 mostra que os pacotes de maior prioridade são escalonados por primeiro, sendo que o peso máximo determina qual será a parcela do enlace reservada para uma determinada classe de tráfego [49].

Existe uma variação deste algoritmo que é baseada em classes e não em fluxos. Nesta variação reserva-se uma fila para cada tipo de classe, sendo que as classes são caracterizadas pela largura de banda, peso e limite máximo de pacotes, além de especificar o tamanho da fila, ou seja, a permissão para acumular número de pacotes em uma mesma fila.

FIGURA 10. ESCALONAMENTO JUSTO PONDERADO [48]



3.1.3.4. Escalonamento Diferenciado (*Class Based Queuing*)

Normalmente, tráfegos sensitivos a atrasos são tratados antes e por isso para os outros tipos tráfegos usam-se filas maiores. O descarte de pacotes pode ser controlado de diversas maneiras: através da detecção randômica antecipada RED (*Random Early Detection*), que usa métodos probabilísticos, para iniciar o descarte quando se atinge um determinado limiar de fila, a fim de aumentar a probabilidade de armazenamento de pacotes com prioridade mais alta. Por exemplo, pacotes que possuam tipos diferentes de serviços são colocados em filas diferentes, e dentro de um dado tipo de serviço, os pacotes de maior prioridade são descartados antes daqueles com prioridade mais baixa.

3.1.4. Controle de Admissão (*Admission Control*)

Este componente implementa o algoritmo de decisão que um roteador utiliza para determinar se um novo fluxo de tráfego pode ser admitido sem impactar nas garantias de QoS contratadas previamente. O controle de admissão é invocado em cada nó para realizar o processo de decisão local sobre aceitação ou rejeição. Para decidir se a solicitação pode ser atendida, normalmente depende de políticas de

reserva ou simplesmente da disponibilidade de recursos. Uma vez que o controle de admissão tenha completado sua função com sucesso, os recursos locais são reservados imediatamente. Compreende dois modos: o modo tradicional, que lembra sobre os parâmetros de serviços de solicitações passadas e realiza o cálculo baseado nos piores casos (piores limiares) de cada serviço. Outro modo, que fornece uma otimização do enlace, é programar o roteador para medir o uso atual de fluxos existentes e, através destas medições, realizar a admissão de novos fluxos.

A necessidade do controle de admissão é parte integrante do modelo do serviço global, porém os detalhes de como o algoritmo opera é um problema de ordem local. Com isso, os fornecedores podem competir entre si, fornecendo diferentes tipos de algoritmos para implementar esta função.

Enquanto o controle de admissão assegura a capacidade adequada ao longo da rota, o escalonador assegura um atraso limitado nas filas, quando a capacidade da rede é compartilhada com outras classes de tráfego com diferentes características de QoS [50].

Como cada fluxo de tráfego necessita de uma certa quantidade de recursos, como largura de banda no enlace e espaço no *buffer* do roteador, para transferir dados da fonte para o destino, o controle de admissão gerencia estes recursos na rede. O objetivo final é computar com a maior precisão possível a região de admissão, pois se um algoritmo negar acesso a um determinado fluxo, que poderia ter sido aceito, os recursos da rede estarão sendo sub-utilizados.

O controle de admissão pode ser: determinístico, estatístico e medido. Os dois primeiros usam uma estimativa pré-determinada, enquanto que o último é baseado em medidas atuais dos parâmetros escolhidos. O modo determinístico usa o cálculo do tipo "pior caso", que não permite nenhuma violação de QoS. É considerado aceitável para fluxos de tráfego contínuos, mas é ineficiente para fluxos em rajadas. Tanto o modelo estatístico quanto o medido fornecem uma pequena probabilidade de violação de QoS, mas asseguram uma utilização maior dos recursos da rede.

3.1.5. Protocolo de Sinalização (*Setup Protocol*)

O protocolo de reserva de recursos interage com o roteamento para estabelecer, no primeiro instante, a melhor rota através da rede. Depois, baseado no mapeamento de QoS e no controle de admissão, em cada módulo local (p.ex.: CPU, memória, dispositivos de entrada/saída, comutadores, roteadores, etc.), os recursos fim-a-fim são alocados. O resultado final é que os mecanismos de controle e gerenciamento de QoS são configurados de forma mais adequada.

O protocolo de sinalização proposto pelo IETF é o RSVP (*Resource Reservation Protocol*). A reserva é inicializada pelo usuário receptor e podem ser de três tipos :

Filtro curinga WF (*Wildcard-Filter*) : a reserva é compartilhada com todas as origens, ou seja, a reserva se propaga em direção a todas as máquinas.

Filtro fixo FF (*Fixed-Filter*) : reserva é individual para uma origem.

Filtro compartilhado explícito SE (*Shared-Explicit*) : a reserva é compartilhada entre as origens que foram pré-selecionadas, ou seja, o receptor pode escolher os transmissores.

A diferença entre o curinga e o compartilhado é que na reserva compartilhada o receptor escolhe um conjunto fixo de transmissores, dentre vários disponíveis na conexão, para compartilhar recursos.

O protocolo de reserva de recursos RSVP [51] é considerado o protocolo de sinalização "de facto" da arquitetura IntServ [44]. Apesar do RSVP ser um complemento do serviço integrado IntServ, trata-se de uma tecnologia independente [12].

O protocolo RSVP não pode ser caracterizado como sendo um protocolo de roteamento, pois além de não transportar carga útil [32], também não fornece nenhum mecanismo para determinar qual a rota mais adequada para acomodar a QoS solicitada pela aplicação [11]. Existe porém uma interação entre o protocolo de reserva de recursos e o roteamento. Esta interação é sumarizada em quatro fatores [30]: primeiro, encontrar a rota que acomode a reserva de recursos; segundo,

encontrar uma rota que possua capacidade suficiente e não reservada para o novo fluxo; terceiro, adaptar-se a uma falha na rota; e finalmente adaptar-se às mudanças no roteador.

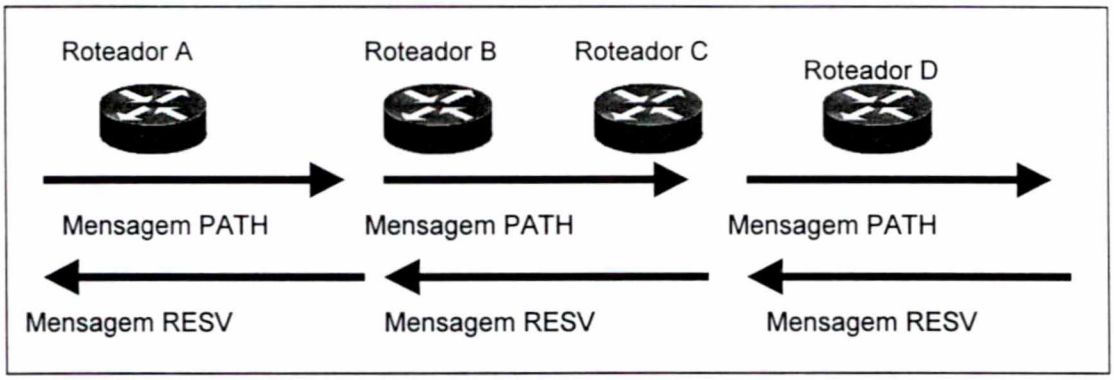
Os itens seguintes destinam-se a resumir o funcionamento do protocolo RSVP :

1) A origem conhece as especificações do tráfego que pode transmitir, mas é o destino que sabe exatamente o que está apto a receber. Portanto, o protocolo RSVP envia uma mensagem, denominada PATH, contendo os parâmetros de especificação de tráfego Tspec (*traffic specification*) ao endereço de destino. Além das especificações TSpec, a mensagem RESV (*reservation*) inclui a especificação de solicitação Rspec (*request specification*), que indica o tipo de serviço integrado que foi solicitado (garantido ou carga controlada) e uma especificação de filtro *FSpec* (*filter specification*), que caracteriza os pacotes, por exemplo, protocolo de transporte e número da porta. Juntos, RSpec e FSpec representam o descritor de fluxos (*flow-descriptor*), que os roteadores utilizam para identificar cada reserva.
2) A primeira função da mensagem PATH é instalar um estado de roteamento reverso em cada roteador, ao longo da rota, e segundo fornecer ao receptor toda a informação sobre as características da origem e fornecer a rota fim-a-fim para que a reserva possa ser realizada [31].
3) Cada roteador, suportando o protocolo RSVP, estabelece um estado de rota que inclui o endereço de origem da mensagem PATH, isto é, o próximo salto em direção ao transmissor.
4) O receptor é responsável por confirmar e estabelecer oficialmente a reserva e propaga em direção ao transmissor, realizando a reserva em cada roteador ao longo do caminho. Quando cada roteador, ao longo da rota, recebe a mensagem RESV, ele usa um processo de controle de admissão para autenticar a solicitação e alocar os recursos necessários. Se a solicitação não pode ser satisfeita (problemas de falta de recursos ou falhas na autenticação), o roteador retorna uma mensagem de erro ao receptor. Caso contrário, envia uma

mensagem RESV ao próximo roteador. Quando o último roteador receber a mensagem RESV e aceitar a solicitação, ele envia uma mensagem de confirmação ao receptor . Observe que o último roteador é aquele que se localiza o mais próximo do transmissor.

Existem dois estilos de protocolos de reserva : o tipo *"hard state"* (HS), também denominado de "orientado à conexão" e o *"soft state"* (SS), também chamado de "não orientado à conexão" [30]. Nos dois tipos, a distribuição multidestino (*multicast*) é realizada usando um estado específico de fluxo em cada roteador ao longo da rota, onde no tipo HS, este estado é criado e removido de forma determinística em cooperação com todos os roteadores envolvidos (exemplo protocolo ST-II *Revised Internet Stream Protocol*) [52]. O protocolo RSVP é do tipo *soft-state* (SS), que usa o estado de reserva armazenado em memória *cache* e é periodicamente atualizada pelas máquinas. Estados que não são usados por um determinado tempo são descartados pelos roteadores [53]. Se a rota muda, as mensagens de atualização instalam automaticamente um novo estado para esta nova rota.

FIGURA 11. PROTOCOLO DE RESERVA DE RECURSOS



A figura 11 ilustra de forma esquemática a transmissão da mensagem PATH contendo a especificação do tráfego do fluxo e a recepção da mensagem RESV. A especificação sobre o fluxo é levada pelo protocolo de sinalização de reserva de recursos, passa pelo controle de admissão, pelo teste de aceitação, e finalmente usa estes parâmetros para configurar o escalonador de pacotes.

3.1.6. Classes de Serviço da Arquitetura IntServ

O modelo de serviços integrados IntServ permite a integração de serviços QoS com os serviços de melhor esforço. O grupo de trabalho de Serviços Integrados do IETF definiu duas classes de serviço: serviço garantido [54] e serviço de carga controlada [55]. Ambos suportam a fusão de fluxos ou agregação de fluxos para melhorar a escalabilidade da rede.

O serviço garantido (*Guaranteed Service*) foi projetado para suportar exigências de aplicações em tempo real, já o serviço de carga controlada (*Controlled-load service*) fornece uma qualidade de serviço similar ao modelo de melhor esforço em uma rede sub-utilizada, sem perdas e atraso. Em caso de sobrecarga da rede, o serviço foi projetado para compartilhar a largura de banda de fluxos agregados de forma controlada.

3.1.6.1. Serviço Garantido

O serviço garantido foi projetado para aplicações intolerantes a atraso RTI (*Real Time Intolerant Applications*) que requerem uma determinada largura de banda mínima e um atraso máximo [31]. Os pacotes não são perdidos, nem experimentam atrasos que ultrapassam os limites estabelecidos. Este tipo de garantia é feito através da reserva de recursos na rede.

Para que um roteador possa invocar o serviço garantido para um determinado fluxo de dados, ele precisa ser informado sobre as características de tráfego do fluxo Tspec juntamente com as características de reserva de recursos Rspec. Nos pontos de acesso, o tráfego deve ser policiado para assegurar conformidade nas especificações de tráfego. Os pacotes que não estão em conformidade são roteados em regime de melhor esforço.

O serviço garantido garante que os datagramas sejam entregues ao destino dentro de um determinado intervalo de tempo e não sejam descartados em caso de congestionamento, isto se o fluxo em questão estiver em conformidade.

3.1.6.2. Serviço de Carga Controlada

Diferente do serviço garantido, o serviço de carga controlada não estabelece garantias quantitativas. O roteador deve receber as especificações de tráfego Tspec para o fluxo, da mesma forma que é feito no serviço garantido, porém não é necessário incluir a taxa de pico. Se o fluxo for aceito, então o roteador compromete-se a oferecer ao fluxo um serviço equivalente ao serviço de melhor esforço, sem congestionamento na taxa combinada.

A diferença mais importante entre os serviços de melhor esforço e de carga controlada, é que o fluxo de carga controlada não se deteriora à medida que a carga da rede aumenta. Já para o serviço de melhor esforço, o fluxo experiementa uma degradação progressiva do serviço (mais atrasos e perdas de pacotes) à medida que a carga da rede aumenta.

Neste serviço de carga controlada, o atraso médio é garantido, mas o atraso fim-a-fim não pode ser determinado. O serviço pode ser implementado de várias formas distintas, uma delas através de algoritmos de escalonamento melhorados. Uma forma de se implementar o serviço de carga controlada para os fluxos regulados é através de mecanismos de escalonamento priorizados, onde a prioridade mais alta seria usada para o serviço de carga controlada e a prioridade menor para o serviço de melhor esforço. O algoritmo de controle de admissão pode ser usado para limitar a quantidade de tráfego posicionado na fila de alta prioridade, baseando-se nos parâmetros de especificação de tráfego do fluxo.

3.2. SERVIÇOS DIFERENCIADOS - DIFFSERV

O modelo de serviços diferenciados [56], também chamado de priorização ou "*Soft QoS*", foi proposto pelo IETF para resolver alguns problemas associados com o serviço integrado IntServ. As características mais importantes do serviço diferenciado, para resolver as limitações do IntServ são:

i) **Diferenciação Grossa (*Coarse Differentiation*):** o serviço diferenciado não diferencia o tráfego por fluxo, ao invés disto, trabalha com um pequeno número de classes bem definidas.

ii) **Classificação de pacotes nas bordas da rede:** Na reserva de recursos realizada pelo protocolo de sinalização RSVP, cada roteador tinha que classificar o pacote para fornecer diferentes níveis de serviço. Para tornar a solução mais escalável, a classificação de pacotes no modelo DiffServ foi alterada para as bordas do domínio. Os roteadores de borda classificam e marcam os pacotes adequadamente. Os roteadores internos simplesmente processam os pacotes baseados nestas marcações, ou seja, os roteadores internos não reconhecem fluxos individuais, eles entendem apenas classes de fluxos agregados. A separação da política de controle e de funções de suporte permite uma evolução independente de cada mecanismo. A maior desvantagem é que o roteador interno não efetua controle de admissão, o que pode causar situações de sobrecarga temporárias na rede em conexões de baixa banda.

iii) **Provisionamento estático:** IntServ requer reserva de recursos e controle de admissão dinâmicos em cada roteador, o que pode gerar em um número muito grande de pacotes de controle. Já no modelo DiffServ, o controle de admissão também foi transferido para as bordas da rede, e além disso não requer reservas dinâmicas, e sim um provisionamento estático a longo prazo para estabelecer acordos de serviço com os usuários.

iv) **Sem garantias absolutas:** o objetivo primordial do modelo DiffServ é monitorar o tráfego entrante no domínio, no nó de entrada, e verificar a conformidade comparando com perfis de serviço pré-definidos. Baseado nisto, os pacotes são marcados como "dentro" ou "fora" da conformidade. Os pacotes marcados com "fora" são descartados pelos roteadores.

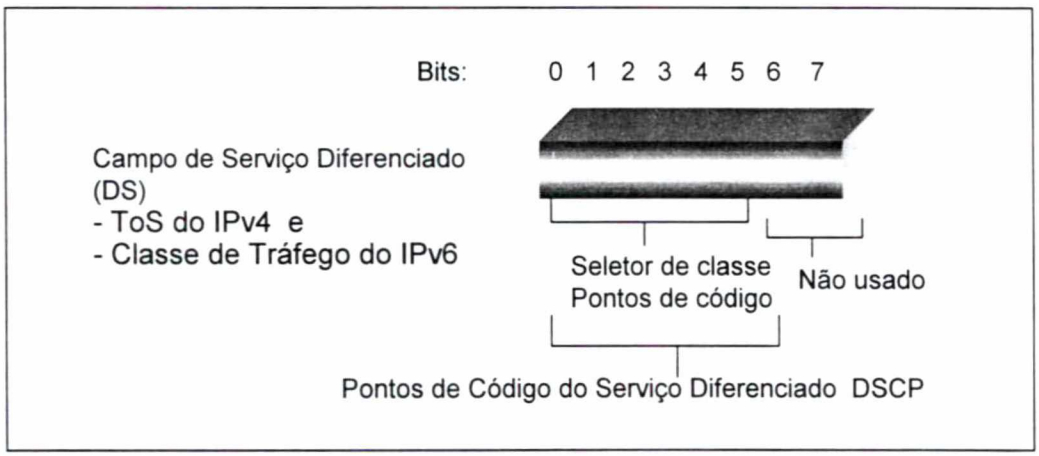
v) **Mais informações no cabeçalho:** o cabeçalho dos pacotes contém informações adicionais como tipo de serviço TOS. Como exemplo de

aplicação, o serviço prêmio pode ser oferecido como um equivalente à uma conexão do tipo CBR (*Constant Bit Rate*) de taxa constante do ATM. Outro exemplo seria um serviço com comportamento em rajadas, no qual sua alocação de banda é realizada dinamicamente.

O serviço diferenciado DiffServ é um modelo de discriminação de serviço de classe agregada (*per-aggregate-class*), pois executa uma classificação grossa do nível de tráfego [57] [58].

DiffServ assume a existência de um contrato de serviço SLA entre as redes que compartilham uma fronteira [32]. O contrato SLA estabelece critérios e políticas, além de definir o perfil do tráfego. Estima-se que o tráfego seja policiado nos pontos de entrada e qualquer tipo de tráfego fora das conformidades com o SLA (p.ex.: acima dos limites superiores estipulados) não receba nenhuma garantia de QoS.

FIGURA 12. PONTOS DE CÓDIGO DO SERVIÇO DIFERENCIADO [57]



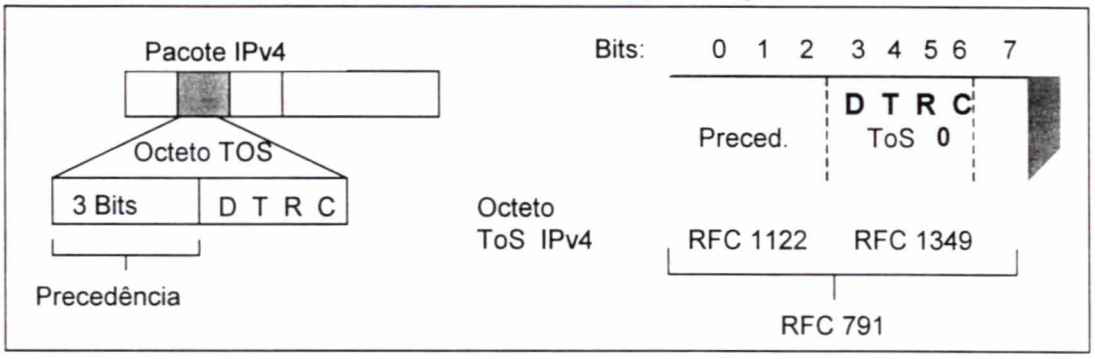
A figura 12 mostra os pontos de código do serviço diferenciado DSCP (*Differentiated Service Code Points*). Eles redefinem o octeto tipo de serviço do IPv4, mas os bits de prioridade são preservados nos pontos de código do seletor de classe.

O modelo é baseado em comportamento por salto PHB, onde datagramas são marcados com pontos de serviço, ou seja, sem a necessidade do estado ser gerenciado fluxo a fluxo e realizar sinalização a cada salto [59]. Os serviços podem

ser fim-a-fim ou intradomínio ou então uma combinação dos dois.

Os campos do serviço diferenciado para o IPv4 corresponde ao campo TOS e no IPv6 ao octeto classe de tráfego [57]. A figura 13 ilustra a estrutura para o IPv4:

FIGURA 13. OCTETO TIPO DE SERVIÇO



O octeto TOS [19] é dividido em dois campos: o campo precedência de 3 bits e o campo tipo de serviço. O campo precedência mostra a prioridade e o tipo de serviço indica o roteamento, onde os protocolos de roteamento devem calcular uma rota quando um bit ToS for configurado. No caso, quando o bit D de *delay* for setado, calcula-se a rota mais curta, quando o bit T de *throughput* é configurado, calcula-se a rota com o maior *throughput*, se R de *Reliable* estiver setado, calcula-se a rota mais confiável e finalmente se C de *Cost* estiver setado, a rota de menor custo e o último bit não é usado.

Os bits em ToS são uma forma que o usuário tem de informar à rede sobre o serviço que precisa, se é tempo real por exemplo, baixo atraso e alta prioridade.

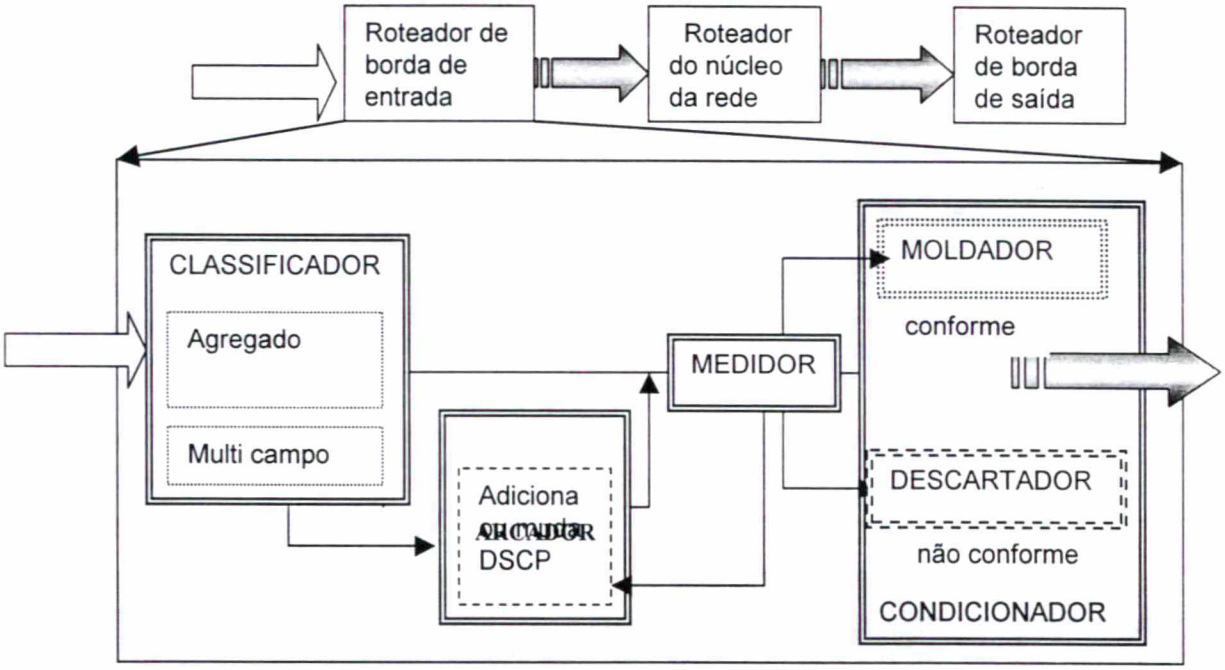
3.2.1. Comportamento por Salto (*Per Hop Behavior*)

O comportamento por salto PHB [56] corresponde à descrição do comportamento do roteamento, por um observador externo, do nó quando aplicado a um fluxo agregado. Os roteadores do núcleo da rede (*core routers*) no modelo DiffServ precisam rotear apenas os pacotes, cujos comportamentos PHB tenham sido especificados. Se apenas um fluxo agregado ocupar um enlace, o comportamento do roteamento dependerá apenas do congestionamento deste enlace. Os PHBs

podem ser definidos em termos de recursos (buffer e banda) ou em termos de prioridade ou ainda em termos de métricas de tráfego, como atraso e perdas. PHBs são implementados nos nós através de mecanismos de gerenciamento de filas (*queuing*) ou de escalonamento de pacotes (*scheduling*).

A figura 14 ilustra os componentes da arquitetura do DiffServ:

FIGURA 14. ARQUITETURA DO SERVIÇO DIFERENCIADO



3.2.2. Classificador de Tráfego

Como já visto no capítulo anterior, os classificadores de pacotes utilizam a informação contida no cabeçalho do datagrama para selecioná-los. Os classificadores enviam os pacotes ao módulo de condicionamento para processamento e sua função além de classificar é autenticar os pacotes à medida que os seleciona. Existem dois tipos: os classificadores de agregados, que selecionam os pacotes baseados nos pontos de código DS (*differentiated service*) e os classificadores multi-campos, que selecionam os pacotes baseados em múltiplos valores de diferentes campos.

3.2.2.1. Condicionador de Tráfego

As quatro tarefas executadas pelo condicionador de tráfego são: medição do tráfego (*metering*), baseado em um mecanismo de balde de fichas, por exemplo, moldagem (*shapping*), policiamento (*policing*) e/ou marcação (*marking*). Finalmente é tomada uma ação para os pacotes que não estão em conformidade com o perfil de tráfego contratado (taxa de transmissão ou rajada maior que o permitido). A ação depende do serviço oferecido, mas, em geral, pode ser o descarte, a suavização (espera em uma fila) ou a remarcação para outro DSCP.

Em suma, o condicionador (*conditioner*) assegura que o tráfego entrando no domínio do serviço diferenciado DS esteja em conformidade com o contrato interdomínios, ou seja, entre o domínio de origem e de destino.

3.2.2.1.1. Medição

O condicionador recebe os pacotes do classificador e usa um "medidor" para medir as propriedades temporais (*temporal properties*) do fluxo e compará-las com o perfil do contrato interdomínios. A especificação de tráfego já foi vista na seção anterior. O resto do processamento é realizado pelos marcadores, moldadores e módulos de policiamento baseados no perfil.

3.2.2.1.2. Marcação

Este módulo marca o pacote setando o campo DS para um valor código apropriado. Assim, o pacote fica pertencendo a um agregado. O marcador pode alterar a marcação de um pacote.

3.2.2.1.3. Moldagem

A modelagem de fluxos (*flow shaping*) regula os fluxos baseados nas especificações fornecidas pelo usuário sobre o desempenho esperado [46]. Ela pode

ser baseada em uma única taxa fixa de *throughput* (p.ex.: taxa de pico) ou algum outro tipo de representação estatística (p.ex.: taxa sustentável). A maior vantagem na modelagem de fluxos é que o tráfego será regulado de forma a não exceder os parâmetros de QoS contratados. Os pacotes podem ser descartados se houver sobrecarga do buffer.

3.2.2.1.4. Policiamento

O policiamento do fluxo (*flow policing*) observa se o nível de QoS contratado está sendo mantido pela provedora de serviços. Se o esquema de modelagem de fluxos foi bem feito na fonte, o mecanismo de policiamento pode facilmente detectar fluxos com comportamento duvidoso. A ação tomada pelo policiamento pode variar desde a aceitação de violações até uma notificação do usuário para efetuar uma renegociação da modelagem de fluxos para um nível de QoS aceitável.

3.2.2.1.5. Descarte

Como o próprio nome diz, descarta pacotes de um fluxo de tráfego para que o perfil fo tráfego esteja em conformidade com o especificado. Os descartadores podem ser implementados como um caso especial de um moldador com tamanho de buffer igual a 0. Pacotes que não estiverem dentro do perfil determinado podem ser descartados ou enviados à rede com precedência de descarte maior.

A técnica tradicional de escalonamento nos roteadores é estabelecer um comprimento máximo (em termos de pacotes) para cada fila e aceitar apenas os pacotes que estão na fila e rejeitar (descartar) os pacotes subsequentes, até a diminuição da fila. Esta técnica é conhecida como "descarte da cauda" (*tail drop)*, assim chamada, pois o pacote que chegar e estiver posicionado na cauda da fila será descartado quando a fila estiver cheia. Em casos de congestionamento, a Internet tem usado por muitos anos este mecanismo, ou seja, a forma como os roteadores devem descartar os pacotes quando a fila fica totalmente cheia.

O resto desta seção foi baseado em [2] e [47].

O descarte de um pacote pode ser interpretado de várias formas: no contexto da Internet, atualmente, o descarte de pacotes é visto pelo TCP como um sinal de congestionamento e, assim quando configurado reduz automaticamente sua carga na rede. Já no contexto de serviços em tempo real, o descarte do pacote está relacionado como uma ação necessária para manter a qualidade de serviço, ou seja, há a redução de atrasos na rede. O fracasso de um pacote pode contribuir para o sucesso de muitos [30].

Descarte da Cauda (*Tail Dropping*)

Como normalmente ocorre, os pacotes permanecem nas filas a espera da transmissão. O descarte da cauda simplesmente descarta os pacotes entrando na rede se a fila estiver cheia. Quando o congestionamento é eliminado, e as filas voltam a ter espaço os pacotes voltam a ser enfileirados. A maior desvantagem deste tipo de descarte é o sincronismo global do TCP, pois todas as máquinas enviam no mesmo instante e param no mesmo instante [2], isto pois os pacotes são descartados de várias máquinas no mesmo momento.

Descarte Randômico Antecipado (*Random Early Dropping*)

O descarte radômico antecipado RED é um algoritmo de gerenciamento de fila ativo. Segundo [44], RED foi projetado para minimizar o comprimento das filas dos roteadores pelo descarte inteligente de pacotes. O algoritmo RED objetiva controlar o tamanho médio da fila, indicando aos terminais quando eles devem diminuir temporariamente a transmissão de pacotes.

Ao contrário dos algoritmos de gerenciamento de filas tradicionais, que descartam os pacotes apenas no momento em que o buffer está cheio, o algoritmo RED descarta os pacotes de forma probabilística, ou seja, a probabilidade do

descarte aumenta à medida que o tamanho médio da fila aumenta. Assim, se a fila estava praticamente vazia no passado recente, o RED não tentará descartar pacotes (a não ser que o buffer esteja cheio). Por outro lado, se recentemente a fila estava relativamente cheia (indicação de possível congestionamento), haverá uma probabilidade maior dos novos pacotes, entrando na rede, serem descartados.

O algoritmo RED consiste de duas partes: estimativa da média do tamanho da fila e decisão de descartar ou não um pacote entrante. A estimativa da média do tamanho da fila calcula o tamanho médio da fila, usando uma média ponderada exponencial ou, em segundo plano, ou seja, não efetuando o cálculo no caminho do roteamento, através de um cálculo similar. A decisão sobre o descarte do pacote corresponde à segunda parte do algoritmo. Esta parte do algoritmo é que resulta em um melhoramento do desempenho de resposta dos fluxos. Os dois parâmetros usados, limite mínimo (*minimum threshold*) e limite máximo (*maximum threshold*), são os limiares mais importantes no processo de decisão. Quando o tamanho da fila for maior que o parâmetro estipulado, o algoritmo RED inicia o descarte dos pacotes de forma randômica.

Descarte Randômico Antecipado Ponderado (*Weighted Random Early Dropping*)

O descarte randômico antecipado ponderado, ou WRED é uma estratégia de descarte randômico antecipado que, adicionalmente, descarta pacotes de baixa prioridade, quando a interface de saída começa a ficar congestionada.

Para ambientes que operam com serviços integrados IntServ, o algoritmo ponderado WRED descarta apenas pacotes que não sejam fluxos designados para efetuar a reserva de recursos e para o ambiente que opera com serviços diferenciados DiffServ, o algoritmo WRED analisa os bits de precedência para decidir as prioridades e então, seletivamente, iniciar o descarte.

Normalmente, o WRED é configurado nos roteadores no núcleo da rede (*core routers*).

3.2.3. Serviço *Default* (DE)

Também conhecido como serviço de melhor esforço. Configurado como serviço default. Todos os seis bits do DSCP são configurados para zero.

3.2.4. Encaminhamento Expedito (EF)

Encaminhamento expedito EF (*Expedited Forwarding*) [60] é definido como sendo um tratamento de roteamento particular para um determinado agregado, onde a taxa inicial, dos pacotes pertencentes ao agregado de qualquer nó, deve ser igual a taxa previamente configurada. Para diminuir as perdas, latência e *jiiter*, as filas são reduzidas na rede. Trata-se de um serviço prêmio com uma taxa quase constante de bits.

Na entrada da rede, o tráfego é condicionado (policiado e moldado). Os roteadores de borda de entrada na rede devem negociar uma taxa menor que a configurada com seus roteadores vizinhos.

O encaminhamento expedito possui um ponto de código (*codepoint*) específico e fornece o maior nível de QoS ao agregado. O valor DSCP é configurado para **101100,** ou seja, o controle de admissão é rígido e todos os pacotes em excesso são descartados.

3.2.5. Encaminhamento Assegurado (AF)

O encaminhamento assegurado AF (*Assured Forwarding*) [61] configura o tráfego conforme seu nível de prioridade, quatro classes e três prioridades de descarte para cada classe. Quatro classes e três prioridades de descarte por classe totalizam doze códigos distintos.

Para cada roteador DS, para cada classe atribui-se uma determinada quantidade de recursos, como banda e *jitter*. Durante os períodos de congestionamento, os pacotes que contiverem precedência de descarte maior terão uma probabilidade

maior de serem descartados. Porém, o tráfego de cada classe é encaminhado independentemente dos pacotes pertencentes a outras classes, ou seja, o nó DS não pode agregar mais de duas classes de tráfego juntas. Nas bordas do domínio DS, o tráfego pode ser condicionado.

3.3. COMUTAÇÃO MULTIPROTOCOLO POR RÓTULOS - MPLS

Os modelos propostos pela indústria, como comutação IP pela Ipsilon [62] e comutação por rótulos da Cisco [63], nortearam a formação do grupo multiprotocolo de comutação por rótulos MPLS em 1997 pela IETF, para estabelecer acordos comuns para a tecnologia. MPLS está sendo atualmente considerada a solução para prover QoS e como solução para engenharia de tráfego [64]. O termo "rótulo" (*label*) foi escolhido devido ao modo como o caminho é estabelecido: pacotes carregam rótulos que podem ser comutados de forma eficiente, pois o equipamento não precisa desempacotar todo o pacote para verificar o endereço de destino no cabeçalho. O termo "multiprotocolo" foi escolhido pois as técnicas de comutação podem ser aplicadas à qualquer protocolo de rede, como por exemplo o IP e IPX ou diretamente sobre a camada de enlace.

No roteamento IP convencional, o próximo salto do pacote é escolhido pelo roteador baseado na informação do cabeçalho do pacote. Na arquitetura MPLS a escolha da melhor próxima rota é composta de duas funções: (a) divisão do conjunto de pacotes em classes de encaminhamento equivalentes, denominadas de FEC (*Forwarding Equivalence Classes*); (b) mapeamento de cada FEC ao próximo salto (roteador ou comutador).

O conceito chave da técnica multiprotocolo de comutação por rótulos MPLS é a separação das funcionalidades do roteador IP em duas partes: encaminhamento e controle [66].

A arquitetura MPLS combina a comutação de circuitos da Rede ATM (por exemplo) e o roteamento de pacotes da rede IP. Na entrada da rede, ou seja, do

domínio MPLS, os pacotes são rotulados (rótulos determinam a QoS) e são comutados pelos rótulos no núcleo da rede. Na saída do domínio, os rótulos são retirados e os pacotes são roteados de forma convencional. Trata-se portanto de uma tecnologia híbrida, onde a informação é comutada de forma rápida no núcleo da rede e nas bordas do domínio efetua-se o roteamento convencional.

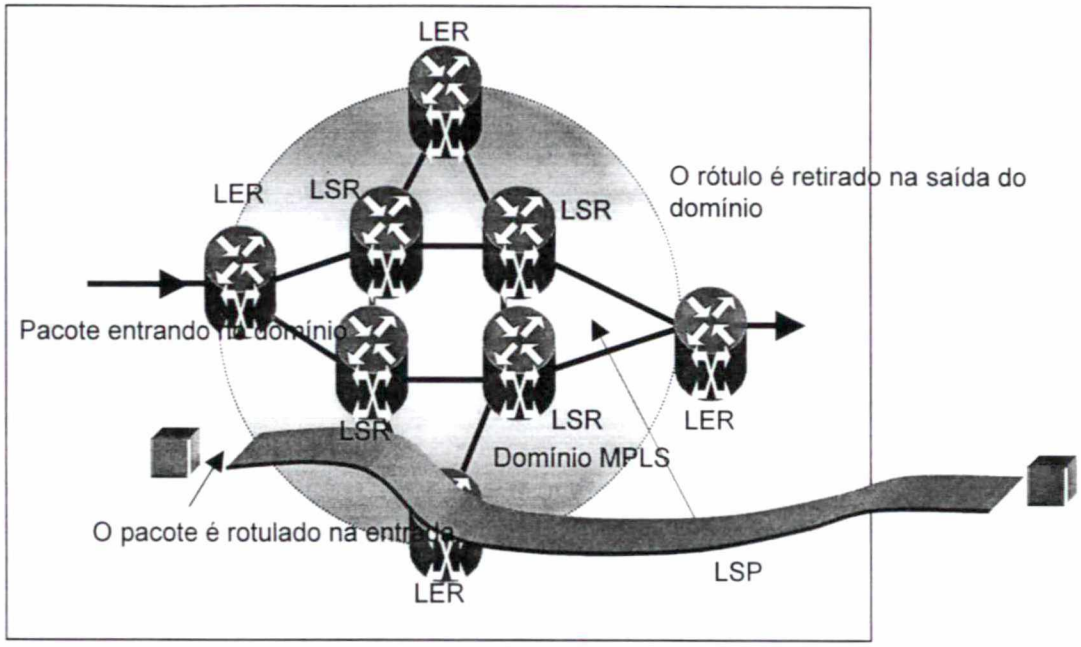
As principais motivações para o desenvolvimento desta arquitetura foram: capacidade de expansão melhorada, ou escalabilidade, desempenho mais rápido no encaminhamento da informação e engenharia de tráfego.

Bruce Davie cita em seu livro [67] que vários fatores se encontram por detrás da tecnologia multiprotocolo de comutação por rótulos MPLS e que qualidade de serviço QoS não foi a razão mais forte. Porém, tem sido usada como modelo arquitetônico de resolução de QoS.

3.3.1. Componentes do Modelo MPLS

A comutação multiprotocolo por rótulos MPLS é parecida, de certa forma, com o Serviço Diferenciado DiffServ, no sentido de que o MPLS também marca o tráfego nas bordas do domínio e o desmarca na saída [68]. Mas, DiffServ usa a marcação para determinar a prioridade do tráfego no roteador, já na arquitetura MPLS as marcações (valores de comprimento fixo de 20 bits) são primordialmente designados para determinar o próximo roteador. MPLS não pode ser controlado pelas aplicações, pois reside unicamente nos roteadores. A figura 15 mostra os componentes da arquitetura.

FIGURA 15. COMPONENTES DA ARQUITETURA MPLS



LER Roteador de rótulo de borda LER (*Label Edge Router*), equipamento situado nas bordas do domínio capaz de utilizar as informações de roteamento para atribuir rótulos aos datagramas e encaminhá-los dentro do domínio MPLS.

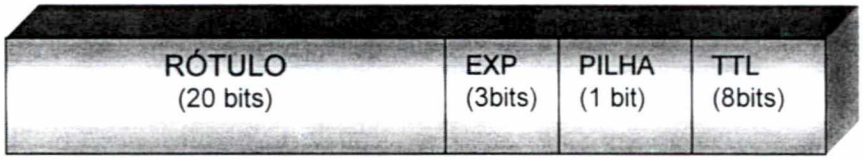
LSR Roteador de comutação de Rótulo LSR (*Label Switch Router*), equipamento localizado no núcleo da rede, capaz de comutar pacotes baseados em seus rótulos. Em muitos, casos o LSR é um comutador ATM modificado.

LSP Caminho comutado por rótulos LSP (*Label Switch Path*), é um caminho específico pelo qual o datagrama é transportado através da rota estabelecida pelos rótulos.

O pacote MPLS possui um cabeçalho que fica entre o cabeçalho da camada de enlace e o cabeçalho da camada de rede. O cabeçalho MPLS compreende um rótulo de 20 bits, um campo denominado experimental, antigamente denominado de classe de serviço CoS de 3 bits, um campo de indicação da pilha de 1 bit (*stack indicator*) e um campo de tempo de vida TTL (*Time to Live*) de 8 bits [68].

O rótulo (*label*) é curto, de tamanho fixo e de significância local, usado para identificar uma classe de encaminhamento equivalente FEC, baseado totalmente ou parcialmente no endereço de destino da camada de rede. É de responsabilidade de cada roteador de comutação de rótulos certificar-se que os rótulos de entrada são únicos. A figura 16 ilustra o formato do cabeçalho MPLS:

FIGURA 16. FORMATO DO CABEÇALHO MPLS



Quando um pacote entra no domínio MPLS, a ele é atribuído um rótulo MPLS, que especifica o caminho que o pacote tomará dentro da rede. A partir dos roteadores comutadores, o pacote é comutado para a interface de saída baseado apenas na informação contida no rótulo. O campo classe de serviço COS é usado para escolher a fila adequada na interface de saída, ou seja, especificar o nível de qualidade de serviço.

Na saída do domínio, o cabeçalho MPLS é removido e a partir de então o pacote é roteado de forma tradicional.

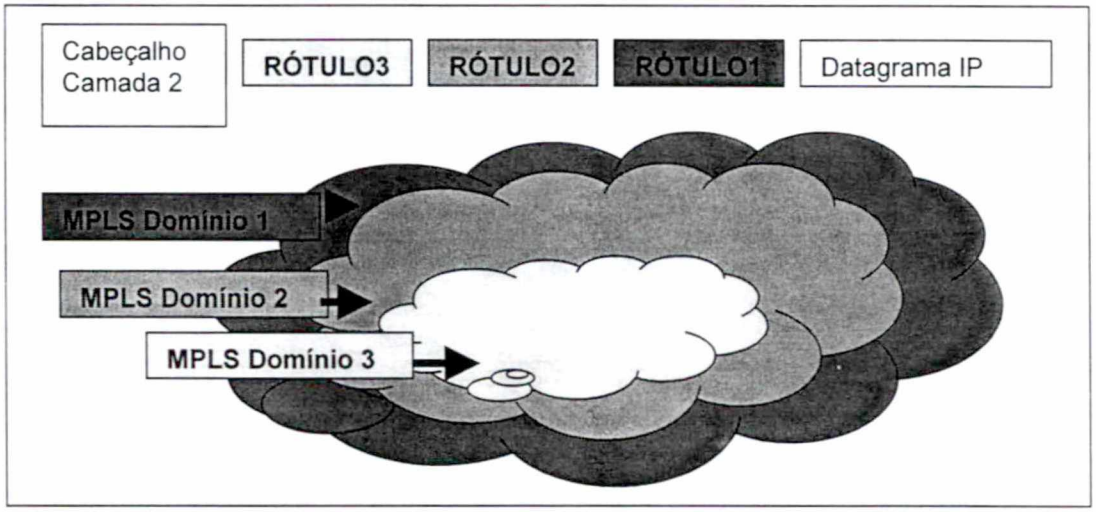
3.3.2. Pilha de Rótulos (*Label Stack*)

Pode ser que ao invés do cabeçalho carregar consigo um único rótulo, o pacote pode transportar vários rótulos, organizados como LIFO (*Last In First Out*), ou seja, o último que entra é o primeiro que sai. Isto é denominado de pilha de rótulos (*label stack*) [69]. O processamento de um pacote rotulado é completamente independente do nível de hierarquia e é sempre baseado no rótulo de cima. Um pacote não rotulado pode ser interpretado como sendo um pacote cuja pilha de rótulos esteja vazia [68].

Os rótulos podem estar encapsulados ou marcados dentro do cabeçalho do

pacote. No roteador os rótulos são numerados de dentro para fora , ou seja, da parte inferior da pilha à parte superior. A pilha de rótulos é importante na implementação de túneis e para determinar hierarquias nos domínios por onde a informação trafega. A figura 17 ilustra três domínios, neste caso, a pilha de rótulos contém três rótulos diferentes, que identificam os domínios.

FIGURA 17. PILHA DE RÓTULOS



Os principais componentes da pilha de rótulos (*label stack*) são [65]:

3.3.2.1. Entrada de Rótulo do Próximo Salto - NHLFE

A entrada de rótulo do próximo salto NHLFE (*Next Hop Label Forwarding Entry*) é usada para encaminhar um pacote rotulado e contém o próximo salto do pacote ou então o tipo de operação a ser executada na pilha de rótulos. As operações podem ser resumidas em [68]: a) Substituir o rótulo de cima por um rótulo novo; b) desempilhar o rótulo da pilha (*pop*); c) empilhar o rótulo de cima por um rótulo novo (*push*) ou mais rótulos novos sobre a pilha; d) Encapsular a camada de enlace de dados para transmitir o pacote; e) Codificar a pilha de rótulos para transmitir o pacote.

3.3.2.2. Mapeamento do Rótulo de Entrada - ILM

rótulo de entrada de um pacote para um conjunto de entradas NHLFE para determinar o próximo salto NHLFE [68].

3.3.2.3. Mapeamento FEC para NHLFE - FTN

O mapeamento da classe equivalente FEC à tabela de entrada do próximo salto NHLFE é denominado de mapeamento FTN (*FEC to NHLFE*). Este mapeamento é usado para rotular e encaminhar pacotes não rotulados [68].

3.3.3. Encaminhamento dos Pacotes

O pacote rotulado é um pacote no qual codificou-se um rótulo. Em alguns casos, os rótulos residem em um cabeçalho de encapsulamento, que existe unicamente com este propósito. Em outros casos, o rótulo pode residir na camada de enlace de dados ou até mesmo no cabeçalho da camada de rede [68]. Quando o roteador de comutação de rótulo LSR recebe o pacote, utiliza o mapeamento ILM para mapear o rótulo ao NHLFE, e com a informação obtida de NHLFE para executar operações na pilha de rótulos e finalmente encaminhar o pacote ao próximo salto, como especificado na tabela NHLFE .

Os pacotes rotulados também possuem um campo definido como tempo de vida do pacote TTL, que funciona essencialmente da mesma maneira que o campo TTL do cabeçalho IPv4, ou seja, em cada salto o valor é decrementado de uma unidade até zero. O pacote MPLS será descartado se seu rótulo for inválido [68].

Caso um roteador de comutação de rótulos LSR receba um pacote não rotulado, ele procurará no cabeçalho informações para poder mapeá-lo a uma classe equivalente FEC, e então usará a FTN para encontrar uma entrada à tabela NHLFE. Esta informação então será usada para encaminhar o pacote ao próximo salto.

3.3.4. Métodos de retenção de Rótulos

Quando um roteador de comutação de rótulos LSR recebe um rótulo para uma determinada classe FEC de outro roteador de comutação de rótulos, então ele pode escolher em manter o relacionamento do rótulo (*label binding*) ou descartar este relacionamento. Existem dois métodos para trabalhar com os rótulos: o modo de retenção liberal, onde os roteadores comutadores de rótulos LSRs, optam por manter o relacionamento entre os rótulos e as classes de encaminhamento [68]. O modo de retenção conservador, onde deve-se estabelecer sempre um novo relacionamento. Os rótulos são liberados no momento em que são recebidos.

3.3.5. Encapsulamento de Rótulos

A escolha da técnica de codificação do rótulo depende do tipo de equipamento usado para encaminhar pacotes rotulados. Se for usado um hardware e software específicos da comutação multiprotocolo por rótulos MPLS para encaminhar os pacotes, a melhor forma de codificar a pilha de rótulos é através de um cabeçalho sanduíche (*shim header)*, inserido entre os cabeçalhos da camada de enlace e da camada de rede. Este cabeçalho sanduíche é apenas um encapsulamento do pacote da camada de rede e que independe de qualquer tipo de protocolo, por isso denominado também de "encapsulamento MPLS genérico" [68].

De forma geral, a arquitetura MPLS suporta caminho comutado por rótulos com diferentes tipos de codificação que serão descritos nas seções a seguir.

3.3.5.1. Encapsulamento ATM

Existem três formas de codificar os rótulos em cabeçalhos de células ATM, presumindo o uso da camada de adaptação AAL5 [70]:

Codificação comutada por circuito virtual (*Switched Virtual Circuit*): usa os campos VPI (*virtual path identifier*) e VCI (*virtual circuit identifier*) para codificar o

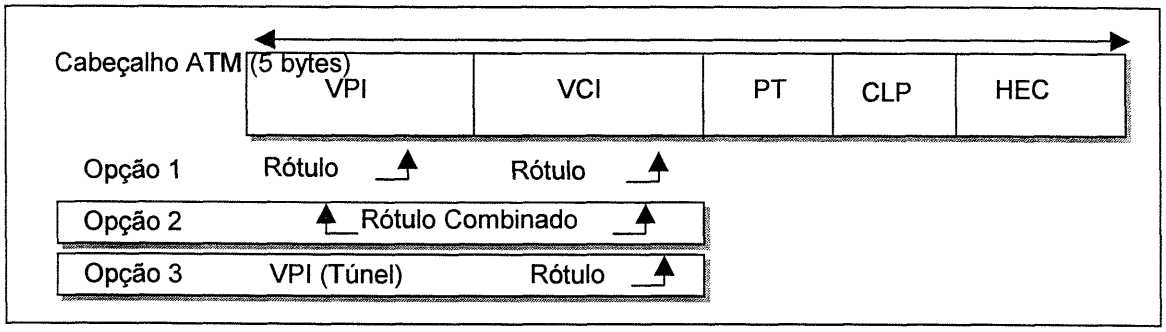
rótulo no topo da pilha de rótulos. Nesta técnica, usada em qualquer tipo de rede, cada caminho comutado por rótulo é estabelecido através de circuitos virtuais comutados. A restrição é que os roteadores de comutação de rótulos ATM LSR não conseguem executar as operações "empilhar e desempilhar " rótulos da pilha.

Codificação comutada por caminho virtual SVP (*Switched Virtual Path*): usa o campo VPI para codificar o rótulo, no topo da pilha, e o campo VCI para codificar o segundo rótulo na pilha. Esta técnica possui algumas vantagens, permite o uso da comutação de caminhos virtuais no ATM, ou seja, os caminhos comutados por rótulos LSP são estabelecidos através da comutação de caminhos virtuais, usando o o protocolo de distribuição de rótulos LDP como protocolo de sinalização do ATM.

Codificação comutada por caminho virtual multidestino: usa o campo VPI para codificar o rótulo no topo da pilha, parte do campo VCI para codificar o segundo rótulo na pilha e, caso necessário, usa o restante do campo VCI para identificar o nó de entrada do caminho comutado por rótulo LSP. Caso houver mais rótulos que devam ser codificados, então usa-se o cabeçalho MPLS genérico.

A figura 18 mostra as três codificações que o ATM usa para formar um rótulo:

FIGURA 18. ENCAPSULAMENTO ATM [71]



Nesta seção faz-se necessário descrever resumidamente a evolução que o protocolo IP teve sobre ATM, que é um protocolo usado em backbones de alta velocidade e que previu na sua concepção níveis diferenciados de QoS em suas camadas de adaptação. Surgiram duas propostas: modelo *Overlay* e o modelo *Peer*.

a) Modelo *Overlay*

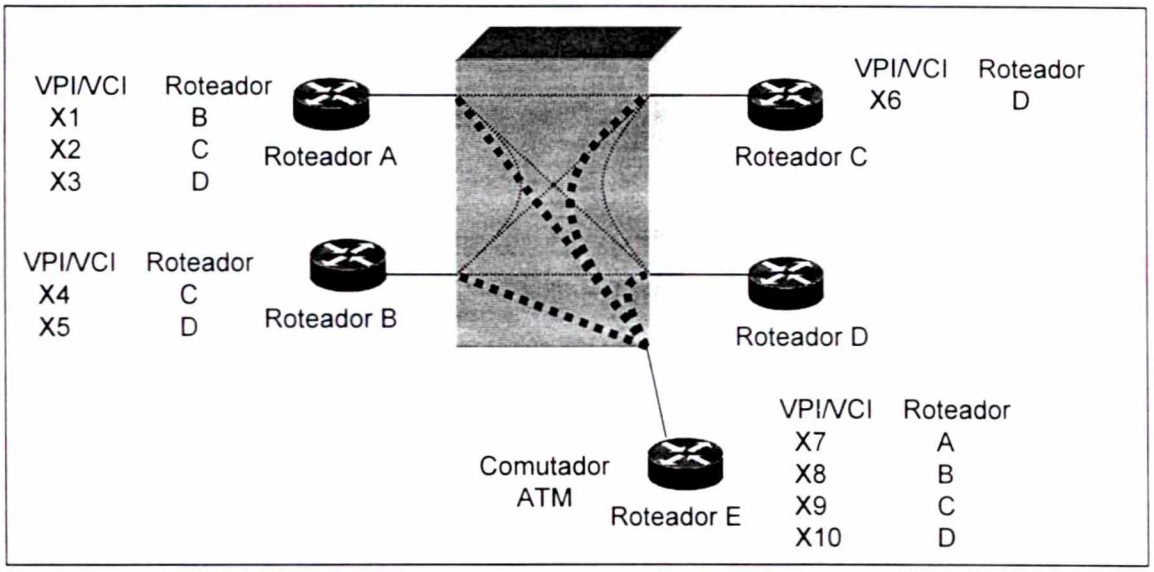
No modelo *Overlay* o protocolo IP é transportado sobre o protocolo ATM através da criação de uma ponte transparente (*flat bridging*) [72] ou na criação de uma rede de malha *overlay* [73]. Este modelo determina como os datagramas e pacotes IP devem ser encapsulados sobre o enlace ATM. A recomendação RFC 1577 da IETF foi a primeira tecnologia padronizada, mais conhecida atualmente como IP Clássico sobre CLIP (*Classical IP e ARP over ATM*) [74]. Esta recomendação descreve a emulação de uma sub-rede IP em uma infra-estrutura ATM.

Neste modelo introduziu-se a concepção de sub-redes IP lógicas LIS (*Logical IP Subnets*) [67], que são interconectadas através de roteadores e o tráfego inter-LIS passa por eles, mesmo havendo um caminho ATM direto entre a fonte e o destino. Para que dois roteadores IP possam estabelecer conexões uns com os outros, há a necessidade de resolver os endereços IP para ATM. Isto é feito através do mecanismo de resolução de endereços ATM. Para reduzir uma etapa no processo de estabelecimento de uma conexão ATM e minimizar o tráfego de difusão na sub-rede, a recomendação RFC 2225 [75] define os procedimentos de conversão de um endereço IP ARP (*Address Resolution Protocol*) estendido diretamente para endereços ATM, através de um servidor denominado de ATMARP (*ATM Address Resolution Protocol*), que responde às solicitações das máquinas sobre os endereços ATM de outras máquinas, que estejam diretamente conectadas à rede ATM através de uma sub-rede IP lógica LIS.

Como os roteadores IP são mais lentos que os comutadores ATM, as operadoras costumam posicionar todos os roteadores dentro de uma mesma LIS para minimizar os saltos IP/ATM. Como conseqüência disto, houve problemas de escalabilidade, pois o número de circuitos virtuais VC (*Virtual Channels*) e o número de nós interiores resultaram na conexão dos roteadores entre si, gerando uma malha completa de conexões [21]. A figura 19 ilustra o problema da malha completa no modelo *overlay*. Suponha quatro roteadores pertencentes à uma sub-rede, para interconectá-los há a necessidade de 6 conexões virtuais. Se mais um roteador

entrar na rede, como é o caso do roteador E, o número de conexões aumenta para 10. Como o número de conexões aumenta a uma ordem de n(n-1)/2 com a introdução de um novo roteador, por isso o problema ficou conhecido como sendo problema N^2.

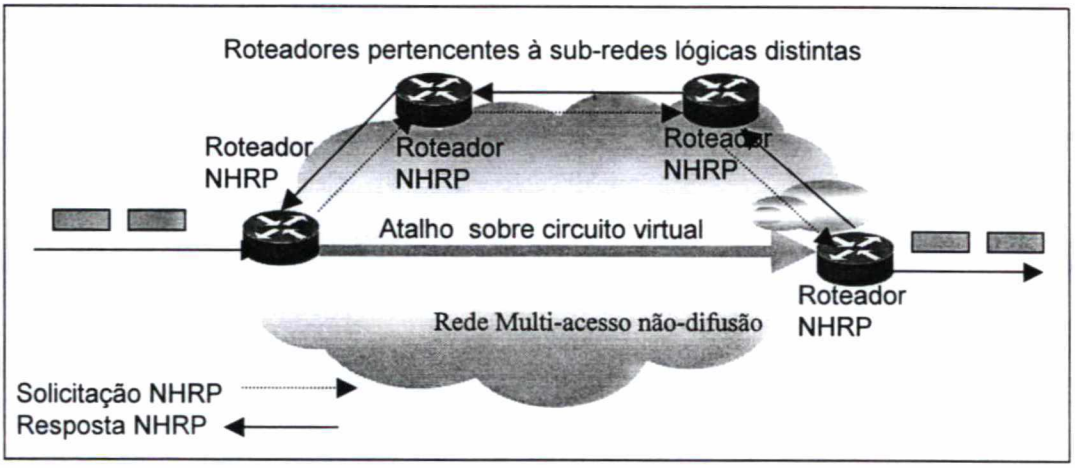
FIGURA 19. MALHA COMPLETA DO MODELO *OVERLAY*



Em configurações com múltiplas LIS, a recomendação RFC2332 [76] definiu um protocolo de resolução de endereço do próximo salto NHRP (*Next Hop Resolution Protocol*), que especifica que um roteador IP pode determinar o endereço ATM de uma máquina pertencente à outra LIS através da utilização de um ou mais servidores de próximo salto NHS (*Next Hop Server*) [77]. A figura 20 mostra um exemplo de solicitação do próximo salto pelos roteadores pertencentes a múltiplas LIS.

A estação fonte determina o próximo salto para a máquina de destino através de processos de roteamento normais, se a informação sobre o endereço destino já estiver disponível na memória cache do servidor. Caso contrário, o servidor cria um pacote de solicitação de resolução NHRP, contendo o endereço de rede do destino.

FIGURA 20. RESOLUÇÃO DO PRÓXIMO SALTO NHRP



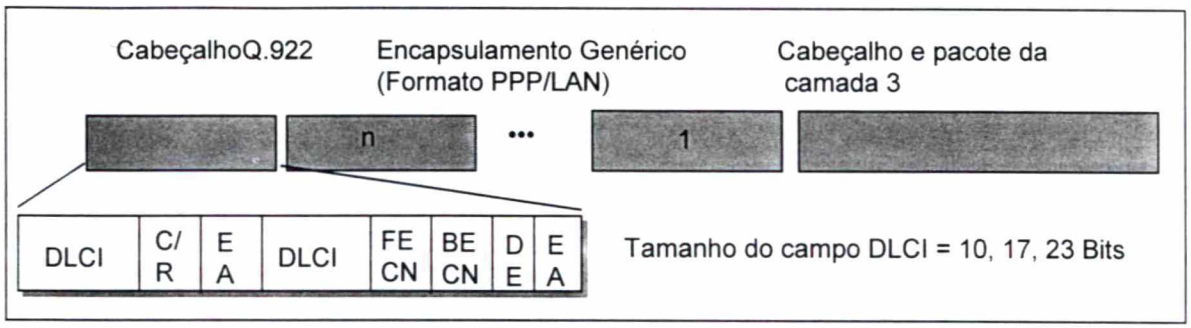
b) Modelo *Peer*

Já no modelo *Peer* o problema N^2 é resolvido, pois não há necessidade de se criar uma malha completa de conexões semipermanentes entre os roteadores. Existe apenas uma conexão do roteador com o comutador que estabelece a conexão com os outros roteadores comutando rótulos, ou seja, os pacotes IP são roteados na bordas do domínio e comutados no núcleo da rede. Um exemplo deste modelo é a arquitetura MPLS.

3.3.5.2. Encapsulamento Frame Relay

O valor usado pelo frame relay para codificar rótulos é o campo DLCI, situado no cabeçalho do quadro Frame Relay. Pode usar 2 ou 4 octetos do endereço Q.922 (10, 17, 23 bytes). A recomendação RFC 2427 [78] descreve a interconexão de múltiplos protocolos sobre frame relay. A figura 21 mostra o encapsulamento do frame relay em MPLS.

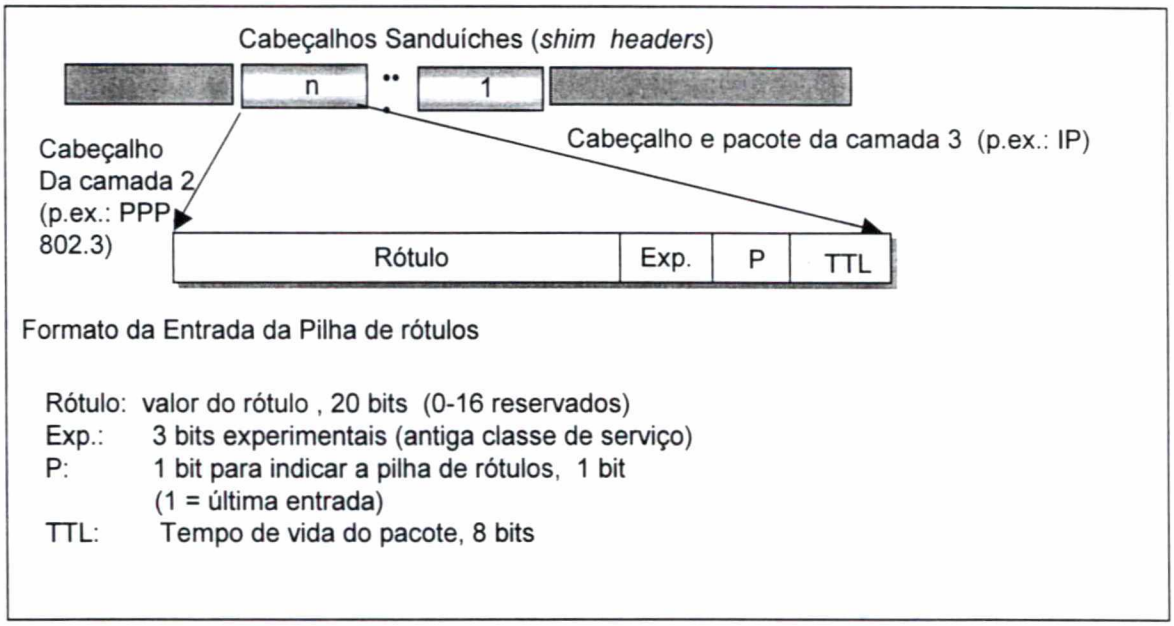
FIGURA 21. ENCAPSULAMENTO FRAME RELAY [71]



3.3.5.3. Encapsulamento - PPP & LAN

A arquitetura MPLS sobre enlaces PPP (*point-to-point protocol*) e LANs (*Local Area Network*) usa o cabeçalho sanduíche, inserido entre os cabeçalhos das camadas 2 e 3, como pode ser visto na figura 22.

FIGURA 22. ENCAPSULAMENTO PPP E LAN [71]



3.3.6. Distribuição do Rótulo

Um aspecto mais complexo da arquitetura MPLS envolve a distribuição e gerenciamento dos rótulos entre os roteadores.

O protocolo de distribuição de rótulos LDP (*Label Distribution Protocol*) é composto de uma série de procedimentos , nos quais o roteador de comutação de rótulos LSR informa outro LSR sobre seu relacionamento rótulo/classe. Dois roteadores LSR que trocam informações sobre estes relacionamentos são denominados nós de distribuição de rótulos (*Label Distribution Peers*), portanto um LSR é considerado como sendo seu adjacente [68] .

O protocolo de distribuição de rótulos foi projetado especificamente com este propósito, mas a arquitetura não assume a existência de um protocolo em especial, sendo que vários protocolos podem ser usados para servirem de distribuidores de rótulos: BGP e RSVP, são exemplos.

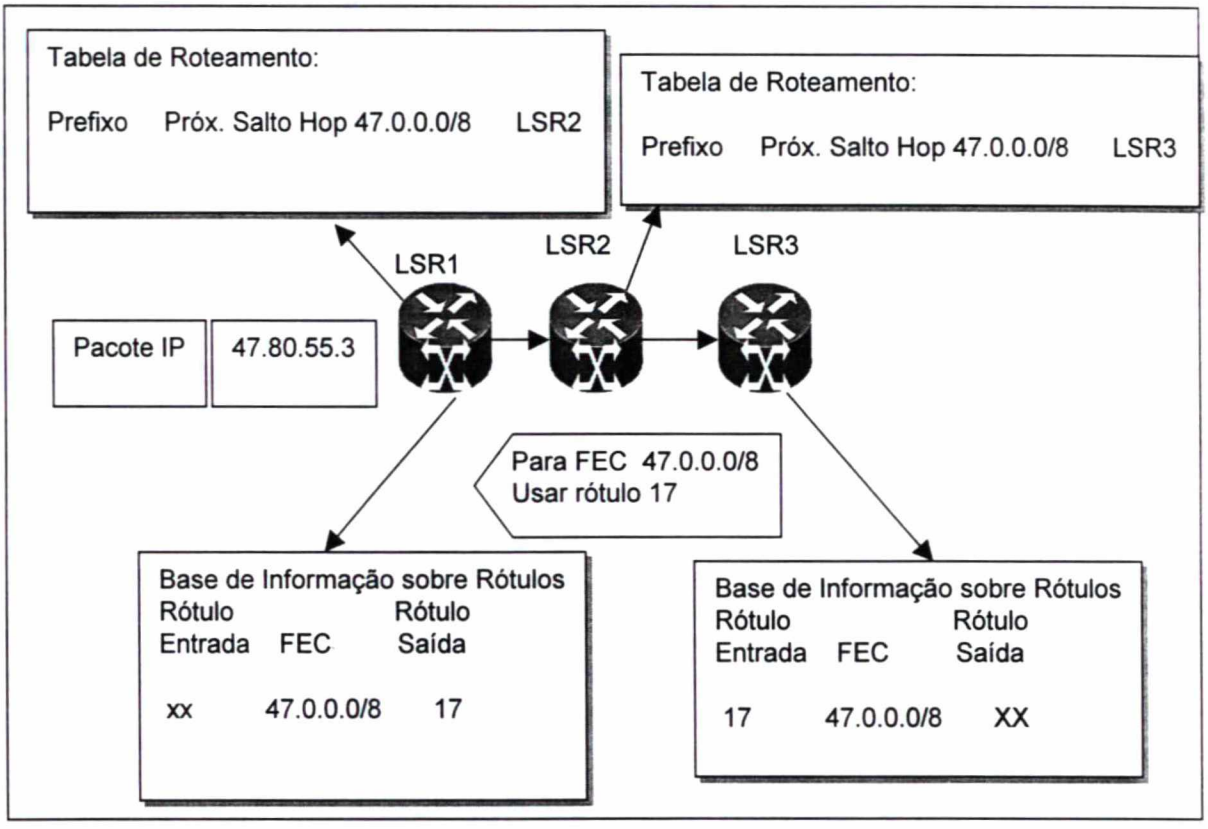
MPLS suporta duas formas de distribuir rótulos em diferentes níveis de hierarquia: distribuição explícita (*Explicit Peering*) e distribuição implícita (*Implicit Peering*). A distribuição explícita distribui os rótulos para um nó através de mensagens de distribuição de rótulos, endereçadas ao nó em questão. Já a distribuição implícita não envia mensagens, mas envia o rótulo a um nível hierárquico superior, este aos nós de distribuição remotos. Estes por sua vez distribuem o rótulo aos nós de distribuição locais, que então propagam a informação. O processo persiste até que a informação alcance seu nó remoto.

3.3.6.1. Através do Protocolo de Distribuição de Rótulos - LDP

A distribuição dos rótulos ocorre para que os roteadores adjacentes possuam a mesma visão sobre o relacionamento rótulo x classe equivalente. O procedimento ocorre da seguinte forma: o LSR cria uma relação entre a FEC e o valor do rótulo; o LSR comunica o LSR adjacente esta relação e o LSR introduz o valor do rótulo em sua tabela.

Quando um roteador de comutação de rótulos LSR solicita explicitamente, de seu próximo salto, uma determinada classe equivalente FEC, cria-se um relacionamento entre um rótulo e a classe. Este método é denominado de distribuição de rótulos sob demanda (*On-Demand Label Distribution*), caso contrário é denominado de distribuição não solicitada (*Unsolicited Label Distribution* [68]. A arquitetura MPLS suporta estes dois métodos, contudo os adjacentes devem concordar previamente sobre qual método usar. A figura 23 ilustra os conteúdos das bases de informação sobre rótulos e a informação contida nas tabelas de roteamento usadas pelos roteadores de comutação de rótulos.

FIGURA 23. DISTRIBUIÇÃO DE RÓTULOS



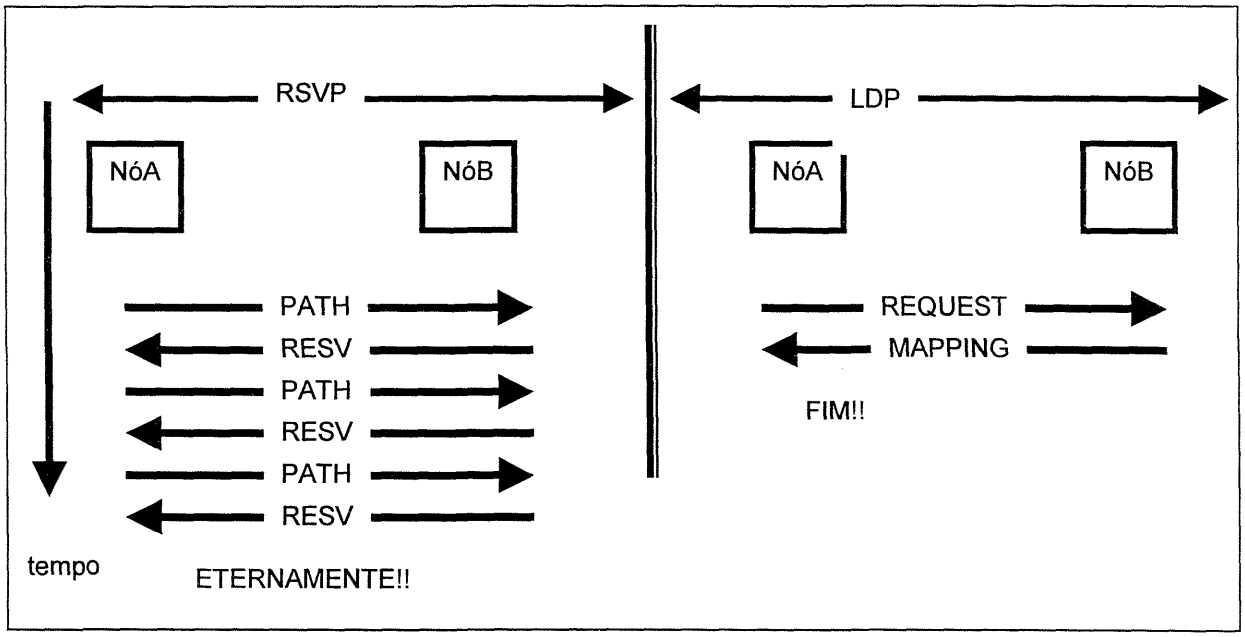
3.3.6.2. Através do Protocolo de Reserva de Recursos RSVP

Apesar do protocolo de reserva de recursos RSVP ter sido projetado para alocar banda para fluxos individuais, também serve para alocar largura de banda para acomodar as necessidades de agregados. Este é o desafio: conhecer as demandas

de banda antecipadamente, para solicitar a reserva de recursos adequadamente.

A figura 24 mostra uma das diferenças entre o protocolo LDP e RSVP e consequentemente sua maior desvantagem, ou seja, em intervalos regulares o RSVP tem que manter-se atualizado, já o LDP não precisa desta constante atualização.

FIGURA 24. ATUALIZAÇÃO DO PROTOCOLO RSVP [71]



3.3.6.3. Através do Protocolo entre Sistemas Autônomos BGP

Quando o protocolo inter–sistema autônomo BGP (*Border Gateway Protocol*) é usado para distribuir uma determinada rota, ele pode ao mesmo tempo ser usado para distribuir rótulos, mapeados para esta rota. O mapeamento de rótulos "pega carona" (*piggybacked*) com o protocolo, ou seja, o rótulo é codificado no campo de informação da camada de rede NLRI (*Network Layer Reachability Information*) e o campo de identificação da família de endereços subsequentes SAFI (*Subsequent Address Family Identifier*) é usado para indicar que o campo NLRI contém um rótulo na mesma mensagem UPDATE [79]. O campo NLRI é codificado na forma <comprimento, rótulo, prefixo>, onde: o campo comprimento indica o comprimento em bits do prefixo mais o rótulo, o campo rótulo corresponde à pilha de rótulos e

finalmente o campo prefixo contém o prefixo do endereço seguido de um trem de bits de preenchimento.

Se dois roteadores comutadores de rótulos LSRs forem adjacentes e nós BGPs ao mesmo tempo, então a distribuição de rótulos pode ser feita sem o auxílio de um protocolo de sinalização adicional.

Caso os nós BGPs não sejam diretamente adjacentes, deve ser estabelecido um caminho comutado por rótulos LSP entre eles.

3.3.7. Seleção da Rota

A seleção da rota refere-se ao método usado para selecionar um caminho comutado por rótulo LSP para uma determinada classe de equivalência FEC [68]. A arquitetura MPLS suporta duas opções :

a) Roteamento salto-a-salto
b) Roteamento Explícito

O roteamento salto-a-salto permite que cada nó escolha de forma independente o próximo salto para cada classe FEC. Este é o modo mais comum atualmente, ou seja, o estabelecimento de um caminho comutado por rótulo salto-a-salto. A rota é selecionada em cada roteador de comutação LSR da mesma forma que no roteamento IP convencional. No roteamento explícito, o roteador de comutação de origem explicitamente especifica a rota desejada.

3.3.8. Fusão (*Merging*)

A fusão de rótulos é a capacidade da rede em encaminhar vários rótulos de entrada a uma classe de equivalência FEC para um mesmo rótulo de saída. Sem esta facilidade, o número de rótulos de saída por classe FEC seria tão grande quanto à quantidade de nós na rede.

Infelizmente, tecnologias como ATM e Frame Relay não suportam a capacidade de fusão de rótulos. No caso do ATM, por exemplo, a tentativa de fusão de rótulos

pode resultar no intercalamento de células de vários pacotes. Devido a estes problemas, criaram-se procedimentos para favorecer a fusão de rótulos no ATM, ou seja, métodos de eliminar o problema de intercalamento de células:

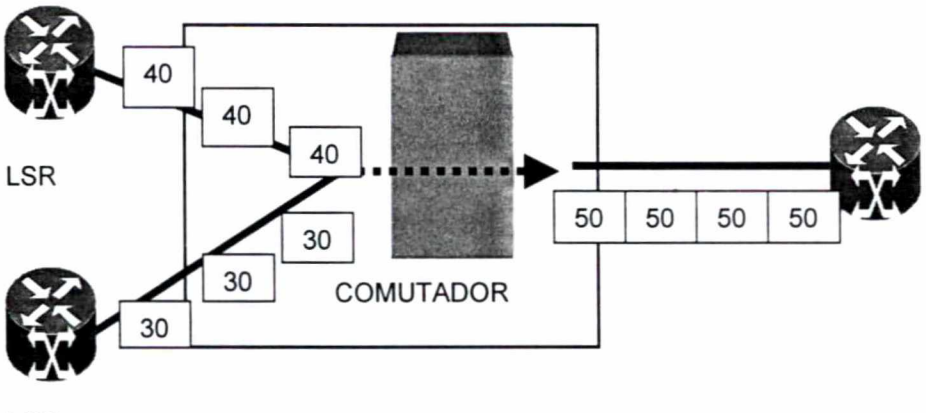
i) Pela fusão de caminhos virtuais, através da codificação multidestino, onde vários caminhos virtuais são unidos em um mesmo caminho virtual, mas os pacotes são diferenciados pelos seus valores VCI.

ii) Pela fusão de circuitos virtuais, onde os comutadores armazenam as células em buffers de um pacote até a chegada completa do mesmo.

iii) A fusão por caminho virtual possui a vantagem de ser compatível com a maioria das implementações de comutadores ATM, porém exige uma coordenação de valores de circuitos virtuais VCI.

A figura 25 mostra dois rótulos, com VPIs distintos (30 e 40), sendo fundidos para um mesmo rótulo.

FIGURA 25. FUSÃO VP



Interface Entrada	Rótulo Entrada	Prefixo do Endereço	Interface Saída	Rótulo Saída
1	40	171.69	3	50
2	40	171.69	3	50

4. IMPLEMENTAÇÃO DE AMBIENTE EXPERIMENTAL

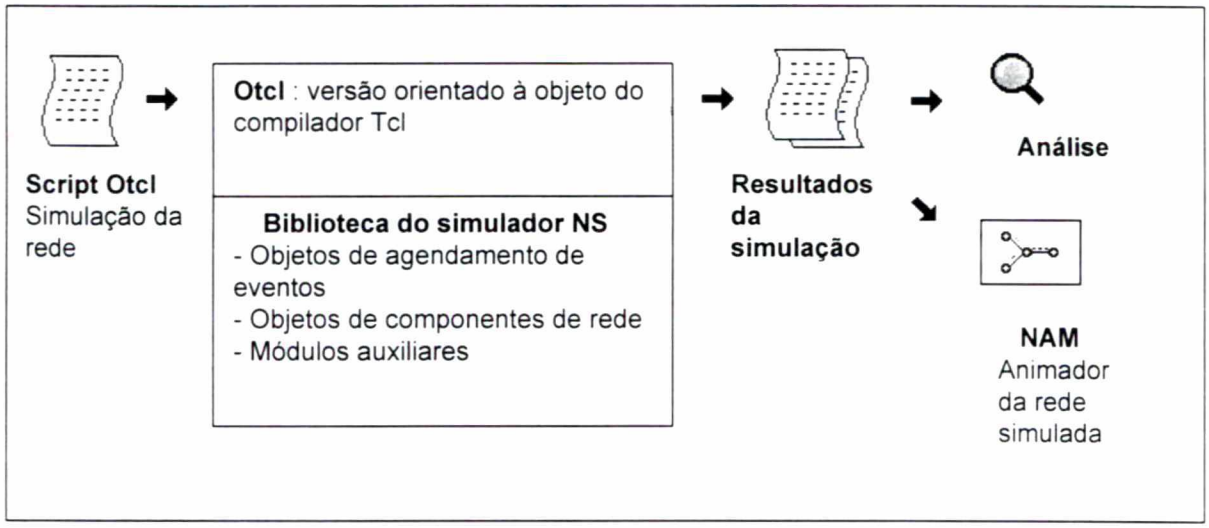
Este capítulo realiza uma análise quantitativa do desempenho das redes IP sem nenhum mecanismo de QoS, rede IP usando um escalonador baseado em classe CBQ/WRR e IP sobre MPLS. Através de experimentos de simulação usando o simulador de rede NS-2 [80], os testes são realizados a fim de demonstrar o desempenho das configurações sob circunstâncias específicas, ou seja, em congestionamento.

4.1. SIMULADOR NS

O simulador NS, desenvolvido pela universidade de Berkeley, simula uma variedade de protocolos do conjunto de protocolos Inter-Redes. Ele implementa protocolos como TCP e UPD, simula comportamento de tráfegos, tais como: FTP , Telnet, Web e CBR, este último usado para aplicações em tempo real. Estão disponíveis algoritmos de roteamento como o Dijkstra e Bellman-Ford. O NS também simula redes locais sem fio (*Wireless Local Networks*) e redes satélites.

A versão 2.1b8 do simulador de rede instalado [80] foi implantado em linguagem em C++ e utiliza o OTcl como interface de comando e configuração [81]. A versão 2 possui duas modificações com relação à versão 1. A primeira mudança é que os objetos mais complexos da versão 1 foram decompostos em componentes mais simples para maior flexibilidade. A Segunda modificação foi a interface de configuração OTcl que é uma versão orientada a objeto da versão Tcl.

FIGURA 26. VISÃO SIMPLIFICADA DO SIMULADOR NS [82]



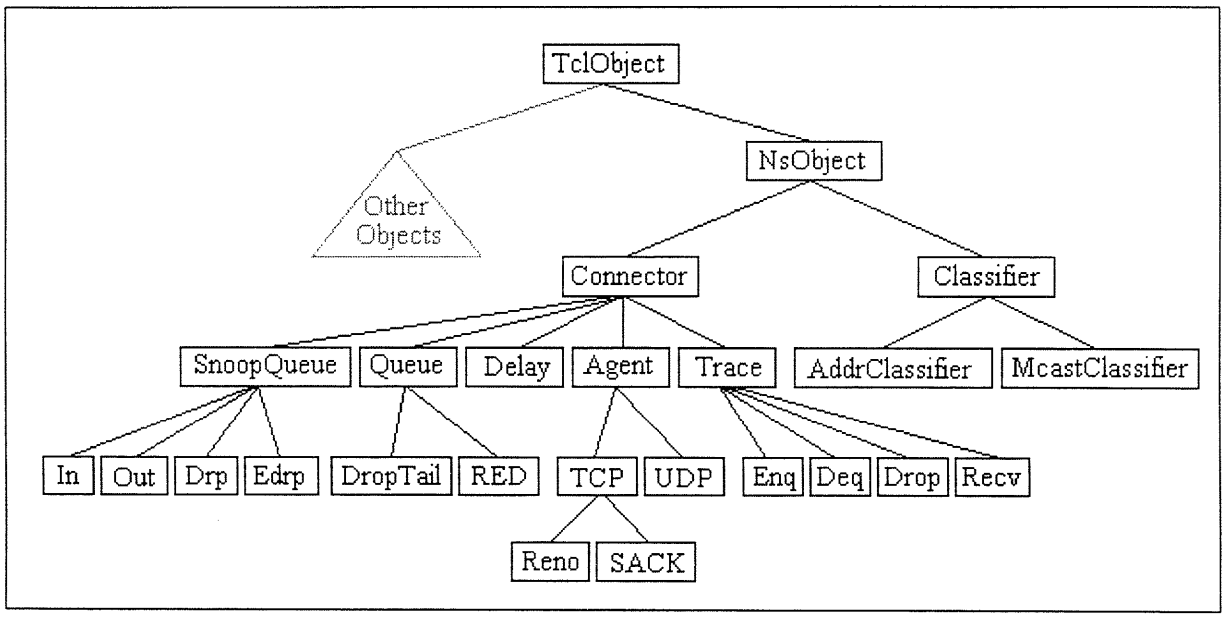
A figura 26 fornece uma visão geral do simulador NS. O compilador OTcl compreende agendadores de eventos de simulação, bibliotecas com os componentes da rede e bibliotecas com módulos auxiliares para configurar a rede desejada [82]. Em outras palavras, para "rodar" o simulador NS, o usuário precisa escrever um script em linguagem Otcl, que depois de compilado inicializa um agendador de eventos, configura a topologia da rede através de objetos de rede e estabelece o início e fim da geração de tráfego, para que o agendador de eventos possa iniciar e finalizar a transmissão de pacotes através da rede. Os resultados da simulação aparecem na forma de arquivos de saída, que podem ser analisados e cujos dados podem ser selecionados e plotados em gráfico. Um dos arquivos de saída, gerado pelo simulador NS, corresponde ao arquivo de entrada para uma interface gráfica, denominada de NAM (*Network Animator*). A interface mostra a topologia criada e a animação dos pacotes sendo transmitidos, recebidos e perdidos.

A raiz da árvore hierárquica do simulador é a classe do objeto Tcl, que é uma superclasse de todos os objetos da biblioteca OTcl (agendadores, componentes de rede, temporizadores e outros objetos incluindo a interface gráfica NAM).

A figura 27 ilustra a árvore com os componentes básicos da rede que são ojetos

subdivididos em duas subclasses: conectores e classificadores. A subclasse conectores, por exemplo, compreende agentes que normalmente são terminais geradores de tráfego TCP e/ou UDP. Outro conector é objeto de enlace que encapsula uma fila e um retardo. Os classificadores são responsáveis pelo encaminhamento dos pacotes.

FIGURA 27. ÁRVORE HIERÁRQUICA [82]



O simulador de rede MPLS para NS-2 é chamado de MNS (*MPLS Network Simulator*) e possibilita a criaçao de domínios MPLS. O simulador suporta comutação por rótulos, protocolo de sinalização LDP e explícito CR-LDP, porém não suporta o protocolo RSVP.

A versão atual do MNS (versão 2) [82] contém uma variável de entrada no objeto do simulador que referencia um classificador denominado de MPLS, que determina se um pacote recebido deve ser rotulado ou não. Se o pacote for rotulado, o classificador MPLS realiza comutação da camada de enlace, caso contrário realiza roteamento da camada de rede.

4.2. SIMULAÇÃO

Existem três opções para realizar a análise do modelo proposto: utilizar modelos analíticos, realizar um experimento prático em uma planta existente ou utilizar um simulador [83]. O método de medição não foi viável, pois a técnica MPLS é relativamente recente e o analítico muito complexo para descrever uma grande rede. Optou-se utilizar neste trabalho a simulação pois possui uma boa precisão, além disso, a liberação de uma versão mais atual motivou a utilização do simulador NS, já que a versão 1 (mais antiga) apresentava muitos erros.

O objetivo desta simulação é comparar quantitativamente o desempenho do tráfego UDP em três configurações distintas:

- rede IP sem nenhuma característica de QoS
- rede IP com escalonador baseado em classe CBQ/WRR
- rede IP sobre MPLS

Para a simulação foi necessária a instalação no computador Pentium 233MHz de 128Mb de RAM e 20G de HD, dos seguintes softwares:

- Sistema operacional LINUX 2.4.7-10, distribuição Red Hat 7.2
- Interface gráfica do Linux: KDE 2.1-10
- Simulador de rede NS: ns-2.1b8
- Interface gráfica do NS: nam-1-snapshot-20011228
- Biblioteca TCL: tcl8.3.3
- Editor gráfico: xgraph 12.1

4.2.1. Topologia da rede IP

A figura 28 mostra a topologia criada para simular as três configurações. A rede simulada consiste de um terminal de origem gerando tráfego UDP que será objeto de análise, um terminal de destino e vários roteadores na nuvem, sendo que a eles estão conectados agentes TCP e UDP para gerar tráfego e com isso criar congestionamento na rede.

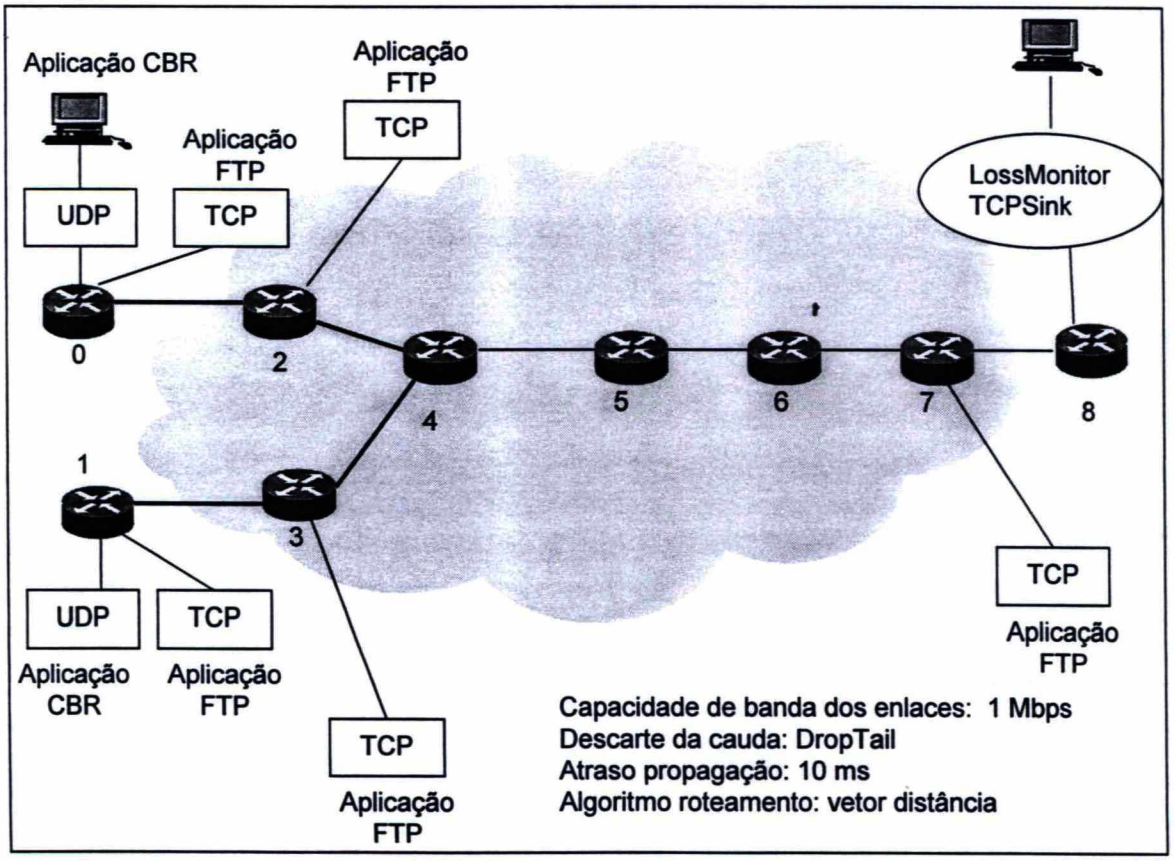
Para a configuração da topologia, utilizaram-se enlaces full-duplex com capacidade de banda de 1Mbps e latência de 10ms. O retardo de propagação de 10 ms não foi baseado em nenhuma característica real de rede, mas é mantido o mesmo valor para se obter uma base comum para análise.

O algoritmo de roteamento usado é o vetor distância, que usa como métrica o valor do custo entre os roteadores para estabelecer em sua tabela de roteamento o caminho mais curto entre origem e destino. Esta escolha é indiferente, ou seja, poder-se-ia escolher outro tipo de algoritmo de roteamento, já que o objeto de estudo não é o roteamento em si.

Os documentos de apoio usados para a geração dos scripts foram o manual ns [81] e o tutorial sobre ns [82].

Os scripts podem ser encontrados no Anexo A.

FIGURA 28. TOPOLOGIA DA REDE SIMULADA



4.2.2. Geração de Tráfego

O tráfego da rede inclui dois tipos de tráfego:

- simulando aplicação FTP usando protocolo TCP
- simulando aplicação de tempo real usando protocolo UDP

A combinação do tráfego UDP e TCP pelo mesmo enlace é para simular o efeito de uma aplicação FTP e de uma aplicação multímidia, respectivamente. O tráfego TCP (transmissão de grandes blocos de dados) usa o máximo da largura de banda disponível no enlace, competindo com uma aplicação de tempo real (transmissão constante de bits em pequenos blocos).

Os agentes de origem TCP e UDP, visualizados na figura 28, não geram dados por si, eles são conectados a módulos de geração de tráfego. O tráfego UDP gerado pela aplicação no terminal de origem (roteador 0) possui uma geração constante de pacotes de tamanho 80 bytes e que são enviados a intervalos de 10 ms, o que representa uma taxa de geração de 64 kbps, que conforme o teorema de Nyquist é suficiente para uma aplicação simulando voz [9]. Escolheu-se a transmissão de voz como um exemplo de aplicação de tempo real e utilizou-se como transporte o protocolo UDP que normalmente é empregado neste tipo de aplicação [85].

A geração do tráfego TCP inicia-se a 0.1 segundo da simulação, onde o tamanho do pacote na simulação foi fixado em 1000 bytes. As configurações como tempo de simulação e o início de geração de tráfego foram mantidas iguais [84]. O tempo da simulação é curto (20 segundos) para reduzir o tempo de processamento da simulação. Neste tempo, espera-se que a rede atinja o estado de operação estável.

A figura 28 também ilustra agentes de destino. O agente denominado de *LossMonitor* é definido como sendo o terminal de recepção de dados e que contém alguns parâmetros para fins de análise, como quantidade de bytes recebidos e pacotes UDP perdidos. O agente de destino TCPSink implementa um módulo receptor de pacotes TCP e para cada pacote TCP recebido é enviado um pacote de confirmação ACK.

Para gerar congestionamento, foram criados 6 níveis de geração de tráfego distintos. Para o nível 0, gerou-se apenas tráfego UDP oriundo do roteador 0. Rodou-se o script e obteve-se o resultado. Depois, no nível 1, além da geração de tráfego UDP, iniciou-se a geração de tráfego UDP do roteador 1 e novamente rodou-se o script anotando-se os resultados obtidos. E assim, realizou-se o mesmo teste para os níveis 2, 3, 4, 5 e finalmente 6. A tabela 6 mostra os seis níveis distintos de geração, o tamanho do pacote e a qual roteador está conectado.

TABELA 6. NIVEIS DE GERAÇÃO DE TRÁFEGO

Nivel	Geração de tráfego	Tamanho pacote	Geradores conectados ao
0	UDP	80 bytes	Roteador 0
1	UDP	1000 bytes	Roteador 1
2	TCP	1000 bytes	Roteador 0
3	TCP	1000 bytes	Roteador 1
4	TCP	1000 bytes	Roteador 2
5	TCP	1000 bytes	Roteador 3
6	TCP	1000 bytes	Roteador 7

Para cada nível de geração com a rede IP, obteve-se uma perda gradual de pacotes UDP gerados pelo roteador 0 (objeto de estudo) no destino (roteador 8). A tabela 7 mostra os seis níveis de geração, a quantidade total de pacotes UDP gerados pelo roteador 0 e a perda apresentada pelo roteador 8 (destino).

TABELA 7. PERCENTUAL DE PERDA

Nivel	Total de pacotes UDP gerados pelo roteador 0	Total de pacotes UDP perdidos	Percentual de perda no destino
0	1988	0	0%
1	1988	0	0%
2	1988	108	5,4%
3	1988	136	6,8%
4	1988	197	9,9%
5	1988	301	15,1%
6	1988	389	19,6%

A partir do nível 4 já se obtém um percentual de perda de pacotes UDP no destino, degradando significativamente a qualidade de serviço para a aplicação, atingindo o nível máximo da simulação, cuja perda é de quase 20% do total gerado.

4.2.3. Rede IP com Escalonador CBQ/WRR

A rede usada para simular a rede IP com o método baseado em classe CBQ/WRR foi a mesma da figura 28, com exceção de algumas implementações específicas do escalonamento, cujo script está mostrado no anexo A desta dissertação.

O método CBQ endereça o particionamento e divisão de banda disponível através de uma estrutura hierárquica de classes. Cada classe possui sua própria fila e banda. Uma classe filho pode emprestar da classe pai quando houver sobra de banda e que adicionalmente permite que seja utilizado um algoritmo de pacotes diferente gerenciando cada fila criada [49]. As classes são atendidas pelo algoritmo WRR, onde cada classe pode transmitir uma certa quantidade de dados. Após a transmissão, o escalonador seleciona a classe seguinte e transmite uma certa quantidade de dados desta classe. Quando uma classe está marcada como fora de limites, ela não será escalonada.

As configurações usadas para implementar o escalonador foram a criação de duas classes: voz e dados. As duas classes compartilham 30% (voz) e 70% (dados) da banda, cuja configuração foi inserida nos roteadores 2, 4, 6 e 8 (figura 28). Para os pacotes UDP, gerados pelo roteador 0, foi definido um identificador de fluxo (id = 0) para que os roteadores possam identificar a classe, classificar e dispor na fila para então escalonar corretamente. Para a classe dados foi definido um identificador de fluxo (id = 1) conectado ao TCP gerado pelo roteador 0, apenas para ilustração.

4.2.4. Topologia da Rede IP sobre MPLS

Para gerar o script com a configuração de rede IP sobre MPLS (Anexo A), os roteadores 2, 3 e 7 da figura 28 foram definidos como LER, ou seja, roteadores de comutação de fronteira ou borda do domínio MPLS. Os roteadores 4, 5 e 6 foram configurados como LSR, ou seja, roteadores de comutação do núcleo do domínio. Além da definição adicional e específica do modelo MPLS, tal como, o tipo de

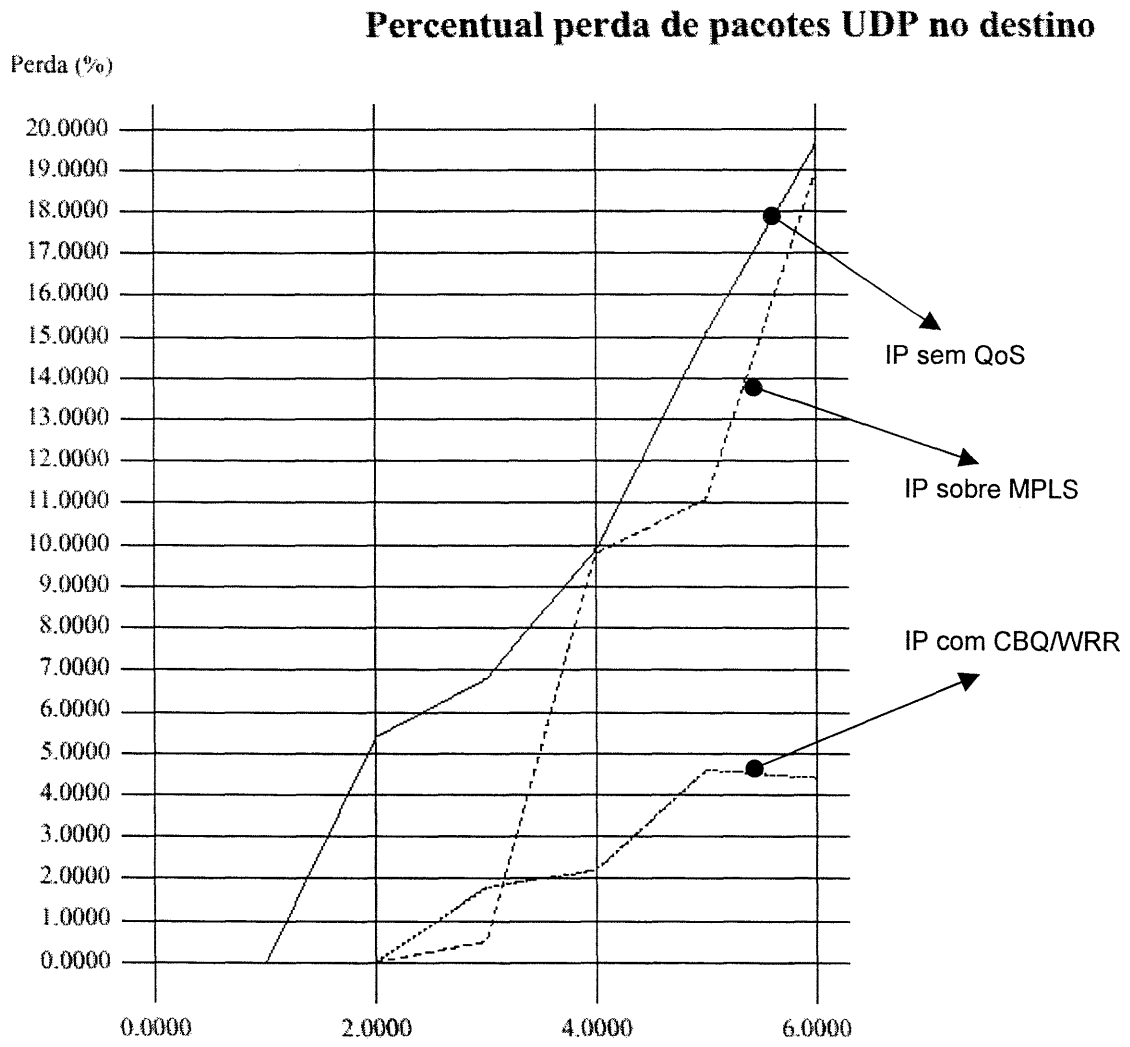
roteamento salto-a-salto ou explícito. Para a simulação foi usado o roteamento salto-a-salto, onde o caminho comutado é estabelecido conforme os pacotes sejam recebidos. Depois que os pacotes atingem o destino, o caminho é conhecido e então os pacotes destinados são comutados pelo caminho comutado.

As demais definições permanecem iguais ao configurado na rede IP sem QoS.

4.2.5. Resultados da Simulação

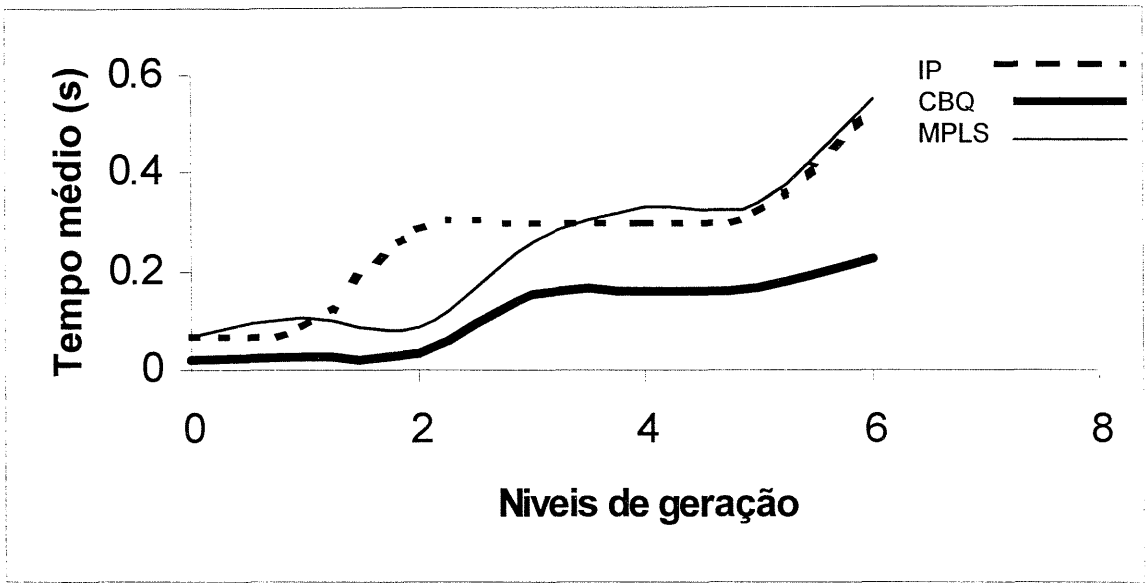
A figura 29 a seguir mostra o percentual de perda para as três configurações. Pode-se observar que o resultado obtido pelo escalonador CBQ/WRR foi muito melhor que a rede IP sem QoS e IP sobre MPLS. Para os níveis 0, 1 e 2, tanto o escalonador como IP sobre MPLS não tiveram nenhuma perda de pacotes. No nível 3, para o escalonador o percentual de perda foi reduzido de 5,4% para 1,8% e para IP sobre MPLS a redução foi maior ainda 0,5%. Para os níveis 5 e 6, que mais comprometem a qualidade de serviço da aplicação, o escalonador obteve um desempenho abaixo de 5%, já para IP sobre MPLS e IP sem QoS foi de 11% e 15%, respectivamente.

FIGURA 29. PERCENTUAL PERDA



Outro parâmetro usado para julgar o desempenho foi o atraso dos pacotes. A figura 30 mostra o atraso médio apresentado para cada nível de geração de tráfego e para as três configurações. A rede IP com escalonador apresentou um desempenho muito melhor com relação ao atraso médio dos pacotes UDP no destino.

FIGURA 30. ATRASO MÉDIO



4.2.6. Conclusão da Simulação

A partir dos experimentos implementados com o simulador NS, constatou-se que para não haver degradação da qualidade de serviço para aplicações em tempo real, apenas a configuração dos roteadores de uma rede como roteadores de comutação MPLS não alteram em muito as perdas de pacotes UDP. Em contrapartida, mecanismos de escalonamento CBQ/WRR aumentam a eficiência e diminuem significantemente a perda de pacotes.

A rede IP sobre MPLS não apresentou um atraso médio constante, pois acreditava-se em uma agilidade no processo de decisão de encaminhamento, ou seja, a consulta à tabela de rótulos para estabelecer o caminho comutado a transmissão de pacotes pelo túnel (a nível de camada 2) reduziria o atraso médio da transmissão. O fato de a topologia conter poucos roteadores pode influenciar neste fator.

Cabe salientar que o tempo de processamento da máquina rodando o simulador NS, foi de apenas 20 segundos. O tempo de simulação poderia ser aumentado e com isso obter uma melhor idéia do desempenho de redes IP sobre MPLS. O

resultado final pode apresentar diferenças significativas, já que o modelo MPLS comuta em seu domínio apenas rótulos na camada 2.

Finalmente, conclui-se que um mecanismo desenvolvido exclusivamente para aumentar a QoS de aplicações em tempo real, como o escalonador CBQ/WRR apresentou resultados muito melhores que a rede IP sobre MPLS.

5. CONCLUSÃO

Neste trabalho foram apresentados mecanismos e modelos de QoS, que integram as mais recentes pesquisas sobre o tema que envolve o provisionamento de qualidade de serviços em redes IP. Para a realização deste foi efetuado um amplo levantamento bibliográfico de artigos de revistas, normas da ITU-T e IETF (através de RFCs), livros, dissertações, teses, informações de fabricantes, além de outras informações disponíveis na Internet. Os principais modelos estudados foram: serviços integrados, serviços diferenciados e comutação por rótulos.

A reserva de recursos do IntServ é dinâmica e efetuada através do protocolo RSVP. Para um número pequeno de fluxos este modelo deve funcionar bem, porém quando o número de fluxos cresce acentuadamente, os roteadores tornam-se muito lentos, pois as tabelas ficam sobrecarregadas, tentando acomodar o grande número de fluxos RSVP (mensagens de controle/sinalização e mensagens periódicas). Este problema de escalabilidade gerou a abordagem do modelo de serviço diferenciado: agregação de fluxos em classes e separação das funções dos roteadores de borda e de núcleo nas grandes redes, denominadas de domínios DS. A tecnologia por rótulos MPLS foi inicialmente projetada com a finalidade de melhorar o desempenho dos roteadores da rede através de um novo paradigma de encaminhamento de pacotes.

A rede estudada reproduz um cenário para simular a transmissão de um sinal de voz através de pacotes UDP. A geração de tráfego foi sendo incrementada até um nível de congestionamento que comprometesse a qualidade de serviço da aplicação. Neste nível, o escalonador CBQ/WRR (30% da banda para voz e 70% para dados) obteve um desempenho melhor que a rede IP sobre MPLS, ou seja, uma perda de apenas 5% de pacotes UDP, já para IP sobre MPLS foi de 11%. Este resultado é esperado pois o escalonador é um mecanismo desenvolvido com a finalidade de prover QoS, já o MPLS possui uma filosofia diferente, ou seja, mudar o paradigma

de encaminhamento de informações. Além disso, outro parâmetro importante para aplicações de voz, atraso médio dos pacotes UDP no destino, não apresentou uma diferença muito grande nas três configurações simuladas.

Cabe salientar que o animador de rede NAM do simulador NS demonstrou ser muito útil na geração dos scripts, especialmente no que tange a geração de tráfego, pois as dificuldades encontradas na execução eram mais facilmente detectáveis e corrigidas com a utilização visual desta interface.

Como observa-se um crescimento exponencial de usuários com aplicativos multimídia, constatou-se que usando-se um mecanismo desenvolvido com a finalidade de melhorar a QoS, como mecanismos de escalonamento CBQ/WRR, apresentam uma eficiência muito melhor que a rede IP sem QoS e que IP sobre MPLS sem nenhuma configuração específica de QoS ou CoS no domínio.

5.1. TEMAS PARA TRABALHOS FUTUROS

Além de aumentar o tempo de simulação e implementar QoS ou CoS no domínio MPLS e verificar o desempenho quanto à perda e atraso de pacotes, sugere-se os seguintes itens de aprimoração, tais como:

- validar a simulação para topologias maiores;
- efetuar uma validação estatística dos dados;
-discutir o modelo de simulação, mostrando o que representa e sua abrangência.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] INFORMATION SCIENCES INSTITUTE; UNIVERSITY OF SOUTHERN CALIFORNIA. RFC 791 - **Internet Protocol**. Internet Engineering Task Force. 1981. Disponível em: <http://www.ietf.org/rfc/rfc791.txt>

[2] BOB, Braden et al. RFC 2309 - **Recommendations on Queue Management and Congestion Avoidance in the Internet**. Internet Engineering Task Force. 1998. Disponível em: <http://www.ietf.org/rfc/rfc2309.txt>

[3] KAPOOR AMAN; RYAN, JOHN. **IP Architecture**. IEEE Communications Magazine., October, 49-56p., 1998.

[4] SOARES, Luiz et al. **Das LANs, MANs e WANs às Redes ATM**. 2a ed. Rio de Janeiro: Campus, 1995. 705 p.

[5] ITU-T STUDY GROUP. ITU-T Recommendation X.200: **Open Systems Interconnection – Basic Reference Model**. ITU-T Recommendation Series X : Data Networks and Open System Communication. , 69 p., 1994.

[6] METZ, Christopher. **At the Core of IP Networks: Link-State Routing Protocols**. IEEE Internet Computing. , September-October, 71-77 p., 1999.

[7] BLACK, Uyless. **QoS in Wide Area Networks**. Prentice Hall, 1999.

[8] COMER, DOUGLAS. **Interligação em rede com TCP/IP, Princípios, Protocolos e Arquitetura**, volume I, 2ª Edição, Tradução da Terceira Edição, Rio de Janeiro, Campus, 1998.

[9] TANENBAUM, Andrew. **Rede de Computadores**. Tradução da 3a Edição ed. Rio de Janeiro: Campus, 1997.

[10] ITU-T STUDY GROUP 2. ITU-T Recommendation E.800: **Terms and Definitions related to Quality of Service and Network Performance including Dependability**. ITU-T Recommendation Series E: Overall Network Operation, Telephone Service, Service Operation and Human Factors. Geneva, 59 p., 1994.

[11] RAJAGOPALAN, Bala et al. RFC 2386 - **A Framework for QoS-based Routing in the Internet**. Internet Engineering Task Force. 1998. Disponível em: <http://www.ietf.org/rfc/rfc2386.txt>

[12] BERNET, Yoram. **The complementary Roles of RSVP and Differentiated Services in the Full-Service QoS Network** . IEEE Communications Magazine. , February, 154-162 p., 2000.

[13] RAHKO, Kauko et al. **Grade of service and human factors** In: 6TH International Symposium on Human Factors in Telecommunications. Stockholm, 1972.

[14] ITU-T STUDY GROUP. ITU-T Recommendation X.641 : **Information Technology – Quality of Service Framework**. ITU-T Recommendation Series X : Data Networks and Open System Communication. Geneva, December, 55 p., 1997.

[15] GOSZTONY et al. **The grade of service in the world wide telephone network**. Telecommunication Journal, 46(IX). 556 – 565 p, 1979.

[16] ITU-T STUDY GROUP. ITU-T Recommendation E.543 : **Grades of Service in Digital International Telephone Exchanges**. ITU-T Recommendation Series E: Overall Network Operation, Telephone Service, Service Operation and Human Factors. Geneva, 1988.

[17] ITU-T STUDY GROUP 2. ITU-T Recommendation E.721 – **Network Grade of Service Parameters and Target Values for Circuit-Switched Services in the Evolving ISDN**. ITU-T Recommendation Series E: Overall Network Operation, Telephone Service, Service Operation and Human Factors. Geneva, May, 11 p., 1999.

[18] FERGUSON, Paul; HUSTON, Geoff. **Quality of Service: Delivering QoS on the**

Internet and in Corporate Networks. John Wiley & Sons, 1998.

[19] ALMQUIST, Philip. **RFC 1349 - Type of Service in the Internet Protocol Suite** . Internet Engineering Task Force. 1992. Disponível em: <http://www.ietf.org/rfc/rfc1349.txt>

[20] HUSTON, Geoff. **Quality of Service - Fact or Fiction?** Telstra. March. 2000. Disponível em: <http://<www.cisco.com/warp/public/~/59/ipj_3-1/ipj_3-1_qos.html>

[21] ARMITAGE, Greenville. **MPLS: The Magic behind the Myths**. IEEE Communications Magazine. , January, 124-131 p., 2000.

[22] DAVIE, Bruce. **MPLS Service Providers to Benefit from New Functionality**. Packet Magazine. 1999. Disponível em: <http://www.cisco.com/warp/public/784/packet/apr99/6.html>

[23] XIPENG, Xiao et al. **Internet QoS : the Big Picture**. IEEE Network. 29 p, 1999.

[24] IIDA, Katsupshi et al. **Performance Evaluation of the Architecture for End-to-End Quality-of-Service Provisioning**. IEEE Communications Magazine. April, 76-81 pp. 2000.

[25] CASNER, Stephen; JACOBSON, Van. RFC 2508 – **Compressing IP/UDP/RTP Headers for Low-Speed Serial Links**. Internet Engineering Task Force. 1999. Disponível em: <http://www.ietf.org/rfc/rfc2508.txt>

[26] ITU STUDY GROUP. ITU-T Recommendation X.642 : **Information Technology - Quality of Service - Guide to Methods and Mechanisms**. ITU-T Recommendation Series X : Data Networks and Open System Communication. Geneva, September, 34 p., 1998.

[27] AURRECOECHEA, Cristina et al. A **Survey of Quality of Service Architectures**. Technical report. University of Lancaster, 33 p., 1995.

[28] CAMPBELL, Andrew. **A Quality of Service Architecture**. Lancaster, 251 p. PhD - Lancaster University. 1996.

[29] KESHAV, S., **Report on the Workshop on Quality of Service Issues in High Speed Networks**. ACM Computer Communications Review, 22(1), 6-15 p., 1993.

[30] BRADEN et al. **RFC 1633 – Integrated Services in the Internet Architecture: an Overview**. Internet Engineering Task Force. 1994. Disponível em: <http://www.ietf.org/rfc/rfc1633.txt>

[31] WHITE, Paul. **RSVP and Integrated Services in the Internet: A Tutorial**. IEEE Communications Magazine. University College London, 35(5), 100-106 p., 1997.

[32] STARDUST White Paper. **QoS Protocols & Architectures**. www.stardust.com. Disponível em: <http://www.stardust.com/qos/whitepapers/protocols.htm>

[33] PAXSON, Vern et al. **RFC 2330 : Framework for IP Performance Metrics**. Internet Engineering Task Force. 1998. Disponível em: <http://www.ietf.org/rfc/rfc2330.txt>

[34] ALMES, Guy et al. **RFC 2679 - A One-way Delay Metric for IPPM**. Internet Engineering Task Force. 1999. Disponível em: <http://www.ietf.org/rfc/rfc2679.txt>

[35] ALMES, Guy et al. **RFC 2681 - A Round-trip Delay Metric for IPPM**. Internet Engineering Task Force. 1999. Disponível em: <http://www.ietf.org/rfc/rfc2681.txt>

[36] ALMES, Guy et al. **RFC 2680 - A One-way Packet Loss Metric for IPPM**. Internet Engineering Task Force. 1999. Disponível em: <http://www.ietf.org/rfc/rfc2680.txt>

[37] MAHDAVI, Jamshid; PAXSON, Vern. **RFC 2678 – IPPM Metrics for Measuring Connectivity**. Internet Engineering Task Force. 1999. Disponível em: <http://ietf.org/rfc/rfc2678.txt>

[38] ITU-T STUDY GROUP 13. ITU-T Recommendation I.380: **Internet protocol data communication service - IP packet transfer and availability performance parameters**. ITU-T Recommendation Series I., Geneva, 33 p., 1999.

[39] JIANG e SCHULZRINNE. **QoS Measurement of Internet Real-Time Multimedia Services**. Multimedia Tools and Applications Journal. New York, Dezembro 1999. Columbia University, 24 p.

[40] KOSTAS, Thomas et al. **Real-Time Voice over Packet Switched Networks**. IEEE Network. IEEE 12(1), 18-27 p., 1998.

[41] LI, Bo et al. **QoS-Enabled Voice Support in the Next-Generation Internet: Issues, Existing Approaches and Challenges**. IEEE Communications Magazine. , April, 54- 61 p., 2000.

[42] KWOK, Timothy. **Residential Broadband Internet Services and Application Requirements**. IEEE Communications Magazine. Microsoft Corporation, 35 (6), 76-83 p., 1997.

[43] LANTING, Cees et al. **Multimedia Services over an Internet: horizontal and vertical aspects of quality of conveyance**. In: INTERWORKING'98 - CONFERENCE,. 6-10 July, 1998. Ottawa, Canada, 1998.

[44] METZ, Chris. **IP QoS: Traveling in First Class on the Internet**. IEEE Internet Computing. , March-April, 84-88 p., 1999.

[45] STEVENS, Richard. **RFC 2001 – TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms**. Internet Engineering Task Force. 1997. January. Disponível em: <http://www.ietf.org/rfc/rfc2001.txt>

[46] PARTRIDGE, Craig. **RFC 1363 – A proposed Flow Specification**. Internet Engineering Task Force. 1992. September. Disponível em: <http://www.ietf.org/rfc/rfc1363.txt>

[47] CISCO. **Cisco Quality of Service (QoS) Networking**. Disponível em: <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm>

[48] SILVA, Adailton. **Qualidade de Serviço em VoIP**. RNP News Generation. , May 4(3), 2000.

[49] PEDROSO, Carlos. **Qualidade de Serviço em Inter Redes IP sobre ATM**. Dissertação de Mestrado. 2000.

[50] GOYAL, Pawan et al. **Integration of Call Signalling and Resource Management for IP Telephony**. IEEE Internet Computing. , May-June, 44-52 p., 1999.

[51] BRADEN et al. **RFC 2205 - Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification**. Internet Engineering Task Force. 1997. September. Disponível em: <http://ietf.org/rfc/rfc2205.txt>

[52] CASNER, Steven et al. **RFC 1190 – ST-II Experimental Internet Stream Protocol Version 2**. Internet Engineering Task Force. 1990. October. Disponível em: <http://ietf.org/rfc/rfc1190.txt>

[53] METZ, Chris. **RSVP: General-Purpose Signaling for IP**. IEEE Internet Computing. , May-June, 95-99 p., 1999.

[54] SHENKER, Scott et al . **RFC 2212 - Specification of Guaranteed Quality of Service**. Internet Engineering Task Force. 1997. September. Disponível em: <http://ietf.org/rfc/rfc2212.txt>

[55] WROCLAWSKI, John. **RFC 2211 - Specification of the Controlled-Load Network Element Service**. Internet Engineering Task Force. September, 1997. Disponível em: <http://www.ietf.org/rfc/rfc2211.txt>

[56] BLAKE, Steven et al. RFC 2475 : **An Architecture for Differentiated Services**. 1998. Disponível em:<http://www.ietf.org/rfc/rfc2475.txt>

[57] NICHOLS, Kathleen et al. **RFC 2474 - Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers**. Internet Engineering Task Force. 1998. Disponível em: <http://www.ietf.org/rfc/rfc2474.txt>

[58] BLACK, David. **RFC 2983 - Differentiated Services and Tunnels**. Internet Engineering Task Force. 2000 . October. Disponível em: <http://www.ietf.org/rfc/rfc2983.txt>

[59] LUOMA, Marko et al. **Quality of service for IP voice services - Is it necessary?** In: PROCEEDINGS OF VOICE, VIDEO AND DATA '98. November 1998 SPIE: , 242-253 p.,1998.

[60] JACOBSON, Van et al. **RFC 2598 – An Expedite Forwarding PHB**. Internet Engineering Task Force. 1999 . June. Disponível em:

<http://www.ietf.org/rfc/rfc2598.txt>

[61] HEINANEN, Juha et al. **RFC 2597 – Assured Forwarding PHB Group**. Internet Engineering Task Force. 1999 . June. Disponível em: <http://www.ietf.org/rfc/rfc2597.txt>

[62] NEWMAN, Peter et al. **RFC 2297 – Ipsilon's General Switch Management Protocol Specification Version 2.0**. Internet Engineering Task Force. 1998. March. Disponível em: <http://www.ietf.org/rfc/rfc2297.txt>

[63] REKHTER, Yakov et al. **RFC 2105 - Cisco Systems' Tag Switching Architecture Overview**. Internet Engineering Task Force. 1997. February. Disponível em: <http://www.ietf.org/rfc/rfc2105.txt>

[64] AWDUCHE, D. et all. **RFC 2702 - Requirements for Traffic Engineering over MPLS**. Internet Engineering Task Force. 1999. September. Disponível em: <http://www.ietf.org/rfc/rfc2702.txt>

[65] ROSEN, Eric et al. **RFC 3032 -.MPLS Label Stack Coding**. Internet Engineering Task Force. 2001. Disponível em: <http://www.ietf.org/rfc/rfc3032.txt>

[66] CHEN, Thomas; OH, Tae. **Reliable Services in MPLS**. IEEE Communications Magazine. December, 58-62 p., 1999.

[67] DAVIE, Bruce; REKHTER, Yakov. **MPLS Technology and Applications** EUA: Morgan Kaufmann., 287 p., 2000.

[68] ROSEN, Eric et al. **RFC 3031 – Multiprotocol Label Switching Architecture**. Internet Engineering Task Force. 2001. Disponível em: <http://www.ietf.org/rfc/rfc3031.txt>

[69] DRURY, David. **Hierarchy via Label stack = Network scalability**. Power point presentation. 1999. 55 p. Disponível em: <www.mplsforum.com>

[70] GROSSMAN, Dan; HEINANEN, Juha. **RFC 2684 – Multiprotocol Encapsulation over ATM Adaptation Layer 5**. Internet Engineering Task Force. 1999. September. Disponível em: <http://www.ietf.org/rfc/rfc2484.txt>

[71] ASHWOOD-SMITH, Peter; JAMOUSSI, Bilel. **MPLS Tutorial and Operational Experiences**. Nortel Networks, 1999.

[72] HEINANEN, Juha. **RFC 1483 – Multiprotocol Encapsulation over ATM Adaptation Layer 5**. Internet Engineering Task Force. 1993. July. Disponível em: <http://www.ietf.org/rfc/rfc1483.txt>

[73] BRAY, Andy. **IP over ATM: a switch for the better?** Telecommunications Magazine. October, 67-71 p., 1998.

[74] LAUBACH, Mark. **RFC 1577 – Classical IP and ARP over ATM**. Internet Engineering Task Force. 1994. January. Disponível em: <http://www.ietf.org/rfc/rfc1577.txt>

[75] LAUBACH, Mark; HALPERN, Joel. **RFC 2225 - Classical IP and ARP over ATM** . Internet Engineering Task Force. 1998. April. Disponível em: <http://www.ietf.org/rfc/rfc2225.txt>

[76] LUCIANI, James et al. **RFC 2332 - NBMA Next Hop Resolution Protocol (NHRP)**. Internet Engineering Task Force. 1998. April. Disponível em: <http://www.ietf.org/rfc/rfc2332.txt>

[77] HUNT, Ray. **Evolving Technologies for New Internet Applications**. IEEE Internet Computing. University of Canterbury, September-October, 16- 26 p., 1999.

[78] BROWN, Caralyn; MALIS, Andrew. **RFC 2427 – Multiprotocol Interconnect over Frame Relay**. Internet Engineering Task Force. 1998. September. Disponível em: <http://www.ietf.org/rfc/rfc2427.txt>

[79] REKHTER, Yakov; ROSEN, Eric. **Draft – Carrying Label Information in BGP-4**. IETF Draft. 2001. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-mpls-bgp4-mpls-05.txt>

[80] BERKELEY UNIVERSITY. **Simulador NS** (Network Simulator) disponível em : <http://www-mash.cs.berkeley.edu/ns/>

[81] FALL, Kevin; VARADHAN Kannan. **NS Manual**. UC Berkeley. November. 2001.

Disponível em: <http://www.isi.edu/nsnam/ns/ns-documentation.html>

[82] CHUNG, Jae; CLAYPOOL, Mark . **NS by Example**, *Technical Report WPI-CS-TR-99-25*, Computer Science Department, Worcester Polytechnic Institute, September, 1999.

[83] JAIN, **The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling**. John Wiley and Sons, New York, NY, 1991.

[84] RAGHAVAN, Srihari. **DiffServ and MPLS: Concepts and Simulation.** Disponível em : http://csgrad.cs.vt.edu/~sraghava/report_raghavan.pdf

[85] CHUNG, Jae; CLAYPOOL, Mark. **Better-Behaved, Better-Performing Multimedia Networking**, *SCS Euromedia Conference (COMTEC)*, Antwerp, Belgium, May 8-10, 2000.

ANEXO A – SCRIPTS GERADOS PARA SIMULAÇÃO

Script para simular uma rede IP sem qualidade de serviço

```
#####################################################################
# Autora: Lilian Barbara Bender Portugal
# Mestranda Telecomunicacoes UFPR/ Fevereiro 2002
# Linux 2.4.7-10 com Red Hat 7.2
# KDE 2.1-10
# Versao Network Simulator:  ns-2.1b8
# Versao Network Animator:   nam-1-snapshot-20011228
# Versao TCL:                tcl8.3.3
# Versao XGRAPH:     xgraph12.1
# Script : ip.tcl
#####################################################################
# Inicializa o objeto de Simulacao
set ns [new Simulator]
# Abertura dos arquivos de saida para NAM e XGRAPH
set nf [open ip.nam w]
$ns namtrace-all $nf
set f0 [open ip_banda_nivel6.tr w]
set f1 [open ip_perda_nivel6.tr w]
set all [open ip_all_nivel6.tr w]
$ns trace-all $all
# Configuracao de variaveis usadas no script
set udp0_flow_id 1
set udp1_flow_id 2
set tcp0_flow_id 3
set tcp1_flow_id 4
set tcp2_flow_id 5
```

```
set tcp3_flow_id 6
set tcp4_flow_id 7
set capacidade_banda 1Mb
set latencia 10ms
set descarte DropTail
set tamanho_pacote_udp0 80
set tamanho_pacote_udp1 1000
set tamanho_pacote_tcp0  1000
set tamanho_pacote_tcp1  1000
set tamanho_pacote_tcp2  1000
set tamanho_pacote_tcp3  1000
set tamanho_pacote_tcp7  1000
set inicio_gravacao 0.0
set inicio_geracao_trafego_udp0   0.1
set inicio_geracao_trafego_udp1   0.1
set inicio_geracao_trafego_tcp0   0.1
set inicio_geracao_trafego_tcp1   0.1
set inicio_geracao_trafego_tcp2   0.1
set inicio_geracao_trafego_tcp3   0.1
set inicio_geracao_trafego_tcp7   0.1
set fim_geracao_trafego_udp0      20.0
set fim_geracao_trafego_udp1      20.0
set fim_geracao_trafego_tcp0      20.0
set fim_geracao_trafego_tcp1      20.0
set fim_geracao_trafego_tcp2      20.0
set fim_geracao_trafego_tcp3      20.0
set fim_geracao_trafego_tcp7      20.0
set intervalo_gravacao 1.0
set tempo_simulacao 20.0
```

```
# Configuracao de cores para o trafego UDP e TCP transmitidos
$ns color $udp0_flow_id Green
$ns color $udp1_flow_id Blue
$ns color $tcp0_flow_id Orange
$ns color $tcp1_flow_id Red
$ns color $tcp2_flow_id Yellow
$ns color $tcp3_flow_id Purple
$ns color $tcp7_flow_id Magenta
# Procedimento de fechamento dos arquivos de saida NAM e XGRAPH
proc finish {} {
global ns nf f0 f1 all
$ns flush-trace
close $nf
close $f0
close $f1
close $all
exec nam ip.nam &
exit 0
}
# Procedimento de gravacao dos dados no arquivo de saida para XGRAPH
proc record {} {
global destino0 f0 f1 intervalo_gravacao
set ns [Simulator instance]
# Ajustar o intervalo de tempo para a gravacao
set time $intervalo_gravacao
# Parametros no destino: bytes recebidos, pacotes perdidos e recebidos
set banda_udp0   [$destino0 set bytes_]
set perdidos0    [$destino0 set nlost_]
# Configurar hora para tempo corrente
```

```
set now [$ns now]
# Calcular largura de banda (em MBit/s) e escrever no arquivo
puts $f0 "$now [expr $banda_udp0/$time*8/1000000]"
puts $f1 "$now [expr $perdidos0]"
# Escrever no arquivo a quantidade de pacotes perdidos
# Zerar os valores no destino
$destino0 set bytes_ 0
$destino0 set nlost_ 0
# Reprogramar o procedimento
$ns at [expr $now+$time] "record"
puts $f0  "YUnitText: Banda (Mbps)"
puts $f0  "XUnitText: Tempo (s)"
puts $f0  "TitleText: Banda UDP no Destino - Nivel  6"
puts $f0  "YUnitText: Perda (pacotes)"
puts $f0  "XUnitText: Tempo (s)"
puts $f0  "TitleText: Perda Pacotes UDP no Destino - Nivel  6"
}
# Topologia
set Node0   [$ns node]
set Node1   [$ns node]
set Node2   [$ns node]
set Node3   [$ns node]
set Node4   [$ns node]
set Node5   [$ns node]
set Node6   [$ns node]
set Node7   [$ns node]
set Node8   [$ns node]
set Node9  [$ns node]
set Node10  [$ns node]
```

```
set Node11  [$ns node]
$ns duplex-link $Node0  $Node2   $capacidade_banda $latencia $descarte
$ns duplex-link $Node1  $Node3   $capacidade_banda $latencia $descarte
$ns duplex-link $Node2  $Node4   $capacidade_banda $latencia $descarte
$ns duplex-llnk $Node3  $Node4   $capacidade_banda $latencia $descarte
$ns duplex-link $Node4  $Node5   $capacidade_banda $latencia $descarte
$ns duplex-link $Node5  $Node6  $capacidade_banda $latencia $descarte
$ns duplex-link $Node6  $Node7   $capacidade_banda $latencia $descarte
$ns duplex-link $Node7  $Node8  $capacidade_banda $latencia $descarte
$ns duplex-link $Node9  $Node2  $capacidade_banda $latencia $descarte
$ns duplex-link $Node10  $Node3  $capacidade_banda $latencia $descarte
$ns duplex-link $Node11  $Node7  $capacidade_banda $latencia $descarte
# Definicao do protocolo de Bellman-Ford
$ns rtproto DV
# Criacao de agentes e aplicacoes para transmitir pacotes
# Geracao trafego UDP conectado a uma aplicacao CBR na fonte 0 para estudo
set Fonte0 [new Application/Traffic/CBR]
$Fonte0 set packetSize_ $tamanho_pacote_udp0
$Fonte0 set interval_ 0.01
set udp0 [new Agent/UDP]
$Fonte0 attach-agent $udp0
$udp0 set fid_ $udp0_flow_id
$ns attach-agent $Node0 $udp0
# Geracao de trafego UDP e TCP para gerar congestionamento
set Src0 [new Application/Traffic/CBR]
$Src0 set packetSize_ $tamanho_pacote_udp1
$Src0 set interval_ 0.01
set udp1 [new Agent/UDP]
$Src0 attach-agent $udp1
```

```
$udp1 set fid_ $udp1_flow_id
$ns attach-agent $Node1 $udp1
set Src1  [new Application/FTP]
set tcp0 [new Agent/TCP]
$tcp0 set fid_ $tcp0_flow_id
$tcp0 set packetSize_ $tamanho_pacote_tcp0
$Src1 attach-agent $tcp0
$ns attach-agent $Node0 $tcp0
set Src2  [new Application/FTP]
set tcp1 [new Agent/TCP]
$tcp1 set fid_ $tcp1_flow_id
$tcp1 set packetSize_ $tamanho_pacote_tcp1
$Src2 attach-agent $tcp1
$ns attach-agent $Node1 $tcp1
set Src3  [new Application/FTP]
set tcp2 [new Agent/TCP]
$tcp2 set fid_ $tcp2_flow_id
$tcp2 set packetSize_ $tamanho_pacote_tcp2
$Src3 attach-agent $tcp2
$ns attach-agent $Node22 $tcp2
set Src4 [new Application/FTP]
set tcp3 [new Agent/TCP]
$tcp3 set fid_ $tcp3_flow_id
$tcp3 set packetSize_ $tamanho_pacote_tcp3
$Src4 attach-agent $tcp3
$ns attach-agent $Node33 $tcp3
set Src5 [new Application/FTP]
set tcp7 [new Agent/TCP]
$tcp7 set fid_ $tcp7_flow_id
```

```
$tcp7 set packetSize_ $tamanho_pacote_tcp7
$Src5 attach-agent $tcp7
$ns attach-agent $Node11 $tcp7
# Criacao de trafego de destino e conexao ao no 8
set destino0  [new Agent/LossMonitor]
set sink0     [new Agent/Null]
set sink1     [new Agent/TCPSink]
set sink2     [new Agent/TCPSink]
set sink3     [new Agent/TCPSink]
set sink4     [new Agent/TCPSink]
set sink5     [new Agent/TCPSink]
$ns attach-agent $Node8 $destino0
$ns attach-agent $Node8 $sink0
$ns attach-agent $Node8 $sink1
$ns attach-agent $Node8 $sink2
$ns attach-agent $Node8 $sink3
$ns attach-agent $Node8 $sink4
$ns attach-agent $Node8 $sink5
$ns connect $udp0 $destino0
$ns connect $udp1 $sink0
$ns connect $tcp0  $sink1
$ns connect $tcp1  $sink2
$ns connect $tcp2  $sink3
$ns connect $tcp3  $sink4
$ns connect $tcp7  $sink5
$ns at $inicio_gravacao                 "record"
$ns at $inicio_geracao_trafego_udp0   "$Fonte0  start"
$ns at $inicio_geracao_trafego_udp1    "$Src0  start"
$ns at $inicio_geracao_trafego_tcp0    "$Src1  start"
```

```
$ns at $inicio_geracao_trafego_tcp1    "$Src2  start"
$ns at $inicio_geracao_trafego_tcp2    "$Src3  start"
$ns at $inicio_geracao_trafego_tcp3    "$Src4  start"
$ns at $inicio_geracao_trafego_tcp7    "$Src5  start"
$ns at $fim_geracao_trafego_udp0       "$Fonte0  stop"
$ns at $fim_geracao_trafego_udp1       "$Src0  stop"
$ns at $fim_geracao_trafego_tcp0       "$Src1  stop"
$ns at $fim_geracao_trafego_tcp1       "$Src2  stop"
$ns at $fim_geracao_trafego_tcp2       "$Src3  stop"
$ns at $fim_geracao_trafego_tcp3       "$Src4  stop"
$ns at $fim_geracao_trafego_tcp7      "$Src5  stop"
$ns at $tempo_simulacao                "finish"
$ns run
```

Script para simular uma rede IP com escalonador CBQ/WRR

```
#####################################################################
# Autora: Lilian Barbara Bender Portugal
# Mestranda Telecomunicacoes UFPR/ Fevereiro 2002
# Linux 2.4.7-10 com Red Hat 7.2
# KDE 2.1-10
# Versao Network Simulator:  ns-2.1b8
# Versao Network Animator:   nam-1-snapshot-20011228
# Versao TCL:                tcl8.3.3
# Versao XGRAPH:     xgraph12.1
# Script : cbq.tcl
#####################################################################
# Inicializa o objeto de Simulacao
set ns [new Simulator]
# Abertura dos arquivos de saida para NAM e XGRAPH
set nf [open cbq.nam w]
$ns namtrace-all $nf
set f0 [open cbq_banda_nivel6.tr w]
set f1 [open cbq_perda_nivel6.tr w]
set all [open cbq_all_nivel6.tr w]
$ns trace-all $all
set capacidade_banda 1Mb
set latencia 10ms
set descarte DropTail
set escalonador CBQ/WRR
set tamanho_pacote_udp0 80
set tamanho_pacote_udp1 1000
set tamanho_pacote_tcp0  1000
```

```
set tamanho_pacote_tcp1  1000
set tamanho_pacote_tcp2  1000
set tamanho_pacote_tcp3  1000
set tamanho_pacote_tcp7  1000
set inicio_gravacao 0.0
set inicio_geracao_trafego_udp0  0.1
set inicio_geracao_trafego_udp1  0.1
set inicio_geracao_trafego_tcp0  0.1
set inicio_geracao_trafego_tcp1  0.1
set inicio_geracao_trafego_tcp2  0.1
set inicio_geracao_trafego_tcp3  0.1
set inicio_geracao_trafego_tcp7  0.1
set fim_geracao_trafego_udp0     20.0
set fim_geracao_trafego_udp1     20.0
set fim_geracao_trafego_tcp0     20.0
set fim_geracao_trafego_tcp1     20.0
set fim_geracao_trafego_tcp2     20.0
set fim_geracao_trafego_tcp3     20.0
set fim_geracao_trafego_tcp7     20.0
set intervalo_gravacao 1.0
set tempo_simulacao 20.0
# Procedimento de fechamento dos arquivos de saida NAM e XGRAPH
proc finish {} {
global ns nf f0 f1 all
$ns flush-trace
close $nf
close $f0
close $f1
close $all
```

```
exec nam wfq.nam &
exit 0
}
# Procedimento de gravacao dos dados no arquivo de saida para XGRAPH
proc record {} {
global destino0 f0 f1 intervalo_gravacao
set ns [Simulator instance]
# Ajustar o intervalo de tempo para a gravacao
set time $intervalo_gravacao
# Parametros no destino: bytes recebidos, pacotes perdidos e recebidos
set banda_udp0   [$destino0 set bytes_]
set perdidos0    [$destino0 set nlost_]
# Configurar hora para tempo corrente
set now [$ns now]
# Calcular largura de banda (em MBit/s) e escrever no arquivo
puts $f0 "$now [expr $banda_udp0/$time*8/1000000]"
puts $f1 "$now [expr $perdidos0]"
# Escrever no arquivo a quantidade de pacotes perdidos
# Zerar os valores no destino
$destino0 set bytes_ 0
$destino0 set nlost_ 0
# Reprogramar o procedimento
$ns at [expr $now+$time] "record"
puts $f0  "YUnitText: Banda (Mbps)"
puts $f0  "XUnitText: Tempo (s)"
puts $f0  "TitleText: Banda UDP no Destino - Nivel 6"
puts $f1  "YUnitText: Perda (pacotes)"
puts $f1  "XUnitText: Tempo (s)"
puts $f1  "TitleText: Perda Pacotes UDP no Destino - Nivel 6"
```

```
}
# Topologia
set Node0   [$ns node]
set Node1   [$ns node]
set Node2   [$ns node]
set Node3   [$ns node]
set Node4   [$ns node]
set Node5   [$ns node]
set Node6   [$ns node]
set Node7   [$ns node]
set Node8   [$ns node]
set Node9   [$ns node]
set Node10  [$ns node]
set Node11  [$ns node]
$ns duplex-link $Node0  $Node2   $capacidade_banda $latencia $descarte
$ns duplex-link $Node1  $Node3   $capacidade_banda $latencia $descarte
$ns simplex-link $Node2  $Node4  $capacidade_banda  $latencia $escalonador
$ns simplex-link $Node4  $Node2  $capacidade_banda  $latencia $descarte
$ns duplex-link $Node3  $Node4   $capacidade_banda $latencia $descarte
$ns simplex-link  $Node4  $Node5  $capacidade_banda  $latencia $escalonador
$ns simplex-link  $Node5  $Node4  $capacidade_banda  $latencia $descarte
$ns simplex-link  $Node5  $Node6  $capacidade_banda  $latencia $escalonador
$ns simplex-link  $Node6  $Node5  $capacidade_banda  $latencia $descarte
$ns simplex-link $Node6  $Node7 $capacidade_banda  $latencia $escalonador
$ns simplex-link $Node7 $Node6  $capacidade_banda  $latencia $descarte
$ns simplex-link  $Node7 $Node8 $capacidade_banda  $latencia $escalonador
$ns simplex-link  $Node8 $Node7 $capacidade_banda  $latencia $descarte
$ns duplex-link $Node9  $Node2  $capacidade_banda $latencia $descarte
$ns duplex-link $Node10  $Node3  $capacidade_banda $latencia $descarte
```

```
$ns duplex-link $Node11  $Node7  $capacidade_banda $latencia $descarte
# Definicoes especificas para o escalonador
set cbqlink24    [$ns link $Node2 $Node4]
set cbqlink45    [$ns link $Node4 $Node5]
set cbqlink56    [$ns link $Node5 $Node6]
set cbqlink67    [$ns link $Node6 $Node7]
set cbqlink78    [$ns link $Node7 $Node8]
set topclass [new CBQClass]
$topclass setparams none 0 1 auto 0 2 0
set voz [new CBQClass]
set fila_voz [new Queue/DropTail]
$voz install-queue $fila_voz
$fila_voz set limit_ 50
$voz setparams $topclass true 0.3 auto 0 1 0
set dados [new CBQClass]
set fila_dados [new Queue/DropTail]
$dados install-queue $fila_dados
$fila_dados set limit_ 50
$dados setparams $topclass false 0.7 auto 1 1 0
$cbqlink24      insert $topclass
$cbqlink24      insert $voz
$cbqlink24      insert $dados
$cbqlink45      insert $topclass
$cbqlink45      insert $voz
$cbqlink45      insert $dados
$cbqlink56      insert $topclass
$cbqlink56      insert $voz
$cbqlink56      insert $dados
$cbqlink67      insert $topclass
```

```
$cbqlink67      insert $voz
$cbqlink67      insert $dados
$cbqlink78      insert $topclass
$cbqlink78      insert $voz
$cbqlink78      insert $dados
$ns color 0 Green
$ns color 1 Blue
$cbqlink24      bind $voz   0
$cbqlink24      bind $dados 1
$cbqlink45      bind $voz   0
$cbqlink45      bind $dados 1
$cbqlink56      bind $voz   0
$cbqlink56      bind $dados 1
$cbqlink67     bind $voz   0
$cbqlink67     bind $dados 1
$cbqlink78     bind $voz   0
$cbqlink78    bind $dados 1
# Definicao do protocolo de Bellman-Ford
$ns rtproto DV
# Criacao de agentes e aplicacoes para transmitir pacotes
# Geracao trafego UDP conectado a uma aplicacao CBR na fonte 0 para estudo
set Fonte0 [new Application/Traffic/CBR]
$Fonte0 set packetSize_ $tamanho_pacote_udp0
$Fonte0 set interval_ 0.01
set udp0 [new Agent/UDP]
$udp0 set fid_ 0
$Fonte0 attach-agent $udp0
$ns attach-agent $Node0 $udp0
# Geracao de trafego UDP e TCP para gerar congestionamento
```

```
set Src0 [new Application/Traffic/CBR]
$Src0 set packetSize_ $tamanho_pacote_udp1
$Src0 set interval_ 0.01
set udp1 [new Agent/UDP]
$Src0 attach-agent $udp1
$ns attach-agent $Node1 $udp1
set Src1  [new Application/FTP]
set tcp0 [new Agent/TCP]
$tcp0 set packetSize_ $tamanho_pacote_tcp0
$tcp0 set fid_ 1
$Src1 attach-agent $tcp0
$ns attach-agent $Node0 $tcp0
set Src2  [new Application/FTP]
set tcp1 [new Agent/TCP]
$tcp1 set packetSize_ $tamanho_pacote_tcp1
$Src2 attach-agent $tcp1
$ns attach-agent $Node1 $tcp1
set Src3  [new Application/FTP]
set tcp2 [new Agent/TCP]
$tcp2 set packetSize_ $tamanho_pacote_tcp2
$Src3 attach-agent $tcp2
$ns attach-agent $Node22 $tcp2
set Src4 [new Application/FTP]
set tcp3 [new Agent/TCP]
$tcp3 set packetSize_ $tamanho_pacote_tcp3
$Src4 attach-agent $tcp3
$ns attach-agent $Node33 $tcp3
set Src5 [new Application/FTP]
set tcp7 [new Agent/TCP]
```

```
$tcp7 set packetSize_ $tamanho_pacote_tcp7
$Src5 attach-agent $tcp7
$ns attach-agent $Node11 $tcp7
# Criacao de trafego de destino e conexao ao no 12
set destino0  [new Agent/LossMonitor]
set sink0     [new Agent/Null]
set sink1     [new Agent/TCPSink]
set sink2     [new Agent/TCPSink]
set sink3     [new Agent/TCPSink]
set sink4     [new Agent/TCPSink]
set sink5     [new Agent/TCPSink]
$ns attach-agent $Node8 $destino0
$ns attach-agent $Node8 $sink0
$ns attach-agent $Node8 $sink1
$ns attach-agent $Node8 $sink2
$ns attach-agent $Node8 $sink3
$ns attach-agent $Node8 $sink4
$ns attach-agent $Node8 $sink5
$ns connect $udp0 $destino0
$ns connect $udp1 $sink0
$ns connect $tcp0  $sink1
$ns connect $tcp1  $sink2
$ns connect $tcp2  $sink3
$ns connect $tcp3  $sink4
$ns connect $tcp7 $sink5
$ns at $inicio_gravacao                 "record"
$ns at $inicio_geracao_trafego_udp0   "$Fonte0  start"
$ns at $inicio_geracao_trafego_udp1    "$Src0  start"
$ns at $inicio_geracao_trafego_tcp0    "$Src1  start"
```

```
$ns at $inicio_geracao_trafego_tcp1    "$Src2  start"
$ns at $inicio_geracao_trafego_tcp2    "$Src3  start"
$ns at $inicio_geracao_trafego_tcp3    "$Src4  start"
$ns at $inicio_geracao_trafego_tcp7    "$Src5  start"
$ns at $fim_geracao_trafego_udp0       "$Fonte0  stop"
$ns at $fim_geracao_trafego_udp1       "$Src0  stop"
$ns at $fim_geracao_trafego_tcp0       "$Src1  stop"
$ns at $fim_geracao_trafego_tcp1       "$Src2  stop"
$ns at $fim_geracao_trafego_tcp2       "$Src3  stop"
$ns at $fim_geracao_trafego_tcp3       "$Src4  stop"
$ns at $fim_geracao_trafego_tcp7       "$Src5  stop"
$ns at $tempo_simulacao                "finish"
$ns run
```

Script para simular uma rede IP sobre MPLS

```
#####################################################################
# Autora: Lilian Barbara Bender Portugal
# Mestranda Telecomunicacoes UFPR/ Fevereiro 2002
# Linux 2.4.7-10 com Red Hat 7.2
# KDE 2.1-10
# Versao Network Simulator:  ns-2.1b8
# Versao Network Animator:   nam-1-snapshot-20011228
# Versao TCL:                tcl8.3.3
# Versao XGRAPH:     xgraph12.1
# Script : mpls.tcl
#####################################################################
# Inicializa o objeto de Simulacao
set ns [new Simulator]
# Abertura dos arquivos de saida para NAM e XGRAPH
set nf [open mpls.nam w]
$ns namtrace-all $nf
set f0 [open mpls_banda_nivel6.tr w]
set f1 [open mpls_perda_nivel6.tr w]
set all [open mpls_all_nivel6.tr w]
$ns trace-all $all
# Configuracao de variaveis usadas no script
set udp0_flow_id 1
set udp1_flow_id 2
set tcp0_flow_id 3
set tcp1_flow_id 4
set tcp2_flow_id 5
set tcp3_flow_id 6
```

```
set tcp7_flow_id 7
set capacidade_banda 1Mb
set latencia 10ms
set descarte DropTail
set tamanho_pacote_udp0 80
set tamanho_pacote_udp1  1000
set tamanho_pacote_tcp0  1000
set tamanho_pacote_tcp1  1000
set tamanho_pacote_tcp2  1000
set tamanho_pacote_tcp3  1000
set tamanho_pacote_tcp7  1000
set inicio_gravacao 0.0
set inicio_geracao_trafego_udp0  0.1
set inicio_geracao_trafego_udp1  0.1
set inicio_geracao_trafego_tcp0  0.1
set inicio_geracao_trafego_tcp1  0.1
set inicio_geracao_trafego_tcp2  0.1
set inicio_geracao_trafego_tcp3  0.1
set inicio_geracao_trafego_tcp7  0.1
set fim_geracao_trafego_udp0    20.0
set fim_geracao_trafego_udp1    20.0
set fim_geracao_trafego_tcp0    20.0
set fim_geracao_trafego_tcp1    20.0
set fim_geracao_trafego_tcp2    20.0
set fim_geracao_trafego_tcp3    20.0
set fim_geracao_trafego_tcp7    20.0
set intervalo_gravacao 1.0
set tempo_simulacao 20.0
# Configuracao de cores para o trafego UDP e TCP transmitidos
```

```
$ns color $udp0_flow_id Green
$ns color $udp1_flow_id Blue
$ns color $tcp0_flow_id Orange
$ns color $tcp1_flow_id Red
$ns color $tcp2_flow_id Yellow
$ns color $tcp3_flow_id Purple
$ns color $tcp7_flow_id Magenta
# Procedimento de fechamento dos arquivos de saida NAM e XGRAPH
proc finish {} {
global ns nf f0 f1 all
$ns flush-trace
close $nf
close $f0
close $f1
close $all
exec nam mpls.nam &
exit 0
}
# Procedimento de gravacao dos dados no arquivo de saida para XGRAPH
proc record {} {
global destino0 f0 f1 intervalo_gravacao
set ns [Simulator instance]
# Ajustar o intervalo de tempo para a gravacao
set time $intervalo_gravacao
# Parametros no destino: bytes recebidos, pacotes perdidos e recebidos
set banda_udp0   [$destino0 set bytes_]
set perdidos0    [$destino0 set nlost_]
# Configurar hora para tempo corrente
set now [$ns now]
```

```
# Calcular largura de banda (em MBit/s) e escrever no arquivo
puts $f0 "$now [expr $banda_udp0/$time*8/1000000]"
puts $f1 "$now [expr $perdidos0]"
# Escrever no arquivo a quantidade de pacotes perdidos
# Zerar os valores no destino
$destino0 set bytes_ 0
$destino0 set nlost_ 0
# Reprogramar o procedimento
$ns at [expr $now+$time] "record"
puts $f0  "YUnitText: Banda (Mbps)"
puts $f0  "XUnitText: Tempo (s)"
puts $f0  "TitleText: Banda UDP no Destino - Nivel 6"
puts $f1  "YUnitText: Perda (pacotes)"
puts $f1  "XUnitText: Tempo (s)"
puts $f1  "TitleText: Perda Pacotes UDP no Destino - Nivel 6"
}
# Definicoes especificas do modelo MPLS
Classifier/Addr/MPLS set data_driven_ 1
Classifier/Addr/MPLS enable-ordered-control
# Topologiaset Node0   [$ns node]
set Node0   [$ns node]
set Node1   [$ns node]
$ns node-config -MPLS ON
set LSR2    [$ns node]
set LSR3    [$ns node]
set LSR4    [$ns node]
set LSR5    [$ns node]
set LSR6    [$ns node]
set LSR7    [$ns node]
```

```
$ns node-config -MPLS OFF
set Node8  [$ns node]
set Node9   [$ns node]
set Node10  [$ns node]
set Node11  [$ns node]
$ns duplex-link $Node0   $LSR2   $capacidade_banda  $latencia $descarte
$ns duplex-link $Node1   $LSR3   $capacidade_banda  $latencia $descarte
$ns duplex-link $LSR2    $LSR4   $capacidade_banda  $latencia $descarte
$ns duplex-link $LSR3    $LSR4   $capacidade_banda  $latencia $descarte
$ns duplex-link $LSR4    $LSR5   $capacidade_banda  $latencia $descarte
$ns duplex-iink $LSR5    $LSR6   $capacidade_banda  $latencia $descarte
$ns duplex-link $LSR6    $LSR7   $capacidade_banda  $latencia $descarte
$ns duplex-link $LSR7    $Node8  $capacidade_banda  $latencia $descarte
$ns duplex-link $Node9    $LSR2  $capacidade_banda  $latencia $descarte
$ns duplex-link $Node10  $LSR3  $capacidade_banda  $latencia $descarte
$ns duplex-link $Node11 $LSR7   $capacIdade_banda  $latencia $descarte

# Configurar os agentes LDP para todos os elementos do dominio MPLS
for {set i 2} {$i < 8} {incr i} {
        for {set j [expr $i+1]} {$j < 8} {incr j} {
        set a LSR$i
        set b LSR$j
        eval $ns LDP-peer $$a $$b
        }
}
# Definicao das cores para as mensagens de sinalizaco do protocolo LDP
$ns ldp-request-color     red
$ns ldp-mapping-color     black
$ns ldp-withdraw-color    magenta
```

```
$ns ldp-release-color      orange
$ns ldp-notification-color yellow
# Configuracao dos Peeres LDP
for {set i 2} {$i < 8} {incr i} {
        set a LSR$i
        for {set j [expr $i+1]} {$j < 8} {incr j} {
        set b LSR$j
        eval $ns LDP-peer $$a $$b
        }
        set m [eval $$a get-module "MPLS"]
        eval set LSR$i $m
        $m enable-reroute "new"
}
# Criacao de agentes e aplicacoes para transmitir pacotes
# Geracao trafego UDP conectado a uma aplicacao CBR na fonte 0 para estudo
set Fonte0 [new Application/Traffic/CBR]
$Fonte0 set packetSize_ $tamanho_pacote_udp0
$Fonte0 set interval_ 0.01
set udp0 [new Agent/UDP]
$Fonte0 attach-agent $udp0
$udp0 set fid_ $udp0_flow_id
$ns attach-agent $Node0 $udp0
# Geracao de trafego UDP e TCP para gerar congestionamento
set Src0 [new Application/Traffic/CBR]
$Src0 set packetSize_ $tamanho_pacote_udp1
$Src0 set interval_ 0.01
set udp1 [new Agent/UDP]
$Src0 attach-agent $udp1
$udp1 set fid_ $udp1_flow_id
```

```
$ns attach-agent $Node1 $udp1
set Src1  [new Application/FTP]
set tcp0 [new Agent/TCP]
$tcp0 set fid_ $tcp0_flow_id
$tcp0 set packetSize_ $tamanho_pacote_tcp0
$Src1 attach-agent $tcp0
$ns attach-agent $Node0 $tcp0
set Src2  [new Application/FTP]
set tcp1 [new Agent/TCP]
$tcp1 set fid_ $tcp1_flow_id
$tcp1 set packetSize_ $tamanho_pacote_tcp1
$Src2 attach-agent $tcp1
$ns attach-agent $Node1 $tcp1
set Src3  [new Application/FTP]
set tcp2 [new Agent/TCP]
$tcp2 set fid_ $tcp2_flow_id
$tcp2 set packetSize_ $tamanho_pacote_tcp2
$Src3 attach-agent $tcp2
$ns attach-agent $Node2 $tcp2
set Src4 [new Application/FTP]
set tcp3 [new Agent/TCP]
$tcp3 set fid_ $tcp3_flow_id
$tcp3 set packetSize_ $tamanho_pacote_tcp3
$Src4 attach-agent $tcp3
$ns attach-agent $Node3 $tcp3
set Src5 [new Application/FTP]
set tcp7 [new Agent/TCP]
$tcp7 set fid_ $tcp7_flow_id
$tcp7 set packetSize_ $tamanho_pacote_tcp7
```

```
$Src5 attach-agent $tcp7
$ns attach-agent $Node11 $tcp7
# Criacao de trafego de destino
set destino0  [new Agent/LossMonitor]
set sink0     [new Agent/Null]
set sink1     [new Agent/TCPSink]
set sink2     [new Agent/TCPSink]
set sink3     [new Agent/TCPSink]
set sink4     [new Agent/TCPSink]
set sink5     [new Agent/TCPSink]
$ns attach-agent $Node8 $destino0
$ns attach-agent $Node8 $sink0
$ns attach-agent $Node8 $sink1
$ns attach-agent $Node8 $sink2
$ns attach-agent $Node8 $sink3
$ns attach-agent $Node8 $sink4
$ns attach-agent $Node8 $sink5
$ns connect $udp0  $destino0
$ns connect $udp1  $sink0
$ns connect $tcp0  $sink1
$ns connect $tcp1  $sink2
$ns connect $tcp2  $sink3
$ns connect $tcp3  $sink4
$ns connect $tcp7 $sink5
$ns at $inicio_gravacao                 "record"
$ns at $inicio_geracao_trafego_udp0   "$Fonte0  start"
$ns at $inicio_geracao_trafego_udp1    "$Src0  start"
$ns at $inicio_geracao_trafego_tcp0    "$Src1  start"
$ns at $inicio_geracao_trafego_tcp1    "$Src2  start"
```

```
$ns at $inicio_geracao_trafego_tcp2    "$Src3  start"
$ns at $inicio_geracao_trafego_tcp3    "$Src4  start"
$ns at $inicio_geracao_trafego_tcp7    "$Src5  start"
$ns at $fim_geracao_trafego_udp0       "$Fonte0  stop"
$ns at $fim_geracao_trafego_udp1       "$Src0  stop"
$ns at $fim_geracao_trafego_tcp0       "$Src1  stop"
$ns at $fim_geracao_trafego_tcp1       "$Src2  stop"
$ns at $fim_geracao_trafego_tcp2       "$Src3  stop"
$ns at $fim_geracao_trafego_tcp3       "$Src4  stop"
$ns at $fim_geracao_trafego_tcp7       "$Src5  stop"
$ns at $tempo_simulacao                "finish"
$ns run
```