

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO

Alexandre Briani Kieling

IMPLEMENTAÇÃO DE UM SISTEMA DE
GERÊNCIA PARA O SERVIÇO DIFERENCIADO
PREMIUM

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

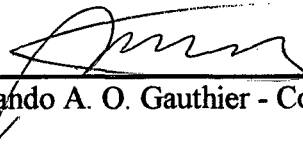
Orientador: Prof. Dr. Carlos Becker Westphall

Florianópolis, fevereiro de 2002

IMPLEMENTAÇÃO DE UM SISTEMA DE GERÊNCIA PARA O SERVIÇO DIFERENCIADO PREMIUM

Alexandre Briani Kieling

Esta dissertação foi julgada adequada para obtenção do título de Mestre em Ciência da Computação na Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.



Prof. Dr. Fernando A. O. Gauthier - Coordenador

Banca Examinadora



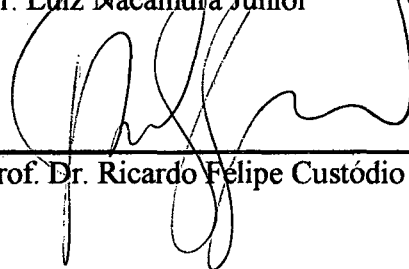
Prof. Dr. Carlos Becker Westphall - Orientador



Prof. Dr. Roberto Willrich



Dr. Luiz Nacamura Júnior



Prof. Dr. Ricardo Felipe Custódio

SUMÁRIO

LISTA DE FIGURAS	IV
LISTA DE TABELAS.....	V
LISTA DE SIGLAS	VI
RESUMO	VIII
ABSTRACT	IX
1 INTRODUÇÃO.....	1
1.1 CONSIDERAÇÕES INICIAIS	1
1.2 OBJETIVOS.....	2
1.2.1 <i>Objetivo Geral</i>	2
1.2.2 <i>Objetivos Específicos</i>	2
1.3 JUSTIFICATIVAS.....	2
1.4 ESTRUTURA DO TRABALHO.....	3
2 QUALIDADE DE SERVIÇO NA INTERNET	5
2.1 DEFINIÇÃO.....	5
2.2 PRINCIPAIS MEDIDAS DE QUALIDADE.....	6
2.2.1 <i>Latência</i>	6
2.2.2 <i>Variação de Latência</i>	6
2.2.3 <i>Perda de Pacotes</i>	7
2.3 MECANISMOS DE QUALIDADE DE SERVIÇO	8
2.3.1 <i>Controle de Admissão</i>	8
2.3.2 <i>Classificação de Pacotes</i>	8
2.3.3 <i>Marcação de Pacotes</i>	8
2.3.4 <i>Enfileiramento</i>	8
2.3.5 <i>Condicionamento de Tráfego</i>	9
2.3.6 <i>Priorização e Escalonamento</i>	10
2.3.7 <i>Controle de Congestionamento</i>	10
2.4 MÉTRICAS DO IPPM.....	11
2.5 CONCLUSÃO.....	16
3 SERVIÇOS DIFERENCIADOS.....	17
3.1 INTRODUÇÃO	17
3.2 ACORDOS DE NÍVEL DE SERVIÇO ENTRE DOMÍNIOS	18
3.3 COMPORTAMENTOS POR NÓ.....	19
3.3.1 <i>EF</i>	19
3.3.2 <i>AF</i>	20
3.4 SERVIÇOS DIFERENCIADOS NA INTERNET2.....	21
3.4.1 <i>Projeto QBone</i>	22
3.4.2 <i>Serviço QPS</i>	22

3.4.3	<i>Serviço QBSS</i>	24
3.5	SERVIÇOS DIFERENCIADOS NO LINUX.....	25
3.5.1	<i>Disciplinas de Enfileiramento</i>	27
3.5.2	<i>Classes</i>	28
3.5.3	<i>Filtros</i>	29
3.5.4	<i>Ferramenta TC</i>	29
3.5.5	<i>Disciplina de Enfileiramento CBQ</i>	32
3.6	CONCLUSÃO.....	32
4	BANDWIDTH BROKER E IMPLEMENTAÇÕES EXISTENTES	33
4.1	SIEMENS	36
4.2	UNIVERSIDADE DE KANSAS.....	37
4.3	GLOBUS.....	38
4.4	UCLA.....	39
4.5	BCIT.....	40
4.6	MERIT	41
4.7	TELLA	41
4.8	COMPARAÇÃO ENTRE OS BANDWIDTH BROKERS	42
4.9	CONCLUSÃO.....	43
5	O SISTEMA DE GERÊNCIA	46
5.1	FUNCIONALIDADES.....	46
5.2	INTERFACE COM O USUÁRIO.....	47
5.3	CONCLUSÃO.....	49
6	IMPLEMENTAÇÃO DO SISTEMA DE GERÊNCIA	50
6.1	TECNOLOGIAS USADAS E JUSTIFICATIVAS.....	50
6.2	ARQUITETURA DO SISTEMA	52
6.3	AGENTE DE DESCOBERTA DE ROTAS	54
6.4	AGENTE DE CONFIGURAÇÃO.....	56
6.5	AGENTE DE MEDIÇÃO.....	57
7	AVALIAÇÃO DO SISTEMA DE GERÊNCIA	60
7.1	AMBIENTE DE TESTES	60
7.2	CADASTRAMENTO DE CLIENTES.....	62
7.3	DESCOBERTA E CONFIGURAÇÃO DOS NÓS DO DOMÍNIO.....	64
7.4	MEDIÇÃO DA QUALIDADE DAS TRANSMISSÕES	66
7.4.1	<i>Medições sem Serviço Premium em Rede sem Tráfego</i>	66
7.4.2	<i>Medições sem Serviço Premium em Rede Congestionada</i>	72
7.4.3	<i>Medições com Serviço Premium em Rede Congestionada</i>	79
7.5	CONCLUSÃO.....	81
8	CONCLUSÃO	83
8.1	CONCLUSÕES SOBRE O SISTEMA DE GERÊNCIA.....	83
8.2	CONCLUSÕES GERAIS	85
8.3	SUGESTÕES PARA TRABALHOS FUTUROS	86
8.3.1	<i>Gerenciamento dos Recursos do Domínio</i>	86
8.3.2	<i>Reserva de Recursos Através de Vários Domínios</i>	86
8.3.3	<i>Gerenciamento do Uso dos Recursos da Rede</i>	87

REFERÊNCIAS BIBLIOGRÁFICAS	88
ANEXO 1 - DESCRIÇÃO DOS ARQUIVOS DA APLICAÇÃO WEB	94
ANEXO 2 - DESCRIÇÃO DAS CLASSES DO AGENTE DE DESCOBERTA DE ROTAS.....	96
ANEXO 3 - DESCRIÇÃO DAS CLASSES DO AGENTE DE CONFIGURAÇÃO	97
ANEXO 4 - DESCRIÇÃO DAS CLASSES DO AGENTE DE MEDIÇÃO	101

LISTA DE FIGURAS

FIGURA 2-1: MÉTODO DO BALDE DE FICHAS	10
FIGURA 2-2: LATÊNCIA <i>ONE-WAY</i>	12
FIGURA 2-3: LATÊNCIA <i>ROUND-TRIP</i>	13
FIGURA 3-1: FLUXO QPS DO CLIENTE PARA O PROVEDOR	24
FIGURA 3-2: PROCESSAMENTO DOS PACOTES DA REDE.....	26
FIGURA 3-3: EXEMPLO DE DISCIPLINA DE ENFILEIRAMENTO COM FILTROS E CLASSES.....	27
FIGURA 4-1: COMPONENTES QUE FORMAM O BANDWIDTH BROKER [NIC99]	33
FIGURA 4-2: DOMÍNIOS <i>DIFFSERV</i> COM BANDWIDTH BROKERS.....	34
FIGURA 5-1: OPERAÇÃO DO SISTEMA	46
FIGURA 5-2: INTERFACE DO GERENTE DE SERVIÇOS DIFERENCIADOS.....	48
FIGURA 6-1: ARQUITETURA DO GERENTE DE SERVIÇOS DIFERENCIADOS.....	54
FIGURA 6-2: AMBIENTE DE TESTES COM A FERRAMENTA <i>TRACEROUTE</i>	55
FIGURA 7-1: AMBIENTE DE TESTES	61
FIGURA 7-2: VISUALIZAÇÃO DOS CLIENTES CADASTRADOS	62
FIGURA 7-3: TESTE DE MEDIÇÃO SEM TRÁFEGO DE <i>BACKGROUND</i>	67
FIGURA 7-4: GRÁFICO DA MEDIÇÃO <i>ONE-WAY</i> SEM TRÁFEGO DE <i>BACKGROUND</i>	68
FIGURA 7-5: GRÁFICO DA MEDIÇÃO <i>ONE-WAY</i> NO SENTIDO INVERSO SEM TRÁFEGO DE <i>BACKGROUND</i>	69
FIGURA 7-6: GRÁFICO DA MEDIÇÃO <i>ONE-WAY</i> COM SINCRONIZAÇÃO A CADA 30 SEGUNDOS E SEM TRÁFEGO DE <i>BACKGROUND</i>	70
FIGURA 7-7: GRÁFICO DA MEDIÇÃO <i>ONE-WAY</i> COM SINCRONIZAÇÃO A CADA 1 SEGUNDO E SEM TRÁFEGO DE <i>BACKGROUND</i>	71
FIGURA 7-8: GRÁFICO DA MEDIÇÃO <i>ROUND-TRIP</i> SEM TRÁFEGO DE <i>BACKGROUND</i>	72
FIGURA 7-9: TESTE DE MEDIÇÃO COM TRÁFEGO DE <i>BACKGROUND</i>	73
FIGURA 7-10: GRÁFICO DAS MEDIÇÕES DE LATÊNCIA <i>ONE-WAY</i> COM TRÁFEGO DE <i>BACKGROUND</i>	74
FIGURA 7-11: MEDIÇÕES <i>ONE-WAY</i> COM SERVIÇO PREMIUM	80

LISTA DE TABELAS

TABELA 3-1: CÓDIGOS DSCP DAS CLASSES AF.....	21
TABELA 4-1: COMPARAÇÃO ENTRE BANDWIDTH BROKERS.....	42
TABELA 6-1: TECNOLOGIAS ESCOLHIDAS PARA O DESENVOLVIMENTO DO SISTEMA	50
TABELA 6-2: RESULTADO DO <i>TRACEROUTE</i> ENTRE 192.168.2.2 E 192.168.3.2.....	55
TABELA 6-3: RESULTADO DO <i>TRACEROUTE</i> ENTRE 192.168.3.2 E 192.168.2.2.....	55
TABELA 6-4: FONTES DE ERRO DE MEDIÇÃO	59
TABELA 7-1: INFORMAÇÕES DO TC SOBRE A CONFIGURAÇÃO BASE NA INTERFACE <i>ETH0</i> DO NÓ LINUX1	63
TABELA 7-2: INFORMAÇÕES DO TC SOBRE A CONFIGURAÇÃO BASE NA INTERFACE <i>ETH1</i> DO NÓ ROTEADOR	64
TABELA 7-3: INFORMAÇÕES DO TC SOBRE OS FILTROS CONFIGURADOS NA INTERFACE <i>ETH0</i> DO NÓ LINUX1	66
TABELA 7-4: INFORMAÇÕES DO TC SOBRE OS FILTROS CONFIGURADOS NA INTERFACE <i>ETH1</i> DO NÓ ROTEADOR	66
TABELA 7-5: ESTATÍSTICAS DA MEDIÇÃO <i>ONE-WAY</i> COM SINCRONIZAÇÃO A CADA 30 SEGUNDOS E SEM TRÁFEGO DE <i>BACKGROUND</i>	70
TABELA 7-6: ESTATÍSTICAS DA MEDIÇÃO <i>ONE-WAY</i> COM SINCRONIZAÇÃO A CADA 1 SEGUNDO E SEM TRÁFEGO DE <i>BACKGROUND</i>	71
TABELA 7-7: ESTATÍSTICAS DA MEDIÇÃO <i>ROUND-TRIP</i> SEM TRÁFEGO DE <i>BACKGROUND</i>	72
TABELA 7-8: ESTATÍSTICAS DO AGENTE DE MEDIÇÃO SEGUNDO O MÉTODO <i>ONE-WAY</i> COM PACOTES TCP E TRÁFEGO DE <i>BACKGROUND</i>	76
TABELA 7-9: ESTATÍSTICAS DO AGENTE DE MEDIÇÃO SEGUNDO O MÉTODO <i>ONE-WAY</i> COM PACOTES UDP E TRÁFEGO DE <i>BACKGROUND</i>	76
TABELA 7-10: ESTATÍSTICAS DO MGEN SEGUNDO O MÉTODO <i>ONE-WAY</i> COM PACOTES UDP E TRÁFEGO DE <i>BACKGROUND</i>	77
TABELA 7-11: ESTATÍSTICAS DO AGENTE DE MEDIÇÃO SEGUNDO O MÉTODO <i>ROUND-TRIP</i> COM PACOTES TCP E TRÁFEGO DE <i>BACKGROUND</i>	78
TABELA 7-12: ESTATÍSTICAS DO AGENTE DE MEDIÇÃO SEGUNDO O MÉTODO <i>ROUND-TRIP</i> COM PACOTES UDP E TRÁFEGO DE <i>BACKGROUND</i>	78
TABELA 7-13: ESTATÍSTICAS DO PING SEGUNDO O MÉTODO <i>ROUND-TRIP</i> COM PACOTES ICMP E TRÁFEGO DE <i>BACKGROUND</i>	79
TABELA 7-14: MEDIÇÕES COM UDP.....	81
TABELA 7-15: MEDIÇÕES COM TCP.....	81

LISTA DE SIGLAS

AF	Assured Forwarding
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BAR	Bandwidth Allocation Request
BB	Bandwidth Broker
CBQ	Class Based Queuing
COPS	Common Open Policy Service Protocol
CSZ	Clark-Shenker-Zhang
DiffServ	Differentiated Services
DS	Differentiated Services
DSCP	Differentiated Services Code Point
DSMARK	Differentiated Services Field Marker
EF	Expedited Forwarding
HTB	Hierarchical Token Bucket
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IOP	Internet Inter ORB Protocol
IntServ	Integrated Services
IP	Internet Protocol
IPPM	Internet Protocol Performance Metrics
JDK	Java Development Kit
JNI	Java Native Interface
JSP	Java Server Pages
MIB	Management Information Base
MTU	Maximum Transmission Unit
NTP	Network Time Protocol
OSPF	Open Shortest Path First

PC	Personal Computer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PHB	Per-Hop Behavior
PIB	Policy Information Base
PQ	Priority Queuing
QoS	Quality of Service
RAP	Resource Allocation Protocol
RED	Random Early Detection
RSVP	Resource Reservation Protocol
SFQ	Stochastic Fair Queuing
SLA	Service Level Agreement
SLS	Service Level Specification
SNMP	Simple Network Management Protocol
TBF	Token Bucket Filter
TC	Traffic Controller
TCP	Transmission Control Protocol
TOS	Type of Service
TTL	Time To Live
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
WFQ	Weighted Fair Queuing

RESUMO

Novos serviços baseados na arquitetura de Serviços Diferenciados estão sendo especificados para suportar aplicações avançadas na Internet. Um exemplo destes serviços é o serviço Premium, o qual possui a capacidade de garantir largura de banda e baixos níveis de latência, variação de latência e perda de pacotes. Devido ao aumento da complexidade da rede gerado por tais serviços, será necessário desenvolver um sistema de gerenciamento sofisticado para controlar o uso da rede. Por esta razão, vários esforços estão sendo direcionados na especificação de um sistema de gerenciamento para os serviços diferenciados. No momento, esta especificação ainda está incompleta, apresentando apenas uma arquitetura básica do sistema e as principais funcionalidades que devem ser implementadas. Este trabalho avalia a situação atual desta arquitetura através da implementação de um sistema de gerenciamento para o serviço Premium baseado na especificação existente. O modo como as funcionalidades do sistema foram desenvolvidas e as dificuldades encontradas são apresentadas em detalhes e as perspectivas futuras na área são comentadas.

ABSTRACT

New services based on the architecture of Differentiated Services are being specified to support advanced applications in the Internet. An example of these services is the Premium service, which possesses the capacity to guarantee bandwidth and low levels of latency, latency variation and packet loss. Due to the increase of the complexity of the net generated by such services, it will be necessary to develop a sophisticated management system to control the use of the net. For this reason, several efforts are being addressed in the specification of a management system for the differentiated services. In the moment, this specification is still incomplete, just presenting a basic architecture of the system and the main functionalities that should be implemented. This work evaluates the current situation of this architecture through the implementation of a management system for the Premium service based on the existent specification. The way the functionalities of the system were developed and the difficulties that had been found are presented in details and the future perspectives in the area are commented.

1 INTRODUÇÃO

1.1 Considerações Iniciais

Com a rápida transformação da Internet em uma infraestrutura comercial, tem surgido muita necessidade por garantias de qualidade que a Internet atual não tem condições de prover. Esta qualidade é representada, principalmente, pela garantia de que as transmissões possuam baixos níveis de latência, variação de latência e perda de pacotes. Para disponibilizar tais níveis de qualidade, a Internet precisará implementar uma arquitetura de Qualidade de Serviço com capacidade de tratar diferentemente cada tráfego.

Nos últimos anos, a arquitetura de Qualidade de Serviço chamada Serviços Diferenciados [BLA98] tem despertado muito interesse no IETF (*Internet Engineering Task Force*) e nos provedores de serviços e equipamentos para a Internet. Esta arquitetura apresenta avanços em relação à arquitetura de Serviços Integrados em termos de escalabilidade e simplicidade. Por esta razão, diversos projetos de avaliação e aprimoramento da arquitetura estão sendo realizados atualmente. Por exemplo, o consórcio Internet2 vem investindo em um projeto chamado QBone [TEI99], o qual tem o propósito de criar um ambiente de testes para os Serviços Diferenciados. O serviço que vem sendo testado com maior ênfase atualmente se chama serviço Premium. Este serviço é razoavelmente simples de implementar e se aplica bem às necessidades atuais de qualidade devido às suas características de baixos níveis de latência, variação de latência e perda de pacotes.

No atual momento, o suporte aos Serviços Diferenciados vem sendo habilitado nas redes locais de diversas instituições de ensino e pesquisa com os objetivos de verificar as características da tecnologia e avaliar suas reais possibilidades de implementação em grande escala. A grande maioria destes testes tem sido realizada através da configuração manual dos nós da rede. Em testes básicos com poucos equipamentos isso não representa um grande problema, porém testes mais sofisticados em redes maiores podem provocar uma grande perda de tempo na realização das configurações. Prevendo a necessidade de um sistema de configuração automática, a arquitetura de Serviços Diferenciados definiu um componente especializado no gerenciamento dos recursos das redes. Este componente

é chamado de *Bandwidth Broker* e sua principal função é viabilizar a disponibilidade de serviços aos usuários da rede. Para isso, o *Bandwidth Broker* deve implementar as seguintes funcionalidades: processamento das requisições de serviço de acordo com as políticas da organização, interação com os nós da rede para a configuração dos mecanismos de controle de tráfego e gerenciamento dos recursos alocados.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é implementar um sistema de gerência para a arquitetura de Serviços Diferenciados. Este sistema é composto por uma parte gerente, baseada no *Bandwidth Broker*, e uma parte agente, composta por sistemas de configuração e medição.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Estudar a arquitetura de Serviços Diferenciados, principalmente as especificações do *Bandwidth Broker* e do serviço Premium;
- Pesquisar implementações do *Bandwidth Broker* e avaliar as tecnologias usadas;
- Implementar um sistema de gerência para o serviço Premium baseado nas especificações do *Bandwidth Broker*;
- Verificar a qualidade do sistema através de testes práticos; e
- Discutir os problemas enfrentados na implementação do sistema e as perspectivas futuras nesta área.

1.3 Justificativas

Na área de Qualidade de Serviço, a arquitetura de Serviços Diferenciados [BLA98] tem evoluído bastante desde sua especificação em 1998, principalmente através dos diversos projetos que vêm sendo realizados por universidades e empresas de diversas partes do mundo. Muito trabalho tem sido realizado no aperfeiçoamento da arquitetura e na especificação de serviços fáceis de implementar e úteis para as necessidades de

qualidade do momento. Mesmo assim, a implementação da arquitetura fora dos ambientes de teste ainda não é possível devido ao fato de existirem diversos detalhes técnicos a resolver. Uma das áreas com menor número de trabalhos e maior número de indefinições é a área de gerenciamento da arquitetura. Esta área é essencial para implementação da arquitetura devido às tarefas que é responsável, como, por exemplo, o controle das políticas de alocação de recursos da rede e a configuração automática dos roteadores para disponibilizar novos serviços aos usuários.

Este trabalho foi realizado na área de gerenciamento da arquitetura de Serviços Diferenciados devido à sua importância na implantação da arquitetura e aos desafios que apresenta. Atualmente, poucos projetos vêm sendo realizados sobre os *Bandwidth Brokers* devido à necessidade de maiores definições sobre a arquitetura do sistema e as tecnologias que devem ser usadas. Porém, projetos em áreas relacionadas ao *Bandwidth Broker* vêm apresentando avanços que poderão ajudar a eliminar as dificuldades de gerenciamento da arquitetura de Serviços Diferenciados. Por esta razão, é importante que novos trabalhos nesta área de gerenciamento de Qualidade de Serviço sejam feitos para acelerar o aprimoramento da arquitetura. Neste trabalho, foi realizada uma pesquisa sobre projetos similares já realizados e atuais esforços com o propósito de avaliar a situação desta área da arquitetura de Serviços Diferenciados. Com base nestes dados, foi realizada a implementação de um sistema básico de gerenciamento para o serviço Premium com o objetivo de obter uma experiência prática e, desta forma, melhor avaliar o *Bandwidth Broker* e suas necessidades de aprimoramento.

1.4 Estrutura do Trabalho

O trabalho está organizado em nove capítulos:

- Capítulo 1: Introdução – Neste capítulo são apresentados os objetivos e justificativas do trabalho;
- Capítulo 2: Qualidade de Serviço na Internet - São apresentadas as principais noções relacionadas com a Qualidade de Serviço;
- Capítulo 3: Serviços Diferenciados - São apresentados os conceitos e a atual situação da tecnologia;

- Capítulo 4: Bandwidth Broker e Implementações Existentes - Este componente da arquitetura de Serviços Diferenciados é apresentado e suas principais implementações são comparadas;
- Capítulo 5: O Sistema de Gerência – Neste capítulo é introduzido o sistema de gerência, com ênfase nas suas funcionalidades.
- Capítulo 6: Implementação do Sistema de Gerência – Neste capítulo são apresentados os detalhes da implementação do sistema, tais como as tecnologias usadas e a arquitetura do sistema;
- Capítulo 7: Avaliação do Sistema de Gerência, onde diversos testes com o sistema são apresentados e discutidos;
- Capítulo 8: Conclusão, onde são apresentadas as conclusões do trabalho realizado e as perspectivas futuras na área.

2 QUALIDADE DE SERVIÇO NA INTERNET

2.1 Definição

Qualidade de Serviço (*Quality of Service* - QoS) [FER98] pode ser definida como o fornecimento de serviços de entrega de dados consistente e previsível. Com QoS, os usuários da rede podem ter certeza que as necessidades de serviço e tráfego podem ser satisfeitas. Habilitar QoS requer a cooperação de todos os elementos da rede, de ponta a ponta. Para prover QoS com um nível de serviço garantido é necessário alocar recursos para fluxos de dados individuais. Além disso, é importante que o tráfego não prioritário não seja privado de fluir pela rede devido às reservas de recursos da rede. As aplicações que usam tecnologias de QoS não podem desabilitar as aplicações Internet comuns.

Na Internet, QoS pode ser bastante útil para dar tratamento preferencial para certos tipos de tráfegos IP (*Internet Protocol*). Com o suporte a QoS através de uma rota, pode-se reduzir bastante os efeitos provocados pela latência nas filas dos roteadores e pelos congestionamentos na rede. Um nó que suporta QoS pode classificar o tráfego em classes de serviço distintas e realizar reserva de recursos para aplicações. Além disso, QoS pode ser usado por instituições para desenvolver e forçar o policiamento do uso da banda da rede.

O protocolo IP e a arquitetura da Internet são baseados no conceito de que datagramas com endereços fonte e destino podem atravessar uma rede de roteadores independentemente, isto é, sem a ajuda do transmissor ou receptor. Porém, existe um preço a pagar por esta simplicidade. A razão pela qual o IP é simples se deve ao fato de não disponibilizar vários serviços. O IP disponibiliza endereçamento e, com isso, habilita a independência de cada datagrama. O IP pode fragmentar os datagramas nos roteadores e remontá-los no receptor para passar por diferentes mídias de rede. Mas o IP não disponibiliza o serviço de entrega confiável de dados. Os roteadores podem descartar datagramas IP sem avisar o transmissor ou o receptor. O IP espera que os protocolos das camadas superiores façam o controle dos datagramas e o retransmitam se necessário, como faz o TCP (*Transmission Control Protocol*). Nem o IP nem os protocolos das camadas superiores podem dar certeza do tempo de entrega ou prover garantias sobre o comportamento dos fluxos de dados.

2.2 Principais Medidas de Qualidade

2.2.1 Latência

Segundo [FER98], a latência de uma transmissão é o tempo entre o envio de uma mensagem de um nó e o recebimento da mesma em outro nó. Isto inclui o tempo gasto nas linhas de transmissão e nos dispositivos da rota de transmissão. As linhas e *hubs* têm um tempo de transmissão relativamente constante. A maior parte da latência é originada dentro dos dispositivos de comunicação (computadores e roteadores). Cada nível na pilha do protocolo TCP/IP dos computadores e roteadores representa um processamento que gera uma certa quantidade de latência. A latência que cada mensagem encontra em cada nível é variável e costuma depender da carga de processamento do dispositivo da rede naquele instante. Nos roteadores, a latência é a quantidade de tempo entre a recepção e retransmissão dos pacotes, também conhecida como tempo de propagação.

A latência tem grande influência sobre a qualidade de aplicações interativas. Aplicações como telefonia e videoconferência pela Internet são exemplos destas aplicações. Até certo limite de latência, o ser humano entende que os dados estão sendo transmitidos e reproduzidos de forma instantânea. Porém, quando a latência das transmissões passa de um limite, esta característica é perdida e a interatividade entre os usuários fica bastante prejudicada. A latência também afeta sistemas computadorizados dependentes de transmissões de informações quase instantâneas, como os sistemas de operação remota em pacientes usando robôs.

2.2.2 Variação de Latência

Em uma rede de pacotes, a variação de latência [FER98] ocorre, principalmente, devido à latência gerada pelo enfileiramento dos pacotes nas interfaces de saída dos roteadores. Como a quantidade de pacotes enfileirados varia de acordo com as variações do tráfego, as latências dos pacotes também variam.

A variação de latência, juntamente com a latência, é particularmente importante em aplicações que necessitam da transmissão contínua de dados para atingir uma qualidade aceitável. Exemplos de aplicações não-interativas que são as transmissões de áudio e

vídeo na Internet. Nestes casos, antes do início das reproduções, os pacotes são armazenados no destino por um certo tempo antes de serem consumidos pela aplicação. Esta técnica é chamada *buffering* e seu objetivo é evitar que a reprodução dos dados seja prejudicada pelas variações de latência dos pacotes. Os pacotes que chegam com latência menor que um valor limite são colocados em um espaço de armazenamento (*buffer*) antes da reprodução e os pacotes que chegam depois são descartados. Quanto maior a variação de latência, maior deverá ser o tamanho do *buffer* para os pacotes. É importante notar que, quanto maior a quantidade de pacotes armazenados, maior será a latência dos mesmos, portanto aumentar o *buffer* não representa a solução do problema. Aplicações deste tipo dependem muito do controle da rede sobre os níveis de variação de latência dos pacotes.

Estudos sobre variação de latência e seu controle, como os apresentados em [VER91], [BAS98] e [MAN98], explicam que, para manter baixo o nível de variação de latência, o tráfego deve ser condicionado nos roteadores.

2.2.3 Perda de Pacotes

A perda de pacotes acontece, principalmente, devido ao descarte de pacotes em roteadores onde os *buffers* não comportam todos os pacotes recebidos. O processo de descarte de pacotes depende de três fatores: condicionamento de tráfego, escalonamento do tráfego na interface de saída e recursos disponíveis no escalonador.

Existem diversas razões porque a perda de pacotes tem grande influência na qualidade de transmissões. Algumas aplicações, como as aplicações de multimídia, não funcionam bem se a taxa de perda de pacotes excede um certo limite. Esta sensibilidade à perda de pacotes aumenta em situações onde as linhas de comunicação têm baixa velocidade. Além disso, protocolos da camada de transporte não conseguem manter altas taxas de transmissão em situações de grande perda de pacotes.

É importante distinguir a perda de um pacote de uma grande latência. Um limite simples seria os 255 segundos da vida de um pacote IP, porém estudos devem ser feitos para chegar a um valor perto do ideal. No QBone [TEI99], o tempo mínimo para concluir que um pacote foi perdido é de dois minutos devido ao fato de ser o valor padrão do TCP e por funcionar muito bem para a maioria das aplicações.

2.3 Mecanismos de Qualidade de Serviço

2.3.1 Controle de Admissão

O controle de admissão é uma parte crucial em qualquer implementação de QoS. Sua principal função é controlar o tráfego que entra na rede. Em redes IP com Serviços Diferenciados, o controle de admissão determina se a requisição de uma conexão pode ser satisfeita pela rede. As principais informações que costumam ser consideradas antes de tomar esta decisão são: a quantidade de tráfego no momento, o perfil de tráfego requisitado e a QoS requisitada.

2.3.2 Classificação de Pacotes

Para disponibilizar a qualidade de serviço requisitada, é importante classificar os pacotes para realizar tratamentos de QoS diferentes. Na Internet, isto pode ser feito com base nas informações existentes no cabeçalho do pacote IP (ex.: endereços fonte/destino e tipo de protocolo) e de pacotes de protocolos de camadas mais altas (ex.: número das portas fonte/destino do TCP ou do UDP - *User Datagram Protocol*). A realização da classificação de pacotes de forma eficiente e consistente é uma questão bastante discutida e pesquisada na área de QoS.

2.3.3 Marcação de Pacotes

Em redes com QoS, os pacotes podem ser marcados para receber um determinado tratamento na rede. A marcação pode ser resultado de um mecanismo de monitoração de tráfego ou de uma discriminação voluntária. Normalmente esta marcação é realizada em algum dos campos do cabeçalho do pacote de dados.

2.3.4 Enfileiramento

No contexto de redes de computadores, enfileiramento é o ato de estocar pacotes para posterior processamento. O enfileiramento costuma ser realizado logo após a

chegada dos pacotes nas interfaces de entrada e logo antes da transmissão pelas interfaces de saída. O gerenciamento de filas é um mecanismo fundamental para realizar a diferenciação de níveis de serviço. A escolha correta do algoritmo de enfileiramento e do tamanho máximo da fila é difícil devido aos diferentes padrões de tráfego presentes na Internet.

2.3.5 Condicionamento de Tráfego

Em redes IP com QoS é necessário especificar o perfil dos tráfegos para decidir como os recursos da rede devem ser alocados. O condicionamento de tráfego certifica que o tráfego entrando através do roteador segue o perfil de tráfego especificado. O volume de tráfego e sua taxa de transmissão são controlados através deste mecanismo. Também pode ser necessário identificar os fluxos de tráfego no roteador de borda (roteador que interliga duas redes) para separá-los e condicioná-los diferentemente. Atualmente, os métodos predominantes para condicionar tráfego são: balde furado (*leaky bucket*) e balde de fichas (*token bucket*). Estes métodos têm propriedades diferentes e são usados para propósitos diferentes.

O método do balde furado é usado para controlar a taxa na qual o tráfego é enviado na rede. Ele tem a capacidade de transformar tráfegos em rajada em tráfegos contínuos. O tamanho do balde e a taxa de transmissão costumam ser configurados pelo usuário e medidos em bytes. Devido à taxa de transmissão ser fixa, este mecanismo não serve para condicionar tráfegos com taxas de transmissão variáveis.

O método do balde de fichas, ilustrado na figura 2-1, define quando o tráfego pode ser transmitido baseado na presença de fichas no balde. Cada ficha representa uma unidade em bytes e os fluxos podem ser transmitidos enquanto existirem fichas. A principal diferença do método balde de fichas em relação ao método balde furado é sua característica de permitir que tráfegos em rajadas continuem sendo transmitidos enquanto existirem fichas no balde. Desta forma, tráfegos em rajada mantêm suas características.

Considerando o envio de um pacote de tamanho b fichas ($b < S$), três cenários podem ser apresentados:

- O balde de fichas está cheio. Neste caso o pacote é enviado e b fichas são removidos do balde.

- O balde de fichas está vazio. Neste caso o pacote deve esperar que b fichas sejam recolocadas no balde antes de enviar.
- O balde de fichas está parcialmente cheio. Existem B fichas no balde. Se $b \leq B$ então o pacote é enviado imediatamente; caso contrário ele deve esperar pelas $b - B$ fichas que faltam.

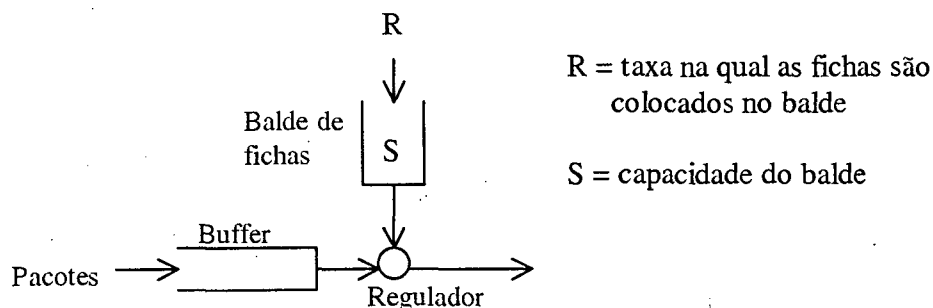


Figura 2-1: Método do balde de fichas

2.3.6 Priorização e Escalonamento

Para satisfazer as necessidades de QoS dos diversos fluxos, os nós precisam ter mecanismos de priorização e escalonamento. Através da priorização, pode-se servir pacotes com alta prioridade antes de pacotes com baixa prioridade em relação ao seu processamento e transmissão através da interface de saída. O tratamento de prioridade também pode influenciar a taxa de perda de pacotes dos diversos fluxos de dados. A priorização de pacotes é a base da diferenciação dos tráfegos na arquitetura de Serviços Diferenciados.

Os nós também precisam ter mecanismos de escalonamento para certificar que cada fluxo obtenha a parte dos recursos que foi prometida, isto é, processamento e largura de banda. O escalonamento também certifica que qualquer recurso disponível seja distribuído justamente.

2.3.7 Controle de Congestionamento

Para que as redes IP com QoS operem de forma eficiente e estável é essencial que existam mecanismos de controle de congestionamento viáveis e robustos. Estes

mecanismos estão relacionados com o controle dos fluxos de pacotes nas filas dos roteadores.

Sem controle de congestionamento, as filas dos roteadores enchem até o limite. A partir daí, os pacotes começam a ser descartados. Quando tais descartes acontecem, o protocolo TCP reduz a taxa de transmissão dos pacotes para tentar diminuir o congestionamento. Este mecanismo é útil, porém não resolve sempre. Existe um mecanismo chamado RED (*Random Early Detection*) que evita os congestionamentos através do descarte randômico de pacotes das filas que estão ficando cheias. Através destes descartes, as transmissões TCP reduzem os tráfegos e o congestionamento é evitado.

2.4 Métricas do IPPM

O grupo de trabalho IPPM (*Internet Protocol Performance Metrics*) do IETF vem desenvolvendo diversos trabalhos relacionados com a medição da performance e confiabilidade das transmissões em redes TCP/IP. Seus principais avanços têm sido representados pela definição de métricas de medição de latência, variação de latência e taxa de perda de pacotes. Estas métricas são apresentadas a seguir, incluindo suas metodologias de medição e as diversas fontes de erro que pode influenciar a confiabilidade dos resultados.

2.4.1.1 Latência One-way

Uma boa forma de medir a latência de transmissões é através da métrica Type-P-One-way-Delay [KAL99a] definida pelo grupo de trabalho IPPM. Esta métrica é apropriada para situações onde se quer medir o tempo de transmissão de pacotes em uma direção específica. Cada amostra é obtida pela contagem do tempo entre a saída do pacote do nó fonte e a chegada do mesmo no nó destino, como ilustrado na figura 2-2. A métrica é composta pelos parâmetros: Src (endereço IP fonte), Dst (endereço IP destino) e T (tempo). Uma amostra de latência de um nó fonte até um nó destino no tempo T é denominada dT, isto é, o nó fonte envia o primeiro bit de um pacote de teste no tempo T e o nó destino recebe o último bit daquele pacote no tempo T+dT.

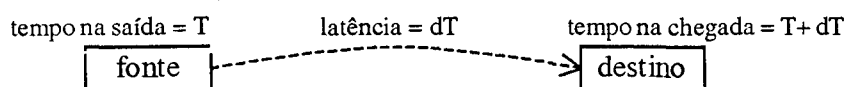


Figura 2-2: Latência *one-way*

Esta métrica é bastante recomendada para aplicações de rede que realizam transmissões em somente um sentido na maioria do tempo. Exemplos destas aplicações podem ser os sistemas de transferência de arquivos (*File Transfer Protocol*) e os sistemas de transmissão ponto-multiponto (*broadcast*) de áudio e vídeo pela Internet. Por suas características, estas aplicações necessitam de boa performance de transmissão em somente um único sentido. Desta forma, a medição da performance da rede é necessária somente no sentido do fluxo de dados.

A metodologia de medição da métrica Type-P-One-way-Delay segue os seguintes passos:

- Certificar que os nós fonte e destino estão bem sincronizados entre si e com o tempo atual;
- Criar um pacote de teste com os endereços Src e Dst no nó fonte. A parte do pacote referente aos dados deve ser preenchida com caracteres aleatórios para evitar a medição de valores menores do que deveria caso haja alguma compressão do pacote;
- Esperar a chegada deste pacote no nó fonte;
- Marcar o tempo atual do sistema neste pacote no nó fonte e transmiti-lo para o nó destino;
- Se o pacote chegar dentro de um período de tempo razoável, o tempo de chegada e o tempo marcado no pacote são determinados o mais rápido possível. Uma estimativa da latência *one-way* pode ser calculada pela subtração dos dois tempos. Se o pacote não chegar dentro de um tempo pré-determinado, a latência *one-way* é denominada indefinida.

A metodologia de medição não especifica o tempo limite de espera pelos pacotes de teste antes de determiná-los perdidos. Na prática, uma boa engenharia de redes e um bom entendimento do tempo de vida dos pacotes são necessários para poder determinar este limite.

A análise de erro de uma implementação do método deve levar em conta o nível de sincronização entre os relógios do nó fonte e nó destino. Já que a latência dos pacotes costuma ter valores bem baixos (alguns milissegundos), é muito importante que os relógios sejam sincronizados com extrema precisão. Duas alternativas para esta sincronização são os sistemas GPS (*Global Positioning Service*) e os servidores NTP.

2.4.1.2 Latência Round-trip

Outra alternativa para medir a latência de transmissões é através da métrica Type-P-Round-trip-Delay [KAL99b]. Na medição de latência *round-trip*, cada amostra é obtida pela contagem do tempo entre a saída do pacote do nó fonte, sua passagem pelo nó destino e sua chegada no mesmo nó fonte, como ilustrado na figura 2-3. Similarmente à métrica Type-P-One-way-Delay, esta métrica é composta pelos parâmetros: Src (endereço IP fonte), Dst (endereço IP destino) e T (tempo).

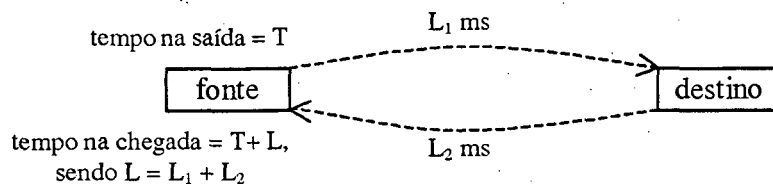


Figura 2-3: Latência *round-trip*

Esta métrica é mais apropriada para situações onde é interessante saber o tempo de ida e volta entre dois pontos. Esta é a realidade em diversas aplicações, inclusive as aplicações interativas, como as videoconferências ponto-a-ponto. Nos casos onde é necessário determinar o valor da latência em somente um sentido, costuma-se dividir os valores medidos por dois.

A metodologia de medição da métrica Type-P-Round-trip-Delay é a seguinte:

- Criar um pacote de teste com os endereços Src e Dst no nó fonte. A parte do pacote referente aos dados deve ser preenchida com caracteres aleatórios para evitar a medição de valores menores do que deveria caso haja alguma compressão do pacote. O pacote deve ter algum identificador de seqüência para controle;

- Esperar a chegada deste pacote no nó destino;
- Inserir o tempo atual do sistema no pacote no nó fonte e transmiti-lo para o nó destino;
- Se o pacote chegar no nó destino, enviar um pacote de resposta com o mesmo conteúdo para o nó fonte o mais rápido possível;
- Se o pacote de resposta chegar dentro de um período de tempo razoável no nó fonte, o tempo de chegada e o tempo marcado no pacote são determinados o mais rápido possível. Uma estimativa da latência *round-trip* pode ser calculada pela subtração dos dois tempos. Se o pacote não chegar dentro de um tempo pré-determinado, a latência é denominada indefinida.

Este método, comparado com o método de medição de latência *one-way*, apresenta algumas vantagens e desvantagens. A principal vantagem é a maior facilidade de implementação, principalmente porque o nó destino deve somente devolver os pacotes recebidos. O maior processamento fica centralizado no nó fonte. Além disso, como a única referência de tempo é o relógio do nó fonte, não é preciso se preocupar com a sincronização entre os nós.

Entre as desvantagens, pode-se citar o problema da assimetria de caminhos nas transmissões pela Internet, isto é, o caminho usado pelos pacotes enviados de um ponto A para um ponto B pode ser diferente do caminho usado pelos pacotes enviados do ponto B para o ponto A. Desta forma, as medições *round-trip* podem medir a performance de dois caminhos distintos juntos. Mesmo que os caminhos sejam simétricos, eles podem ter diferenças de performance, principalmente em situações onde há reserva de recursos em algum sentido. Em diversos casos, as reservas de recursos para as aplicações são realizadas somente um sentido.

2.4.1.3 Variação de Latência Instantânea

Para medir a variação de latência de transmissões, o IPPM especificou a métrica Type-P-One-way-ipdv [DEM99]. De acordo com esta métrica, em um fluxo de pacotes, a variação de latência é medida pela diferença da latência *one-way* de um pacote e a latência *one-way* do próximo pacote do fluxo. Tal definição assume que a fonte transmite

o fluxo de pacotes de forma periódica e a uma taxa baixa para não influenciar os outros tráfegos da rede.

Em geral, o procedimento para medição da variação de latência é o seguinte:

- Transmitir um fluxo de teste da fonte para o destino de acordo com uma distribuição de Poisson;
- Marcar cada pacote de teste com um número sequencial para evitar perda ou reordenação de pacotes;
- Aplicar a metodologia da métrica Type-P-One-Way-Delay para cada par (P_n , P_{n+1}) de pacotes de teste consecutivos;
- Armazenar o valor de latência de cada pacote de teste no destino do fluxo;
- Na recepção do pacote P_{n+1} , subtrair a latência do pacote P_n da latência do pacote P_{n+1} ;
- O valor resultante é a variação de latência;
- Se um ou ambos pacotes não chegarem no destino dentro de um tempo determinado, então a variação de latência deve ser tratada como indefinida;
- Se um pacote for duplicado pelo caminho, o pacote deve ser contado como recebido e a primeira cópia determina o valor de latência;
- Pacotes não devem ser fragmentados.

2.4.1.4 Perda de Pacotes

A métrica de medição de perda de pacotes definida pelo IPPM se chama Type-P-One-way-Packet-Loss [KAL99c]. Ela requer que os pacotes de medição sejam enviados de acordo com a metodologia da métrica Type-P-One-way-Delay. Segundo sua definição, os pacotes transmitidos que chegam no destino são representados pelo valor zero e os pacotes que se perdem são representados pelo valor um.

A metodologia da métrica Type-P-One-way-Packet-Loss é composta pelos seguintes passos:

- Certificar que os nós fonte e destino estão sincronizados entre si;
- Criar um pacote de teste com os endereços Src e Dst no nó fonte;
- Esperar a chegada deste pacote no nó destino;

- Marcar o tempo atual do sistema neste pacote no nó fonte e transmiti-lo para o nó destino;
- Se o pacote chegar no destino, a métrica Type-P-One-way-Packet-Loss é zero.
- Se o pacote não chegar dentro do tempo limite, a métrica é um.

Existem duas situações onde um pacote é determinado perdido: quando o pacote não chega no destino ou quando a diferença entre o tempo de saída e o tempo de chegada do pacote é maior que o tempo limite de espera. Se os nós não estiverem bem sincronizados, a definição da latência dos pacotes pode ser feita erroneamente, prejudicando a determinação dos pacotes perdidos. Nestes casos, pacotes que chegam dentro do limite podem ser considerados perdidos. Para que as medições sejam precisas, os nós devem estar sincronizados e o valor limite de espera deve ser alto o suficiente para que os pacotes que chegam no destino sejam raramente considerados perdidos.

Os recursos da máquina receptora dos pacotes podem influenciar a medição de pacotes perdidos. Nos casos onde os recursos do sistema estão no limite, alguns pacotes podem ser descartados e contados perdidos, mesmo chegando em um tempo razoável. Por esta razão, os equipamentos usados nas medições devem lidar bem com as situações de congestionamento na interface de rede ou outro tipo de limite de recursos, como os *buffers*.

2.5 Conclusão

Em rede com QoS, o controle de congestionamento para tráfegos prioritários requer a criação de filas, o direcionamento dos pacotes para estas filas através de mecanismos de filtragem e classificação e o escalonamento dos pacotes para transmissão. Existem vários protocolos de enfileiramento disponíveis e cada um permite uma forma de criação de filas diferente, proporcionando diversos níveis de diferenciação de tráfego e ordens de transmissão.

A qualidade das transmissões pode ser medida através de métricas de medição, como as definidas pelo grupo de trabalho IPPM. Estas métricas são suficientes para avaliar as transmissões ponto-a-ponto baseado nos parâmetros: latência, variação de latência e perda de pacotes.

3 SERVIÇOS DIFERENCIADOS

3.1 Introdução

Na última década, diversos grupos do IETF (*Internet Engineering Task Force*) desenvolveram especificações de mecanismos de QoS para redes IP. Muito desse desenvolvimento foi focado na arquitetura de Serviços Integrados [BRA94], a qual requer que todos os roteadores sejam capazes de disponibilizar QoS diferentemente para cada fluxo de dados. O problema é que este processamento diferenciado para cada fluxo aumenta muito a carga dos roteadores, principalmente os que ficam no centro das grandes redes (*backbones*). Em 1998, foi desenvolvida a arquitetura de Serviços Diferenciados (Differentiated Services - DiffServ) [BLA98] [MET00], cujo principal objetivo seria resolver o problema de escalabilidade encontrado pelos Serviços Integrados. Sua arquitetura define que os roteadores centrais devem operar somente com grupos de fluxos cujas necessidades de QoS são similares. Em vez de prover níveis de QoS diferentes para cada fluxo, os Serviços Diferenciados provêem níveis de QoS diferentes para cada grupo de fluxos, diminuindo a complexidade da arquitetura.

Os Serviços Diferenciados podem disponibilizar vários serviços com diferentes níveis de QoS. De acordo com as necessidades, as aplicações escolhem um dos serviços disponíveis. Todas as aplicações que escolhem o mesmo serviço têm seus fluxos de dados agregados e, desta forma, tratados igualmente pelos roteadores. A associação entre os fluxos e os serviços é feita através de uma nova informação que é colocada no cabeçalho dos pacotes IP. O DiffServ renomeou o campo TOS (*Type of Service*) do IPv4 e o campo *Traffic Class* do IPv6 para DS (*Differentiated Services*) [NIC98] e definiu um código diferente para cada serviço disponível, chamado DSCP (*Differentiated Services Codepoint*) [BLA98]. Cada DSCP corresponde a um tratamento de retransmissão diferente, chamado PHB (*Per Hop Behavior*) [NIC98]. De acordo com os códigos DSCP, os roteadores da rede classificam os pacotes em diferentes classes de retransmissão, cujos recursos alocados dependem das políticas de QoS da rede.

Os Serviços Diferenciados definiram o conceito de domínio como um conjunto contíguo de nós que operam com uma política de provisão de serviços e implementam um conjunto de comportamentos de retransmissão. Na prática, estes domínios são

representados pelas redes das empresas, universidades, provedores de serviço, etc. Para que os fluxos recebam QoS fim a fim, deve haver cooperação entre os diversos domínios por onde passam estes fluxos.

A complexidade dos Serviços Diferenciados é colocada nos roteadores folha. Estes roteadores realizam a conexão entre a rede dos usuários e os roteadores centrais. Por representarem o primeiro roteador por onde passa o tráfego dos usuários, devem realizar a classificação, o policiamento, a marcação e o condicionamento de cada fluxo. Após serem marcados e condicionados, os pacotes são transmitidos para outros roteadores, onde são agregados com pacotes com mesma marcação provindos de outras fontes. Nestes roteadores, onde o tráfego é maior e o processamento é mais elevado, somente a classificação e priorização do tráfego são necessárias. Cada agregação de fluxos recebe o tratamento definido pelo PHB correspondente ao DSCP marcado nos pacotes. A latência das filas dos roteadores e o aumento do número de fluxos das agregações podem aumentar a variação de latência de alguns fluxos. Para manter a qualidade dos serviços, o tráfego deve ser condicionado nos roteadores que interconectam os domínios DiffServ.

3.2 Acordos de Nível de Serviço entre Domínios

Acordos de Nível de Serviço (*Service Level Agreements – SLAs*) são contratos de longa duração entre domínios adjacentes. Eles garantem que o nível de serviço oferecido por um domínio a uma agregação de fluxos será mantido pelo outro domínio. Em geral, os SLAs são contratos complexos que possuem tanto informações técnicas como informações não técnicas, incluindo garantias de disponibilidade e performance da rede, regras de policiamento e condicionamento do tráfego, modelos de pagamento, entre outras questões.

Os detalhes técnicos dos SLAs são chamados de Especificações de Nível de Serviço (*Service Level Specifications – SLSs*). Elas especificam as agregações de fluxos e suas correspondentes regras de classificação, medição, marcação, condicionamento e descarte. Também fazem parte das SLSs os parâmetros de performance, como taxa de transmissão, variação de latência e taxa de perda de pacotes. De forma prática, uma SLS define que uma agregação de fluxos, que entra em um domínio por uma linha específica e obedece a certas condições, será tratada de acordo com o PHB adequado.

Através da configuração de SLAs entre os domínios DiffServ e o cuidadoso condicionamento das fontes de tráfego dentro dos domínios, a arquitetura de Serviços Diferenciados é capaz de prover serviços fim-a-fim através de diferentes redes administradas separadamente, como a Internet. Além disso, os acordos de serviço dos Serviços Diferenciados são relativamente simples e não diferem muito dos acordos de serviço já existentes.

3.3 Comportamentos por Nó

Os comportamentos por nó (*Per Hop Behaviors* - PHBs) podem ser especificados em termos de prioridades de uso de recursos (ex.: *buffer* e largura de banda) ou em termos de características de tráfego observáveis (ex.: latência e perda de pacotes). Os PHBs são definidos em termos de características de comportamento relevantes às políticas da rede e não em termos de mecanismos de implementação. A existência do suporte a grupos PHB em nós de um domínio DiffServ habilita a divisão eficaz dos recursos dos nós e das conexões entre os nós. Quando um ou mais PHBs só têm sentido quando especificados juntos, eles são denominados um grupo PHB. Além disso, é comum que mais de um grupo PHB seja implementado em um nó e utilizado dentro de um domínio DiffServ.

3.3.1 EF

O PHB EF (*Expedited Forwarding*) [JAC99] pode ser usado em domínios DiffServ para habilitar serviços fim-a-fim com baixa perda de pacotes, baixa latência, baixa variação de latência e largura de banda garantida.

O PHB EF é definido como um tratamento de retransmissão para a agregação de fluxos (com DSCP 101110) onde a taxa máxima de chegada do tráfego deve ser menor que a taxa mínima de saída em um mesmo nó. Tal exigência deve ser satisfeita porque as filas crescem quando a taxa de chegada do tráfego excede a taxa de saída, provocando variações de latência no tráfego. O tráfego EF deve receber a taxa estipulada independentemente da intensidade de qualquer outro tráfego existente no nó. Além disso, uma taxa limite máxima também deve ser configurada para garantir que a largura de

banda não seja injustamente usada. Tal comportamento é atingido através do policiamento e condicionamento do tráfego.

3.3.2 AF

O PHB AF (*Assured Forwarding*) [HEI99] foi sugerido para aplicações que necessitam de uma confiabilidade melhor que o serviço Melhor Esforço. Em uma aplicação típica, uma empresa usa as linhas de um provedor para interconectar a matriz com as filiais distribuídas em várias regiões e quer uma garantia que os pacotes IP transmitidos tenham alta prioridade contanto que não seja excedida a taxa de transmissão limite combinada. Além disso, é permitido à empresa gerar tráfego em excesso, mas com a condição que este tráfego será tratado com menor prioridade.

Quatro classes de serviço (classes AF) foram definidas com diferentes níveis de garantia de transmissão. De acordo com o serviço desejado pelo cliente, o tráfego é classificado para uma das classes AF. Os pacotes de uma classe AF não podem, em qualquer circunstância, ser agregados a pacotes de outras classes. Cada nó DiffServ deve alocar um mínimo de recursos para cada classe AF implementada. Estas classes podem usar mais recursos que o mínimo configurado caso existam recursos disponíveis.

Dentro de cada classe, existem três precedências de descarte, como pode ser visto na tabela 3-1. Em caso de congestionamento, a precedência determina a importância de cada pacote e, desta forma, influi no comportamento de descarte. Em redes com baixa ocorrência de congestionamentos, pode ser interessante implementar somente dois níveis de probabilidade de descarte em cada classe AF.

Roteadores de borda em um domínio DiffServ podem realizar ações de condicionamento dos tráfegos AF que entram e saem. Tais ações podem incluir a modelagem do tráfego, o descarte ou a alteração da precedência de descarte dos pacotes e a remarcação de pacotes para outras classes AF. É importante ressaltar que o condicionador de tráfego não deve reordenar os pacotes de um fluxo específico. Todos os pacotes, estejam dentro ou fora do perfil, são colocados em uma fila para evitar a retransmissão fora de ordem.

	Classe 1	Classe 2	Classe 3	Classe 4
Baixa precedência de descarte	001010	010010	011010	100010
Média precedência de descarte	001100	010100	011100	100100
Alta precedência de descarte	001110	010110	011110	100110

Tabela 3-1: Códigos DSCP das classes AF

3.4 Serviços Diferenciados na Internet2

O projeto Internet2 vem sendo desenvolvido através da cooperação de diversas universidades, empresas e organizações. Um dos principais objetivos da Internet2 é criar uma grande rede de computadores que possa disponibilizar a Qualidade de Serviço necessária para o uso de aplicações de rede mais avançadas, como as usadas para o ensino à distância ou para o controle remoto de instrumentos em operações médicas. Após muita pesquisa, o grupo de trabalho em QoS da Internet2 chegou à conclusão que, atualmente, os Serviços Diferenciados representam a tecnologia de QoS que melhor atende as necessidades da Internet2. Estas necessidades, discutidas em detalhes em [TEI98], podem ser resumidas nos seguintes pontos:

- Suportar aplicações avançadas: os Serviços Diferenciados disponibilizam serviços com diferentes características e níveis de qualidade;
- Ser interoperável: os domínios DiffServ têm boa interoperabilidade porque somente um pequeno grupo de PHBs é implementado e porque os *Bandwidth Brokers* centralizam a administração, abstraindo a maioria das diferenças;
- Ser escalável: a arquitetura de Serviços Diferenciados foi projetada para suportar um grande número de fluxos em conexões de alta velocidade através da agregação destes fluxos e realização de policiamento somente nos roteadores de borda;
- Ser administrável: cada domínio DiffServ é livre para configurar seus procedimentos administrativos (autenticação, autorização e contabilidade), entretanto os acordos bilaterais com os domínios adjacentes devem ser respeitados;
- Prover uma estrutura de medição: os fluxos devem ser medidos para a certificação de que os contratos de performance estão sendo respeitados; e

- Se desenvolver de forma incremental: uma grande qualidade da arquitetura Serviços Diferenciados é que ela pode oferecer serviços úteis antes de estar totalmente implementada.

3.4.1 Projeto QBone

Dentro da Internet2, QBone representa o projeto de um ambiente de testes para os Serviços Diferenciados, formado pela interconexão das redes de algumas instituições ligadas à Internet2. O objetivo do projeto é estudar todos os detalhes necessários para a implementação da arquitetura de Serviços Diferenciados. Neste ambiente, os Serviços Diferenciados são implementados, analisados e refinados por engenheiros de rede e pesquisadores que trabalham em colaboração com os desenvolvedores de novas aplicações de rede. Por se tratar de uma tecnologia bastante nova, ainda existem várias questões que precisam ser definidas. Por exemplo, ainda não está claro como as reservas de recursos entre os domínios devem ser feitas, quais protocolos de comunicação devem ser usados ou como suportar transmissões multiponto.

3.4.2 Serviço QPS

O serviço QPS (*QBone Premium Service*) foi desenvolvido pela Internet2 com o propósito de ser o primeiro serviço implementado no ambiente de testes do projeto QBone. Sua definição é baseada no PHB EF devido à sua simplicidade e capacidade de oferecer baixos níveis de latência, variação de latência e perda de pacotes, além de garantir a disponibilidade da largura de banda reservada para os fluxos. Outra vantagem é que a grande maioria de roteadores existentes implementa algum mecanismo de QoS que pode ser usado para implementar o PHB EF.

Uma reserva QPS é composta pelos seguintes parâmetros: endereço IP e porta do nó fonte, endereço IP e porta do nó destino, protocolo, tempo de início, tempo de término, taxa máxima de transmissão, MTU (*Maximum Transmission Unit*) e variação de latência. Cada reserva representa a disponibilidade por tempo limitado do serviço QPS a uma aplicação para transmissão unidirecional de dados entre dois pontos que podem estar no mesmo domínio DiffServ ou em domínios diferentes. O fluxo de dados deve ser

condicionado através de um TBF (*Token Bucket Filter*) com taxa de fichas igual à taxa máxima de transmissão e tamanho do balde igual ao MTU.

Durante as transmissões, os pacotes dos fluxos com QPS são marcados com o código DSCP do serviço e condicionados para a taxa de transmissão alocada. A marcação e o condicionamento podem ser feitos no primeiro roteador que o fluxo passar (roteador folha) ou no próprio nó fonte, caso ele tenha estas funcionalidades implementadas. Todos os outros roteadores devem usar o código DSCP marcado nos pacotes para priorizar o tráfego QPS sobre os outros tráfegos da rede.

A figura 3-1 mostra como um fluxo do QPS é estabelecido dentro de um domínio DiffServ e enviado através da ligação com outro domínio. Os pacotes saem do nó fonte sem marcação. Assume-se que o roteador folha já esteja configurado para classificar e marcar os pacotes para o QPS (código DSCP 101110) e realizar o condicionamento do fluxo de acordo com as taxas de transmissão e rajada estabelecidas. Com exceção dos roteadores de borda, os roteadores restantes do domínio apenas classificam os pacotes em um dos dois serviços disponíveis (QPS ou Melhor Esforço). Os roteadores de borda processam dois tipos de tráfegos: o tráfego que está saindo do domínio e o tráfego que está entrando. O tráfego de saída pode ser condicionado através de um Acordo de Nível de Serviço (*Service Level Agreement - SLA*) entre os dois domínios. O tráfego de entrada é policiado para certificar que esteja dentro do limite combinado e, caso exceda este limite, é descartado.

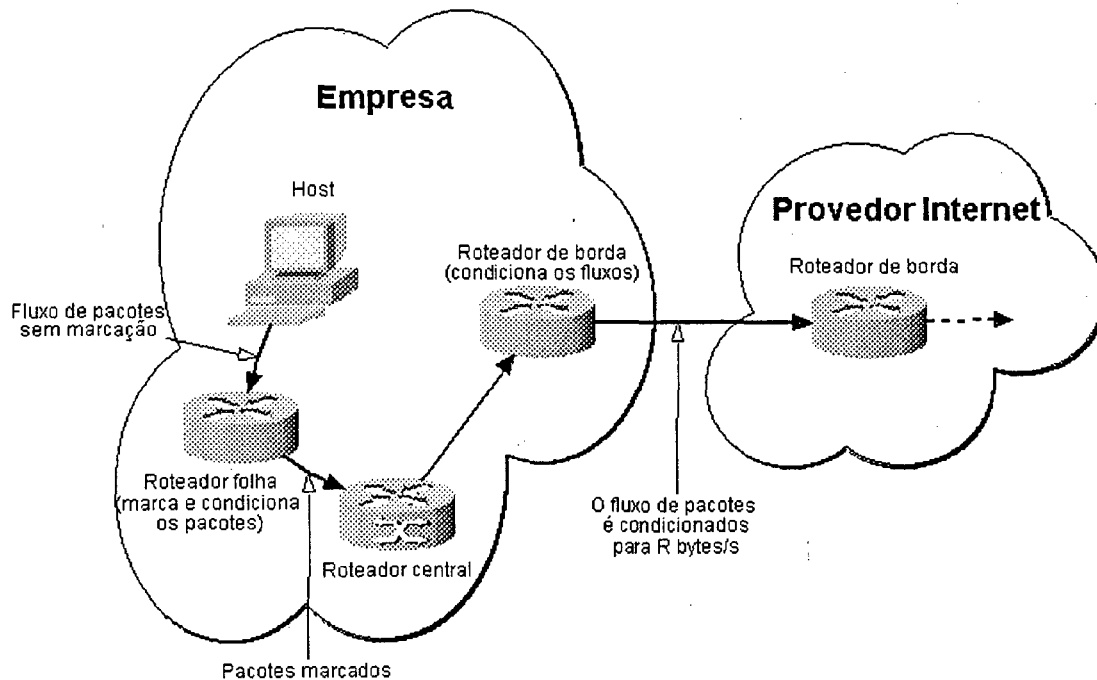


Figura 3-1: Fluxo QPS do cliente para o provedor

3.4.3 Serviço QBSS

O serviço QBSS (*QBone Scavenger Service*) é visto como uma subclasse do serviço Melhor Esforço. Ele utiliza uma pequena parte da largura de banda da rede que pode variar dependendo da quantidade de tráfego do momento. Quando a capacidade da rede estiver sendo subutilizada, o serviço QBSS pode consumir mais largura de banda. Este serviço proporciona uma qualidade menor que o serviço Melhor Esforço, portanto é aplicado somente a aplicações tolerantes a grandes perdas, latências e variações de latência. Exemplos de aplicações adequadas a este serviço seriam:

- Novos tipos de aplicações distribuídas que utilizam a capacidade livre das redes da mesma forma que aplicações computacionais distribuídas utilizam a capacidade livre das CPUs; e
- Aplicações de transferência de grandes quantidades de dados usando TCP, como FTP anônimo, sistemas de *backup*, etc.

3.5 Serviços Diferenciados no Linux

Atualmente, os esforços para prover níveis de serviço de QoS na Internet estão concentrados na arquitetura de Serviços Diferenciados. Para investigar os reais benefícios desta arquitetura, experiências com aplicações reais e implementações de Serviços Diferenciados estão sendo realizadas em diversas partes do mundo. Devido à dificuldade de implementação de novas funcionalidades no hardware de roteadores dedicados, vários trabalhos de pesquisa vêm sendo baseados em implementações de software para plataforma PC (*Personal Computer*). Esta plataforma tem algumas limitações de performance, mas, por outro lado, é bastante flexível pelo fato do roteamento ser feito totalmente em software.

Este trabalho, especificamente, é baseado na implementação dos Serviços Diferenciados para o sistema operacional Linux. O Linux roda em plataforma PC e representa um dos sistemas operacionais com maiores avanços na área dos Serviços Diferenciados, como apresentado em [SAL99], [STR00] e [BRA00]. Desde o *kernel* versão 2.1, o Linux disponibiliza uma grande variedade de funções de controle de tráfego. Nos documentos [ALM99a] e [ALM99b], é apresentado o projeto de implementação dos recursos de controle de tráfego no *kernel* do Linux.

Os princípios básicos envolvidos na implementação de QoS no Linux são mostrados na figura 3-2. Esta figura mostra como o *kernel* processa os dados recebidos da rede e como gera os pacotes para serem enviados na rede. Na entrada, o demultiplexador examina os pacotes para determinar se são destinados ao nó local ou não. Caso sejam destinados ao nó local, são passados para as camadas superiores da pilha de protocolos. Caso contrário, são passados ao bloco “transmissão”, o qual também pode receber das camadas superiores dados gerados localmente. O processo de transmissão inclui várias ações: a seleção da interface de saída, a seleção do próximo elemento da rede, encapsulamento, etc. Após este processo, os dados são enfileirados nas interfaces de saída. É exatamente neste ponto que o controle de tráfego é realizado. A decisão se os pacotes devem ser enfileirados ou descartados, a definição da ordem na qual os pacotes são enviados e a limitação da taxa de envio dos pacotes são alguns exemplos de possíveis ações de controle de tráfego.

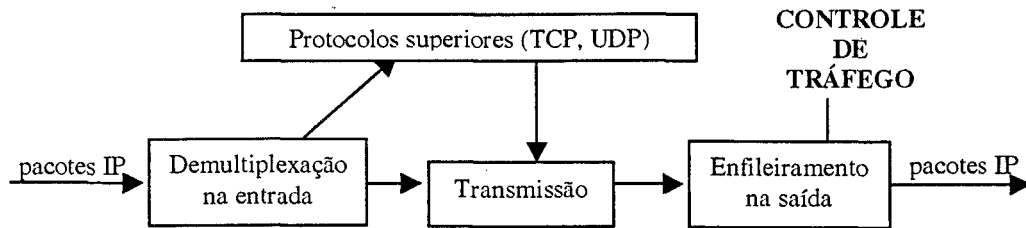


Figura 3-2: Processamento dos pacotes da rede

A implementação do controle de tráfego no *kernel* do Linux consiste, principalmente, de quatro componentes conceituais:

- Disciplinas de enfileiramento;
- Classes (dentro de uma disciplina de enfileiramento);
- Filtros; e
- Policiamento.

Devido a sua flexibilidade, o controle de tráfego do Linux pode ser usado para construir combinações complexas de disciplinas de enfileiramento, classes e filtros. Cada interface de rede tem uma disciplina de enfileiramento associada que controla como os pacotes enfileirados devem ser tratados. Uma disciplina de enfileiramento bem simples pode consistir de uma única fila, onde todos os pacotes são colocados em ordem de chegada e, então, processados o mais rápido possível pela interface. Disciplinas de enfileiramento mais elaboradas, como a da figura 3-3, podem conter filtros para distinguir entre classes de pacotes diferentes e processar cada classe de forma específica. É importante notar que mais de um filtro pode mapear para a mesma classe.

Uma disciplina de enfileiramento é sempre anexada à raiz do dispositivo. Todos os pacotes que saem do dispositivo passam por esta disciplina de enfileiramento. Cada disciplina de enfileiramento pode ter diversos filtros. Nestes filtros, os pacotes são testados de acordo com as regras de filtragem e, quando aceitos, são passados para o destino do filtro específico. Se não existir filtros anexados ao nó ou se todos os filtros falharem, então o pacote é enfileirado. Dependendo da disciplina de enfileiramento escolhida e do conjunto de parâmetros o pacote pode ser enviado imediatamente, enfileirado para posterior processamento ou descartado.

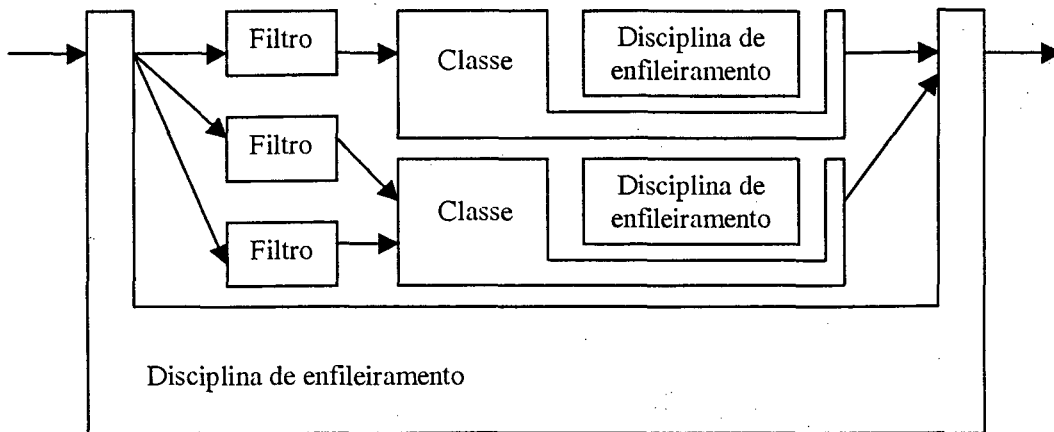


Figura 3-3: Exemplo de disciplina de enfileiramento com filtros e classes

3.5.1 Disciplinas de Enfileiramento

As disciplinas de enfileiramento formam o bloco básico para o suporte a QoS no Linux. Cada interface de rede tem uma fila associada. Existem nove tipos de disciplinas de enfileiramento suportadas atualmente no *kernel* versão 2.4 do Linux. Elas são:

- FIFO (*First-In First-Out*);
- TBF;
- SFQ (*Stochastic Fair Queuing*);
- PQ (*Priority Queuing*);
- CBQ (*Class Based Queueing*);
- HTB (*Hierarchical Token Bucket*);
- CSZ (Clark-Shenker-Zhang);
- DSMARK (*Differentiated Services Marker*); e
- RED.

As disciplinas de enfileiramento são identificadas por um identificador <principal:secundário>, onde o número secundário é zero para filas. Os números são usados para associar as classes às disciplinas de enfileiramento.

As classes e as disciplinas de enfileiramento são componentes bastante ligados. A presença de classes e suas semânticas são propriedades fundamentais das disciplinas de enfileiramento. Em contraste, os filtros podem ser combinados livremente com as classes e disciplinas de enfileiramento. Nem todas as disciplinas de enfileiramento são associadas a classes. Por exemplo, a TBF não tem nenhuma classe associada a si.

Uma das maiores vantagens do suporte a QoS no Linux é a flexibilidade de configuração das disciplinas de enfileiramento e classes. Cada disciplina de enfileiramento pode ter zero ou mais classes. Tais classes usam outra disciplina de enfileiramento para guardar os pacotes, a qual pode ter outras classes. Esta é a flexibilidade que faz do suporte a QoS no Linux tão original.

As funções suportadas nas diversas disciplinas de enfileiramento são: *enqueue*, *dequeue*, *requeue*, *drop*, *init*, *reset*, *destroy* e *dump*. A descrição de cada função pode ser vista em [RAD99].

3.5.2 Classes

Normalmente, cada classe possui uma fila, mas também é possível que várias classes compartilhem a mesma fila. Quando a função *enqueue* da disciplina de enfileiramento é chamada, os filtros desta disciplina de enfileiramento são aplicados em ordem de prioridade para determinar a classe na qual o pacote pertence. Dentro da classe selecionada existe outra disciplina de enfileiramento, portanto a função *enqueue* da classe é chamada. Os pacotes que não foram selecionados por nenhum filtro costumam ser atribuídos a alguma classe padrão.

Existem duas formas para identificar uma classe: via o identificador da classe, o qual é especificado pelo usuário, ou via o identificador interno, o qual é usado dentro do *kernel*. Este identificador interno é atribuído pela disciplina de enfileiramento. O identificador de classe, similarmente ao identificador de disciplina de enfileiramento, é estruturado na forma <principal:secundário>. O número principal corresponde à disciplina de enfileiramento na qual a classe pertence e o número secundário identifica a classe com relação às outras classes da disciplina de enfileiramento.

Nem todas as disciplinas de enfileiramento suportam classes. As que suportam classes são: CBQ, DSMARK, CSZ e PQ. O resto das disciplinas de enfileiramento não suporta classes.

3.5.3 Filtros

Os filtros são usados para classificar os pacotes de acordo com certas propriedades, por exemplo, o campo TOS no cabeçalho do pacote IP, os endereços, os números das portas, etc. São chamados quando a função *enqueue* da disciplina de enfileiramento é executada. As disciplinas de enfileiramento usam filtros para distribuir os pacotes entre as classes disponíveis.

Os filtros são mantidos em listas de filtros, as quais são ligadas às classes ou às disciplinas de enfileiramento, dependendo da forma como a disciplina de enfileiramento foi projetada. As listas de filtros são ordenadas por prioridade em ordem ascendente. Além disso, as entradas nas listas são numeradas pelo protocolo pelo qual se aplicam. Em uma mesma lista de filtros, os filtros para o mesmo protocolo devem ter prioridades diferentes.

Os principais filtros implementados no Linux são:

- Route: classifica os pacotes de acordo com a tabela de roteamento; e
- u32: classifica os pacotes com os endereços IP fonte e destino, as portas TCP/UDP fonte e destino, o campo TOS do cabeçalho IP e o tipo de protocolo usado na camada de transporte.

3.5.4 Ferramenta TC

TC (*Traffic Controller*) é uma aplicação de configuração dos mecanismos de controle de tráfego da plataforma Linux que pode ser usada para configurar serviços da arquitetura de Serviços Diferenciados. TC pode ser usado para configurar vários tipos de filas, classes e filtros e associar estas configurações com as interfaces do computador. Por se comunicar com as funções de rede do *kernel*, esta ferramenta somente pode ser executada por usuários com permissão de *root*.

A operação do TC é realizada através da linha de comando. Seus recursos de descrição de erro são bastante limitados. A descrição de erros de sintaxe é razoável, porém a descrição de erros ocorridos no *kernel* é bem pobre. Para cada comando processado, a ferramenta recebe do *kernel* um número indicando sucesso ou erro. Nos casos de erro, existem três descrições diferentes que podem ser apresentadas ao usuário:

RTNETLINK answers: No such file or directory

RTNETLINK answers: File exists

RTNETLINK answers: Invalid argument

A primeira mensagem de erro costuma significar que um identificador inexistente foi referenciado. A segunda costuma significar que foi tentado adicionar alguma coisa onde o identificador já estava em uso. A última mensagem representa todos os outros erros possíveis. Normalmente não há indicação do que ocorreu, portanto somente a releitura da ajuda e a realização de novas tentativas com pequenas alterações nos parâmetros poderão ajudar.

- Criação de Disciplina de Enfileiramento

Diversas disciplinas de enfileiramento (*queuing disciplines*) podem ser configuradas através da ferramenta TC. Por exemplo, para adicionar uma disciplina de enfileiramento do tipo CBQ na raiz da interface *Fast Ethernet* eth1, o seguinte comando deve ser usado:

```
tc qdisc add dev eth1 handle 1:0 root cbq bandwidth 100Mbit avpkt 1000 mpu 64
```

1 2 3 4 5 6

1. Esta parte diz que se quer adicionar uma disciplina de enfileiramento.
2. O dispositivo no qual se quer adicionar a disciplina de enfileiramento.
3. Este é o identificador que se quer dar à disciplina de enfileiramento.
4. Esta parte significa que se quer anexar à raiz do dispositivo. Somente uma disciplina de enfileiramento pode ser a raiz. Outras alternativas são “ingress” (para pacotes que entram) e “parent CLASS” (a qual especifica o pai desta disciplina de enfileiramento).
5. Este campo especifica o tipo da disciplina de enfileiramento.
6. Estes parâmetros configuram a classe que é automaticamente criada quando se cria uma disciplina de enfileiramento. Os parâmetros mudam de acordo com o tipo de disciplina de enfileiramento escolhida.

- Criação de Classe

Após criar as disciplinas de enfileiramento, normalmente são adicionadas classes a estas disciplinas de enfileiramento para representar os tipos de dados que se deseja

condicionar. Por exemplo, com a disciplina de enfileiramento acima, para criar uma classe restrita a somente 2Mbit, o seguinte comando deve ser usado:

```
tc class add dev eth1 parent 1:0 classid 1:1 cbq ...
      1       2       3       4       5
... bandwidth 2Mbit avpkt 1000 prio 1 rate 1Mbit maxburst 10 bounded isolated
      6
```

1. Esta parte indica o desejo de criar uma classe.
2. O dispositivo no qual se quer adicionar a classe.
3. O identificador do pai da classe. Este parâmetro pode ter significados especiais dependendo da disciplina de enfileiramento usada.
4. O identificador da classe. O identificador principal sempre é o mesmo do pai.
5. O tipo do escalonador da classe. Deve ser sempre o mesmo da disciplina de enfileiramento pai.
6. Os parâmetros para o algoritmo de escalonamento da classe.

- Criação de Filtro

O próximo passo é configurar filtros para que os dados sejam direcionados para as classes corretas. Um filtro bastante usado para aplicações individuais é o u32. Por exemplo, uma configuração deste filtro seria:

```
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 ...
      1       2       3       4       5   6
... match ip dst 10.0.0.0/24 flowid 1:1
      7
```

1. Indica a criação de uma fila.
2. O dispositivo no qual se quer criar o filtro.
3. O pai é a classe na qual este filtro vai ser anexado.
4. O protocolo que será filtrado.
5. Significa que este filtro será processado antes dos filtros com prioridade maior que 1.
6. Indica o tipo do filtro.
7. Mapeia todos os pacotes com destino 10.0.0.0/24 para a classe 1:1.

3.5.5 Disciplina de Enfileiramento CBQ

CBQ [FLO95, FLO96, RIS98] é uma disciplina de enfileiramento bastante flexível baseada em classes de serviço. Após o tráfego ser classificado, são aplicadas as características de cada classe ao tráfego. Uma classe pode conter um fluxo ou uma agregação de fluxos representando aplicações, usuários, departamentos ou servidores diferentes. A classe pode ser definida por informações do cabeçalho do pacote IP (ex.: endereço fonte/destino), por protocolo (ex.: TCP ou UDP) ou por aplicação (ex.: http, ftp, telnet, etc.).

CBQ é uma variação da PQ onde várias filas podem ser definidas. A preferência para cada fila pode ser configurada indicando a preferência do serviço e a quantidade de tráfego enfileirado, medido em bytes, que deve ser processado em cada rotação. Para cada classe, é configurada a taxa limite de largura de banda que pode ser usada. Caso o limite seja ultrapassado, pode-se condicionar o tráfego para a taxa limite ou permitir que a banda seja usada caso haja largura de banda disponível de outras classes.

3.6 Conclusão

A inteligência que a Internet precisa implementar para poder gerenciar seus recursos de tal forma que as transmissões na Internet sejam controladas para receber a qualidade que merecem está sendo especificada em arquiteturas de QoS. O grau de complexidade de uma especificação deste tipo é tão grande que ainda levarão vários anos para que surja uma arquitetura pronta para implementação.

Dentre as arquiteturas de QoS para a Internet que vêm sendo desenvolvidas nos últimos anos, a arquitetura de Serviços Diferenciados vem apresentando os maiores avanços. Sua proposta de disponibilizar serviços com diferentes níveis de qualidade, de acordo com a necessidade de cada momento, e gerenciar os recursos da Internet através domínios administrativos é bastante avançada e com grande potencial de crescimento.

4 BANDWIDTH BROKER E IMPLEMENTAÇÕES EXISTENTES

O *Bandwidth Broker* (BB) [NIC99] é um componente da arquitetura dos Serviços Diferenciados que surgiu com a necessidade de se ter maior controle sobre os recursos disponíveis dentro dos domínios DiffServ. A idéia do BB ainda é bastante recente e as implementações que já foram realizadas ainda precisam de muito aprimoramento. No entanto, os trabalhos a respeito do BB caminham com o firme propósito de padronizar este componente e lançá-lo como um avançado sistema de gerenciamento para os domínios DiffServ.

Segundo [NEI01], o BB deve ter uma arquitetura parecida com a apresentada na figura 4-1. Pode haver situações onde algumas funcionalidades sejam implementadas por outros sistemas e apenas disponibilizadas para o BB, como o roteamento e a monitoração da rede. Os principais componentes dos BBs são as interfaces de comunicação com os roteadores, aplicações e usuários da rede, o sistema de configuração de roteadores e o gerenciamento dos recursos da rede de acordo com políticas pré-definidas.

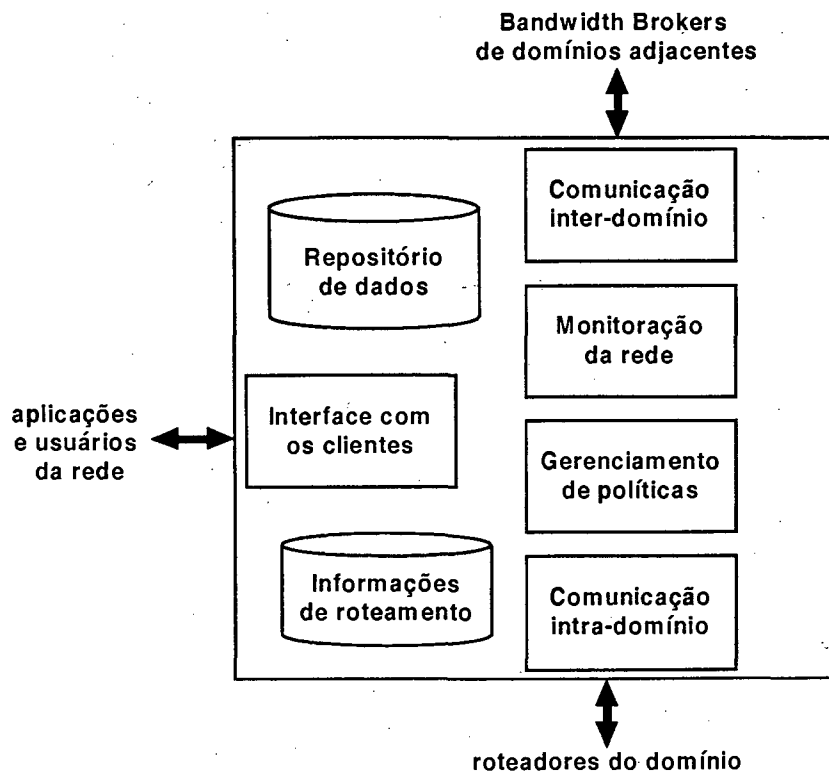


Figura 4-1: Componentes que formam o Bandwidth Broker [NIC99]

Dentro de seu domínio, o BB deve disponibilizar interfaces de operação para as aplicações e usuários da rede poderem requisitar alocações de recursos do domínio. Estas requisições podem ser feitas manualmente no sistema ou através de mensagens de algum protocolo de comunicação, como o RSVP (*Resource Reservation Protocol*). Caso não existam recursos disponíveis, as requisições de alocação são rejeitadas. A comunicação com os nós do domínio também deve existir para que seja possível configurar os mecanismos de QoS destes dispositivos. Os protocolos de comunicação já usados para este fim incluem o COPS (*Common Open Policy Service Protocol*) [DUR00], o *Diameter*, o SNMP (*Simple Network Management Protocol*) e alguns sistemas proprietários.

Externamente, o BB é responsável pela configuração e manutenção dos Acordos de Nível de Serviço (SLAs) com os BBs dos domínios adjacentes para tratar corretamente do tráfego de dados que entra e sai pelos roteadores de borda. Quando há necessidade de alocação de recursos para um fluxo específico, uma requisição de serviço é feita ao BB por algum usuário local ou BB adjacente. Após autenticar o cliente, o sistema verifica se existem recursos suficientes para atender a requisição. Se existirem, estes recursos são alocados e as especificações do fluxo são armazenadas. O BB configura os roteadores do domínio para oferecer ao fluxo de dados específico as características do serviço requisitado. Por motivos de segurança, é crucial que somente o BB possa configurar os roteadores. A figura 4-2 ilustra domínios DiffServ com BBs.

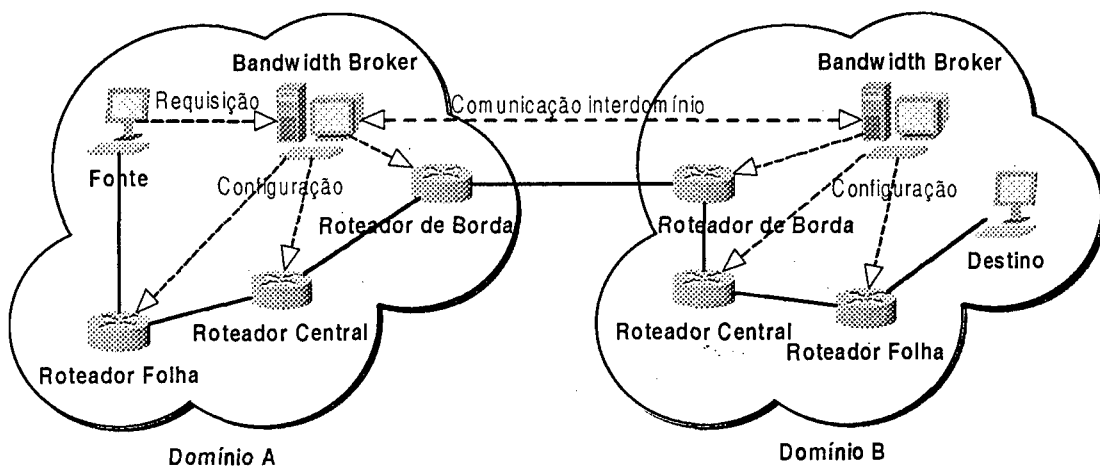


Figura 4-2: Domínios DiffServ com Bandwidth Brokers

O BB deve ter acesso a informações de roteamento para determinar a localização dos roteadores e os caminhos que os fluxos usam para atravessar o domínio. Além de informações de roteamento, o BB requer um repositório para os dados processados por seus componentes. A seguir são citadas algumas informações que podem ser necessárias nos BBs:

- Informações sobre os SLSs com outros domínios;
- Alocações correntes e recursos disponíveis;
- Comandos de configuração dos roteadores;
- Mapeamento de serviços para mecanismos de QoS;
- Políticas de reserva;
- Performance da rede; e
- Autenticação de clientes.

Por serem tão importante na implementação dos Serviços Diferenciados, os BBs têm sido bastante estudados no ambiente de testes do QBone. Os integrantes do QBone planejam introduzir este sistema em seus domínios de forma incremental. Para isso, foram definidas três fases de desenvolvimento:

- Fase 0: Os BBs operam somente dentro do domínio DiffServ. As funcionalidades implementadas são as interfaces de requisição de recursos e a configuração automática dos roteadores do domínio para marcação e condicionamento dos fluxos dos usuários. A configuração de SLSs nos roteadores entre os domínios é feita manualmente pelo administrador da rede.
- Fase 1: Após os BBs ficarem estáveis e estiverem implementados em um bom número de domínios, será incorporado um protocolo de comunicação interdomínio com a capacidade de descobrir se os recursos requisitados ao BB estão disponíveis em todos os domínios que compõem o caminho do fluxo.
- Fase 2: Os BBs desta fase incorporam a capacidade de configuração dinâmica dos roteadores de borda de diversos domínios para satisfazer as necessidades dos usuários. Para que esta fase seja implementada, diversas questões deverão ser esclarecidas, como o método de ajuste dinâmico de SLSs, a integração do protocolo de sinalização com o protocolo de roteamento, o impacto do protocolo de sinalização na carga da rede e capacidade de suportar comunicações multiponto.

Um pequeno grupo de instituições têm investido no desenvolvimento de BBs nos últimos anos. As tecnologias usadas para a implementação destes BBs são as mais variadas. Os protocolos de comunicação e sistemas operacionais usados também diferem bastante. É importante ressaltar que muitos destes sistemas são apenas protótipos, porém alguns poderão virar produtos futuramente. Nas seções seguintes são descritos alguns destes trabalhos que estão na fase 0, isto é, operam somente dentro de um domínio. No momento, alguns estudos e desenvolvimentos da fase 1 estão começando a ser realizados, como no trabalho descrito em [KHA01].

4.1 Siemens

Em Munique, Alemanha, a Siemens desenvolveu um BB [STE00] através de um projeto fundado pela União Européia, chamado AQUILA. Este projeto teve como contribuintes operadoras de serviços de Internet (Bertelsmann, T-Nova e Helsinki Telephone Corporation) e universidades (Technical University Dresden e National Technical University Athens).

O sistema, desenvolvido em Java, oferece uma interface que pode ser acessada por aplicações dos usuários ou por BBs de outros domínios. Através do uso de CORBA¹ (*Common Object Request Broker Architecture*), o BB é acessível por qualquer plataforma, independente do sistema operacional e linguagem de programação. O BB disponibiliza duas interfaces gráficas de operação para os administradores e usuários da rede. Existe uma interface em HTML (*Hiper Text Markup Language*) desenvolvida com Java Servlets² e uma interface em Java Swing³. Ambas interfaces se comunicam com o servidor Web através do protocolo HTTP (*Hiper Text Transfer Protocol*). O servidor Web se comunica com o BB através do protocolo IIOP (*Internet Inter ORB Protocol*).

Nos roteadores de borda, o sistema realiza classificação, marcação e policiamento do tráfego. Em todos os roteadores, a retransmissão dos pacotes é feita de acordo com o sistema de enfileiramento baseado em classes (CBQ). Caso os roteadores façam parte de uma rede ATM (*Asynchronous Transfer Mode*), o sistema aplica funcionalidades do

¹ CORBA é uma arquitetura aberta definida pela OMG com o objetivo de possibilitar o trabalho conjunto entre aplicações através de uma rede de computadores.

² Servlets são módulos que estendem os servidores Web, possibilitando a geração dinâmica de páginas HTML.

ATM para melhorar a qualidade das transmissões. Além disso, o sistema ainda implementa funcionalidades de medição e contabilidade.

A implementação segue uma arquitetura distribuída, onde existe um Agente de Controle de Recursos (ACR) e vários Agentes de Controle de Admissão (ACA). Os Agentes de Controle de Admissão são responsáveis pelas seguintes tarefas:

- Receber as requisições de QoS;
- Checar a identidade e as permissões dos usuários;
- Checar a disponibilidade dos recursos e aceitar ou rejeitar as requisições;
- Configurar os roteadores de borda com as políticas adequadas;
- Requisitar mais recursos ao ACR; e
- Liberar recursos que não são mais necessários.

O Agente de Controle de Recursos implementa as seguintes funcionalidades:

- Limitar o total de tráfego prioritário dentro de um domínio;
- Controlar os elementos da rede para adaptar as reservas de QoS com a carga da rede;
- Aumentar ou diminuir as reservas de QoS entre domínios através da comunicação com os ACAs; e
- Distribuir a largura de banda disponível entre os ACAs, desta forma as requisições de QoS dos usuários podem ser aceitas por estes ACAs sem interação com o Agente de Controle de Recursos;

4.2 Universidade de Kansas

O BB desenvolvido pela Universidade de Kansas [SRE99] foi baseado no documento [REI98] e nos trabalhos da BCIT (*British Columbia Institute of Technology*). Sua implementação foi baseada em uma arquitetura cliente-servidor, composta por base de dados (MySQL), servidor e interface de operação. O servidor e o banco de dados rodam em plataforma Linux. A comunicação entre as aplicações e o BB é realizada através de *sockets* TCP. O BB se comunica com os roteadores da marca Cisco através de

³ Swing é uma biblioteca de classes para criação da interface de aplicações.

mecanismos de *telnet* automatizados. Os mecanismos de QoS usados nos roteadores são o CAR (*Committed Access Rate*) e o WRED.

A base de dados armazena todos os dados relacionados com a funcionalidade do BB dentro de seu domínio. Estes dados incluem SLAs, BARs e mapeamentos de alocações para códigos DSCP. Dados são inseridos na base de dados sempre que uma requisição é aceita.

O servidor é o sistema que realiza a interação entre a base de dados e os diversos clientes. Quando iniciado, o servidor se conecta com os roteadores do domínio, realiza configurações básicas e fica esperando pelas requisições de reserva de recursos. Suas funções são:

- Executar as políticas de alocação para cada requisição recebida dos clientes;
- Se uma requisição for válida, inserir seus dados na base de dados;
- Configurar o roteador de borda quando um SLA é aceito; e
- Configurar os roteadores internos quando uma BAR é aceita.

Existem duas interfaces de operação: uma interface de linha de comando e uma interface HTML. Ambas interfaces de operação possibilitam aos clientes e operadores do sistema interagir com o servidor para configurar SLAs e requisitar Serviços Diferenciados, mais especificamente o serviço Premium.

4.3 Globus

O Projeto Globus⁴ vem investindo em um projeto de pesquisa chamado GARA (*Globus Architecture for Reservation and Allocation*) [ROY00] [SAN00], cujo objetivo é desenvolver uma infraestrutura para suportar requisições de QoS em diversos tipos de recursos, como largura de banda em redes de computadores, CPUs e discos rígidos. Esta infraestrutura contém um BB que implementa muitas das funcionalidades propostas pelo grupo de trabalho Internet2 QoS, tais como controle de admissão e configuração de roteadores para realizar marcação de pacotes, além de funcionalidades adicionais como, por exemplo, autenticação via chave pública. As requisições de reserva de recursos podem ser realizadas através de uma interface desenvolvida em Java ou através de uma API (*Application Programming Interface*). Após serem aceitas, as reservas podem ser

modificadas, canceladas e monitoradas. No momento, somente o estado da reserva é monitorado, porém existem planos para que seja implementado um sistema de monitoração da qualidade dos serviços recebidos pelas aplicações.

O sistema roda em diversas plataformas: Solaris, IRIX, Linux e AIX. Similarmente ao BB da Universidade de Kansas, este BB também configura roteadores da marca Cisco através de mecanismos de *telnet* automatizados. Os mecanismos de QoS usados nos roteadores são o WFQ (*Weighted Fair Queuing*) e o CAR (*Committed Access Rate*). Somente os roteadores folha são configurados pelo BB. Assume-se que os outros roteadores sejam configurados estaticamente.

4.4 UCLA

A UCLA (*University of California at Los Angeles*) tem um laboratório de pesquisas em Internet que vem desenvolvendo diversos trabalhos na área de RSVP e Serviços Diferenciados. Um destes trabalhos foi a implementação de um BB [TER00] composto por um módulo de interação com o banco de dados e um servidor COPS para a comunicação com os roteadores do domínio.

O banco de dados usado é o MySQL. Ele armazena informações sobre todos os fluxos que requisitam um serviço diferenciado ao BB. Estes fluxos podem ter sido gerados no mesmo domínio do BB ou em outro domínio DiffServ. Para cada fluxo, são armazenadas as seguintes informações:

- Interface de entrada no domínio;
- Interface de saída do domínio;
- Recursos requisitados expressados em parâmetros de um TBF;
- Tempo de início da reserva;
- Tempo de término da reserva; e
- Informações auxiliares.

O BB inclui um sistema Web para maior facilidade de acesso ao banco de dados. Através deste sistema, o administrador da rede pode operar sobre as informações dos

⁴ <http://www.globus.org>

fluxos armazenadas. Este sistema Web é baseado na linguagem PHP, cuja integração com o servidor Web Apache e o banco de dados MySQL é bastante fácil.

Após armazenar as informações de alocação dos fluxos, o BB processa estes dados para determinar o total de recursos que deve ser alocado e se comunica com os roteadores para configurar os parâmetros de policiamento e condicionamento. A comunicação é realizada através do protocolo COPS. Neste protocolo, o servidor, também chamado de PDP (*Policy Decision Point*), mantém as informações de configuração relacionadas aos recursos do domínio e instala estas configurações nos clientes, também chamados de PEPs (*Policy Enforcement Points*). Na arquitetura do BB da UCLA, o BB implementa o servidor COPS e os roteadores implementam os clientes COPS.

4.5 BCIT

O Grupo de Tecnologia da Informação do Centro de Tecnologia da BCIT (*British Columbia Institute of Technology*) desenvolveu um projeto de implementação de um BB para a rede CA *net II [BBT98a] [BBT98B] [BBT98C]. Sua arquitetura é composta dos seguintes componentes: banco de dados, servidor, interfaces de linha de comando, servidor de autenticação e servidor administrativo.

O banco de dados MySQL é usado para armazenar todos os dados relacionados à funcionalidade do BB dentro de seu domínio, incluindo SLAs e Requisições de Alocação de Banda (*Bandwidth Allocation Requests – BARs*).

O servidor é um sistema multiusuário que interage com o banco de dados, as interfaces de linha de comando e o servidor de autenticação. Através do servidor, os clientes das interfaces de linha de comando e o servidor de autenticação acessam as informações armazenadas no banco de dados. Todas as informações recebidas que requisitam alguma modificação no banco de dados são processadas de acordo com as políticas do domínio e gravadas em um arquivo. O servidor é configurado através de um arquivo de configuração e de uma série de arquivos que definem as primitivas de cada roteador usado dentro do domínio.

A interface de linha de comando e a interface gráfica disponibilizam uma série de comandos para os operadores interagirem com o servidor. Estes comandos são transmitidos ao servidor através do protocolo BBTP (*Bandwidth Broker Transfer*

Protocol), um protocolo desenvolvido pela própria BCIT. De acordo com o resultado da execução dos comandos, uma mensagem de sucesso ou falha é retornada ao operador.

4.6 Merit

A provedora de acesso a Internet Merit desenvolveu um BB [EVA99] composto por diversos componentes. O módulo de políticas recebe as requisições de recursos dos clientes através do protocolo *Diameter*⁵ e verifica se estão de acordo com as políticas de alocação do domínio. As requisições aceitas são passadas para o módulo de alocação, cuja função é configurar os roteadores pertencentes à rota de transmissão. O módulo de alocação acessa o sistema de roteamento OSPF⁶ (*Open Shortest Path First*) do domínio para determinar as rotas dos fluxos. Após o processamento de cada requisição, uma mensagem de sucesso ou falha é retornada ao requerente. O BB não aceita requisições que ultrapassem o limite de recursos definidos pelas políticas de alocação. Além de requisitar a alocação de novos recursos, os clientes também podem requisitar a modificação ou liberação dos recursos que já possuem.

Os roteadores suportados são PCs rodando FreeBSD com o pacote ALTQ⁷ instalado. Trabalhos futuros talvez sejam desenvolvidos com roteadores Cisco. O policiamento, marcação e condicionamento dos fluxos são realizados nos roteadores folha através do mecanismo CBQ. Os roteadores centrais são configurados estaticamente também com CBQ para diferenciar o tráfego através do DSCP de cada pacote.

4.7 Telia

O BB da Telia [SHC99] disponibiliza o serviço Premium para seus clientes. As requisições do serviço podem ser feitas pelos usuários através de uma interface Web ou por outras aplicações através de *sockets* TCP. O BB possibilita a programação do tempo de início e fim de cada alocação de recursos. Por isso, o BB mantém todas as informações

⁵ Diameter é um protocolo de autenticação, autorização e contabilidade definido pelo Grupo de Trabalho AAA do IETF

⁶ OSPF é um protocolo de roteamento baseado no estado das conexões.

⁷ ALTQ (*Alternate Queueing for BSD UNIX*) é um pacote que contém diversas filas (CBQ, WFQ, RED, etc.), o protocolo RSVP e a base da arquitetura DiffServ.

relacionadas com as reservas e com o estado dos roteadores do domínio. Estes roteadores são descobertos através dos protocolos OSPF e SNMP e configurados via SNMP. Tanto o BB quanto os roteadores rodam em PCs com sistema operacional FreeBSD.

4.8 Comparação Entre os Bandwidth Brokers

Os *Bandwidth Brokers* desenvolvidos até hoje apresentam grandes diferenças em relação às tecnologias usadas na implementação de suas funcionalidades. A principal razão para essa diversidade de tecnologias se deve, principalmente, à inexistência de uma especificação completa sobre a operação do sistema e a dúvidas sobre as tecnologias mais adequadas para este tipo de sistema. Desta forma, cada projeto pode ser visto como uma proposta diferente sobre como o sistema deve ser implementado. Uma comparação entre as tecnologias escolhidas pelos principais projetos de BB existentes é apresentada na tabela 4.1. As características comparadas foram: forma de comunicação entre usuários e BB, forma de comunicação entre BB e roteadores, tipo de roteadores usados, PHB disponíveis, sistema operacional do BB e mecanismos de QoS usados nos roteadores. A comunicação entre BBs não foi implementada por nenhum projeto devido à falta de definições sobre o assunto na época do desenvolvimento.

	Comunicação entre clientes e BB	Comunicação entre BB e roteadores	Tipo de Roteadores	Mecanismos de QoS	PHBs disponíveis
Siemens	HTTP / IOP	IOP	Estações de trabalho Sun	CBQ	EF
Univ. de Kansas	HTTP / sockets TCP	telnet automatizado	Cisco	CAR e WRED	EF e AF
Globus	RSVP	telnet automatizado	Cisco	WFQ e CAR	EF
UCLA	HTTP	COPS	PCs com FreeBSD	CBQ	EF
BCIT	BBTP	telnet manual	Cisco	?	EF
Merit	<i>Diameter</i>	<i>Sockets</i> TCP	PCs com FreeBSD	CBQ	EF
Telia	HTTP / <i>Sockets</i> TCP	SNMP	PCs com FreeBSD	?	EF

Tabela 4-1: Comparação entre Bandwidth Brokers

4.9 Conclusão

A comunicação entre os usuários e o BB é uma área onde há muitas opções de protocolos e ainda não existe um consenso sobre qual é o melhor. Os protocolos de comunicação já usados para requisição de recursos são o BBTP, o RSVP, o *Diameter*, o IOP e o HTTP, além de implementações proprietárias baseadas em *sockets* TCP. Dos sete BBs apresentados, três disponibilizam uma interface Web para os usuários. Nestes casos, as requisições vão via HTTP até o servidor Web e via algum outro protocolo até o BB. Esta é uma boa opção devido à facilidade de acesso às aplicações Web, isto é, de qualquer ponto da Internet é possível interagir com o BB. O protocolo RSVP tem sido foco de pesquisa ultimamente para avaliar suas possibilidades de integração com a arquitetura de Serviços Diferenciados. O objetivo desta integração seria aproveitar as melhores características das arquiteturas IntServ (*Integrated Services*) e DiffServ, isto é, a capacidade de configuração do protocolo RSVP e a escalabilidade da estratégia de priorização da arquitetura DiffServ. O protocolo *Diameter* também é uma opção interessante na comunicação entre usuários e BBs devido às suas capacidades de autenticação, autorização e contabilidade.

Outra comunicação importante dos domínios DiffServ ocorre entre o BB e os roteadores. Para esta comunicação também já foram testadas várias formas de comunicação: IOP, COPS, SNMP, *sockets* TCP e telnets automatizados. Nos trabalhos da Universidade de Kansas, BCIT e Globus a comunicação com roteadores Cisco foi realizada através de telnets automatizados. A razão do uso desta estratégia se deve ao uso da interface de linha de comando para configurar os equipamentos. Em outros projetos, onde os roteadores foram estações de trabalho ou PCs, houve maior flexibilidade na escolha do mecanismo de comunicação. As implementações com *sockets* TCP se deveram, principalmente, à rapidez de desenvolvimento, porém, como os telnets automatizados, não representam padrões de comunicação. Por outro lado, os protocolos COPS e SNMP, especificados pelo IETF, estão representando as maiores apostas nesta área no momento.

O grupo de trabalho SNMPConf do IETF vem realizando esforços para adaptar o protocolo SNMP à arquitetura de gerenciamento de políticas definida pelo grupo de trabalho *Policy Framework*, também do IETF. O objetivo é demonstrar que o SNMP

pode ser usado para o gerenciamento de configurações. Os trabalhos que vêm sendo realizados compreendem recomendações sobre como as configurações de políticas devem ser realizadas e especificações de MIBs necessárias para facilitar estas configurações. No momento, o grupo de trabalho SNMPCConf vem trabalhando em cooperação com o grupo de trabalho DiffServ do IETF no desenvolvendo de uma MIB (*Management Information Base*) para possibilitar a configuração e monitoração dos dispositivos da arquitetura de Serviços Diferenciados através do protocolo SNMP.

Os grupos de trabalho *Policy Framework* e RAP (*Resource Allocation Protocol*) do IETF vem concentrando seus trabalhos na criação de uma arquitetura de gerenciamento que possibilite representar, gerenciar, compartilhar e reusar políticas de alocação de recursos. Quando necessário, estas políticas de alocação de recursos são traduzidas em instruções de configuração para serem transmitidas aos dispositivos da rede através do protocolo COPS. As informações de configuração, usuários, dispositivos e aplicações são armazenadas em um sistema de diretórios LDAP (*Lightweight Directory Access Protocol*) para que várias aplicações possam compartilhar estes dados. No grupo de trabalho DiffServ está sendo especificada uma PIB (*Policy Information Base*) para o gerenciamento das políticas da rede através do protocolo COPS. A PIB DiffServ pode ser vista como uma abstração da MIB DiffServ, cujas informações armazenadas são, basicamente, regras de policiamento para os serviços diferenciados.

Os roteadores usados nos desenvolvimentos dos BBs se resumem a equipamentos Cisco ou computadores com software de roteamento instalado. Dos sete projetos pesquisados, três usaram roteadores Cisco, três usaram PCs com sistema operacional FreeBSD e um usou estações de trabalho Sun. Os PCs são uma boa opção para a criação de pequenas redes porque são razoavelmente baratos e toda a funcionalidade necessária pode ser realizada em nível de software. Por outro lado, os roteadores da marca Cisco ou de outra grande empresa de equipamentos de comunicação fazem parte de praticamente todas as redes de médio e grande porte. É importante que ambos tenham bom suporte a QoS porque a arquitetura de Serviços Diferenciado depende do suporte de cada roteador da rede.

Outra característica das implementações de BB analisadas se refere aos mecanismos de QoS que foram usados nos roteadores para implementar o PHB EF. Entre os projetos que usaram roteadores Cisco, os mecanismos usados foram o CAR (*Committed Access*

Rate), o WFQ e o WRED. O CAR realiza classificação, medição e policiamento de tráfego. O WFQ e o WRED são filas que podem ser usadas com o CAR para implementar os Serviços Diferenciados. No Unix, todos os projetos usam a fila CBQ para implementar o PHB EF. CBQ é um sistema de filas bastante avançado que não requer o auxílio de outro mecanismo para implementar a classificação, marcação, condicionamento e enfileiramento do tráfego.

O único serviço que está sendo desenvolvido atualmente é o QPS (QBone Premium Service), baseado no PHB EF. A tendência é que o serviço QPS seja o primeiro serviço diferenciado disponível para os usuários da Internet2.

5 O SISTEMA DE GERÊNCIA

5.1 Funcionalidades

O sistema de gerência desenvolvido neste trabalho implementa as principais funcionalidades de um BB necessárias para disponibilizar o serviço Premium em um domínio DiffServ: interface com o usuário, configuração dos nós do domínio e medição ponto-a-ponto. A operação do gerenciador é feita exclusivamente através de uma interface Web, possibilitando o acesso de qualquer computador da rede. O gerenciador se comunica com os agentes de descoberta de rotas e de configuração localizados nos nós do domínio para disponibilizar o serviço Premium requisitado pelos usuários. Além disso, o sistema também se comunica com os agentes de medição localizados nos computadores dos clientes com o objetivo de coletar informações sobre a qualidade do serviço disponibilizado. Uma visão geral de como o sistema opera é apresentada na figura 5-1.

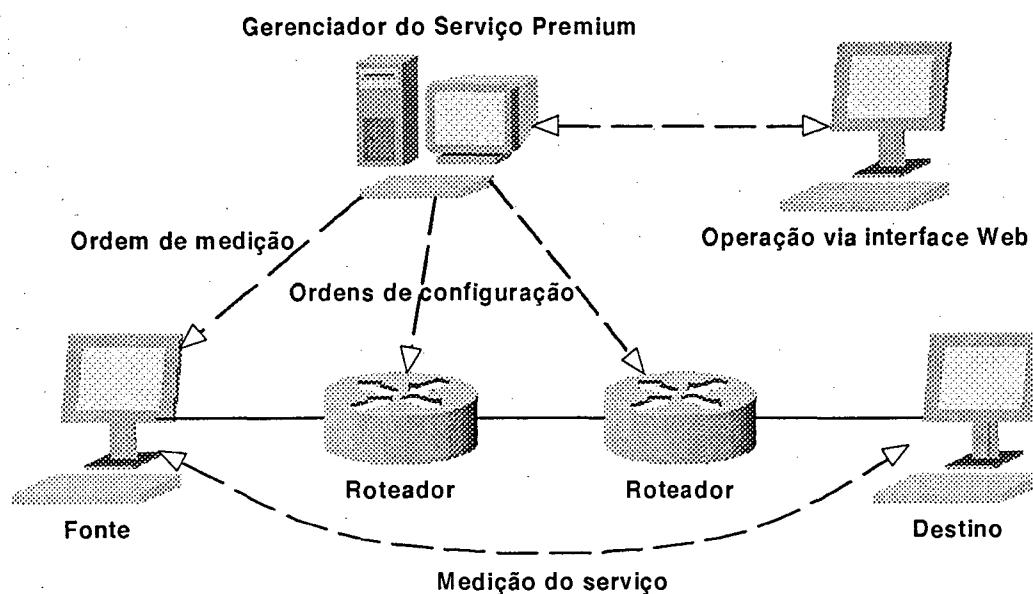


Figura 5-1: Operação do sistema

Através de uma interface Web, os usuários da rede têm a possibilidade de realizar requisições do serviço Premium, monitorar a qualidade das transmissões e monitorar a quantidade de banda alocada em cada roteador do domínio. Todos os roteadores do

domínio devem ter um agente de configuração rodando. Este agente costuma ser chamado de PEP (*Policy Enforcement Point*) e representa a parte do sistema que trabalha diretamente com os mecanismos de controle de tráfego dos roteadores. Os computadores dos usuários devem ter o agente de descoberta de rotas, o agente de configuração para marcar os pacotes Premium e o agente de medição para realizar as medições ponto-a-ponto.

5.2 Interface com o Usuário

A tela de operação do sistema, apresentada na figura 5-2, se divide em quatro áreas: formulário de requisição do serviço Premium, formulário de requisição de medição, lista de clientes e lista de roteadores. Para requisitar a disponibilização do serviço Premium a um determinado fluxo, o usuário deve especificar o endereço IP do nó fonte, o endereço IP do nó destino e a largura de banda que deverá estar sempre disponível. O armazenamento e checagem das políticas de alocação do domínio são funcionalidades que ainda não existem no sistema. Por esta razão, a configuração dos roteadores é realizada sempre que os dados fornecidos forem válidos e o nó fonte estiver cadastrado como cliente do sistema. O sistema tem dois tipos de configuração para os nós do domínio, o computador fonte do tráfego é configurado com um marcador de pacotes para o DSCP do serviço Premium e um condicionador de tráfego para não deixar o usuário transmitir mais do que requisitou. Os roteadores por onde o fluxo vai passar aumentam a largura de banda disponível para o serviço Premium de acordo com o valor requisitado pelo usuário. Em caso de sucesso na operação, é apresentada uma mensagem de confirmação na tela e a alocação é incluída na lista de alocações do cliente. É importante salientar que os usuários devem cadastrar seus computadores como clientes do gerenciador através da ativação dos agentes antes de fazer qualquer requisição do serviço Premium para suas transmissões.

Gerente de Serviços Diferenciados

Domínio UFSC

Requisição de Serviço Premium

Endereço IP Fonte: Endereço IP Destino: Largura de Banda: Kb

Requisição de Medição

Endereço IP Fonte: Endereço IP Destino: Protocolo: TCP UDP Tamanho dos pacotes: bytes Método: One-way Round-trip

Clientes Cadastrados e Respectivas Alocações

Cliente	Destino	Banda Alocada	Latência min/med/max (ms)	Variação de Latência min/med/max (ms)	Parotes Perdidos (%)	
192.168.2.2	192.168.3.1	100	1.451/4.161/17.243	0/0.568/7.912	0	<input type="button" value="desalocar"/>
192.168.3.2						<input type="button" value="desalocar"/>

Rotadores Cadastrados

Interface	Banda Alocada para o Serviço Premium
192.163.2.1	0
192.163.3.1	100

Figura 5-2: Interface do gerente de Serviços Diferenciados

O segundo formulário da tela é usado para verificar a garantia efetiva do serviço através da requisição de medições entre dois nós clientes que já estão usufruindo a qualidade do serviço Premium. O sistema de medição permite a escolha do tamanho dos pacotes, tipo de protocolo de transporte e método de medição. Esta flexibilidade é muito importante para a obtenção de resultados precisos porque possibilita a realização das medições com pacotes similares aos usados nas transmissões do cliente. Por exemplo, se o cliente estiver transmitindo pacotes UDP de 1024 bytes para o destino e ambos nós estiverem sincronizados com um servidor NTP (*Network Time Protocol*) [MIL92], é indicado que a medição use pacotes UDP de 1024 bytes e o método de medição *one-way*. Caso os nós não estejam sincronizados, deve ser usado o método *round-trip*.

A terceira área compreende uma tabela com todos os clientes do sistema e suas respectivas alocações. No momento que o usuário inicia o agente de configuração em seu computador, este agente cadastra o computador como cliente do gerenciador e fica esperando as mensagens de configuração. Juntamente com cada cliente, são apresentados

os endereços dos destinos de suas transmissões, a quantidade de banda reservada para cada fluxo e os resultados da última medição realizada. Na última coluna da tabela, existe um *link* para requisitar a desalocação dos recursos entre o cliente e cada um de seus destinos.

A quarta e última parte do sistema foi criada para apresentar informações sobre as interfaces dos roteadores do domínio. Para cada interface, o sistema apresenta a quantidade de largura de banda alocada para o serviço Premium.

5.3 Conclusão

O sistema de gerência para o serviço Premium proporciona a seus usuários uma interface de operação com diversas funcionalidades. A requisição do serviço Premium ao sistema é bastante simples, facilitando a operação por qualquer usuário da rede. A requisição de medição requer o conhecimento do usuário em relação a: tamanhos de pacotes, protocolos de transmissão e formas de sincronização de relógios. As informações apresentadas na tela sobre os recursos reservados são de grande valia para os administradores de rede.

6 IMPLEMENTAÇÃO DO SISTEMA DE GERÊNCIA

6.1 Tecnologias Usadas e Justificativas

A definição da arquitetura do sistema foi, em parte, baseada nas características de outros sistemas similares, apresentados na seção 4. As opções de tecnologia disponíveis e o tempo limitado para o desenvolvimento foram os principais fatores que influenciaram a definição da arquitetura do sistema. A tabela 6-1 apresenta as tecnologias escolhidas para o desenvolvimento do sistema.

Tecnologias usadas no sistema	
Comunicação com os usuários	HTTP
Comunicação com os roteadores	<i>Sockets</i> TCP
Roteadores	PCs com Linux
PHBs disponíveis	EF
Mecanismos de QoS usados nos roteadores	CBQ e RED

Tabela 6-1: Tecnologias escolhidas para o desenvolvimento do sistema

O sistema foi desenvolvido como uma aplicação Web, portanto, o protocolo usado para realizar a comunicação entre os usuários e o sistema foi o HTTP. A linguagem de programação usada no desenvolvimento do sistema Web foi o JSP (*Java Server Pages*), devido à sua fácil integração com outros módulos do sistema desenvolvidos em Java. Como o sistema é centralizado e os usuários estão distribuídos pelo domínio DiffServ, o fato do sistema ter interface Web facilita muito o seu acesso. Através de qualquer computador conectado a rede, é possível realizar requisições de recursos ou monitorações da qualidade das transmissões. Quando a arquitetura de Serviços Diferenciados evoluir mais e as aplicações de rede suportarem esta tecnologia, algum protocolo específico poderá ser usado para as aplicações requisitarem serviços disponíveis na rede. O protocolo RSVP é visto como uma boa alternativa para esta situação.

A comunicação entre o gerenciador e os nós DiffServ foi implementada através de *sockets* TCP devido à sua simplicidade e facilidade de uso. Foram definidas algumas mensagens para o gerenciador requisitar a execução de ações nos agentes. Os agentes de descoberta de rotas recebem o endereço IP do nó destino para traçar uma rota. Os agentes

de configuração nos nós folhas recebem o endereço IP do destino para configurar um classificador e a largura de banda para configurar um condicionador. Os agentes de configuração nos nós centrais recebem o endereço IP de uma de suas interfaces e a largura que deve ser reservada aos pacotes Premium. Os agentes de medição recebem os endereços IP fonte e destino, o protocolo de transporte, o tamanho do pacote e o método de medição.

Outras opções de comunicação são os protocolos COPS e SNMP. O protocolo COPS segue o modelo cliente/servidor para o controle de políticas e se adapta muito bem na gerência dos Serviços Diferenciados. Existe um *Internet-Draft* no grupo de trabalho DiffServ que define uma PIB para gerenciamento de políticas em domínios DiffServ [FIN01] e sugere que o COPS seja o protocolo de transmissão das políticas para os dispositivos da rede. O protocolo SNMP vem fazendo parte de trabalhos recentes do grupo de trabalho DiffServ na definição de uma MIB para dispositivos que implementam DiffServ [BAK01]. Futuramente, o SNMP poderá ser usado tanto para monitoração como para configuração dos roteadores. A razão de não usar o protocolo COPS na comunicação entre o sistema e os roteadores se deveu à complexidade do protocolo e, conseqüentemente, ao grande acréscimo de tempo que seria inserido no cronograma de desenvolvimento.

Os tipos de roteadores disponíveis para o desenvolvimento do trabalho foram PCs com sistema operacional Linux. O uso de PCs como roteadores não representa a realidade da Internet porque estes equipamentos são limitados em relação à velocidade de processamento de pacotes. Os PCs costumam ser usados como *gateways* entre redes locais e redes centrais. A interconexão de grandes redes é quase totalmente realizada através de roteadores proprietários. Neste trabalho, o uso de PCs com Linux foi bastante satisfatório porque o Linux é um sistema operacional com um *kernel* muito avançado em relação a mecanismos de controle de tráfego. O Linux disponibiliza diversos tipos de filas e total suporte aos Serviços Diferenciados. A fila mais avançada é a CBQ, capaz de realizar todas as funcionalidades necessárias para a implementação dos Serviços Diferenciados: classificação, marcação, enfileiramento e condicionamento de pacotes. Outra alternativa seria usar uma combinação de mecanismos de controle de tráfego, como uma fila PQ para priorizar parte do tráfego e um condicionador TBF para limitar a taxa de transmissão.

O serviço implementado neste trabalho foi o serviço Premium. Ele é baseado no PHB EF e tem como objetivos principais: assegurar uma taxa de transmissão máxima e manter a variação de latência do tráfego bem baixa. Este comportamento só é atingido se todos os roteadores por onde passam os pacotes estão devidamente configurados. É por isso que um sistema de gerenciamento é tão necessário para controlar as requisições do serviço Premium e as reservas de recursos em cada roteador do domínio.

6.2 Arquitetura do Sistema

O gerenciador é composto por uma parte servidora e uma parte cliente. A parte servidora é representada por uma aplicação Web responsável pela disponibilização do serviço Premium através do gerenciamento dos recursos do domínio. A parte cliente roda nos nós do domínio e é responsável pelas configurações e medições. Uma visão geral desta arquitetura distribuída é apresentada na figura 6-1.

A aplicação Web necessita de um servidor de aplicações com suporte à tecnologia JSP. Durante o desenvolvimento e teste do sistema, foi usado o servidor Tomcat⁸ versão 3.2. A tecnologia JSP é direcionada para o desenvolvimento de aplicações Web e possibilita o uso de todas as funcionalidades Java existentes no JDK (*Java Development Kit*). A aplicação Web foi desenvolvida em plataforma Linux, porém pode rodar em qualquer sistema operacional que possua um servidor de aplicação com suporte a JSP e um JDK versão 1.2 ou superior. Exemplos de classes do JDK utilizadas neste sistema são a classe *HashTable* para armazenamento das reservas e a classe *Socket* para comunicação com os agentes.

O agente de descoberta de rotas utiliza a aplicação *Traceroute* para determinar a rota entre dois nós do domínio. Este agente, implementado em Java, recebe requisições de descoberta de rota geradas pela aplicação Web através de *sockets* TCP. A execução do *Traceroute* é realizada através da invocação de uma função localizada em uma biblioteca em C++. O uso desta biblioteca é necessário devido à impossibilidade do Java em acessar recursos nativos do sistema operacional.

O gerenciador implementado neste trabalho suporta a configuração de dois tipos de dispositivos dentro do domínio: os nós folhas e os nós centrais. Os nós folhas são

⁸ <http://jakarta.apache.org/tomcat>

representados pelos computadores dos usuários que se cadastram como clientes do sistema e fazem o papel de fonte ou destino das transmissões de pacotes Premium. Os nós centrais são representados pelos roteadores do domínio que interligam os nós folhas. Em comunicações interdomínio, os roteadores que interligam dois domínios são chamados de roteadores de borda. Eles possuem a função de garantir os SLAs entre os domínios. A configuração de roteadores de borda e a comunicação com BBs de outros domínios são funcionalidades não suportadas pelo gerenciador.

O agente de configuração é formado por um sistema em Java que recebe as ordens de configuração da aplicação Web via *sockets* TCP e usa uma biblioteca em C++ para configurar classificadores, marcadores e condicionadores no Linux. Ambos clientes e roteadores devem rodar este agente de configuração, porém cada um usa um grupo diferente de funções de configuração pelo fato de realizarem configurações distintas, isto é, os clientes usam classificadores, marcadores e condicionadores para fluxos de dados específicos e os roteadores usam classificadores e condicionadores para a agregação dos fluxos de pacotes Premium.

O agente de medição é formado por um sistema transmissor e um sistema receptor. O transmissor usa Java para receber as ordens de medição via *sockets* TCP e C++ para enviar os pacotes de medição ao receptor. O C++ é utilizado no agente de medição devido à necessidade do sistema em processar os pacotes rapidamente para não prejudicar a precisão das medições. O receptor é implementado totalmente em C++ e não realiza nenhuma comunicação com a aplicação Web. Sua execução é representada por quatro processos esperando mensagens do transmissor para realizar medições de acordo com uma das combinações de protocolo (TCP ou UDP) e método de medição (*one-way* ou *round-trip*).

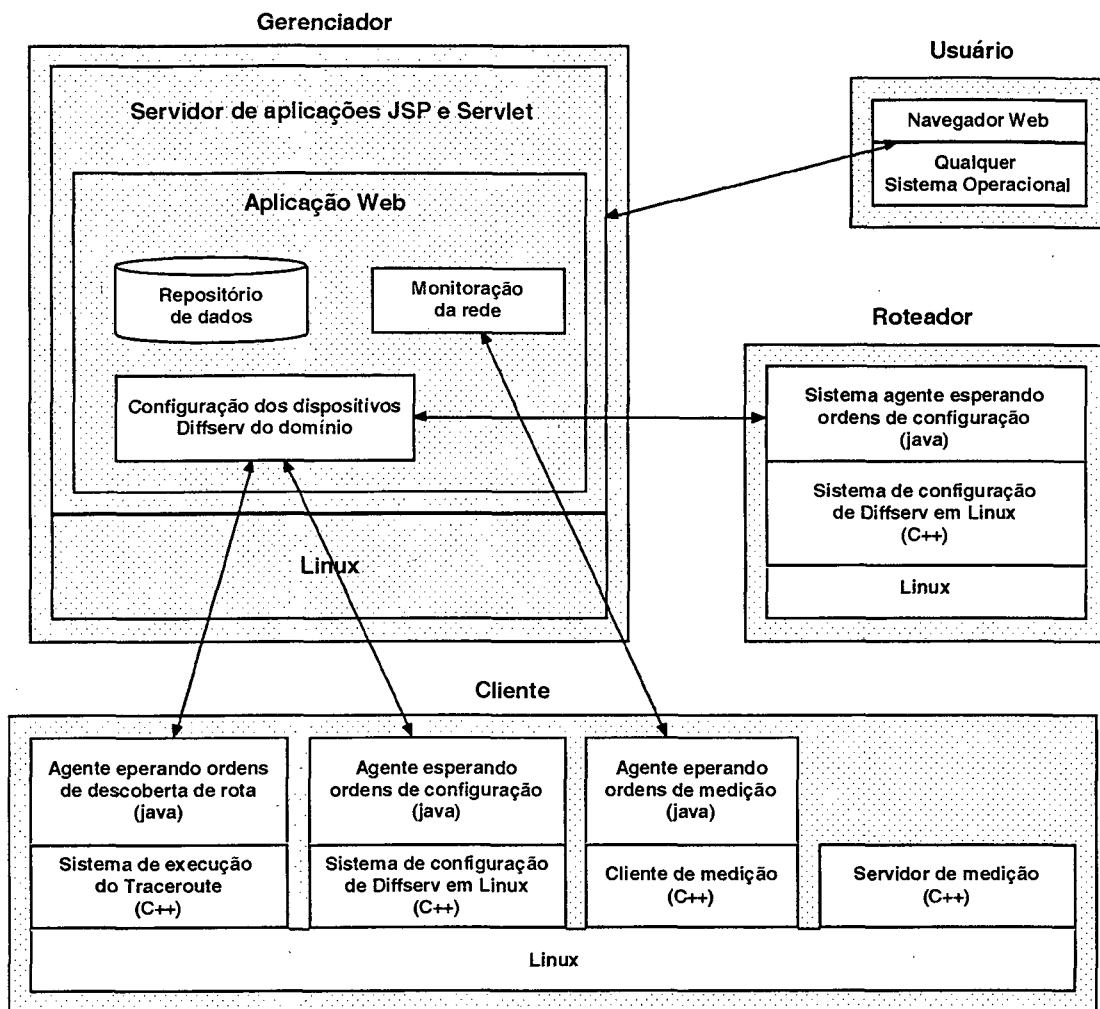


Figura 6-1: Arquitetura do gerente de Serviços Diferenciados

6.3 Agente de Descoberta de Rotas

A estratégia de descoberta de rotas utilizada neste trabalho foi o uso da ferramenta *Traceroute* por agentes localizados nos nós folha do domínio. Esta ferramenta, como o próprio nome diz, traça uma rota entre o nó em que está rodando e outro nó da rede informado como parâmetro. O agente de descoberta de rotas faz a interface entre o gerenciador e o *Traceroute*.

As configurações de priorização do tráfego Premium são realizadas nas interfaces de saída dos nós pertencentes à rota de transmissão entre a fonte e o destino. Por exemplo, as configurações do serviço Premium para transmissões do nó 1 para o nó 3 da figura 6-2 devem ser realizadas na interface 192.168.2.2 do nó 1 e na interface 192.168.3.1 do nó 2.

Com o objetivo de verificar quais interfaces dos nós da rede o *Traceroute* apresenta, a ferramenta foi executada para traçar a rota entre os nós 1 e 3. O resultado apresentado na tabela 6-2 listou a interface de entrada dos pacotes no nó 2 (192.168.2.1), sendo que a interface que receberá configurações é a interface de saída (192.168.3.1). Para gerar as informações corretas, o *Traceroute* deve ser executado no sentido inverso das transmissões de pacotes Premium, gerando os resultados da tabela 6-3. Obviamente, estes dados devem ser lidos também no sentido inverso.

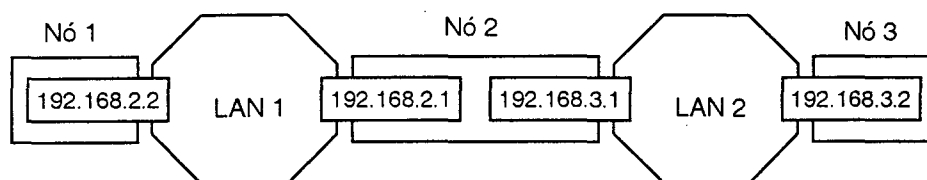


Figura 6-2: Ambiente de testes com a ferramenta *Traceroute*

```

# traceroute 192.168.3.2
traceroute to 192.168.3.2 (192.168.3.2), 30 hops max, 40 byte packets
 1 node2.deggy (192.168.2.1) 0.805 ms 0.248 ms 0.162 ms
 2 node3.deggy (192.168.3.2) 2.748 ms 2.187 ms 2.573 ms
  
```

Tabela 6-2: Resultado do *Traceroute* entre 192.168.2.2 e 192.168.3.2

```

# traceroute 192.168.2.2
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 40 byte packets
 1 node2.deggy (192.168.3.1) 0.802 ms 0.554 ms 0.445 ms
 2 node1.deggy (192.168.2.2) 2.801 ms 2.173 ms 2.241 ms
  
```

Tabela 6-3: Resultado do *Traceroute* entre 192.168.3.2 e 192.168.2.2

O agente de descoberta de rotas é implementado em Java e, através do JNI, usa uma biblioteca de funções escrita em C++ para executar a ferramenta *Traceroute*. Ele espera por requisições da aplicação gerente em uma porta específica do sistema. Para cada requisição recebida, o *Traceroute* é executado e seu resultado é transmitido ao gerente. O resultado do *Traceroute* é processado adequadamente para gerar uma lista de endereços organizada na ordem correta antes de ser enviado ao gerenciador.

6.4 Agente de Configuração

O agente de configuração é representado por um sistema em Java que usa uma biblioteca de rotinas escrita em C++ para acessar os mecanismos de controle de tráfego do Linux. Basicamente, a operação do sistema é esperar as requisições de configuração da aplicação gerente e, de acordo com o tipo de configuração, executar uma das rotinas disponíveis.

Os nós clientes e os roteadores possuem agentes de configuração com funcionalidades distintas, de acordo com o tipo de configuração que deve ser realizada em cada dispositivo. Estas diferentes configurações são ilustradas nas figuras 6-3 e 6-4. Os clientes necessitam de classificadores, marcadores e condicionadores para cada fluxo de pacotes Premium. Os roteadores necessitam de um único classificador e condicionador para a agregação de todos os fluxos de pacotes Premium que passam por cada uma de suas interfaces.

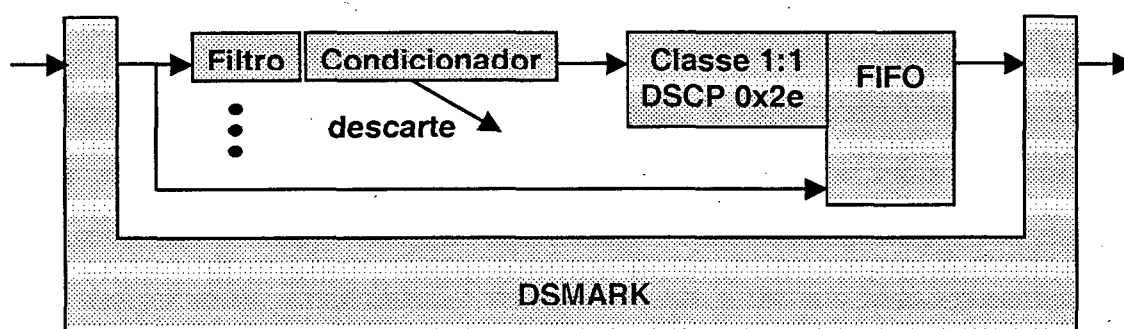


Figura 6-3: Configuração nos clientes

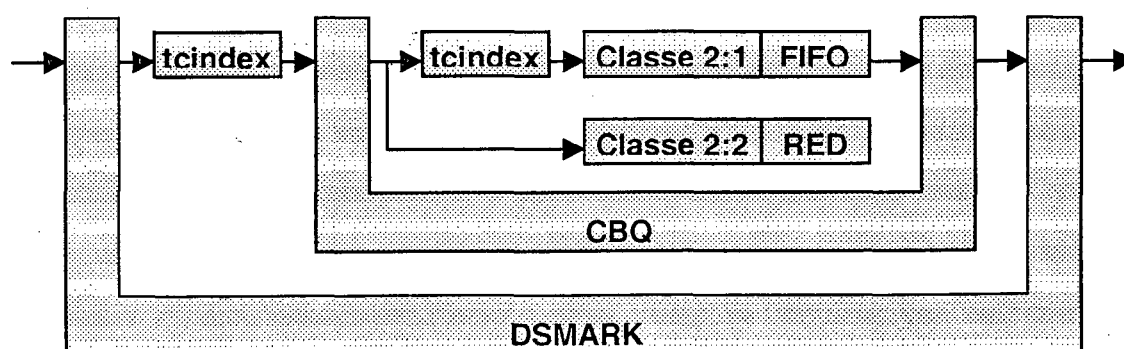


Figura 6-4: Configuração nos roteadores

6.5 Agente de Medição

Os agentes de medição possuem praticamente toda sua funcionalidade implementada em C++. Existe apenas uma interface em Java para o recebimento das requisições de medição geradas pela aplicação gerente. A linguagem C++ é mais adequada para este tipo de aplicação devido à sua maior velocidade de processamento em comparação com a linguagem Java, resultando em medições com maior precisão. No final das medições, a interface Java devolve à aplicação gerente os resultados obtidos.

As medições são realizadas com intervalos definidos por uma distribuição de Poisson [KIN93], onde a qualidade da amostragem depende desta distribuição estatística $G(t)$ [BIL92]. Normalmente, este método afeta menos a rede e gera uma estimativa mais imparcial da métrica em questão. Uma das poucas desvantagens é que a análise dos resultados se torna mais complicada devido às medições não ocorrerem em intervalos fixos. Por exemplo, uma distribuição de Poisson com valor médio 500 poderia ser formada pelos seguintes valores: 508, 529, 500, 502, 530, 478, 534, 515, 517 e 477.

Nas medições através do método *One-way*, o protocolo NTP é usado para realizar a sincronização dos relógios dos dispositivos da rede. Na Internet existe uma ampla disponibilidade de servidores NTP. No Brasil, a Rede Nacional de Pesquisa (RNP) vem implantando uma hierarquia de servidores NTP que já conta com diversos servidores secundários. Segundo [MIL92], os servidores NTP provêm sincronização com precisão de um milissegundo em redes locais e algumas dezenas de milissegundos em redes de longa distância.

Através de estudos sobre as metodologias de medição das métricas definidas pelo IPPM, notou-se que o procedimento de medição para a métrica *Type-P-One-way-Delay* é praticamente repetido nos procedimentos de medição da variação de latência e perda de pacotes. O fato de a medição da variação de latência depender da medição da latência é bastante óbvio, para calcular seus valores é necessário já ter os valores de latência de um fluxo contínuo de pacotes de teste. Além disso, concluiu-se que a medição da perda de pacotes também pode ser realizada com o mesmo fluxo de pacotes. Devido às características em comum entre as métricas, decidiu-se implementar as métricas de uma forma que um mesmo fluxo de pacotes de teste pudesse ser usado para medir latência, variação de latência e perda de pacotes. O sistema de medições implementa as métricas de

latência Type-P-One-way-Delay e Type-P-Round-trip-Delay. Com base nestas medições, as métricas variação de latência e perda de pacotes são determinadas. O grupo de trabalho IPPM não define métricas de variação de latência ou perda de pacotes baseadas em transmissões de ida e volta (*round-trip*), porém o sistema também implementa estas funcionalidades.

No sistema de gerência implementado, a aplicação gerente requisita a um de seus clientes a medição ponto-a-ponto com outro cliente. Os parâmetros informados ao agente de medição do cliente são os seguintes:

- Endereço IP do nó fonte;
- Endereço IP do nó destino;
- Número de pacotes que deverão ser enviados;
- Tamanho dos pacotes; e
- Tempo máximo de espera antes de considerar os pacotes perdidos.

A definição de alguns dos parâmetros acima é baseada em valores definidos pela Internet2 para a arquitetura do Qbone [TEI99]. O tempo máximo de espera por um pacote antes de considerá-lo perdido é dois minutos. Outra característica do Qbone seguida nesta implementação diz respeito ao valor médio da distribuição de Poisson [KIN93] usada para definir o intervalo de tempo entre o envio dos pacotes. Cada pacote deve ser transmitido com um intervalo de aproximadamente 500 milissegundos, pois se acredita que este valor seja eficiente para captar o comportamento da rede. É importante que o intervalo não seja muito pequeno para que o tráfego de teste não perturbe os outros tráfegos ou exceda suas reservas da rede. No caso de intervalos muito grandes, a leitura dos comportamentos da rede pode ser prejudicada.

Para cada pacote de teste enviado durante a medição, são coletados os seguintes valores:

- TempoSaida - o tempo do relógio do nó fonte no momento do envio do pacote; e
- TempoChegada - o tempo do relógio do nó destino no momento do recebimento do pacote.

Os valores de latência são armazenados em uma lista indexada pelo número de seqüência dos pacotes. Nesta lista, os pacotes perdidos são representados através do menor valor suportado por dados do tipo *long*: -2147483648. Desta forma, é possível que uma única lista de latências de transmissão contenha todas as informações necessárias

para a geração de gráficos e estatísticas da medição com relação a latência, variação de latência e perda de pacotes.

Pode haver diversas condições de erro no procedimento de medição. Algumas situações são previstas e evitadas para que o nível de precisão dos valores medidos não seja afetado, porém outras situações são muito difíceis de serem evitadas. Na tabela 6-4 são apresentadas as principais fontes de erro e suas respectivas soluções.

Fonte de erro	Solução
Um pacote não chega no destino dentro do tempo máximo de espera.	O sistema considera perdidos os pacotes que não chegam dentro de dois minutos.
Um pacote chega no destino corrompido.	Os pacotes corrompidos são tratados pelo sistema como pacotes perdidos.
Um pacote é duplicado pelo caminho.	A primeira cópia do pacote é processada normalmente e as outras são desconsideradas.
Os pacotes chegam fragmentados.	Esta situação não ocorre porque o sistema limita o tamanho dos pacotes transmitidos em 1500 bytes (redes Ethernet).
Os relógios dos nós não estão sincronizados adequadamente.	Todos os nós folhas do domínio devem ser clientes NTP. Os nós sem esta funcionalidade devem realizar medições segundo o método <i>round-trip</i> .

Tabela 6-4: Fontes de erro de medição

7 AVALIAÇÃO DO SISTEMA DE GERÊNCIA

A avaliação do sistema de gerência foi realizada através de testes práticos com o sistema em um ambiente criado para simular um domínio DiffServ. Neste ambiente, o gerente foi instalado em um servidor de aplicações e os agentes de descoberta de rota, configuração e medição foram instalados nos nós da rede com suporte aos Serviços Diferenciados. Após sua instalação, o sistema foi colocado para rodar e foi verificado se todos os nós com agentes rodando tinham se cadastrado corretamente no gerenciador. Através da requisição do serviço Premium ao gerenciador para transmissões entre dois nós do domínio, localizados em sub-redes diferentes, foi verificada a capacidade do sistema em descobrir e configurar os nós do domínio que fazem parte das rotas de transmissão. A capacidade do sistema em disponibilizar aos usuários o serviço Premium foi avaliada através da comparação dos resultados de medições realizadas antes e depois da requisição do serviço.

7.1 Ambiente de testes

Os equipamentos e aplicativos disponíveis para a montagem do ambiente de testes foram os seguintes:

- 3 computadores rodando Linux Mandrake versão 8.1 com suporte aos Serviços Diferenciados;
- 3 computadores rodando Windows 98 sem suporte aos Serviços Diferenciados;
- 2 *hubs* Ethernet de 8 portas;
- Servidor de aplicações Tomcat⁹ versão 3.2;
- Aplicativo de sincronização de relógio NTP versão 4.1.0;
- Aplicativo de medição Ping disponibilizado no Linux Mandrake versão 8.1;
- Aplicativo de medição MGEN¹⁰ versão 3.2; e
- Aplicativo de geração de tráfego Iperf versão 1.2.

⁹ Tomcat é um servidor de aplicações com suporte às tecnologias Java Servlet e Java Server Pages.
<http://jakarta.apache.org/tomcat/index.html>.

¹⁰ <http://manimac.itd.nrl.navy.mil/MGEN/>

A montagem do ambiente de teste foi baseada, parcialmente, nas informações contidas no artigo [MCW00]. Neste trabalho, duas redes locais com velocidade de transmissão de 10 Mb/s foram criadas com o auxílio dos *hubs* e um dos computadores rodando Linux foi colocado entre elas para realizar a função de roteador, como apresentado na figura 7-1. Um dos computadores com Windows 98 recebeu a instalação do servidor de aplicações e da aplicação gerente implementada. O computador Linux1 foi usado como cliente do sistema de gerenciamento, sendo responsável pela requisição do serviço Premium para suas transmissões ao computador Linux2. Os outros dois computadores com Windows 98 foram usados para a geração de tráfegos de *background*, quando necessário.

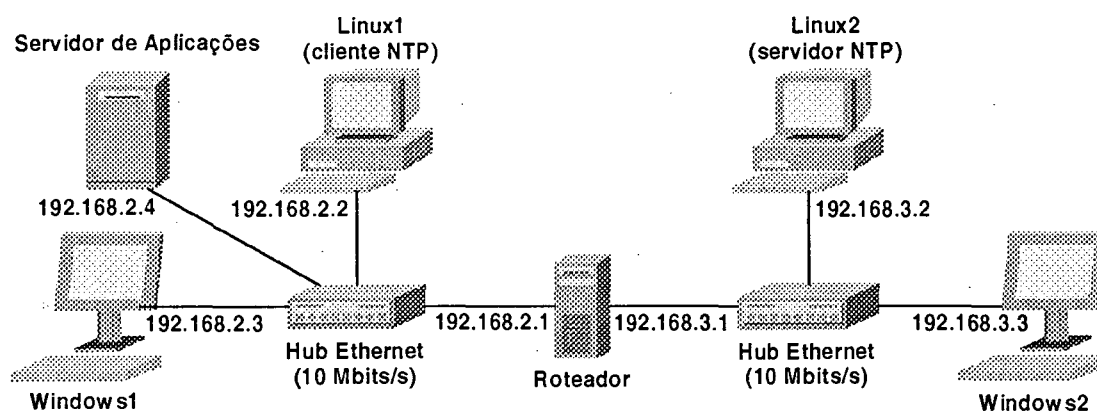


Figura 7-1: Ambiente de testes

O gerenciador implementado neste trabalho é dividido em quatro aplicações: uma aplicação gerente com interface HTML, um agente de descoberta de rotas, um agente de configuração e um agente de medição. A aplicação gerente foi instalada no servidor de aplicações Java e ficou disponível para a requisição de recursos. O agente de configuração foi instalado nos computadores Linux1, Linux2 e roteador. O agente de descoberta de rotas e o agente de medição foram instalados nos computadores Linux1 e Linux2. Ambos nós também receberam a instalação do pacote NTP para poderem sincronizar seus relógios e, desta forma, obter boa precisão nas medições pelo método *one-way*. O computador Linux2 funcionou como servidor NTP.

7.2 Cadastramento de Clientes

O agente de configuração é responsável pelo cadastramento do nó com a aplicação gerente e pela configuração de filas, classes e filtros nas interfaces para que o tráfego Premium receba um tratamento diferenciado do tráfego Melhor Esforço. Existem dois tipos de agentes de configuração, um para os nós folhas e outro para os nós centrais. Os nós folhas são representados pelos computadores dos usuários da rede e os nós centrais são representados pelos roteadores. De acordo com a especificação dos Serviços Diferenciados, estes dois tipos de nós devem receber configurações diferenciadas: os nós folha precisam de classificadores baseados nos endereços IP dos pacotes, marcadores e condicionadores de tráfego e os nós centrais precisam de classificadores baseados no código DSCP dos pacotes e de filas com diferentes prioridades.

A verificação do cadastramento dos nós do domínio pelo sistema foi realizada com facilidade devido ao fato de a aplicação gerente apresentar a lista de todos os nós folhas cadastrados na tabela “Clientes Cadastrados e Respectivas Alocações” e a lista de todos os nós centrais cadastrados na tabela “Roteadores Cadastrados”. Através da figura 7-2, é possível verificar que, após a ativação do agente de configuração nos nós da rede, os seguintes nós se cadastraram no sistema: nó folha Linux1 com interface 192.168.2.2, nó folha Linux2 com interface 192.168.3.2 e nó central Roteador com interfaces 192.168.2.1 e 192.168.3.1. No primeiro momento, aparecem somente as interfaces dos nós. Informações adicionais são apresentadas após as requisições de alocação de recursos e medição das transmissões.

Clientes Cadastrados e Respectivas Alocações					
Cliente	Destino	Banda Alocada	Latência (min/med/max)	Variação de Latência (min/med/max)	Pacotes Perdidos (%)
192.168.2.2					
192.168.3.2					

Roteadores Cadastrados	
Interface	Banda Alocada para o Serviço Premium
192.168.2.1	0
192.168.3.1	0

Figura 7-2: Visualização dos clientes cadastrados

Além de se cadastrarem no gerente, os agentes de configuração também realizam uma configuração base nas interfaces dos nós em que estão rodando. Nos nós folhas, esta configuração base cria as filas e classes especificando a marcação e condicionamento que os pacotes Premium deverão receber. A criação dos filtros para os fluxos de dados específicos é deixada para depois. A tabela 7-1 mostra as informações apresentadas pela ferramenta TC sobre suas configurações na interface eth0 (192.168.2.2) do nó Linux1 após a ativação do agente de configuração. É possível verificar que as configurações criaram uma disciplina de enfileiramento marcadora de pacotes DiffServ, uma classe especificando o código DSCP que deve ser marcado nos pacotes e um filtro do tipo u32.

```
# tc qdisc show dev eth0
qdisc dsmark 1: dev eth0 indices 0x0040

# tc class show dev eth0
class dsmark 1:1 parent 1: mask 0x03 value 0xb8

# tc filter show dev eth0
filter parent 1: protocol ip pref 1 u32
filter parent 1: protocol ip pref 1 u32 fh 1: ht divisor 1
filter parent 1: protocol ip pref 1 u32 fh 800: ht divisor 1
```

Tabela 7-1: Informações do TC sobre a configuração base na interface eth0 do nó Linux1

Nos nós folha, a configuração base cria as filas, classes e filtros especificando o tratamento que os pacotes Premium e Melhor Esforço deverão receber e deixa a especificação da largura de banda alocada para o serviço Premium para futuras configurações baseadas nas requisições dos usuários. A tabela 7-2 mostra as informações apresentadas pela ferramenta TC sobre suas configurações na interface eth1 (192.168.3.1) do nó Roteador após a ativação do agente de configuração. A mesma configuração é realizada em todas as interfaces do nó. Pelas informações apresentadas, é possível verificar que as configurações criaram quatro disciplinas de enfileiramento: uma do tipo DSMARK, uma do tipo CBQ, uma do tipo PFIFO e uma do tipo RED. A disciplina de enfileiramento DSMARK foi colocada na raiz da interface para ler o código DSCP dos pacotes. Os pacotes passam então para a disciplina de enfileiramento CBQ para serem processados de acordo com este código DSCP. Os pacotes Premium vão para a disciplina

de enfileiramento PFIFO e os pacotes Melhor Esforço vão para a disciplina de enfileiramento RED. A configuração das classes especifica a taxa de transmissão máxima e a prioridade de cada disciplina de enfileiramento. A disciplina de enfileiramento PFIFO começou com taxa de transmissão limite de 1 bit por segundo (a especificação de 0 bit por segundo não é permitida) e prioridade 1 (máxima). A disciplina de enfileiramento RED foi limitada em 5 Mbits por segundo e recebeu prioridade 7.

```
# tc qdisc show dev eth1
qdisc red 8002: limit 60 Kb min 15 Kb max 45 Kb
qdisc pfifo 8001: limit 5p
qdisc cbq 2: rate 10 Mbit (bounded, isolated) prio no-transmit
qdisc dsmark 1: indices 0x0040 set_tc_index

# tc class show dev eth1
class cbq 2: root rate 10 Mbits (bounded, isolated) prio no-transmit
class cbq 2:1 parent 2: leaf 8001: rate 1 bps (bonded, isolated) prio 1
class cbq 2:2 parent 2: leaf 8002: rate 5 Mbit prio 7

# tc filter show dev eth1
filter parent 1: protocol ip pref 1 tcindex hash 0 mask 0x00fc shift 2 fall_through
```

Tabela 7-2: Informações do TC sobre a configuração base na interface eth1 do nó Roteador

Pelos dados apresentados pelo gerenciador e pela ferramenta TC, é possível concluir que os agentes de configuração se cadastraram no gerenciador e realizaram uma configuração base nas interfaces de comunicação corretamente. O cadastramento dos agentes no gerenciador possibilitou, então, a continuação da avaliação do sistema através da requisição do serviço Premium para transmissões entre os nós Linux1 e Linux2.

7.3 Descoberta e Configuração dos Nós do Domínio

A funcionalidade mais importante dos BBs é a configuração automática dos nós da rede de acordo com as requisições dos usuários e as políticas de alocação de recursos do domínio. Para realizar estas configurações, o sistema deve ter a capacidade de descobrir os nós do domínio pertencentes às rotas de transmissão. Se a transmissão for entre nós do

mesmo domínio, a rota é composta pelos nós fonte e destino do tráfego e por alguns roteadores centrais. Se a transmissão for entre nós de domínios diferentes, a rota é composta por roteadores conectados a domínios adjacentes (roteadores de borda) e roteadores centrais do domínio.

A pesquisa sobre BBs apresentada na seção 4 não encontrou muitos dados sobre estratégias de descoberta de rotas. Dos sete BBs pesquisados, apenas dois especificaram a estratégia de descoberta de rotas. Ambos se comunicavam com roteadores OSPF para obter estas informações. Esta solução me pareceu bastante eficiente, porém restrita somente a redes com este protocolo de roteamento. A estratégia usada neste trabalho não foi encontrada em nenhum outro sistema, porém demonstrou ser bastante simples e portátil. O gerenciador utilizou a ferramenta *Traceroute* disponível nos nós da rede para descobrir as rotas de transmissão.

Nesta avaliação do sistema, foi requisitado ao gerenciador o serviço Premium para transmissões com fonte em Linux1 e destino em Linux2. Como os pacotes são diferenciados no momento da transmissão pelos nós da rede, as interfaces que necessitaram de configuração foram a interface eth0 (192.168.2.2) do nó Linux1 e a interface eth1 (192.168.3.1) do nó Roteador. Nas tabelas 7-3 e 7-4, as informações de configuração apresentadas pela ferramenta TC confirmam que o gerenciador descobriu os nós do domínio e realizou as configurações necessárias para disponibilizar o serviço Premium. No nó Linux1, foi configurado um filtro u32 para pacotes com fonte em 192.168.2.2 (c0a80202 em hexadecimal) e destino em 192.168.3.2 (c0a80302 em hexadecimal). Este filtro condicionou a taxa de transmissão dos pacotes em 100 Kbits/s e redirecionou-os para a classe 1:1, a qual define a marcação dos pacotes Premium. No nó Roteador, a configuração foi mais simples. A classe dos pacotes Premium (classe 2:1) foi alterada para permitir a transmissão de até 100 Kbits por segundo.

```
# tc filter show dev eth0
```

```

filter parent 1: protocol ip pref 1 u32
filter parent 1: protocol ip pref 1 u32 fh 1: ht divisor 1
filter parent 1: protocol ip pref 1 u32 fh 800: ht divisor 1
filter parent 1: protocol ip pref 1 u32 fh 800::800 order 2048 key ht 800 bkt 0.flowid 1:1
  police 1 action drop rate 100 Kbit burst 2 Kb mtu 2Kb
  match c0a80202/ffffffff at 12
  match c0a80302/ffffffff at 16

```

Tabela 7-3: Informações do TC sobre os filtros configurados na interface eth0 do nó Linux1

```

# tc class show dev eth1
class cbq 2: root rate 10 Mbits (bounded, isolated) prio no-transmit
class cbq 2:1 parent 2: leaf 8001: rate 100 Kbit (bonded, isolated) prio-1
class cbq 2:2 parent 2: leaf 8002: rate 5 Mbit prio 7

```

Tabela 7-4: Informações do TC sobre os filtros configurados na interface eth1 do nó Roteador

7.4 Medição da Qualidade das Transmissões

Foram realizados alguns testes com o agente de medição com o objetivo de observar se a implementação foi realizada corretamente e seus resultados têm boa precisão. A estratégia tomada para verificar a precisão da implementação foi comparar seus resultados com os resultados obtidos com ferramentas de terceiros. Partiu-se do pressuposto que estas ferramentas de terceiros foram bem implementadas e sempre geram resultados com boa precisão. Desta forma, a obtenção de resultados semelhantes com o sistema implementado permitiria afirmar que, sob as condições de congestionamento testadas, este sistema também gera resultados com boa precisão. Todas as medições foram realizadas sob as mesmas condições de rede e os computadores envolvidos não possuíam carga extra que pudesse influenciar os resultados.

7.4.1 Medições sem Serviço Premium em Rede sem Tráfego

Este primeiro teste serviu para avaliar a precisão dos resultados obtidos pelo agente de medição em uma rede sem tráfego. Os únicos computadores que participaram do teste foram o Linux1 e o Linux2, como pode ser visto na figura 7-3. Por não haver outras transmissões durante os testes, os pacotes de medição foram transmitidos com a maior

velocidade possível do ambiente e, conseqüentemente, suas latências de transmissão foram bem baixas. Com esta situação de tráfego, foi possível verificar a qualidade da implementação em relação ao tempo de processamento gasto durante as medições, porque processamentos desnecessários prejudicam a precisão dos resultados.

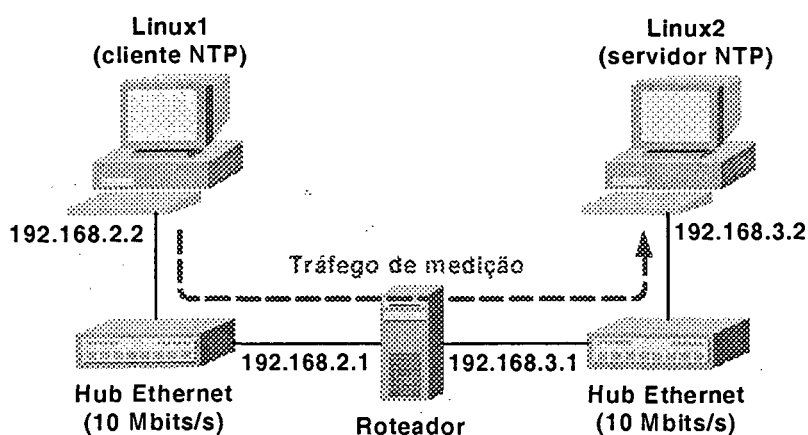


Figura 7-3: Teste de medição sem tráfego de *background*

Os pacotes de medição foram enviados do computador Linux1 (endereço IP 192.168.2.2) para o computador Linux2 (endereço IP 192.168.3.2). Todas as medições foram realizadas por cinco minutos com pacotes de 1400 bytes. O teste foi dividido em dois grupos de medições: um seguindo o método *round-trip* e outro seguindo o método *one-way*.

As medições seguindo o método *one-way* foram realizadas pelo agente de medição utilizando os protocolos TCP e UDP e pela ferramenta MGEN utilizando o protocolo UDP. A ferramenta MGEN é bastante utilizada para a realização de medições com pacotes UDP. Sua escolha foi devido ao fato de ser bastante conhecida e possuir características presentes no agente de medição: capacidade de definir o tamanho dos pacotes, capacidade de transmitir os pacotes em intervalos definidos por uma distribuição de Poisson e capacidade de determinar o tempo de medição. Ambos agente de medição e MGEN transmitiram os pacotes a uma frequência de transmissão de aproximadamente dois pacotes por segundo seguindo uma distribuição de Poisson. O agente de medição usou as portas 7002 e 7002 para o envio e recepção dos pacotes TCP e as portas 8002 e 8003 para o envio e recepção dos pacotes UDP.

A sincronização necessária para medições *one-way* foi realizada através da configuração do nó destino como servidor NTP e do nó fonte como cliente NTP. Desta forma, foi possível atingir um altíssimo nível de sincronização entre os nós, com diferença de menos de 1 milissegundo. Na Internet, as sincronizações não são tão precisas devido à distância entre os clientes e servidores NTP, porém costumam ser sincronizações aceitáveis para a maioria das aplicações. Nesta avaliação do agente de medição usando o método de medição *one-way*, a sincronização entre os relógios foi realizada antes do início das medições.

O gráfico da latência dos pacotes, exibido na figura 7-4, mostra que os relógios dos nós participantes da medição foram perdendo a sincronia durante os cinco minutos de teste. No início das medições, logo após a sincronização entre os nós, os resultados representavam os valores corretos de latência, algo em torno de 2.8 milissegundos. Após um minuto de medição os valores ficaram negativos, representando uma total perda de sincronia entre os relógios dos nós fonte e destino. Ambos agente de medição e MGEN reportaram este fenômeno.

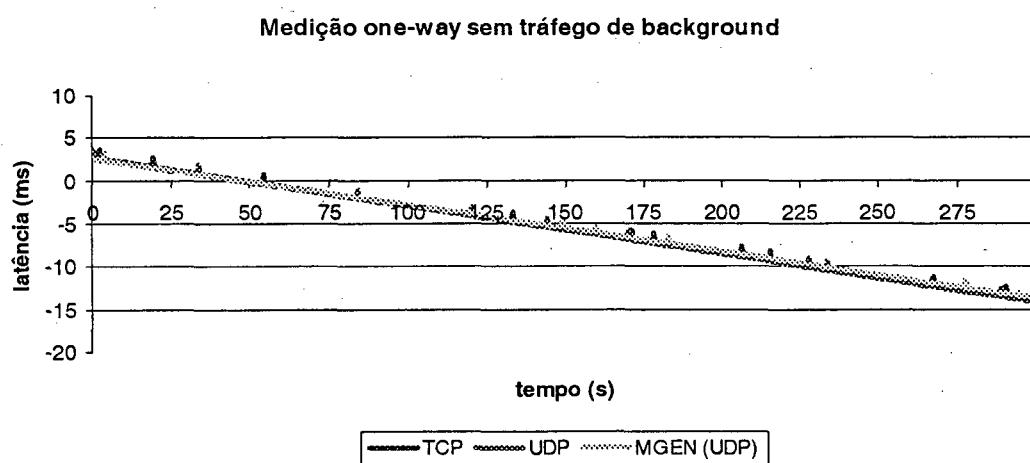


Figura 7-4: Gráfico da medição *one-way* sem tráfego de *background*

Para confirmar que o fenômeno registrado pelo teste anterior foi realmente perda de sincronização, as mesmas medições foram realizadas no sentido inverso. O computador Linux1 foi colocado como destino dos pacotes de medição gerados pelo computador Linux2. O gráfico da figura 7-5 apresenta os dados deste teste. Nota-se que, como no teste anterior, os valores medidos foram perdendo a precisão de forma constante com o passar do tempo. A diferença é que a inversão no sentido das transmissões provocou a inversão

no sentido da linha do gráfico. Quando Linux1 foi a fonte das transmissões, as medições de latência geraram resultados decrescentes, e quando Linux2 foi a fonte das transmissões, as medições de latência geraram resultados crescentes. Este comportamento levou a conclusão que o relógio de Linux1 estava funcionando mais rapidamente que o relógio de Linux2.

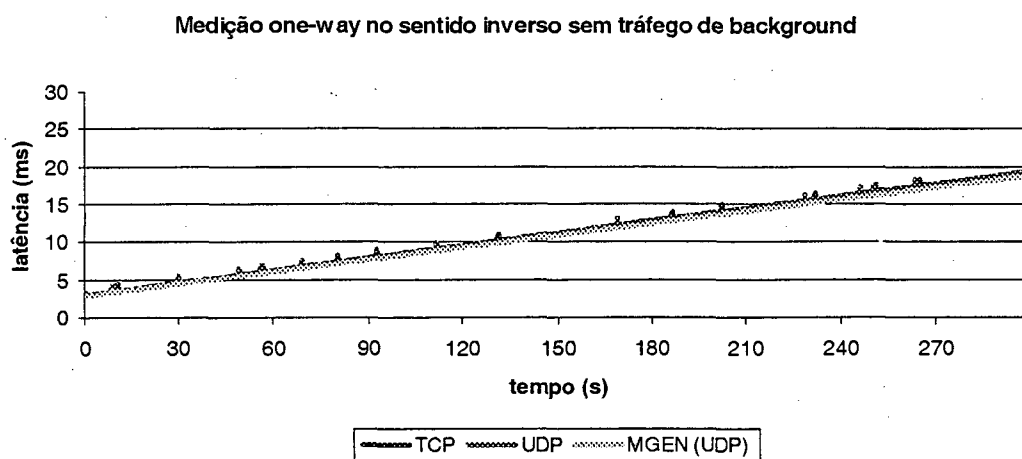


Figura 7-5: Gráfico da medição *one-way* no sentido inverso sem tráfego de *background*

Existem duas formas para melhorar a precisão dos resultados gerados nos testes anteriores. Segundo [KAL99a], se a diferença entre os relógios dos nós é conhecida, este valor pode ser somado ao valor de latência medido para gerar um resultado mais preciso. Como esta diferença entre os relógios se altera com o tempo, seu valor deve ser determinado periodicamente. Em vários casos, esta variação da diferença entre os relógios pode ser representada através de uma função linear. Na prática, esta não parece ser a melhor solução porque seria necessário calcular a taxa de perda de sincronização para cada par de nós de medição. Este cálculo teria que ser realizado antes de cada medição ou teria que ser armazenado e gerenciado pelo sistema gerente.

Uma forma mais simples e fácil de melhorar a precisão das medições é realizar a sincronização dos nós em intervalos fixos durante o processo de medição. Desta forma, quanto menor o intervalo entre as sincronizações, menor o erro inserido nos resultados das medições. Por exemplo, a figura 7-6, apresenta o gráfico da medição *one-way* entre Linux1 e Linux2 com sincronizações a cada 30 segundos. As estatísticas geradas por estes dados mostram resultados bem mais precisos do que os anteriores, com a média das latências ficando em torno de 2.5 milissegundos, como pode ser visto na tabela 7-5.

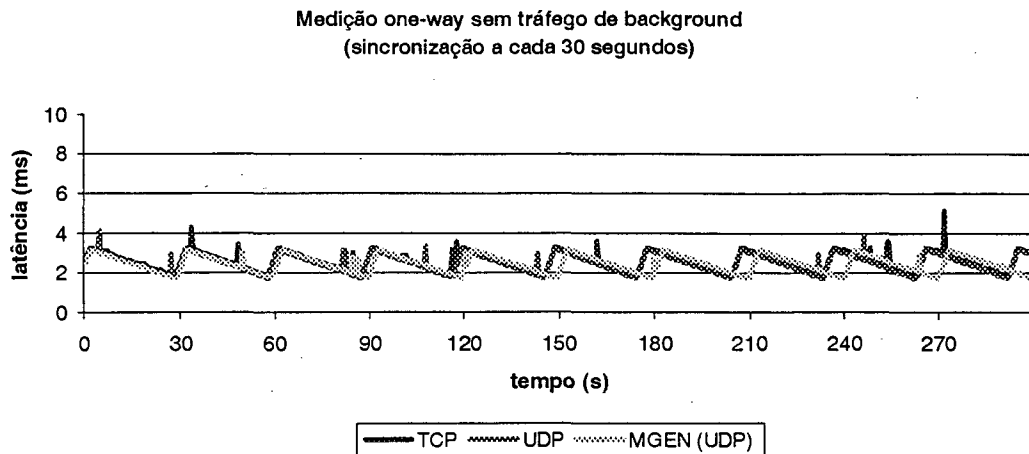


Figura 7-6: Gráfico da medição *one-way* com sincronização a cada 30 segundos e sem tráfego de *background*

	TCP	UDP	MGEN (UDP)
Latência média	2,590	2,477	2,452
Varição de latência média	0,088	0,080	0,063
Porcentagem de pacotes	0 %	0 %	0 %

Tabela 7-5: Estatísticas da medição *one-way* com sincronização a cada 30 segundos e sem tráfego de *background*

Para alcançar alta precisão nas medições com o método *one-way*, foi decidido realizar os próximos testes com sincronizações a cada um segundo. Na Internet, um intervalo tão pequeno não seria possível porque os clientes NTP estão em quantidade muito maior e o tráfego de suas sincronizações seria muito grande. Porém, sincronizações a cada trinta segundos já diminuem bastante o erro inserido nas medições. Este intervalo entre as sincronizações pode variar entre as redes que compõem a Internet, dependendo do nível de tráfego nos roteadores e da localização do servidor NTP mais próximo. A sincronização a cada um segundo entre os nós Linux1 e Linux2 praticamente impede a perda de precisão nas medições, como pode ser visto na figura 7-7 e na tabela 7-6.

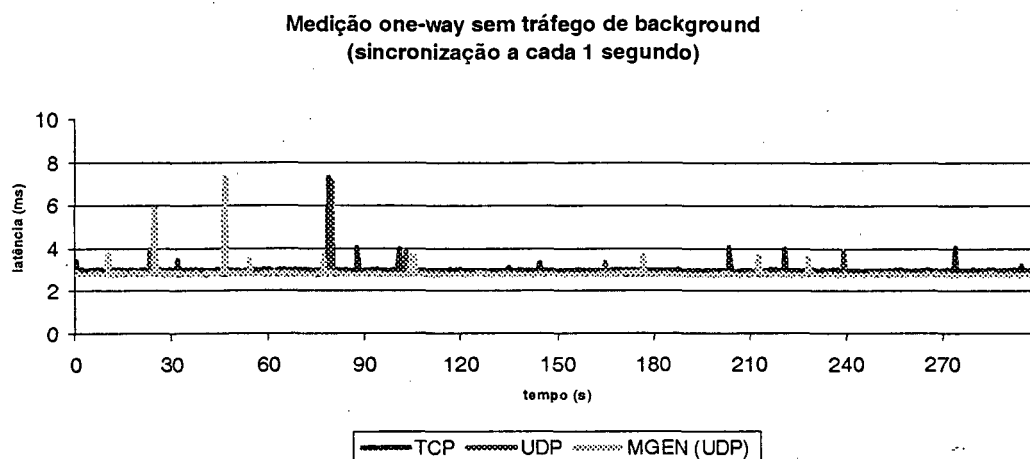


Figura 7-7: Gráfico da medição *one-way* com sincronização a cada 1 segundo e sem tráfego de *background*

	TCP	UDP	MGEN (UDP)
Latência média	2,970	2,823	2,827
Variação de latência média	0,111	0,064	0,078
Porcentagem de pacotes	0 %	0 %	0 %

Tabela 7-6: Estatísticas da medição *one-way* com sincronização a cada 1 segundo e sem tráfego de *background*

Para o método de medição *round-trip*, a ferramenta usada para comparação foi o Ping. Esta ferramenta é bastante usada para testar a comunicação entre nós da rede e verificar a latência e a taxa de perda de pacotes das transmissões. As razões de sua escolha foram a sua grande disponibilidade e suas capacidades de definir o tamanho dos pacotes de medição, o intervalo de tempo entre cada transmissão e o tempo total de medição. O agente de medição transmitiu os pacotes a uma frequência de transmissão de aproximadamente dois pacotes por segundo seguindo uma distribuição de Poisson. Por não implementar esta funcionalidade, o Ping foi configurado para transmitir os pacotes entre intervalos fixos de 0.5 segundos. Com medições de cinco minutos, cada ferramenta de medição enviou um total de 600 pacotes. Os pacotes de medição foram enviados do computador Linux1 para o computador Linux2. O agente de medição usou as portas 7000 e 7001 para o envio e recepção dos pacotes TCP e as portas 8000 e 8001 para o envio e recepção dos pacotes UDP. O fato de o Ping usar o protocolo de transmissão ICMP não foi encarado como um problema porque o roteador não tinha nenhuma configuração que priorizasse um protocolo sobre o outro.

O gráfico com os valores de latência medidos pelo agente de medição, usando TCP e UDP, e pela ferramenta Ping, usando ICMP, é apresentado na figura 7-8. Pode ser visto que as linhas do gráfico praticamente se sobrepõem mostrando valores de latência próximos de 5.7 milisegundos nas três medições. A constância dos resultados já era esperada pelo fato de não haverem outras transmissões de dados na rede. Na tabela 7-7, as estatísticas geradas com os resultados das medições ajudam a mostrar a grande proximidade entre os resultados gerados pelo agente de medição e pelo Ping. Os resultados com o protocolo de transporte TCP foram um pouco acima dos resultados com os protocolos UDP e ICMP. Acredita-se que isso tenha ocorrido porque o protocolo TCP utiliza diversos recursos de controle de fluxo que não existem nos outros protocolos, como as mensagens de confirmação e o controle de congestionamento.

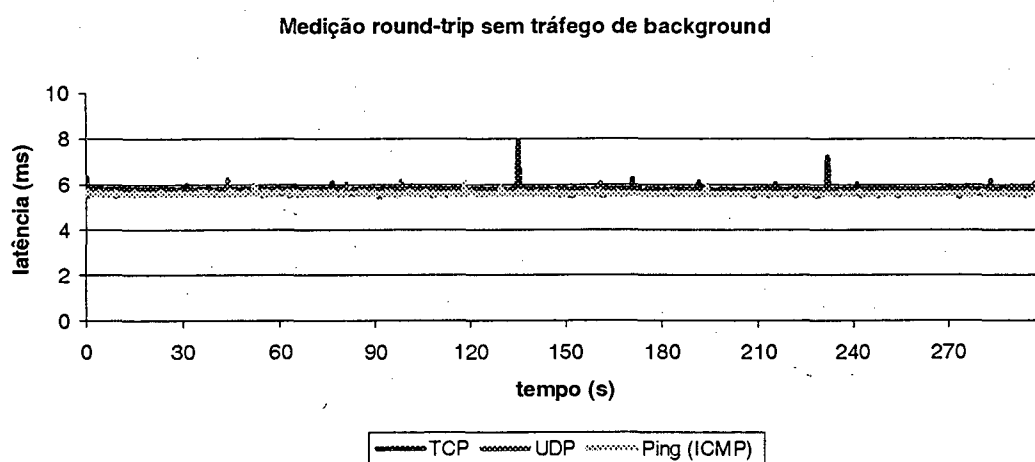


Figura 7-8: Gráfico da medição *round-trip* sem tráfego de *background*

	TCP	UDP	PING (ICMP)
Latência média	5,772	5,619	5,605
Variação de latência média	0,128	0,091	0,069
Porcentagem de pacotes	0 %	0 %	0 %

Tabela 7-7: Estatísticas da medição *round-trip* sem tráfego de *background*

7.4.2 Medições sem Serviço Premium em Rede Congestionada

O segundo teste foi realizado com a rede congestionada. Seu objetivo foi verificar o comportamento do agente de medição em uma rede com altos níveis de latência, variação

de latência e perda de pacotes. Para provocar esta situação, foram gerados dois tráfegos de *background* no ambiente através da ferramenta Iperf. Como apresentado na figura 7-9, um tráfego de *background* de 8 Mb/s foi gerado entre o roteador e o nó Windows2 e outro tráfego de *background* de 3 Mb/s foi gerado entre os nós Windows1 e Windows2. O tráfego de 8 Mb/s gerado diretamente no roteador teve o objetivo de congestionar a interface com endereço IP 192.168.3.1. Este tráfego representa a chegada de diversos fluxos de pacotes em diferentes interfaces e sua retransmissão por uma única interface. Sua consequência no roteador foi gerar uma grande fila de pacotes na saída da interface, provocando um aumento no tempo de transmissão dos pacotes de medição que por ali passavam. Porém, este único tráfego de *background* não foi suficiente para encher a fila e, conseqüentemente, gerar o descarte de alguns pacotes. Para conseguir este comportamento, foi necessário gerar um segundo tráfego de *background* com fonte em Windows1 e destino em Windows2. Após realizar tentativas com diferentes taxas de transmissão, verificou-se que um tráfego adicional de 3 Mb/s atravessando o roteador no mesmo sentido era suficiente para exceder o limite de transmissão da interface do roteador e provocar o descarte de uma quantidade razoável de pacotes.

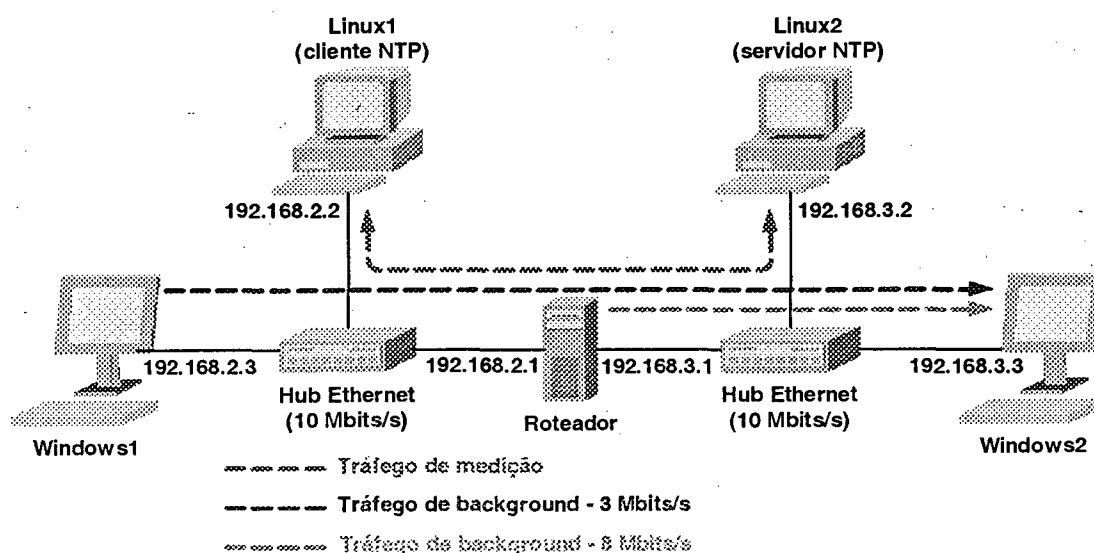


Figura 7-9: Teste de medição com tráfego de *background*

Igualmente ao teste sem tráfego de *background*, as medições foram realizadas entre os nós Linux1 e Linux2 durante cinco minutos com pacotes de 1400 bytes. O agente de

medição foi comparado ao Ping nas medições segundo o método *round-trip* e ao MGEN nas medições segundo o método *one-way*. Devido à grande variabilidade do tempo de transmissão dos pacotes neste teste, cada medição foi realizada dez vezes para proporcionar uma comparação de resultados mais clara.

Com o objetivo de visualizar a variabilidade do tempo de transmissão dos pacotes, os valores de latência da primeira medição de cada ferramenta foram usados para gerar dois gráficos. O primeiro gráfico, representando as medições através do método *one-way*, é apresentado na figura 7-10. As medições com pacotes UDP e ICMP apresentaram um comportamento bem parecido, registrando valores de latência entre 15 e 140 milissegundos. Diferentemente, as medições com pacotes TCP registraram valores de latência entre 15 e 505 milissegundos. Esta diferença é explicada pelo fato de o protocolo TCP retransmitir os pacotes perdidos. Por esta razão, os pacotes TCP que foram descartados pelo roteador durante o teste tiveram que ser retransmitidos, apresentando latência bem maior que os demais.

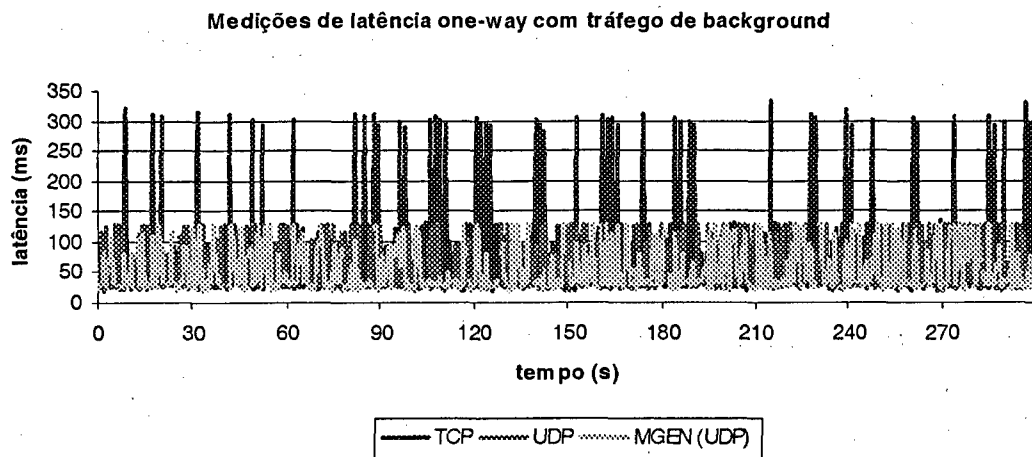


Figura 7-10: Gráfico das medições de latência *one-way* com tráfego de *background*

Através da visualização do gráfico da figura 7-10 e da análise dos dados que o formam, é possível verificar de forma bastante clara que os valores de latência dos pacotes de medição estiveram sempre entre o limite inferior de 15 milissegundos e o limite superior de 140 milissegundos. O tráfego de *background* injetado na rede não permitiu que os pacotes de medição fossem transmitidos em tempo menor que 15 milissegundos e não conseguiu provocar latência de transmissão maior que 140

milisegundos. Baseado nestas informações, surge o questionamento sobre como deve ser medida a latência de pacotes TCP. Segundo [KAL99a] e [KAL99b], as medições de latência devem representar o tempo entre a saída do pacote da fonte e sua chegada no destino. Porém, a metodologia de medição apresentada nestes documentos não especifica se a latência das transmissões TCP deve incluir o tempo de retransmissão dos pacotes perdidos. Em transmissões TCP, quando pacotes perdidos são retransmitidos, o tempo de transmissão dos dados é maior que o tempo de transmissão do novo pacote (pacote de retransmissão). Para os usuários da rede, o importante é saber quanto tempo a rede está levando para transmitir os dados. Por esta razão, o sistema desenvolvido neste trabalho mede o tempo de transmissão dos dados, não dos pacotes.

As tabelas 7-8, 7-9 e 7-10 apresentam, respectivamente, as estatísticas das medições *one-way* com o agente de medição usando pacotes TCP e UDP e o MGEN usando pacotes UDP. Devido ao comportamento diferenciado do protocolo TCP em relação à perda de pacotes, não é possível comparar os resultados das medições com TCP com os resultados das medições com UDP. Nota-se que as estatísticas das dez medições de cada tabela são bastante similares devido ao tráfego de *background* ter sido constante durante todas as medições. Nota-se também que a latência média das medições com TCP foi aproximadamente 20 milisegundos maior que a latência média das medições com UDP. Como a perda de pacotes gera o aumento da latência e as dez medições apresentaram valores de latência média bem próximos, conclui-se que o número de pacotes perdidos não variou muito entre as medições.

As medições *one-way* com pacotes UDP realizadas pelo agente de medição e pela ferramenta MGEN apresentaram resultados muito similares, demonstrando que o agente de medição também funciona bem em redes congestionadas e com perda de pacotes. É possível verificar que a latência mínima, a latência média e a latência máxima registradas pelas duas ferramentas foram praticamente iguais: 19, 54 e 132 milisegundos, respectivamente. O agente de medição e o MGEN também registraram valores muito próximos de variação de latência mínima e a variação de latência máxima. Estranhamente, os valores de variação de latência média registrados tiveram considerável diferença: os resultados do agente de medição ficaram por volta de 47 milisegundos e os resultados do MGEN ficaram por volta de 34 milisegundos. As medições de perda de pacotes apresentaram algumas pequenas variações entre cada

medição, porém, na média, resultaram em valores bem próximos. Acredita-se que pequenas variações no número de pacotes perdidos de cada medição sejam normais.

Agente de medição - TCP - *one-way*

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	média	máxima	
1	19,545	74,075	333,034	0,040	88,687	309,161	0
2	19,420	75,781	334,215	0,022	90,168	309,375	0
3	19,400	76,374	406,263	0,095	93,115	376,182	0
4	19,020	75,345	505,503	0,036	91,178	474,837	0
5	19,049	75,607	396,569	0,055	90,526	367,079	0
6	15,973	71,702	354,523	0,101	84,008	328,898	0
7	18,459	78,971	393,534	0,095	98,293	368,612	0
8	19,229	76,678	441,986	0,035	94,197	414,483	0
9	19,575	76,124	362,474	0,156	92,074	336,349	0
10	19,178	76,110	379,108	0,030	92,154	353,080	0
M	18,885	75,677	390,721	0,066	91,440	363,806	0

Tabela 7-8: Estatísticas do agente de medição segundo o método *one-way* com pacotes TCP e tráfego de *background*

Agente de medição - UDP - *one-way*

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	média	máxima	
1	20,493	53,782	130,563	0,013	47,102	108,403	8,7
2	19,156	55,249	132,270	0,106	48,283	109,325	8,2
3	20,098	53,680	130,186	0,097	47,027	108,771	8,2
4	20,706	55,268	135,510	0,003	48,864	107,763	7
5	20,233	55,483	130,088	0,038	49,792	107,666	6,8
6	19,024	53,028	130,726	0,108	47,998	109,237	7,5
7	19,261	55,044	137,706	0,152	49,128	112,509	9,2
8	19,217	54,480	130,049	0,240	46,100	107,640	7,3
9	20,446	53,428	134,578	0,072	47,094	107,080	9,8
10	20,514	53,066	129,872	0,083	44,901	106,336	8,5
M	19,915	54,251	132,155	0,091	47,629	108,473	8,1

Tabela 7-9: Estatísticas do agente de medição segundo o método *one-way* com pacotes UDP e tráfego de *background*

MGEN - UDP - *one-way*

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	Mínima	média	máxima	mínima	média	máxima	
1	19,325	53,003	129,261	0,022	32,946	108,978	8,5
2	19,205	52,794	132,060	0,030	33,713	109,677	4,8
3	18,952	52,583	132,269	0,011	33,283	108,223	11,3
4	19,005	57,363	136,586	0,019	34,903	109,246	5,3
5	19,285	54,181	131,485	0,007	34,464	109,241	5,8
6	19,543	57,028	130,434	0,071	33,121	108,321	8
7	19,051	55,389	137,369	0,044	35,828	109,442	8,5
8	19,007	54,460	130,898	0,021	34,921	109,197	12,3
9	18,761	54,928	132,179	0,030	33,819	109,028	8,3
10	17,908	55,512	134,735	0,061	36,278	108,543	13,6
M	19,004	54,724	132,728	0,032	34,328	108,990	8,6

Tabela 7-10: Estatísticas do MGEN segundo o método *one-way* com pacotes UDP e tráfego de *background*

As tabelas 7-11, 7-12 e 7-13 apresentam, respectivamente, as estatísticas das medições *round-trip* com o agente de medição usando pacotes TCP e UDP e o Ping usando pacotes ICMP. Devido ao tráfego de *background* deste teste ter sido gerado somente no sentido de ida para gerar o descarte de pacotes em uma das interfaces do roteador, o tempo gasto pelos pacotes na volta foi bastante baixo. Através da comparação dos resultados de cada método de medição, verifica-se que, em média, as latências dos pacotes nas medições *round-trip* foram apenas 3 milissegundos maiores que as latências dos pacotes nas medições *one-way*.

As dez medições realizadas com o agente de medição usando pacotes UDP e com o Ping usando pacotes ICMP apresentaram resultados muito próximos. Na média, o agente de medição registrou uma latência média de 56,224 milissegundos, uma variação de latência média de 46,528 milissegundos e uma porcentagem de perda de pacotes de 7,3 %. Em comparação, o Ping registrou uma latência média de 57,903 milissegundos uma variação de latência média de 48,081 milissegundos e uma porcentagem de perda de pacotes de 8 %. A diferença entre os valores de latência e variação de latência é menor que 2 milissegundos e a diferença entre as porcentagens de perda de pacotes é menor que 1 %, demonstrando que o agente de medição pode gerar resultados com precisão similar à ferramenta Ping.

Agente de medição - TCP - *round-trip*

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	média	máxima	
1	21,045	76,543	393,156	0,014	89,832	366,257	0
2	21,078	79,124	344,575	0,074	94,314	320,737	0
3	21,348	76,211	372,872	0,026	88,287	350,309	0
4	20,879	73,013	378,614	0,006	81,648	351,146	0
5	20,308	73,940	390,895	0,062	85,145	358,067	0
6	23,469	81,402	359,970	0,008	93,290	334,153	0
7	23,693	81,380	366,914	0,008	92,211	337,762	0
8	23,600	76,203	368,467	0,007	81,028	339,746	0
9	22,014	77,278	336,227	0,007	84,836	303,060	0
10	23,464	81,301	406,964	0,002	92,740	372,381	0
M	22,090	77,639	371,865	0,021	88,333	343,362	0

Tabela 7-11: Estatísticas do agente de medição segundo o método *round-trip* com pacotes TCP e tráfego de *background*Agente de medição - UDP - *round-trip*

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	média	máxima	
1	23,460	57,438	134,480	0,008	48,707	109,304	8
2	23,652	58,431	136,657	0,054	48,563	109,344	7,2
3	23,527	58,770	135,195	0,085	49,707	110,226	6,8
4	23,546	58,259	138,955	0,011	48,272	111,206	7,5
5	23,546	58,301	136,024	0,073	48,139	112,478	7,5
6	23,481	55,259	136,214	0,381	45,130	109,581	6,3
7	23,555	53,162	137,627	0,130	43,659	108,419	7,8
8	23,561	54,463	134,896	0,258	44,426	110,732	7
9	23,515	53,935	134,963	0,359	44,625	108,245	7,5
10	23,661	54,225	134,874	0,039	44,048	110,721	7,2
M	23,550	56,224	135,988	0,140	46,528	110,026	7,3

Tabela 7-12: Estatísticas do agente de medição segundo o método *round-trip* com pacotes UDP e tráfego de *background*

Ping - ICMP - *round-trip*

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	média	máxima	
1	22,481	56,773	135,554	0,060	45,158	108,398	9,5
2	23,492	58,393	136,026	0,160	48,980	110,948	7,3
3	23,372	57,690	136,406	0,243	47,731	111,245	8,2
4	23,458	59,325	136,030	0,146	50,740	109,076	6,2
5	23,481	57,466	134,765	0,046	47,011	109,328	8,2
6	23,373	58,751	135,113	0,124	50,044	110,689	7,7
7	22,955	58,357	140,825	0,058	47,731	108,563	8,2
8	18,435	57,655	136,655	0,166	49,020	110,057	7,7
9	23,608	58,609	137,451	0,052	46,872	109,448	6,5
10	23,393	56,010	144,416	0,220	47,525	109,988	10,5
M	22,805	57,903	137,324	0,127	48,081	109,774	8

Tabela 7-13: Estatísticas do PING segundo o método *round-trip* com pacotes ICMP e tráfego de *background*

7.4.3 Medições com Serviço Premium em Rede Congestionada

A avaliação da qualidade das transmissões após a disponibilização do serviço Premium pelo gerenciador foi realizada através da comparação dos resultados das medições realizadas em três diferentes situações: serviço Melhor Esforço em rede sem tráfego, serviço Melhor Esforço em rede congestionada e serviço Premium em rede congestionada. As medições na rede sem tráfego e na rede congestionada já foram realizadas nas seções 9.4.1 e 9.4.2, não sendo necessário suas repetições. O mesmo tráfego de *background* usado para congestionar a rede na seção 9.4.2 foi usado na avaliação do serviço Premium configurado automaticamente pelo gerenciador. Como visto, o tráfego de *background* composto por uma transmissão de 8 Mbits/s do roteador para o nó Linux2 e outra transmissão de 3 Mbits/s do nó Linux1 para o nó Linux2 foi eficaz na geração de altos níveis de latência, variação de latência e perda de pacotes nas transmissões entre os nós Linux1 e Linux2. Já que o serviço Premium disponibiliza baixos níveis de latência, variação de latência e perda de pacotes, tal tráfego de *background* pareceu ser bastante adequado para a avaliação.

O procedimento de medição foi o mesmo das avaliações anteriores: transmissão de pacotes de medição de 1400 bytes do nó Linux1 para o nó Linux2 durante cinco minutos. Como o serviço Premium foi disponibilizado em somente um sentido, o método de medição utilizado foi o *one-way*. Caso não fosse possível sincronizar os nós de medição,

o método de medição teria que ser o *round-trip*. Para motivos de avaliação, as medições foram realizadas com pacotes UDP e TCP. Em situações normais utiliza-se apenas o protocolo das transmissões do cliente. O gráfico da figura 7-11 apresenta a latência dos pacotes transmitidos durante os cinco minutos de medição. Nota-se que, com o serviço Premium, as transmissões não sofreram o atraso gerado pela interface congestionada do roteador, comprovando que os pacotes foram marcados corretamente no nó Linux1 e priorizados no roteador de acordo com a especificação do PHB EF. Para confirmar a precisão das medições do gerenciador e proporcionar mais dados sobre a qualidade das transmissões após a configuração do serviço Premium, foi realizada uma medição também com a ferramenta MGEN.

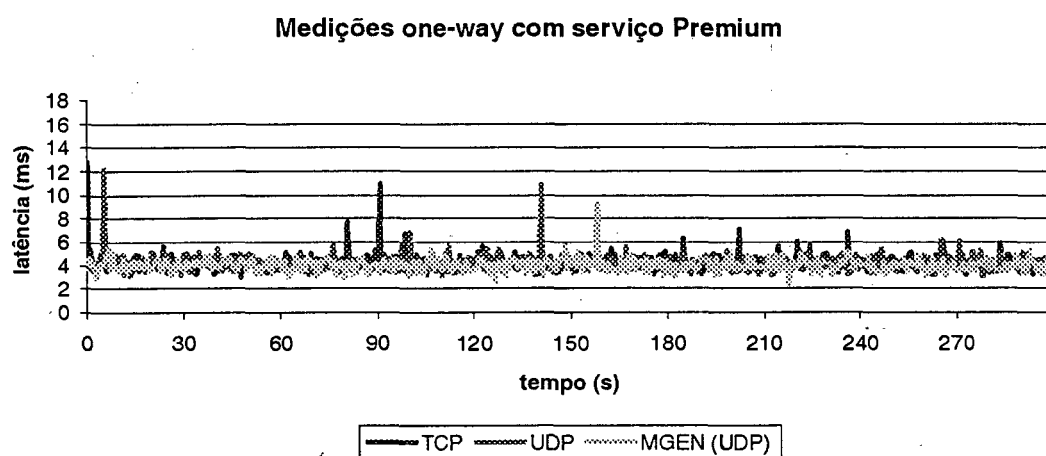


Figura 7-11: Medições *one-way* com serviço Premium

Para uma melhor avaliação da qualidade do serviço Premium disponibilizado automaticamente pelo gerenciador, os resultados das medições realizadas em cada uma das três situações de tráfego criadas no ambiente de testes foram agrupados em duas tabelas de acordo com o protocolo de transmissão usado. A tabela 7-14 apresenta os resultados das medições com pacotes UDP e a tabela 7-15 apresenta os resultados das medições com pacotes TCP.

Os resultados das medições na rede sem tráfego representam a melhor qualidade de transmissão possível no ambiente de testes com pacotes de 1400 bytes: latência de 3 milissegundos, variação de latência de 0,1 milissegundo e nenhuma perda de pacotes. Por outro lado, os resultados das medições com a rede congestionada representam uma

baixíssima qualidade de transmissão entre os nós Linux1 e Linux2. O tráfego de *background* injetado na rede foi maior que a capacidade de transmissão da interface 192.168.3.1 do roteador, provocando o descarte de aproximadamente 8 % dos pacotes e o grande aumento da latência e variação de latência dos pacotes.

O serviço Premium promete baixos níveis de latência, variação de latência e perda de pacotes. Baseado nas informações das tabelas 7-14 e 7-15 é possível concluir que a configuração automática realizada pelo gerenciador foi correta porque as transmissões entre os nós Linux1 e Linux2 apresentaram baixos níveis de latência e variação de latência e nenhuma perda de pacotes, demonstrando que os pacotes foram marcados no nó fonte (Linux1) e priorizados em relação aos pacotes do tráfego de *background* na interface congestionada do roteador.

Medições com UDP

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	média	máxima	
Serviço Melhor Esforço em rede sem tráfego	2,755	2,823	7,130	0	0,064	4,423	0
Serviço Melhor Esforço em rede congestionada	19,915	54,251	132,155	0,091	47,629	108,473	8,1
Serviço Premium em rede congestionada	2,451	4,161	12,243	0	0,568	7,922	0

Tabela 7-14: Medições com UDP

Medições com TCP

	Latência (ms)			Variação de latência (ms)			Perda de pacotes (%)
	mínima	média	máxima	mínima	Média	máxima	
Serviço Melhor Esforço em rede sem tráfego	2,866	2,970	7,326	0	0,111	4,346	0
Serviço Melhor Esforço em rede congestionada	18,885	75,677	390,721	0,066	91,440	363,806	0
Serviço Premium em rede congestionada	3,027	4,312	12,670	0	0,566	8,271	0

Tabela 7-15: Medições com TCP

7.5 Conclusão

A descoberta de rotas foi uma funcionalidade do sistema que teve que ser realizada de uma forma diferente de qualquer outra pesquisada. A comunicação com roteadores

operando com o protocolo de roteamento OSPF para descobrir as rotas pareceu ser uma estratégia muito restrita porque outros protocolos de roteamento são usados na Internet. A utilização da ferramenta *Traceroute* por este sistema se apresentou bastante simples e eficiente. Além disso, o *Traceroute* está disponível em diversos sistemas operacionais e a parte do seu resultado que interessa é somente a rota dentro do domínio, não sendo prejudicado por possíveis filtros de pacotes localizados em outros domínios da Internet.

O sistema de gerência desenvolvido neste trabalho implementou a comunicação entre o sistema gerente e os agentes através de *sockets* TCP. Esta estratégia foi bem sucedida devido à facilidade e flexibilidade deste recurso de comunicação, porém não seguiu nenhum padrão de comunicação. A especificação dos BBs ainda não define qual protocolo de comunicação deve ser usado para este fim.

O sistema de medições implementado possibilitou a avaliação da qualidade das transmissões com bastante precisão devido à sua fidelidade às especificações do grupo de trabalho IPPM do IETF. Além disso, a possibilidade do usuário especificar o tamanho dos pacotes, o protocolo e o método de medição foram funcionalidades essenciais para a geração de bons resultados. As medições de acordo com o método *one-way* demonstraram a total dependência do método em relação à precisão da sincronização entre os nós de medição. Este tipo de medição ponto-a-ponto provou ser uma funcionalidade muito importante no sistema devido à necessidade dos usuários em se certificar que o serviço disponibilizado está de acordo com o combinado.

8 CONCLUSÃO

8.1 Conclusões sobre o Sistema de Gerência

As especificações que o QBone vem fazendo sobre o *Bandwidth Broker* auxiliaram este trabalho a desenvolver um sistema de gerência próximo do que é esperado. Houve várias dúvidas na forma de implementação devido ao grande número de indefinições que ainda existem sobre o assunto, principalmente em relação às comunicações entre os agentes e o gerente. A comunicação entre BBs não foi abordada neste trabalho devido ao grau de complexidade inserido por tal funcionalidade e à grande indefinição que ainda existe sobre o assunto. Mesmo assim, o sistema implementado mostrou ser bastante eficaz na disponibilização do serviço Premium aos seus usuários.

Os recursos de controle de tráfego do sistema operacional Linux funcionaram bastante bem durante a avaliação do sistema. Através das disciplinas de enfileiramento usadas, principalmente a CBQ, foi possível configurar o serviço Premium de forma simples e eficiente. O mapeamento das ordens de configuração para comandos da ferramenta TC funcionou corretamente. Através da estratégia de realizar configurações base, ficou simples adicionar classificadores e alterar a largura de banda limite dos dispositivos DiffServ do domínio.

O sistema de gerência desenvolvido neste trabalho implementou a comunicação entre o sistema gerente e os agentes através de *sockets* TCP. Esta estratégia foi bem sucedida devido à facilidade e flexibilidade deste recurso de comunicação, porém não seguiu nenhum padrão de comunicação. A especificação dos BBs ainda não define qual protocolo de comunicação deve ser usado para este fim, porém, esta definição parece estar próxima devido aos trabalhos que vêm sendo realizados pelo IETF na área de gerenciamento de políticas. Os dois candidatos são o protocolo COPS e o protocolo SNMP. Os grupos de trabalho *Policy Framework* e RAP vêm definindo uma arquitetura de gerenciamento de políticas que usa o protocolo COPS para a transmissão das informações de configuração aos nós da rede. O protocolo COPS possui características bem adequadas à transmissão de informações de configuração, porém é um protocolo desconhecido pela grande maioria dos profissionais da área de redes e vem recebendo pouca atenção dos fabricantes de roteadores. O protocolo SNMP, por outro lado, já vem

sendo usado na gerência de rede por vários anos e é bastante conhecido pelos profissionais da área, porém nunca foi usado para realizar configurações devido, principalmente, à sua falta de segurança. O grande uso do SNMP sempre foi a leitura de variáveis das MIBs e a transmissão de mensagens de alerta ao sistema gerente. Com a definição do SNMP versão 3, foram adicionadas importantes funcionalidades de segurança que tornaram possível o uso do protocolo na configuração de dispositivos da rede. O grupo de trabalho SNMPCConf do IETF vem trabalhando na adaptação do protocolo SNMP para o gerenciamento de políticas com bastante sucesso. Comparando os dois protocolos, o SNMP tem maior potencial de crescimento nesta área devido ao seu amplo suporte nas redes de computadores existentes.

A descoberta de rotas foi uma funcionalidade do sistema que teve que ser realizada de uma forma diferente de qualquer outra pesquisada. A comunicação com roteadores operando com o protocolo de roteamento OSPF para descobrir as rotas pareceu ser uma estratégia muito restrita porque outros protocolos de roteamento são usados na Internet. A utilização da ferramenta *Traceroute* por este sistema se apresentou bastante simples e eficiente. Além disso, o *Traceroute* está disponível em diversos sistemas operacionais e a parte do seu resultado que interessa é somente a rota dentro do domínio, não sendo prejudicado por possíveis filtros de pacotes localizados em outros domínios da Internet.

O sistema de medições implementado possibilitou a avaliação da qualidade das transmissões com bastante precisão devido à sua fidelidade às especificações do grupo de trabalho IPPM do IETF. Além disso, a possibilidade do usuário especificar o tamanho dos pacotes, o protocolo e o método de medição foram funcionalidades essenciais para a geração de bons resultados. As medições de acordo com o método *one-way* demonstraram a total dependência do método em relação à precisão da sincronização entre os nós de medição. Este tipo de medição ponto-a-ponto provou ser uma funcionalidade muito importante no sistema devido à necessidade dos usuários em se certificar que o serviço disponibilizado está de acordo com o combinado. Foi possível concluir que os trabalhos do grupo IPPM estão bastante avançados, proporcionando metodologias de medição muito claras e eficazes. Além de medições ativas, outras formas de monitoração serão necessárias para gerenciar as redes com QoS, tais como medições passivas, sistemas de alarme e contabilidade dos recursos utilizados.

8.2 Conclusões Gerais

Na comunidade de usuários da Internet, é muito desejado que a rede mundial de computadores evolua a um nível onde seja possível escolher serviços com diferentes níveis de qualidade, de acordo com a necessidade de cada momento. Após uma ampla pesquisa sobre os projetos que estão sendo realizados na área de gerência de Qualidade de Serviço e o desenvolvimento de um sistema de gerência para o Serviço Diferenciado Premium, foi possível concluir que, para disponibilizar a tão desejada Qualidade de Serviço aos seus usuários, a Internet terá que se transformar em uma rede inteligente, capaz de gerenciar seus recursos de tal forma que as transmissões na Internet sejam controladas para receber a qualidade que merecem. A inteligência que a Internet precisa implementar está sendo especificada em arquiteturas de QoS e arquiteturas de gerenciamento de políticas, porém existem muitas incertezas e problemas que devem ser resolvidos nestas especificações. O grau de complexidade de uma especificação deste tipo é tão grande que ainda levarão vários anos para que surja uma arquitetura pronta para implementação.

Dentre as arquiteturas de QoS para a Internet que vêm sendo desenvolvidas nos últimos anos, a arquitetura de Serviços Diferenciados vem apresentando os maiores avanços. Sua proposta de gerenciar os recursos da Internet através de BBs distribuídos pelos domínios administrativos da rede é bastante avançada e possui grande potencial de crescimento. A especificação do *Bandwidth Broker* ainda está incompleta, porém várias funcionalidades do sistema já estão definidas, possibilitando a implementação de protótipos para testar sua aplicação no gerenciamento dos recursos de domínios DiffServ.

Os *Bandwidth Brokers* pesquisados neste trabalho apresentam idéias bastante diversificadas sobre a forma de gerenciar os recursos e as tecnologias que devem ser usadas. Esta experimentação de novas idéias e tecnologias é muito importante na área de pesquisa, porém é necessário que exista uma ou mais instituições na liderança dos trabalhos para que as boas idéias sejam transformadas em especificações e as más idéias sejam descartadas. Com o intuito de harmonizar as diferentes propostas de BBs, o projeto QBone da Internet2 vem trabalhando na definição de uma arquitetura básica e de um grupo de requerimentos que deverão ser atendidos nos próximos projetos.

Os BBs representam a base do sistema de gerência da arquitetura de Serviços Diferenciados e, por esta razão, deverão receber mais atenção após a definição das questões mais básicas da arquitetura. As principais questões que estão pendentes na definição dos BBs são a forma de organização e armazenamento das políticas e os protocolos de comunicação entre o *Bandwidth Broker* e os nós do domínio e entre BBs de domínios adjacentes. Como foi demonstrado neste trabalho, atualmente já é possível desenvolver sistemas de gerência para serviços diferenciados restritos a somente um domínio e usando uma mistura de soluções padronizadas e proprietárias. A especificação da arquitetura está acontecendo vagarosamente através de projetos desenvolvidos por inúmeras instituições de pesquisa. Espera-se que, um dia, a Internet possa implementar esta ou outra arquitetura de QoS para poder disponibilizar serviços com maior qualidade.

8.3 Sugestões para Trabalhos Futuros

8.3.1 Gerenciamento dos Recursos do Domínio

Ainda não foi definida a melhor estratégia de configuração para os nós dos domínios DiffServ. Esta estratégia envolve a definição de como armazenar as políticas do domínio, como transmiti-las para os nós DiffServ e como realizar a configuração. Atualmente, existem duas propostas nesta área: o grupo de trabalho *Policy Framework* propõe o uso do protocolo COPS para a comunicação e de diretórios LDAP para o armazenamento das políticas; o grupo de trabalho SNMPCConf propõe o uso do protocolo SNMP para a comunicação e de uma MIB DiffServ para o armazenamento das políticas. Seria interessante fazer uma comparação detalhada sobre as propostas e avaliar qual é mais adequada à arquitetura de Serviços Diferenciados.

8.3.2 Reserva de Recursos Através de Vários Domínios

A arquitetura de Serviços Diferenciados não especifica em detalhes como deve ser realizada a reserva de recursos através de vários domínios. Esta é uma questão bastante complexa e ainda não existe consenso sobre como as reservas devem ser realizadas. Dependendo da estratégia de sinalização, podem surgir grandes problemas de

escalabilidade ou garantia de qualidade. A realização de trabalhos de comparação entre diferentes opções de sinalização, como em [GUN99], ou implementação e avaliação de alguma proposta específica seria bastante útil para o avanço da arquitetura.

8.3.3 Gerenciamento do Uso dos Recursos da Rede

Os sistemas de gerência de QoS precisam oferecer funcionalidades para a contabilidade e cobrança dos recursos usados por cada usuário. Isso envolve a criação de um sistema integrado de contas de usuários, recursos disponibilizados, taxa de cobrança, etc. Um sistema deste tipo pode ficar bem complexo dado que os recursos utilizados pelos usuários poderão estar distribuídos por domínios DiffServ localizados em diferentes países. A arquitetura de Serviços Diferenciados necessita de desenvolvimento na área de contabilidade e cobrança.

REFERÊNCIAS BIBLIOGRÁFICAS

- [ALM99a] ALMESBERGER, Werner; SALIM, Jamal Hadi; KUZNETSOV, Alexey. **Differentiated Services on Linux**. Junho 1999. Disponível em: <ftp://icaftp.epfl.ch/pub/linux/diffserv/misc/dsid-01.ps.gz>. Acesso em: 5 fevereiro 2000. Trabalho não publicado.
- [ALM99b] ALMESBERGER, Werner. **Linux Network Traffic Control – Implementation Overview**. Abril 1999. Disponível em: <ftp://icaftp.epfl.ch/pub/people/almesber/pub/tcio-current.ps.gz>. Acesso em: 5 fevereiro 2000. Trabalho não publicado.
- [BAK01] BAKER, F.; CHAN, K.; SMITH, A. **Management Information Base for the Differentiated Services Architecture**. Internet-Draft, Internet Engineering Task Force, julho 2001. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-diffserv-mib-16.txt>. Acesso em: 10 dezembro 2000.
- [BAS98] BASHANDY, Ahmed; CHONG, Edwin; GHAFOR, Arif. Network Modeling and Jitter Control for Multimedia Communication Over Broadband Network. In: IEEE INFOCOM'99, 559, 1999, New York. **Proceedings...** [S.I.]: IEEE Press, 1999. 1585 p. 559-566.
- [BBT98a] BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY (Canada). Technology Centre. Group for Advanced Information Technology. **CA*netII Differentiated Services - Bandwidth Broker High Level Design**. British Columbia, novembro 1998. 16 p.
- [BBT98b] BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY (Canada). Technology Centre. Group for Advanced Information Technology. **CA*netII Differentiated Services - Bandwidth Broker Transfer Protocol**. British Columbia, outubro 1998. 19 p.
- [BBT98c] BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY (Canada). Technology Centre. Group for Advanced Information Technology. **CA*netII Differentiated Services - Bandwidth Broker System Specification**. British Columbia, outubro 1998. 16 p.
- [BIL92] I. Bilinskis; A. Mikelsons. **Randomized Signal Processing**. Londres: Prentice Hall International, 1992.

- [BLA98] BLAKE, Steven, et al. **An Architecture for Differentiated Services**. Request for Comments: 2475, Internet Engineering Task Force, dezembro 1998.
- [BRA94] BRADEN, R.; CLARK, D.; SHENKER, S. **Integrated Services in the Internet Architecture: An Overview**. Request for Comments: 1633, Internet Engineering Task Force, junho 1994.
- [BRA00] BRAUN, Torsten; et al. A Linux Implementation of a Differentiated Services Router. 5TH IFIP TC6 INTERNATIONAL SYMPOSIUM, INTERWORKING 2000, 1938, 2000, Bergen, Noruega. **Proceedings...** [S.I.]: Springer, 2000. 412 p. 302-315.
- [DEM99] DEMICHELIS, C.; CHIMENTO, P. **IP Packet Delay Variation Metric for IPPM**. Internet Draft, Internet Engineering Task Force, novembro 2001.
- [DUR00] DURHAM, D., et al. **The COPS (Common Open Policy Service) Protocol**. Request for Comments: 2597, Internet Engineering Task Force, janeiro 2000.
- [EVA99] EVANS, Joseph; RAO, Dhananjaya. **Bandwidth Broker Implementation Version 1: Specifications for Internet2-Qbone-BB Operability Event**. Disponível em: <<http://www.merit.edu/internet/working.groups/i2-qbone-bb/inter-op-old/old-new/ku-bb.htm>>. Acesso em: 22 abril 2001.
- [FER98] FERGUSON, P.; HUSTON, G. **Quality of Service: delivering QoS on the Internet and in corporate networks**. USA: John Wiley & Sons, 1998.
- [FIN01] FINE, M., et al. **Differentiated Services Quality of Service Policy Information Base**. Internet-Draft, Internet Engineering Task Force, novembro 2001. Disponível em: <<http://www.ietf.org/internet-drafts/draft-ietf-diffserv-pib-05.txt>>. Acesso em: 10 dezembro 2001.
- [FLO95] FLOYD, Sally. **Notes on CBQ and Guaranteed Service**. Julho 1995. Disponível em: <<http://www.icir.org/floyd/papers/guaranteed.ps>>. Acesso em: 1 março 2001.
- [FLO96] FLOYD, Sally. **Notes on Class-Based Queueing: Setting Parameters**. Setembro 1995. Disponível em:

<<http://www.icir.org/floyd/papers/params.ps.Z>>.

- [GUN99] GUNTER, Manuel; BRAUN, Torsten. Evaluation of Bandwidth Broker Signaling. In: SEVENTH ANNUAL INTERNATIONAL CONFERENCE ON NETWORK PROTOCOLS, 1999, Toronto, Canada. **Proceedings...** Piscataway: IEEE Press, 1999. 360 p. 145-152.
- [HEI99] HEINANEN, Juha, et al. **Assured Forwarding PHB Group**. Request for Comments: 2597, Internet Engineering Task Force, junho 1999.
- [JAC99] JACOBSON, Van; NICHOLS, Kathleen; PODURI K. **An Expedited Forwarding PHB**. Request for Comments: 2598, Internet Engineering Task Force, junho 1999.
- [KAL99a] KALIDINDI, S.; ALMES, G.; ZEKAUSKAS, M. **A One-way Delay Metric for IPPM**. Request for Comments: 2679, Internet Engineering Task Force, setembro 1999.
- [KAL99b] KALIDINDI, S.; ALMES, G.; ZEKAUSKAS, M. **A Round-trip Delay Metric for IPPM**. Request for Comments: 2681, Internet Engineering Task Force, setembro 1999.
- [KHA01] KHALIL, Ibrahim; GUNTER, Manuel; BRAUN, Torsten. Implementation of a Bandwidth Broker for Dynamic End-to-End Capacity Reservation over Multiple Diffserv Domains. In: IFIP/IEEE INTERNATIONAL CONFERENCE ON MANAGEMENT OF MULTIMEDIA NETWORKS AND SERVICES, 4., 2001, Chicago, USA. **Proceedings...** Piscataway: IEEE Press, 2001. 380 p. 160-168.
- [KIE01] KIELING, Alexandre B.; WESTPHALL, Carlos B.; VIEIRA, Elvis M. Uma Biblioteca de Classes para Medição de Serviços Diferenciados. In: WORKSHOP RNP2, 3., 2001, Florianópolis. **Anais...** [S.I. : s. n.], UFSC, 2001.
- [KIN93] KINGMAN, J. F. C. **Poisson Processes**. Oxford: Clarendon Press, 1993.
- [MAN98] MANSOUR, Yishay; PATT-SHAMIR, Boaz. Jitter Control in QoS Networks. In: 39th ANNUAL IEEE SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 5., 1998, Palo Alto. **Proceedings...** [S.I.]:

IEEE Press, 1998. 885 p. 559-566.

- [MCW00] McWherter, David T.; Sevy, Jonathan; Regli, William C. Building an IP Network Quality-of-Service Testbed. **IEEE Internet Computing**, Piscataway, v. 4, n. 4, p. 65-73, julho-agosto 2000.
- [MET00] METZ, Chris. Differentiated Services. **IEEE MultiMedia**, Piscataway, v. 7, n. 3, p. 84-90, julho-setembro 2000.
- [MIL92] MILLS, David. **Network Time Protocol (Version 3) Specifications, Implementations and Analysis**. Request for Comments: 1305, Internet Engineering Task Force, março 1992.
- [NEI01] NEILSON, Rob, at al. **A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment**. Julho 2001. Disponível em: <http://www.merit.edu/working.groups/i2-qbone-bb/doc/BB_Req7.doc>. Acesso em: 10 novembro 2001. Trabalho não publicado.
- [NIC98] NICHOLS, Kathleen, at al. **Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers**, Request for Comments: 2474, Internet Engineering Task Force, dezembro 1998.
- [NIC99] NICHOLS, Kathleen; JACOBSON, Van; ZHANG, Lixia. **A Two-bit Differentiated Services Architecture for the Internet**. Request for Comments: 2638, Internet Engineering Task Force, julho 1999.
- [RAD99] RADHAKRISHNAN, Saravanan. **Linux – Advanced Networking Overview Version 1**. Disponível em <<http://qos.ittc.ukans.edu/howto>>. Acesso em: 21 junho 2001.
- [REI98] REICHMEYER, Francis, at al. **A Two-Tier Resource Management Model for Differentiated Services Networks**. Novembro 1998. Disponível em: <<http://irl.cs.ucla.edu/papers/2-tier-draft.ps>>. Acesso em: 25 janeiro 2001.
- [RIS98] RISSO, Fulvio; GEVROS, Panos. Operational and Performance Issues of a CBQ router. **Computer Communication Review**, New York, v. 24, n. 5, p. 47-58, outubro, 1999.

- [ROY00] ROY, Alain. GARA: A Quality of Service Architecture. In: FIRST JOINT INTERNET2 / DOE QOS WORKSHOP, 1., 2000, Houston, USA. **Proceedings...** [S.l. : s. n.], 2000. Disponível em: <<http://www.internet2.edu/qos/houston2000/proceedings/Roy/20000209-QoS2000-Roy.pdf>>. Acesso em: 10 março 2001.
- [SAL99] SALIM, Jamal Hadi. Kernel Korner: IP Bandwidth Management. **Linux Journal**. USA, n. 62, 1 junho 1999. Disponível em <<http://www.linuxjournal.com/article.php?sid=3369>>. Acesso em: 22 março 2000.
- [SAN00] SANDER, Volker, et al. A Differentiated Services Implementation for High-Performance TCP Flows. In: TERENA NETWORKING CONFERENCE, 2000, Lisboa, Portugal. **Proceedings...** [S.l. : s. n.], 2000. Disponível em: <<http://www.cdt.luth.se/~olov/publications/TWQoS-99a.pdf>>. Acesso em: 21 abril 2001.
- [SHC99] SCHELÉN, Olov, et al. Performance of QoS Agents for Provisioning Network Resources. In: IFIP INTERNATIONAL WORKSHOP ON QUALITY OF SERVICE, 7., 1999, Londres, Inglaterra. **Proceedings...** [S.l. : s. n.], 1999. Disponível em: <<http://www.terena.nl/tnc2000/proceedings/4B/4b3.pdf>>. Acesso em: 21 novembro 2001.
- [SRE99] SREEKANTAN, Deepak; RAO, Dhananjaya. **Implementation of a Bandwidth Broker System for Resource Management in Differentiated Services**. Disponível em: <<http://www.itc.ku.edu/~kdrao/845>>. Acesso em: 1 março 2001.
- [STE00] STELZL, Rudi. The Siemens Bandwidth Broker. In: FIRST JOINT INTERNET2 / DOE QOS WORKSHOP, 2000, Houston, USA. **Proceedings...** [S.l. : s. n.], 2000. Disponível em <<http://www.internet2.edu/qos/houston2000/proceedings/Stelzl/20000209-QoS2000-Stelzl.pdf>>. Acesso em: 1 março 2001.
- [STR00] STRICKLEN, Michael; CUMMINGS, Bob; MCCLELLAN, Stan. Linux and the Next Generation Internet. **Linux Journal**. USA, n. 72, 1 abril 2000. Disponível em: <<http://www.linuxjournal.com/article.php?sid=3928>>. Acesso em: 10 abril 2000.

- [TEI98] TEITELBAUM, B. QoS Requirements for Internet2. In: INTERNET2: JOINT APPLICATIONS/ENGINEERING QOS WORKSHOP, 1998, Santa Clara, USA. **Anais...** [S.l. : s. n.], 1998. Disponível em: <<http://www.internet2.edu/qos/may98Workshop/html/requirements.html>>. Acesso em: 13 julho 2000.
- [TEI99] TEITELBAUM, B. **Qbone Architecture (v1.0)**. Agosto 1999. Disponível em: <<http://www.internet2.edu/qos/wg/papers/qbArch/index.html>>. Acesso em: 9 dezembro 2000. Trabalho não publicado.
- [TER00] TERZIS, Andreas. The Role of BB in Traffic Engineering. In: FIRST JOINT INTERNET2 / DOE QOS WORKSHOP, 2000, Houston, USA. **Proceedings...** [S.l. : s. n.], 2000. Disponível em <<http://www.internet2.edu/qos/houston2000/proceedings/Terzis/20000209-QoS2000-Terzis.pdf>>. Acesso em: 1 março 2001.
- [VER91] VERMA, Dinesh; ZHANG, Hui; FERRARI, Domenico. Delay Jitter Control for Real-Time Communication in a Packet Switching Network. In: IEEE TRICOM '91, 2, 1991, Chapel Hill, USA. **Proceedings...** [S.I.]: IEEE Press, 1991. 512 p. 35-43.

ANEXO 1 - Descrição dos Arquivos da Aplicação Web

PEPEdgeInit.jsp

O arquivo PEPEdgeInit.jsp realiza o cadastramento dos agentes de configuração localizados nos computadores dos clientes do sistema. Os agentes de configuração, quando inicializados, invocam este script no servidor de aplicações informando o endereço IP do computador no qual estão rodando. O script adiciona estes endereços na lista de clientes, possibilitando que requisições do serviço Premium para transmissões com fonte nestes computadores sejam aceitas pelo sistema.

PEPEdgeDeInit.jsp

O arquivo PEPEdgeDeInit.jsp realiza o descadastramento dos agentes de configuração localizados nos computadores dos clientes do sistema. Ele é invocado quando os agentes de configuração estão terminando sua execução.

PEPCoreInit.jsp

O arquivo PEPCoreInit.jsp realiza o cadastramento dos agentes de configuração localizados nos roteadores. Os agentes de configuração, quando inicializados, invocam este script no servidor de aplicações informando o endereço IP das interfaces do roteador no qual estão rodando. O script adiciona estes endereços na lista dos roteadores cadastrados. A partir deste momento, o roteador está habilitado para receber ordens de configuração.

PEPCoreDeInit.jsp

O arquivo PEPCoreDeInit.jsp realiza o descadastramento dos agentes de configuração localizados nos roteadores. Ele é invocado quando os agentes de configuração estão terminando sua execução.

PDP.jsp

O arquivo PDP.jsp contém o código JSP e HTML necessário para apresentar os formulários de requisição de serviço, requisição de medição e listagem de clientes e roteadores. Através de recursos de persistência fornecidos pelo servidor de aplicação, a aplicação Web mantém os objetos de dados disponíveis entre requisições dos clientes. Além disso, o script também é responsável pelo processamento das requisições de alocação e desalocação de serviço. Através de *sockets* TCP, o script envia ordens de configuração para os agentes da rede. O resultado de cada operação é apresentado ao cliente através de uma mensagem abaixo do formulário de requisição.

Cada alocação realizada é armazenada em um objeto da classe HashTable. Este objeto serve para armazenar os dados relativos às alocações e medições de cada cliente. A classe Hashtable armazena dados através de uma chave e um valor. Neste sistema, o IP do cliente foi usado como chave e um vetor de *strings* contendo informações sobre cada alocação do cliente foi usado como valor. Estas informações são formadas pelo IP do destino, a largura de banda alocada e os seguintes valores da última medição: latência mínima, média e máxima, variação de latência mínima, média e máxima e porcentagem de pacotes perdidos.

PDPmeasure.jsp

O arquivo PDPmeasure.jsp é usado para processar requisições de medição. Se todos os parâmetros forem válidos, o script se conecta com o agente de medição presente no cliente e envia os parâmetros da medição que deve ser feita. Após o término da medição, o script recebe os resultados e salva estes valores no objeto Hashtable.

ANEXO 2 - DESCRIÇÃO DAS CLASSES DO AGENTE DE DESCOBERTA DE ROTAS

Traceroute

- startListener

```
public boolean startListener()
```

Este método implementa um servidor que espera por conexões do gerenciador na porta 49151. Cada conexão estabelecida com o agente representa uma execução da ferramenta *Traceroute*. Após receber do gerenciador o endereço do destino, o agente invoca o método nativo *execTraceroute*, cuja única funcionalidade é executar a ferramenta *Traceroute* do sistema e escrever a saída em um arquivo.

- getRoute

```
public String[] getRoute()
```

Este método lê o arquivo com o resultado do *Traceroute* e gera uma lista com os endereços IP que formam a rota.

Retorna uma lista de endereços IP.

ANEXO 3 - DESCRIÇÃO DAS CLASSES DO AGENTE DE CONFIGURAÇÃO

PEPedge

- Init

```
public boolean Init()
```

Este método invoca o método nativo *initDiffserv* para realizar a configuração inicial de DiffServ no computador cliente e se comunica com a aplicação gerente para cadastrar este cliente.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- DeInit

```
public boolean DeInit()
```

Este método invoca o método nativo *deInitDiffserv* para remover a configuração DiffServ do computador cliente e se comunica com a aplicação gerente para remover o cadastro deste cliente.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- startListener

```
public boolean startListener()
```

Este método espera por conexões na porta 49150 e, de acordo com os parâmetros enviados, invoca o método nativo *addMarker* para instalar um marcador de pacotes para um novo fluxo ou o método nativo *delMarker* para remover um marcador de pacotes já existente.

- initDiffserv

```
public native boolean initDiffserv()
```

Este método cria uma disciplina de enfileiramento do tipo DSMARK em uma determinada interface. Esta disciplina de enfileiramento implementa a marcação de pacotes e serve de base para a configuração dos classificadores u32 e condicionadores de tráfego.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- `deInitDiffserv`

```
public native boolean deInitDiffserv()
```

Este método remove todos os mecanismos de controle de tráfego existentes na interface.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- `addMarker`

```
public native boolean addMarker(String src, String dest, String rate)
```

Este método adiciona um marcador para um fluxo de dados entre *src* e *dest*, condicionado a uma taxa de transmissão máxima *rate*.

Parâmetros:

src – o endereço IP fonte;

dest – o endereço IP destino;

rate – a largura de banda.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- `delMarker`

```
public native boolean delMarker(String src, String dest, String rate)
```

Este método remove um marcador existente configurado para um fluxo de dados entre *src* e *dest* e com taxa de transmissão máxima *rate*.

Parâmetros:

src – o endereço IP fonte;

dest – o endereço IP destino;

rate – a largura de banda.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

PEPcore

- `Init`

```
public boolean Init()
```

Este método invoca o método nativo *initDiffserv* para realizar a configuração inicial de DiffServ no roteador e se comunica com a aplicação gerente para cadastrar este roteador.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- DeInit

```
public boolean DeInit()
```

Este método invoca o método nativo *deInitDiffserv* para remover a configuração DiffServ do roteador e se comunica com a aplicação gerente para remover o cadastro deste roteador.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- startListener

```
public boolean startListener()
```

Este método espera por conexões na porta 49150 e, de acordo com os parâmetros enviados, invoca o método nativo *setClassifier* para alterar a largura de banda que deve ser reservada para o serviço Premium.

- initDiffserv

```
public native boolean initDiffserv()
```

Este método cria uma disciplina de enfileiramento do tipo CBQ para classificar e priorizar os pacotes Premium.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- deInitDiffserv

```
public native boolean deInitDiffserv()
```

Este método remove todos os mecanismos de controle de tráfego existentes na interface.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

- setClassifier

```
public native boolean setClassifier(String rate)
```

Este método configura a largura de banda que deve ser reservada ao serviço Premium.

Retorna verdadeiro se a operação foi realizada com sucesso e falso caso contrário.

ANEXO 4 - DESCRIÇÃO DAS CLASSES DO AGENTE DE MEDIÇÃO

O sistema de medições [KIE01] é composto por quatro classes: `Random`, `RandonPoissonDistribution`, `TCPPProbe` e `UDPPProbe`. As duas primeiras são usadas para a geração de números randômicos de acordo com uma distribuição de Poisson. As outras duas são usadas para medições de transmissões TCP e UDP, respectivamente. Este sistema de medições é parte integrante dos agentes de medição e é executado de acordo com as requisições enviadas do gerenciador para o agente de medição. O resultado das medições é representado por uma lista contendo os valores de latência dos pacotes de teste. O agente de medição retorna ao gerenciador a lista de valores para posterior processamento pelo sistema de estatísticas, onde são gerados os valores desejados, como, por exemplo, latência mínima, média e máxima.

Classe `Random`

A classe `Random` implementa a funcionalidade necessária para a geração pseudo-randômica de números e caracteres através do auxílio das funções `rand` e `srand` da biblioteca de classes padrão do C++.

- `Random`

```
public Random()
```

Cria um novo objeto `Random` e semeia o gerador de números randômicos (`srand`) com o tempo corrente. Se o gerador não fosse semeado com valores diferentes, todas as instâncias gerariam a mesma seqüência de resultados.

- `nextDouble`

```
public double nextDouble(void)
```

Retorna o próximo número pseudo-randômico do tipo `double` de uma seqüência distribuída uniformemente entre 0.0 e 1.0.

- `nextChar`


```
public char nextChar(void)
```

Retorna o próximo caractere pseudo-aleatório de uma sequência.

RandomPoissonDistribution

Esta classe gera valores de acordo com uma distribuição de Poisson e é usada pelas classes de medição (TCPProbe e UDPProbe) para determinar o intervalo de tempo entre o envio de cada pacote em uma medição de latência. A grande vantagem do uso deste método, ao invés de intervalos de valor fixo, se deve à possibilidade de evitar um comportamento periódico nas transmissões, o qual pode diminuir a imparcialidade das amostras caso seja gerada uma sincronização.

- RandomPoissonDistribution

```
public RandomPoissonDistribution(long mean)
```

Cria um novo gerador de distribuição de Poisson com um determinado valor médio.

Parâmetros:

mean – o valor médio da distribuição de Poisson.

- NextLong

```
public long nextLong()
```

Retorna o próximo valor da distribuição de Poisson no formato *long*.

TCPProbe

As principais funcionalidades implementadas na classe *TCPProbe* são os métodos de medição de latência de transmissões TCP. Estes métodos seguem uma arquitetura cliente-servidor e possibilitam a medição da performance da rede de acordo com as métricas: latência *round-trip* e latência *one-way*. Para a geração e transmissão de pacotes TCP, foram utilizados *sockets* do tipo `SOCK_STREAM`. Através deste tipo de *sockets*, são criados dois canais de comunicação através de dois pares de portas, um para a troca de informações de controle e outro para a transmissão dos pacotes de medição.

- TCPProbe

```
public TCPProbe()
```

Cria um novo *probe* de medição para transmissões TCP.

- *probe*

```
public float probe(char *dest, unsigned short sport, unsigned short
    dport, unsigned long segNumber, unsigned short segSize)
```

Realiza medições de acordo com a métrica Type-P-Round-trip-Delay [KAL99b].

Deve ser executado no nó gerador dos pacotes de medição.

Parâmetros:

dest – o endereço do nó destino;

sport – a porta fonte dos pacotes;

dport – a porta destino dos pacotes;

segNumber – o número de pacotes que serão transmitidos; e

segSize – o tamanho destes pacotes.

Retorna o tempo gasto pelo método para realizar a medição.

O método cria dois canais de comunicação com o nó endereçado pelo parâmetro *dest*: um canal de controle e um canal para os pacotes de medição. O canal de controle é criado em portas pré-definidas no sistema. O canal para os pacotes de medição é criado nas portas especificadas pelos parâmetros *sport* e *dport*. Através do canal de controle, o método *probe* se comunica com o método *dispatch* do nó destino com a finalidade de informar o tamanho (*segSize*) e o número de pacotes (*segNumber*) que serão transmitidos. Usando o método *nextChar* da classe *Random*, é criada uma lista de caracteres do tamanho de *segSize* para ser inserida em cada pacote. É importante frisar que esta lista é formada por caracteres gerados randomicamente para evitar que o tempo de medição seja menor do que deveria ser devido a alguma técnica de compressão aplicada durante o caminho.

Com o auxílio do método *nextLong* da classe *RandomPoissonDistribution*, cada pacote é transmitido entre intervalos que seguem uma distribuição de Poisson com média de 500 milissegundos. Cada pacote que retorna tem seus tempos de envio e recepção subtraídos para determinar o valor de latência. Este valor é, então, colocado em uma lista na posição representada pelo identificador de seqüência do pacote. As posições dos pacotes perdidos na lista são preenchidas com o valor -2147483648.

- `dispatch`

```
public float dispatch(unsigned short port)
```

Realiza medições de acordo com a métrica Type-P-Round-trip-Delay [KAL99b].

Deve ser executado no nó receptor dos pacotes de medição.

Parâmetros:

`port` – a porta de recepção dos pacotes.

Retorna o identificador do processo que está executando o método.

Este método é contatado pelo método *probe* para a realização de uma medição. Ele é informado do tamanho e número de pacotes que serão transmitidos e se prepara para a recepção. Cada pacote recebido é imediatamente retransmitido para que a latência de processamento não influa muito no valor final da medição.

- `probe2`

```
public float probe2(char *dest, unsigned short sport, unsigned short
    dport, unsigned long segNumber, unsigned short segSize)
```

Realiza medições de acordo com a métrica Type-P-One-way-Delay [KAL99a].

Deve ser executado no nó gerador dos pacotes de medição.

Parâmetros:

`dest` – o endereço do nó destino;

`sport` – a porta fonte dos pacotes;

`dport` – a porta destino dos pacotes;

`segNumber` – o número de pacotes que serão transmitidos; e

`segSize` – o tamanho destes pacotes.

Retorna o tempo gasto pelo método para realizar a medição.

Similarmente ao método *probe*, o método *probe2* cria dois canais de comunicação com o nó do endereço *dest*: um canal de controle e um canal para os pacotes de medição. Neste caso, as medições são realizadas em um sentido para o método *dispatch2* do nó destino, não havendo recepção de nenhuma informação até que todos os pacotes tenham sido enviados. Somente no final da medição, o método *dispatch2* transmite a lista de latências resultante da medição para o método *probe* através do canal de controle.

- `dispatch2`

```
public float dispatch2(unsigned short port)
```

Realiza medições de acordo com a métrica Type-P-One-way-Delay [KAL99a]. Deve ser executado no nó receptor dos pacotes de medição.

Parâmetros:

`port` – a porta de recepção dos pacotes.

Retorna o identificador do processo que está executando o método.

O método *dispatch2* é contatado pelo método *probe2* para a realização de uma medição. Ele é informado do tamanho e número de pacotes que serão transmitidos e se prepara para a recepção. Cada pacote recebido tem sua latência imediatamente calculada através da subtração do tempo corrente do sistema pelo tempo de envio existente dentro do pacote. Estes valores de latência são colocados em uma lista indexada pelo identificadores dos pacotes. Somente no final da medição esta lista é transmitida para o gerador dos pacotes de medição.

UDPProbe

A classe UDPProbe foi desenvolvida para realizar os mesmos tipos de medições da classe TCPProbe, porém com pacotes UDP. Seus métodos têm a mesma assinatura dos métodos da classe TCPProbe, possibilitando o fácil uso de ambas as classes para realização de medições através das métricas Type-P-Round-trip-Delay e Type-P-One-way-Delay.

A criação e transmissão dos pacotes UDP é feita através de *sockets* do tipo SOCK_DGRAM. Um detalhe importante é que o canal de comunicação para informações de controle continua sendo criado com *sockets* SOCK_STREAM devido à garantia de recebimento disponibilizada pelo protocolo TCP. O uso do protocolo UDP nas transmissões dos pacotes exigiu que os métodos de medição implementassem algum mecanismo de controle de tempo de espera. O procedimento padrão do comando *recvfrom* dos *sockets* SOCK_DGRAM é esperar indefinidamente pela chegada de algum pacote. Porém, a metodologia de medição seguida nesta implementação define que, após algum limite de tempo pré-determinado, o sistema deve desistir da recepção do pacote e determiná-lo perdido. Este controle foi implementado através do comando *setsockopt*, o qual serve para alterar diversas características dos *sockets*. Desta forma, através dos parâmetros adequados, foi possível limitar o tempo de espera do *socket*.

Após a criação da lista que receberá os valores de latência dos pacotes, cada posição é preenchida com o valor -2147483648, o qual caracteriza um pacote perdido. A detecção de *timeout* dos pacotes é realizada através da checagem do valor de retorno do comando de recepção *recvfrom*. Se o valor for menor que zero, indicando um erro, e este valor for igual a constante *EAGAIN*, isto significa que o sistema ficou esperando a chegada de um pacote até o limite de tempo configurado. Neste caso, o sistema apenas começa uma nova iteração no laço. Como as posições na lista de resultados são preenchidas com o valor de pacote perdido antes do início das medições, não é necessário fazer nada nesta situação. Apenas os pacotes que chegam inteiros e dentro do limite de tempo têm suas latências inseridas na lista.

- UDPProbe

```
public UDPProbe()
```

Cria um novo *probe* de medição para transmissões UDP.

- probe

```
public float probe(char *dest, unsigned short sport, unsigned short
    dport, unsigned long segNumber, unsigned short segSize)
```

Realiza medições de acordo com a métrica Type-P-Round-trip-Delay [KAL99b].

Deve ser executado no nó gerador dos pacotes de medição.

Parâmetros:

dest – o endereço do nó destino;

sport – a porta fonte dos pacotes;

dport – a porta destino dos pacotes;

segNumber – o número de pacotes que serão transmitidos; e

segSize – o tamanho destes pacotes.

Retorna o tempo gasto pelo método para realizar a medição.

Este método tem a mesma funcionalidade do método *probe* da classe *TCPProbe*, com a única diferença que os pacotes de medição são transmitidos através do protocolo UDP.

- dispatch

```
public float dispatch(unsigned short port)
```

Realiza medições de acordo com a métrica Type-P-Round-trip-Delay [KAL99b].

Deve ser executado no nó receptor dos pacotes de medição.

Parâmetros:

port – a porta de recepção dos pacotes.

Retorna o identificador do processo que está executando o método.

Este método tem a mesma funcionalidade do método *dispatch* da classe TCPProbe, com a única diferença que os pacotes de medição são transmitidos através do protocolo UDP.

- probe2

```
public float probe2(char *dest, unsigned short sport, unsigned short
    dport, unsigned long segNumber, unsigned short segSize)
```

Realiza medições de acordo com a métrica Type-P-One-way-Delay [KAL99a].

Deve ser executado no nó gerador dos pacotes de medição.

Parâmetros:

dest – o endereço do nó destino;

sport – a porta fonte dos pacotes;

dport – a porta destino dos pacotes;

segNumber – o número de pacotes que serão transmitidos; e

segSize – o tamanho destes pacotes.

Retorna o tempo gasto pelo método para realizar a medição.

Este método tem a mesma funcionalidade do método *probe2* da classe TCPProbe, com a única diferença que os pacotes de medição são transmitidos através do protocolo UDP.

- dispatch2

```
public float dispatch2(unsigned short port)
```

Realiza medições de acordo com a métrica Type-P-One-way-Delay [KAL99a].

Deve ser executado no nó receptor dos pacotes de medição.

Parâmetros:

port – a porta de recepção dos pacotes.

Retorna o identificador do processo que está executando o método.

Este método tem a mesma funcionalidade do método *dispatch2* da classe TCPProbe, com a única diferença que os pacotes de medição são transmitidos através do protocolo UDP.