CENTRO UNIVERSITÁRIO DE BRASÍLIA – UNICEUB
FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA - FAET
CURSO DE ENGENHARIA DA COMPUTAÇÃO

JOSÉ JACKSON MACHADO BACELAR JÚNIOR

METODOLOGIA PARA ANÁLISE DE TRÁFEGO EM REDES MPLS

Brasília – DF
2008

JOSÉ JACKSON MACHADO BACELAR JÚNIOR

METODOLOGIA PARA ANÁLISE DE TRÁFEGO EM REDES MPLS

Monografia apresentada como requisito parcial para a conclusão do curso de bacharelado em Engenharia da Computação do Centro Universitário de Brasília – UniCEUB.

Prof. Orientador Marco Antônio Araujo.

Brasília – DF

2008

JOSÉ JACKSON MACHADO BACELAR JÚNIOR

METODOLOGIA PARA ANÁLISE DE TRÁFEGO EM REDES MPLS

Aprovado por:

Prof. Orientador Marco Antônio Araújo
(Orientador)

Prof.
(Membro)

Prof.
(Membro)

Brasília – DF

Agradecimentos

Agradeço aos meus colegas e amigos que muito me ajudaram e contribuíram para tornar essa caminhada mais fácil e divertida.

Agradeço ao Professor Marco Antônio Araujo, cujas sugestões contribuíram muito para a qualidade deste trabalho.

Dedicatória

Dedico esse trabalho aos meus pais e a toda a minha família. Em especial ao Gustavo e à Vera cujo carinho e amor me deram força para enfrentar as dificuldades desse caminho.

RESUMO

A evolução das redes de computadores tornou esses ambientes bastante complexos. Atualmente, são interligadas diferentes redes, compostas por uma variedade muito grande de equipamentos com distintas funções. Ao mesmo tempo, diferentes serviços de dados, áudio e vídeo passaram a compartilhar os mesmos recursos da rede. Novas tecnologias, como a *Multi Protocol Label Switching* (MPLS), foram desenvolvidas com o objetivo de otimizar a utilização desses recursos. Paralelamente ao aumento de sua complexidade, as redes passaram a exercer um papel fundamental para as empresas. Essas empresas demandam garantias para a qualidade dos serviços a custos reduzidos. Nesse cenário, cresce a importância da gerência de redes, que necessita de mecanismos para uma análise detalhada dos dispositivos instalados na rede. Nesse sentido, o objetivo deste trabalho, de cunho acadêmico, é apresentar uma proposta de metodologia para a análise de desempenho de redes MPLS. Para o embasamento desta proposta de metodologia, foi feito um levantamento dos recursos exigidos pelos principais serviços providos pelas redes de computadores com métricas de SLA e garantias de QoS. Também, foi elaborada uma análise da tecnologia MPLS e da teoria do gerenciamento de redes. Com o objetivo de testar a metodologia proposta, foi realizado um estudo de caso onde foi montado um protótipo de rede MPLS. Na análise do protótipo foi constatado que, quando da utilização de um circuito de baixa velocidade, o tempo de resposta estava superior à velocidade do circuito. Porém, quando a velocidade foi aumentada o tempo de resposta não acompanhou a evolução do circuito. Na verificação do QoS do protótipo, os fluxos prioritários obtiveram pouca vantagem sobre os demais. De um modo geral, observou-se que as redes MPLS precisam ser estudadas com mais profundidade para que se possa atingir o seu potencial.

Palavras-chave: Redes MPLS; Análise de desempenho; Gerência de redes; Metodologia para análise de redes MPLS.

ABSTRACT

The development of the computers networks made this kind of environment very complex. Today, different networks are interconnected, with grate variety o equipments with distinct functions. At the same time, different services of data, audio and video are now sharing the same resources of the network. New technologies, as *Multi Protocol Label Switching* (MPLS), were deployed with the objective of optimize the utilization of those resources. Simultaneously the improvement of its complexity, the networks became essential to the enterprises. The enterprises need warranties for the quality of the services at low costs. In this scenario, increases the importance of the network management and, it needs mechanisms to make detailed analyses of the network resources. In this way, the objective of this academic work is to present a methodology to make MPLS network performance analyses. This proposal methodology is based on the study of the resources needed by the most important services provided by the computers networks, including SLA metrics and QoS warranties. Also, were analyzed the MPLS technology and the network management theory. In the purpose to test the proposal methodology, a case study was made, with a MPLS network prototype. The prototype's analises showed that when slower velocity circuit was used, the answer time was over the circuit speed. However, when the speed was impruved the answer time did not increased as well. When QoS was checked the priority flows had few advantage over the others. In general, we can say that the MPLS networks need deeper studies to ative its potential.

Keywords: MPLS network; Performance analyses; Network management; MPLS network analyses methodology.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ARS - *Remedy Action Request System*

ATM - *Asynchronous Transfer Mode*

CGI - *Common Gateway Interface*

CMISE/CMIP - *Common Management Information Service Element/Common Management Information Protocol*

CoS - *Class of Service*

DSCP *Diffserv Code Point*

EXP - Experimental

FEC - *Forwarding Equivalence Class*

FIB - *Forwarding Information Base*

HTML - *HyperText Markup Language*

IETF - *Internet Engineering Task Force*

INTSERV - *Integrated Services Architecture*

IP - *Internet Protocol*

ISO - *International Organization for Standardization*

MPLS - *Multi Protocol Label Switching*

MPLS TE MIB - *MultiProtocol Label Switching Traffic Engineering MIB*

NMS - *Network Management Station*

LER - *Label Edge Router*

LiME - *Linux based MPLS Emulator*

LSP - *Label Switched Path*

MIB - *Managment Information Bases*

NS2 - *Network Simulator*

NVRAM - *Non-Volatile Random Access Memory*

OID - *Object Indentifier*

OSI - *Open Systems Interconnection*

PLR - *Packet Loss Rate*

PVC - *Permanent Virtual Circuit*

QoS - Qualidade de Serviço / *Quality of Service*

RMON - *Remote Monitoring*

RFC - *Request for Comments*

RIB - *Routing Information Base*

RSVP - *Resource ReSerVation Protocol*

RTT -*Round-Trip Time*

SLA - *Service Level Agreement*

SNMP - *Simple Network Management Protocol*

TCP - *Transmission Control Protocol*

TFTP - *Trivial File Transfer Protocol*

TMN - *Telecommunications Management Network*

TTL - *Time To Live*

UDP - *User Datagram Protocol*

VoIP - Voz sobre IP

WEB - *World Wide Web*

CAPÍTULO 1. INTRODUÇÃO

As redes de computadores nos últimos anos se tornaram um dos mais importantes meios de comunicação da humanidade. Em sua evolução, passaram a fornecer os mais variados tipos de serviço que vão desde a diversão até transações comerciais. Utilizam-se as redes, entre outras coisas, para fazer ligações telefônicas com o VoIP, fazer compras, fazer transações bancárias. As empresas, também, utilizam as redes para oferecerem seus serviços, para trocar dados com suas filiais ou ainda para fazer vídeos-conferência. Todos esses serviços compartilham os mesmos recursos da rede, cada um com suas necessidades específicas. Para tentar compatibilizar esses diferentes tipos de demandas, com as limitações dos recursos, novas tecnologias foram desenvolvidas. Dentre essas novas tecnologias, está a Multi Protocol Label Switching (MPLS) que classifica o tráfego IP, estabelece prioridades de acordo com as características dos serviços e cria túneis com reserva de banda para a passagem dos fluxos. Dessa forma, se propõe a dar um novo salto no atendimento dos interesses dos clientes e na otimização dos recursos das redes.

O objetivo principal deste trabalho é a proposição de uma metodologia para auxiliar administradores de redes MPLS a analisarem o desempenho dessas redes. Para tanto, é necessário saber: qual o objetivo da análise; qual a complexidade e o tamanho da rede; que dados precisam ser coletados; qual é a fonte desses dados; que sistema de coleta se adapta melhor a essa realidade e que ferramenta vai ser utilizada para organizar esses dados e permitir que o administrador tenha informações para embasar intervenções corretas na rede. O gerente da rede precisa saber se cada tipo de serviço está sendo tratado de acordo com suas necessidades. Por exemplo, se os serviços críticos estão sendo priorizados. Outro item a ser investigado é a capacidade da rede em termos de taxa de transmissão. Todo esse conhecimento só vai ser atingido se o gerente adotar uma metodologia para a análise do desempenho da rede. Para testar essa metodologia, foi realizado um estudo de caso onde foi criado um protótipo de rede MPLS e foi desenvolvida uma ferramenta para verificação do desempenho dessa rede.

O problema ocorre porque, normalmente, os administradores não estudam, de maneira sistemática, as redes contratadas. Quando se contrata uma rede de dados, deseja-se que todos os seus serviços sejam bem atendidos. No momento dessa contratação, devem ficar estabelecidos os parâmetros de qualidade dessa rede, tais como, largura mínima de banda,

tempo máximo de indisponibilidade, taxa máxima de perda de pacotes. Esse item do contrato é chamado de Acordo de Nível de Serviço ou SLA. Nesse processo, muitos chegam até o SLA. Porém, o administrador da rede necessita de uma metodologia para análise do desempenho dessa rede inclusive para verificar se o SLA está sendo cumprido conforme contratado.

A medição da rede pode trazer vários benefícios. Com a medição é possível, por exemplo, verificar se a rede está bem dimensionada. Caso a rede esteja subdimensionada, ela não está atendendo adequadamente. Por outro lado, se estiver superdimensionada, provavelmente o custo está acima do benefício. Outra justificativa para a medição é verificar se o desempenho da rede instalada está de acordo com o SLA contratado. É preciso conhecer a rede para melhorá-la e o mercado precisa de instrumentos para fazer esse tipo de aferição.

Para alcançar o objetivo proposto, este trabalho foi dividido em seis capítulos: o primeiro é a apresentação do trabalho; o segundo apresenta uma visão da evolução das redes, as métricas de SLA, mostra como as redes oferecem qualidade para diferentes serviços e apresenta, também, uma parte teórica do gerenciamento de redes; o terceiro trata da rede MPLS, nesse capítulo tem uma visão geral da tecnologia MPLS, seu funcionamento básico e suas principais características; no quarto capítulo é proposta uma metodologia, em sete etapas, para análise de redes MPLS; no quinto capítulo é apresentado um estudo de caso para teste da metodologia proposta. Para tanto, foi criado um protótipo de rede MPLS e desenvolvida uma ferramenta para a medição do desempenho dessa rede; o sexto capítulo trás as conclusões desse trabalho.

CAPÍTULO 2. EVOLUÇÃO DAS REDES

As redes de computadores evoluíram muito nos últimos 25 anos. Nesse período, houve um aumento explosivo do número de redes e de usuários. Paralelamente a esse aumento, houve uma mudança no perfil do tráfego. Muitos serviços novos, como voz e vídeo conferência, com necessidades diferentes de recursos de rede, passaram a compartilhar esse ambiente. Com o tempo, surgiu a necessidade de otimizar a utilização dos recursos das redes com o objetivo de harmonizar a convivência das diferentes demandas dos novos e dos antigos serviços com a estrutura da rede existente. Para tanto, novas tecnologias, como IP com QoS e MPLS, surgiram para dar tratamento diferenciado aos serviços de acordo com suas necessidades. Por outro lado, os usuários das redes passaram a exigir a boa qualidade dos serviços.

No início da internet os pontos interligados não eram muitos, apenas as universidades e organizações governamentais utilizavam esse tipo de recurso. Em 1980, no mundo existiam somente dez redes conectadas com um total de cem computadores. Nesse início o IP cumpriu papel fundamental para a interconexão de redes diferentes. “Nos últimos vinte e cinco anos, uma tecnologia foi criada para tornar possível interconectar muitas redes físicas divergentes e operá-las como uma unidade coordenada” [1].

Jon Postel, um dos principais pioneiros da internet, escreveu em 1987: “Sistemas de comunicação por computador e redes atualmente são separados e fragmentados. O objetivo da interconexão e interligação de redes, ter uma única rede de comunicação de computador poderosa, é fundamental para o projeto do TCP/IP” [1]. Essa frase, dita há mais de vinte anos, pelo homem que durante muitos anos foi o único editor das RFC, demonstra a importância que o IP representou para o enorme crescimento da internet. *Request for Comments* (RFC) são documentos oficiais da *Internet Engineering Task Force* (IETF) que descrevem padrões, protocolos, serviços, recomendações operacionais para a internet.

A quantidade de redes cresceu muito desde 1980. Em 2005 esse número já tinha atingido um milhão de redes e cem milhões de computadores interligados. Atualmente, empresas de todos os tamanhos têm suas redes privadas. Redes metropolitanas, continentais e mundiais são utilizadas para conectar os dispositivos das grandes empresas nacionais e multinacionais. Nesse período, a internet se tornou o elo que interconecta a todos.

Esse crescimento estrondoso não se deu apenas em termos quantitativos de redes e máquinas. Os tipos de dados trafegados, também, mudaram de forma expressiva.

Naquela época, apenas pacotes de dados circulavam na rede. Nesse tempo, as redes IP ofereciam apenas a possibilidade de encaminhamento de pacotes baseado em protocolos do menor esforço. Nessa situação, não havia garantias de entrega dos pacotes nem tratamento diferenciado para os vários tipos de serviço. Como esclarecem Evans e Filsfils [2], todo o tráfego era tratado de maneira igual.

Hoje, diferentes serviços trafegam nas redes corporativas e na internet, cada um com sua exigência específica em termos de retardo, variação de retardo, perda de pacotes e largura de banda. O VoIP, por exemplo, não exige mais do que 64Kbps de largura de banda, mas, é extremamente sensível ao retardo. Para tratar essa nova realidade, o protocolo IP evoluiu e, em seu QoS passou a classificar o tráfego e estabelecer prioridades de encaminhamento conforme as necessidades dos serviços. A nova tecnologia MPLS, também, surgiu como um reforço para a engenharia de tráfego tratar, de maneira mais apropriada, as necessidades dos serviços e o limite dos recursos das redes.

Se por um lado, tecnologias são desenvolvidas para otimizar a utilização dos recursos finitos das redes. Por outro, os clientes que utilizam as redes querem receber serviços com qualidade. Para tanto, os contratos já devem prever as métricas claras a serem atingidas para o cumprimento do acordo de nível de serviço (SLA). Por sua vez, os contratantes devem possuir mecanismos para a medição da rede e conferência do cumprimento do SLA.

2.1. Análise de Tráfego

“As aplicações variam em como elas usam a rede, como elas se comportam durante congestionamento na rede, e a importância que têm para sua organização” [6]. Por exemplo, os aplicativos de tempo real, normalmente são sensíveis ao retardo. Dentro desse grupo, existem aqueles serviços pouco exigentes quanto à largura de banda como VoIP e transações bancárias. Ainda dentro desse grupo, a vídeo-conferência, além da sensibilidade ao retardo, depende de uma largura de banda maior para atingir boa qualidade de sinal. Outros serviços como a transferência de grandes arquivos são tolerantes ao retardo; já a largura de banda, depende do tempo que esse serviço tem para se completar.

Fazem parte do tráfego da internet e das redes corporativas serviços de voz, vídeos-conferência, vídeos sob demanda, transações bancárias, troca de arquivos de dados,

correio eletrônico etc., cada um com sua característica, cada um com suas necessidades de largura de banda, retardo, variação de retardo, perda de pacote.

Voz sobre IP (VoIP) é um aplicativo sensível ao retardo. Esse tipo de serviço exige banda constante e baixo retardo. A conversação interativa que ocorre no VoIP tem como característica a pouca necessidade de banda. Dependendo da forma de digitalização, uma conversa utilizando a tecnologia VoIP pode necessitar de uma banda de 64Kbps ou menos. Porém, por se tratar de uma atividade com interatividade, o retardo pode ser crítico. Nesse caso, o retardo pode ser considerado bom se ficar abaixo de 150ms. Porém, a conversação poderá ser inviabilizada se ultrapassar os 400ms. A aplicação VoIP é bastante sensível à variação de retardo, mas, essa dificuldade pode ser compensada pelos *jitter buffers.* Os usuários da VoIP podem não perceber a perda de um pacote se essa ocorrer isoladamente. Contudo, caso mais de um pacote seja perdido seguidamente, haverá interferência na qualidade do serviço [2].

Algumas características do Vídeo sobre IP são similares às de voz. Vídeo, também, é sensível ao retardo. A variação de retardo pode ser administrada com o uso de *buffer* desde que seja garantido que esse tipo de fluxo não enfrentará problemas provocados por congestionamento na rede. Para tanto, é necessário que exista banda suficiente, garantida para esse serviço. Da mesma forma que em voz, a banda reservada para o vídeo deve ser calculada considerando o pico do tráfego esperado. Se a velocidade do circuito é capaz de absorver o tráfego e tenha a máxima taxa de erro admitida pequena, supõe-se que a perda de pacotes não afetará a qualidade do vídeo. A banda necessária para a transmissão de vídeo depende essencialmente do padrão de resolução adotado. Um vídeo de alta definição pode demandar uma banda de até 20 Mbps. Já um vídeo de resolução baixa, utilizado em aplicações de dispositivos manuais móveis, pode funcionar entre 50 e 200 Kbps.

Na vídeo-conferência, os arquivos de voz e vídeo são enviados em canais lógicos separados. Dessa forma, as exigências de voz são as mesmas especificadas para VoIP e as de vídeo são semelhantes às do vídeo sobre IP. Embora, a codificação do vídeo use um padrão de menor definição e tenha menor necessidade de banda [2].

Apesar de sua grande variedade, as aplicações de dados podem ser divididas basicamente em aplicações interativas e não interativas. De qualquer forma, é fundamental conhecer as características do aplicativo para que seja feita uma análise de suas necessidades.

Um exemplo de tráfego de dados não interativo é o utilizado para fazer cópias de segurança entre servidores ou centros de processamento. Esse serviço depende da taxa de

transferência de dados para que se complete no prazo. Normalmente, esses serviços que dependem da taxa de transferência usam TCP, tendo em vista as garantias que esse protocolo da camada de transporte prove. Outros aplicativos, como o *Trivial File Transfer Protocol* - TFTP, que dependem da taxa de transferência, mas não necessitam das garantias e do controle feitos pelo TCP podem utilizar o UDP [2].

O próprio TCP, dinamicamente, verifica o retardo por meio do *Round-Trip Time* - RTT, e ajusta a operação para atingir a máxima taxa de transmissão sem sobrecarregar a rede. A variação do retardo praticamente não interfere no TCP. Nesse caso, a perda de pacotes é controlada, pelo TCP, com o reenvio de pacotes não recebidos. Naturalmente, a taxa de transferência será influenciada pelos itens acima e pela capacidade do circuito de ligação das pontas [2].

No caso das aplicações de dados com interatividade, é difícil caracterizar um padrão de demanda em termos de retardo, taxa de retardo, banda ou perda de pacotes. "As implementações específicas de aplicações interativas podem variar, o impacto que as características da rede, como retardo, têm sobre elas também podem variar" [2]. Entretanto, trata-se de uma aplicação de tempo real sensível a altas taxas de perda de pacotes e de retardo. Uma aplicação onde o tempo de resposta é abaixo de 0.1 segundo passa a impressão de reação instantânea ao usuário. Entre 0.1 e 1 segundo o cliente pode perceber o retardo se a aplicação não tiver mecanismos para manter seu fluxo. Em um tempo de 1 a 10 segundos, a aplicação necessariamente tem que ter formas de manter o cliente atento. Em uma pesquisa realizada em 1997 por Peter Bickford ficou comprovado que metade dos usuários abandonam páginas da WEB após espera de 8.5 segundos [2]. Esse tipo de aplicação acontece, por exemplo, nas transações bancárias ou no comércio eletrônico, onde clientes interagem com os servidores de uma empresa. Nesse tipo de transação, outro item a ser observado é a disponibilidade. Esse fator é considerado crítico para os negócios de uma organização. Um cliente não atendido implica em um negócio não realizado.

Outro aspecto fundamental a ser considerado é o grau de importância que um determinado serviço tem para um usuário ou para uma empresa, sendo necessário identificar até que ponto um aplicativo pode afetar o negócio de uma empresa. Por exemplo, o sítio de uma empresa com vendas pela internet tem que está disponível o tempo todo. Preferencialmente, o retardo na comunicação entre os clientes e os servidores dessa empresa não deve ser perceptível ao cliente. Caso ocorra, deve ser controlado para não afetar diretamente a transação e o negócio. Outro tipo, muito comum, de serviço crítico é a série de transações financeiras feitas entre os clientes e bancos. Esses tipos de serviço, comumente

chamados de “missão crítica”, têm de ser tratados prioritariamente em relação a outros de pouca importância para a organização [2].

2.2. SLA

O acordo de nível de serviço (*Service Level Agreement* - SLA) é a parte do contrato, entre um cliente e um prestador de serviço, que especifica as métricas de qualidade de serviço a serem cumpridas pelo fornecedor. O uso do SLA é uma prática bastante comum nos contratos que envolvem os serviços de telecomunicações. No momento da contração, as métricas a serem observadas devem ser bem definidas. Por exemplo, nesse documento deverá constar o tempo de resposta máximo admitido para um determinado serviço ou, o tempo de recuperação de um determinado tipo de circuito. Essas métricas devem ficar bem claras para ambas as partes, além de, devidamente documentadas. Essa prática facilita o relacionamento futuro entre o cliente e o prestador de serviços.

Em complementação ao SLA, o cliente deve possuir mecanismos de medição do desempenho da rede para averiguar se o SLA estabelecido está sendo cumprido devidamente pelo fornecedor. Evans & Filsfils apresentam as principais métricas a serem observadas [2]:

a) Retardo da Rede - é a diferença de tempo entre o envio de um pacote e a sua recepção. Para aplicativos muito rígidos em termos de retardo, como o VoIP, o retardo é medido em um único sentido. Ou seja, a diferença de tempo entre um ponto de saída de um pacote e o tempo de chegada no outro ponto. A RFC 2679 define as métricas para medição desse tipo de serviço. Por outro lado, aplicativos menos sensíveis ao retardo como os que usam TCP, são medidos no percurso completo de ida e volta, também conhecido como RTT. Nessa medição, é contado o tempo que o pacote leva da saída de um ponto até chegar ao seu destino e voltar ao ponto original, descontado o tempo de processamento da ponta. A RFC 2681 define as métricas de medição para RTT;

b) Variação no retardo - é a taxa de variação do retardo. Normalmente, considera-se a variação do retardo em um único sentido para dois pacotes consecutivos. Ou seja, a relação da diferença observada entre o

retardo de um pacote e o retardo do pacote seguinte. Também, costuma-se medir a variação da média do mínimo ou do máximo retardo. A RFC 3393 trata dessa métrica;

c) Perda de pacote - a perda de pacotes é caracterizada pelo descarte de pacote de dados entre um ponto de entrada da rede e um ponto de saída da rede. Normalmente, provocado pela não recepção de um determinado pacote em um tempo predefinido. As RFC 2680 e 3357 definem as métricas para a medição da Taxa de Perda de Pacote – PLR;

d) Entrega desordenada de pacotes - devido à possibilidade que os pacotes têm de trilhar caminhos diferentes entre a origem e o destino, esses pacotes podem chegar desordenados ao destino. Quanto maior for a taxa de desordem maior será o impacto da reordenação;

e) Banda de um circuito - é a quantidade de bits por segundo (*bps*) que um circuito de dados pode transportar;

f) Disponibilidade da rede - é o tempo que uma conectividade está disponível entre um ponto de entrada e um ponto de saída da rede. A garantia da conectividade, por si só, não garante a disponibilidade de um aplicativo. Muitas vezes, pode haver conectividade, mas o aplicativo não se viabiliza devido a, por exemplo, altas taxas de perda de pacotes. As métricas de verificação da conectividade estão especificadas na RFC 2678;

g) Disponibilidade do serviço - é o tempo em que um serviço está disponível entre um ponto de entrada e um ponto de saída da rede.

2.3. QoS

Qualidade de Serviço (QoS) é utilizado pelas redes atuais para tratar o tráfego de forma diferenciada. Dessa forma, é possível gerenciar os recursos da rede objetivando o atendimento das necessidades dos diversos tipos de serviço ou SLA. O QoS, também, é usado para tornar a utilização das redes mais eficiente, ou seja, aumentar a satisfação do cliente e aumentar a utilização da rede com conseqüente redução de custo.

> O problema central é a utilização: quando uma rede possui recursos suficientes para todo o tráfego, as restrições da QoS são desnecessárias; quando o tráfego excede a capacidade da rede, nenhum sistema de QoS pode satisfazer às demandas de todos os usuários. Na verdade, os proponentes que defendem QoS alegam que os mecanismos de QoS complexos atingem dois objetivos. Primeiro, dividindo os recursos existentes entre mais usuários, eles tornam o sistema mais 'justo'. Segundo, modelando o tráfego de cada usuário, eles permitem que a rede opere em uma taxa de utilização mais alta sem risco de um colapso [1].

A implementação de QoS pode ser vista de forma positiva. Primeiro, porque torna possível a contratação de serviços em redes IP com garantias de SLA. Segundo, o tratamento diferenciado do tráfego possibilita o pagamento do serviço de acordo com as exigências do SLA de cada aplicativo. "A rede é um recurso valioso que é compartilhado e tem capacidade finita" [6]. A utilização de QoS evita a imposição de redução do tráfego e também pode adiar a necessidade de aumento de banda.

A utilização de QoS envolve vários mecanismos para garantir que a rede trate o tráfego conforme suas características de SLA. Esses mecanismos podem ser divididos em "plano de dados" e "plano de controle". A tecnologia MPLS separa o plano de dados do plano de controle para permitir que as decisões de roteamento sejam tomadas baseado em critérios e não apenas no endereço de destino [2].

2.3.1. Plano de Dados

Os mecanismos do plano de dados do QoS são implementados nos roteadores que compõem a rede e servem para determinar as políticas de encaminhamento dos pacotes na rede. Para tanto, os fluxos precisam ser classificados e marcados. Além disso, esses fluxos devem receber informações das políticas de policiamento, enfileiramento e descarte de pacotes características da classe de serviço ao qual pertencem [2].

a) Classificação - é o processo de enquadramento dos fluxos de dados em determinadas classes de serviço CoS. Cada classe de serviço recebe um tratamento específico na rede;

b) Marcação - a marcação é a atribuição de valores DSCP no cabeçalho IP e EXP no MPLS. Assim, o tráfego pode ser identificado facilmente [2];

c) Policiamento - o policiamento serve para garantir que um fluxo de tráfego não ultrapasse a taxa máxima estabelecida [2]. O policiamento é feito na borda da rede e serve para garantir que o tráfego não ultrapasse o projetado. O policiamento, juntamente com a modelagem, pode ser usado para a obtenção de maiores taxas de transferência de uma determinada classe de tráfego;

d) Enfileiramento e Escalonamento - é o mecanismo do QoS que serve para a priorização. À medida que os pacotes chegam a um roteador de borda eles são enfileirados para posterior liberação na rede. Caso não haja qualquer tipo de congestionamento na rede, os pacotes são liberados na ordem em que chegam. Havendo a possibilidade de concorrência na rede, os roteadores têm mecanismos para exercer uma priorização de acordo com o estabelecido na classificação; e,

e) Descarte de pacotes - caso a taxa de chegada de pacotes seja maior que a capacidade de absorção da banda subseqüente, a memória do *buffer* vai se esgotar e pacotes terão de ser descartados. Nesse ponto também é possível o estabelecimento de priorização. Assim, os pacotes das classes prioritárias seriam os últimos a entrarem em risco de descarte.

2.3.2. Plano de Controle

Esse mecanismo faz o controle do estado da rede. É um mecanismo baseado em *software* que faz o controle de admissão e a reserva de recursos para um determinado fluxo. O protocolo mais utilizado para a reserva de recurso é o *Resource ReSerVation Protocol* - RSVP. No contexto da *integrated services architecture* (INTSERV), o protocolo RSVP é utilizado para fazer a reserva de recurso da rede para um fluxo e o controle da

admissão. Já no contesto da engenharia de tráfego MPLS, o RSVP é usado para o controle da admissão e para a criação de túneis MPLS.

O plano de controle é parte da arquitetura do roteaodor voltada para o mapeamento da rede. Ou ainda, é ligado às informações contidas na tabela de roteamento ou, *Routing Information Base* (RIB), que, normalmente contém a lista de endereços de destino e a *interface* de saída do roteador a ser utilizada. O plano de controle, também, pode definir o descarte de certo tipos de pacote, mas, principalmente, pode dar tratamento preferencial para pacotes com sinalização de QoS [2]. Algumas implementações utilizam, em separado, a *Forwarding Information Base* (FIB) que é alimentada pelo plano de controle e utilizada pelo plano de dados.

2.4. Gerenciamento de Redes

O aumento do tamanho e da complexidade das redes de computadores provocou o aparecimento de uma nova disciplina que é o gerenciamento de redes. A diversidade de equipamentos, aplicativos e serviços que compõem essas redes gerou a necessidade da criação de metodologias e ferramentas que viabilizam a sua administração. Nesse sentido, a *International Organization for Standardization* (ISO) definiu cinco áreas de gerenciamento de redes: gerencia de falhas; gerência de contabilidade; gerência de configuração; gerência de segurança e gerencia de desempenho. Apesar de estar baseado no modelo OSI de redes, esse padrão conceitual é amplamente aceito e adotado.

A gerência de falhas deve detectar o incidente, isolar os dispositivos com problema para que estes não interfiram no funcionamento do restante da rede e, se possível, fazer sua recuperação imediata. O registro da detecção do incidente e de sua solução pode auxiliar em trabalhos pró-ativos de prevenção de falhas. O grau de eficiência da gerência de falhas é inversamente proporcional à percepção dos usuários à ocorrência de incidentes na rede.

Cabe à gerência de contabilidade medir a utilização dos recursos da rede. Essa medição serve para regular o uso desses recursos evitando-se a lentidão ou indisponibilidade causada pelas tentativas além de sua capacidade. O conhecimento da utilização dos recursos possibilita a criação de políticas de acesso a determinados recursos. Como, por exemplo, a restrição de acesso a usuários ou grupos de usuários. Em outros casos, pode ficar comprovado

que a limitação não é a solução e assim, a solução é o aumento da capacidade instalada. Para se chegar a esses resultados, devem ser medidos todos os recursos importantes da rede.

A gerência de configuração é responsável pelo funcionamento adequado, no sistema, de diferentes equipamentos e aplicativos. Para tanto, escreve, muda, coleta, guarda e restaura configurações dos recursos do tipo: roteadores; estações de trabalho; servidores; switches e outros. Todos esses dispositivos têm uma variedade de informações disponíveis sobre eles que devem ser configuradas e gerenciadas para que possam contribuir com todo o seu potencial para o melhor funcionamento do sistema.

Os três pilares que devem pautar o gerenciamento da segurança são: confidencialidade da informação; integridade dos dados e disponibilidade do sistema. Para atingir esses objetivos a empresa deve ter uma política de segurança abrangente que vai além dos limites da rede de computadores. Internamente na rede, o gerente de segurança deve controlar e registrar os acessos aos recursos e informações consideradas importantes ou confidenciais [4].

O gerenciamento do desempenho da rede é importante não só para garantir a qualidade de serviço necessária às aplicações, como também para assegurar que esta possa ser atingida com os menores custos [3]. Através do gerenciamento de desempenho, pode ser verificado o nível de eficiência da rede em relação, por exemplo, ao investimento que foi feito para a sua instalação. Ou ainda, se está de acordo com o contratado, no caso da terceirização do serviço.

O desempenho da rede pode indicar, em cada dispositivo ou segmento, entre outras variáveis: a largura de banda disponível; a vazão; o percentual de utilização; taxa de erros; perda de pacotes; tempo de resposta e a qualidade dos serviços. No caso específico das redes MPLS, podem ser verificados, também, dados sobre o comportamento dos túneis estabelecidos nessa tecnologia. A coleta e análise desses dados podem mostrar a tendência a um problema, possibilitando sua solução antes que os serviços sejam afetados. O gerenciamento do desempenho, também, pode ser pró-ativo no sentido utilizar simuladores para projetar a expansão ou a construção de novas redes.

2.4.1. Padrões de gerenciamento

O crescimento das redes de computadores com o aumento da quantidade e variedade de equipamentos, protocolos, aplicativos e serviços, levou à necessidade do estabelecimento de critérios para garantir a eficiência da gestão dessas redes. Entre os principais padrões que começaram a ser desenvolvidos desde a década de 1980 destacam-se: o Common Management Information Service Element/Common Management Information Protocol (CMISE/CMIP); o Remote Monitoring (RMON); o Simple Network Management Protocol (SNMP) e o Telecommunications Management Network (TMN) [3].

O CMIP/CMIS é um modelo especificado pela ISO que tem como principais funções a monitoração, gestão de falhas, gestão de configuração e gestão de desempenho. Sua utilização é limitada por ser baseado em redes do modelo OSI.

O padrão RMON opera no modelo cliente/servidor. Dispositivos de monitoração, chamados de sonda, são utilizados para a coleta de informações e analise de pacotes. Essas sondas agem como servidores e os aplicativos de gerenciamento de rede que se comunicam com eles são considerados clientes. O fato desses dispositivos processarem os dados coletados, reduz o tráfego na rede. Nesse sistema, as informações só são transmitidas para o aplicativo de gerenciamento quando solicitadas. Por suas características, o RMON é utilizado na monitoração baseada em fluxo. Enquanto o SNMP é usado para o gerenciamento de baseado em dispositivos.

O SNMP é um padrão de gerenciamento aberto especificado pelo IETF (RFC 1067, RFC 1157, RFC 1901 e RFC 2571). Esse protocolo trabalha na camada de aplicação do modelo TCP/IP. Foi desenvolvido como uma adaptação do modelo OSI para as redes TCP/IP e é o padrão mais utilizado atualmente. O SNMP permite ao administrador manipular e monitorar todos os componentes de uma rede [3] [4].

O SNMP usa cinco mensagens básicas (Get, GetNext, GetResponse, Set e Trap) para a comunicação entre o gerente e o agente. As mensagens Get e GetNext são usadas pelo gerente para solicitar informação a uma variável especifica. Quando o agente recebe uma mensagem Get ou GetNext enviará uma mensagem GetResponse ao gerente com a informação solicitada ou com a indicação do erro que impediu o processamento do pedido. Do mesmo modo, a mensagem Set permite ao gerente solicitar que uma mudança seja feita no valor de uma variável. O agente, então, responderá, com uma GetResponse, que a mudança foi realizada ou que houve erro. A mensagem Trap permite ao agente informar, espontaneamente um evento importante. Dessa forma, a maioria das mensagens (Get, GetNext e Set) são usadas somente pelo gerente SNMP. A mensagem Trap é a única que pode

ser iniciada pelo agente. Esta é a mensagem usada pelo *Remote Telemetry Unit* (RTU) para comunicar um alarme ao gerente [3] [5].

A quantidade reduzida de comandos é só uma das razões para o SNMP ser considerado simples. Outro fator simplificador é o fato de estar baseado em um link de comunicação sem conexão, não supervisionado (UDP). Isso levou à sua vasta utilização, especialmente, na *Internet Network Management Framework*. Com essa ferramenta, o SNMP é considerado robusto pela independência que existe entre os gerentes e os agentes. Em outras palavras, se o gerente falha o agente continua funcionando e vice versa.

A arquitetura SNMP é baseada no conceito de gerente/agente, que permite a gerentes e agentes se comunicarem com o objetivo de acessar os objetos gerenciáveis. A figura abaixo mostra a arquitetura básica do sistema SNMP.

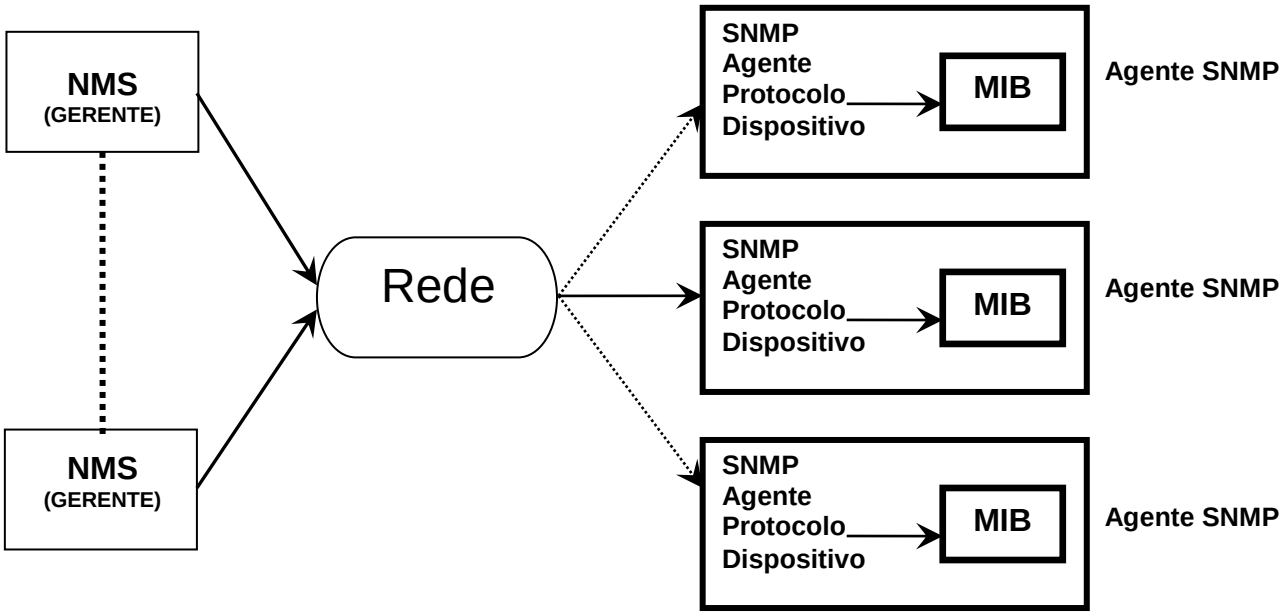


Figura 1. Modelo da arquitetura de gerenciamento SNMP

O protocolo SNMP é composto de:

Agente - nós gerenciados: disponibiliza a interface entre o gerente e o equipamento a ser gerenciado tais como pontes, hubs, roteadores, servidores de rede ;

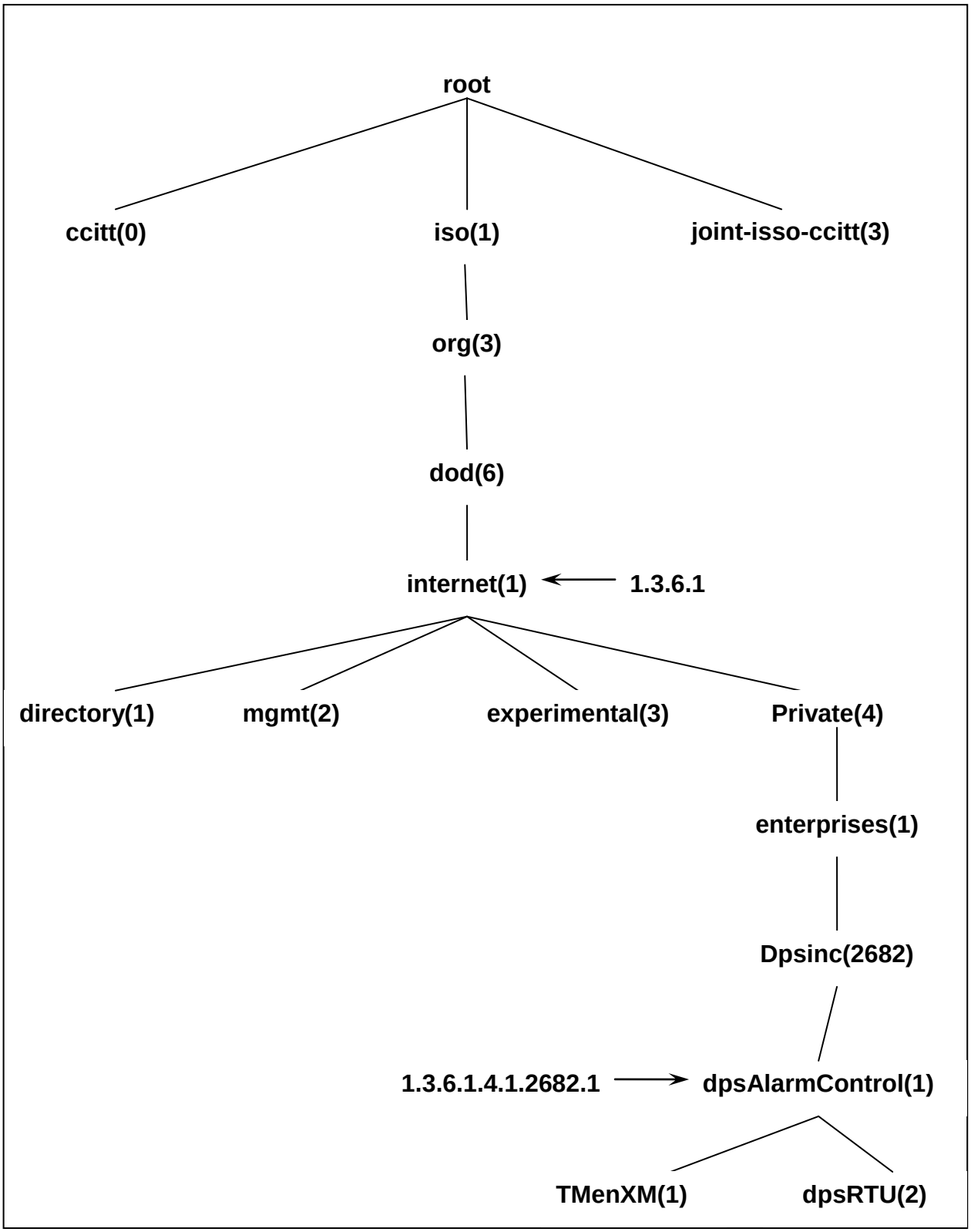
Gerentes - estações de gerenciamento (Network Management Station - NMS): disponibiliza a interface entre o administrador e o sistema de gerenciamento;

Objetos: os objetos gerenciáveis podem ser equipamentos, parâmetros de configuração ou ainda estatísticas de desempenho. Esses objetos são ordenados nas MIB.

Informações de gerenciamento - MIB

O gerente e o agente usam a MIB e alguns comandos para a troca de informações. A MIB é organizada em uma estrutura de árvore com variáveis individuais. Um identificador de objetos (*Object Indentifier* - OID) é usado para localizar cada variável dentro da MIB e nas mensagens SNMP. A MIB lista o identificador do objeto desejado em cada elemento gerenciado numa rede SNMP. O gerente SNMP só consegue monitorar dispositivos que ele já tenha registrado o arquivo de suas MIB. Cada elemento SNMP gerencia objetos específicos e cada objeto tem suas características específicas. Cada característica de um objeto tem o seu próprio identificador [5]. Como, por exemplo, o 1.3.6.1.4.1.2682.1 que pode ser localizado facilmente na árvore da figura 2, abaixo.

Figura 2. Estrutura de árvore da MIB [5]



A MIB associa cada OID a um rótulo e vários outros parâmetros ligados a um objeto. Quando o gerente SNMP quer saber o valor de uma determinada característica de um objeto, ele monta um pacote Get com o OID referente à característica desejada. O dispositivo recebe a requisição e procura o OID na MIB. Se o OID é localizado, o pacote de resposta é

montado e enviado com o valor da característica do objeto solicitado. Caso contrário, o pacote de resposta comunica o erro.

O gerenciamento de redes, também, pode utilizar equipamentos ou aplicativos conhecidos como sniffers para capturar e analisar o conteúdo do tráfego da rede. Cada segmento da rede a ser monitorado deve ter a presença de um desses dispositivos. Eles atuam de forma promíscua e podem obter dados completos sobre o tráfego que passa pelo segmento onde estão localizados. Com esse tipo de gerência, podem ser obtidas informações como o número de pacotes transmitidos, o tamanho dos pacotes, o tempo de resposta e o tráfego por protocolo.

Outro mecanismo bastante utilizado para estudos é simulação de redes virtuais. Para tanto, determinados programas permitem a criação de ambientes de teste para a simulação de uma infinidade de situações referentes ao tema redes de computadores. Nesses ambientes, são verificados o comportamento desde redes simples, redes complexas, equipamentos, circuitos de dados e até diferentes serviços que usam uma determinada rede. Essa prática facilita, por exemplo, o dimensionamento de uma nova rede ou o estudo da melhor forma de expandir uma rede já instalada. A principal característica desse recurso é o baixo custo para a criação quase ilimitada de diferentes cenários.

CAPÍTULO 3. REDES MPLS

As tecnologias de redes evoluíram para atender às necessidades dos novos serviços. Essa evolução busca, por um lado, a satisfação dos clientes e, por outro, a otimização dos recursos. Antigamente os SLA eram alcançados criando-se uma estrutura para cada tipo de serviço. “Provedores de serviço e empresas costumavam construir e manter redes separadas para transportar o seu tráfego de voz, vídeo, missão crítica e missão não-crítica” [10]. Com o desenvolvimento de novas tecnologias, o emprego do QoS permitiu o tratamento diferenciado de serviços em uma mesma rede. Isso aconteceu com as redes ATM, com IP e mais recentemente com a nova tecnologia MPLS.

3.1. Origem da Tecnologia MPLS

A tecnologia MPLS surge num momento que as redes estavam tomando conta da realidade de todos os ambientes. Os tipos de serviços que passaram a transitar pelas redes também estavam mudando muito. Nessa evolução das redes, surgiu a necessidade de tratar esses serviços diferenciadamente. Da mesma forma, tenta-se otimizar a utilização dos recursos e a redução de custos. A tecnologia MPLS tem características que a possibilita preencher uma lacuna deixada pelas redes IP e ATM principalmente. A RFC 2702, que cria a tecnologia MPLS, diz que o principal objetivo da engenharia de tráfego da internet é facilitar operações de rede eficientes e seguras enquanto, ao mesmo tempo otimiza a utilização dos recursos de rede e o desempenho do tráfego. Esses são os desafios para a tecnologia MPLS.

Multi Protocol Label Switching - MPLS - é uma tecnologia orientada para a conexão que utiliza a comutação por pacotes. Essa tecnologia surgiu na década de 90. “A *Ipsilon Corporation* foi uma das primeiras empresas a criar produtos que combinavam IP e *switches* de *hardware*; eles usaram *switches* ATM, chamaram sua tecnologia de comutação IP” [1]. Outras empresas, também, lançaram produtos semelhantes chamados, por exemplo, de comutação de *tags*, comutação *layer* 3 e comutação de rótulos. Em 1999 IETF unificou várias tecnologias que deu origem à MPLS [RFC 2702]. Essa tecnologia está situada entre as camada 2 e 3 do modelo OSI. Embora seja denominada *Multi Protocol*, praticamente, só trabalha com o IP.

O tratamento do tráfego na rede MPLS é baseado no rótulo que os pacotes IP recebem na entrada do domínio MPLS. Esse rótulo tem tamanho fixo com 32 *bits* divididos em: 20 *bits* Valor do rótulo; 3 *bits* EXP (Experimental). Alguns fabricantes usam para mapear o QoS do IP; 1 *bit* Final da pilha; e, 8 *bits Time To Live* - TTL. O valor do rótulo indica o seu comportamento até o próximo salto, onde esse rótulo é substituído por um novo que vale para o salto seguinte. Os roteadores MPLS não lêem o conteúdo IP do pacote, mas, somente, o rótulo MPLS.

A tecnologia MPLS, em muitos aspectos é semelhante ao modelo das redes *Asynchronous Transfer Mode* - ATM. Ambas são orientadas à conexão e oferecem funcionalidades de QoS. A tecnologia ATM, também, cria um circuito virtual (PVC) entre as duas pontas antes de se iniciar a troca de dados. Esses PVCs são incluídos "a partir de uma fonte de tráfego para um destino. Isso significa que você tem controle mais minucioso sobre o fluxo de tráfego na rede" [8]. O ATM trabalha na camada 2 e é utilizado para o transporte de IP e MPLS. Nesse sentido, as redes ATM provaram ser bastante úteis para conexões de longa distância e são bastante utilizadas no núcleo das redes de muitos provedores de telecomunicação.

Para Osborne e Simba [8] um dos grandes inconvenientes das redes ATM é que elas são implementadas formando-se uma malha completa de PVCs ATM entre um conjunto de roteadores. Nessa situação, a queda de um roteador ou de um enlace provoca problemas complexos de inundação capazes de derrubar toda a rede. Já as redes MPLS não necessitam da formam a malha completa. Evitam essa situação tratando o roteamento a cada salto. Com isso, as redes MPLS se adéquam melhor a situações complexas com grandes quantidades de nós.

Um dos aspectos que tornam a tecnologia MPLS bastante robusta é sua capacidade de complementar a tecnologia IP. Sem dúvida alguma, a tecnologia IP é uma das principais responsáveis pela evolução que as redes chegaram até hoje. No início, porém, as possibilidades do IP eram limitadas, a única garantia era a entrega do pacote no seu destino. Nessa época, o QoS não era utilizado e os pacotes de qualquer serviço recebiam o mesmo tratamento. Posteriormente, a tecnologia IP evoluiu, passou a utilizar o QoS permitindo a classificação do tráfego inicialmente em 8, e depois em 64 categorias [8]. O QoS do pacote IP é reconhecido e utilizado pelo roteador de borda *Label Edge Router* – LER para classificar o tráfego a ser introduzido na rede MPLS.

Para classificar um pacote, o LER verifica o conteúdo do cabeçalho da camada de rede e, normalmente baseado no endereço de destino, vincula o pacote a uma *Forwarding*

Equivalence Class (FEC). Para fazer esse vínculo, o LER usa uma lista de acesso pré-existente em sua configuração. Além do endereço de destino, o LER pode considerar tanto o conteúdo dos três *bits* do IP *Precedence*, quanto os seis do *Diffserv Code Point* - DSCP no mapa de classes do MPLS. Para realizar essa tarefa, o vínculo é feito a partir da porta de destino do pacote.

Apesar da evolução do QoS, as redes IP continuam limitadas em relação à escolha do caminho a ser trilhado pelos pacotes. Conforme Sales e Rolla [9], as redes IP têm limitações por serem orientadas ao destino, a origem do tráfego não controla a seleção do caminho. "O algoritmo baseado no menor custo seleciona os caminhos mais curtos para preencher a tabela de roteamento. Caso os caminhos mais curtos estejam sobrecarregados, outros caminhos de maior comprimento não serão utilizados mesmo se estiverem subutilizados" [9]. Apesar das redes IP dominarem completamente o cenário de hoje, esse problema de encaminhamento não é fácil de ser solucionado no mundo IP. Esse problema pode ser mostrado melhor com a utilização da figura 3. Caso dois fluxos cheguem ao mesmo tempo nos roteadores R8 e R1 com destino ao R5 e o cálculo indicar o R2, R3 e R4 como o melhor caminho, os dois fluxos seguirão por esse caminho, que poderá ficar sobrecarregado, enquanto o caminho R2, R6, R7 e R4 ficará subutilizado.

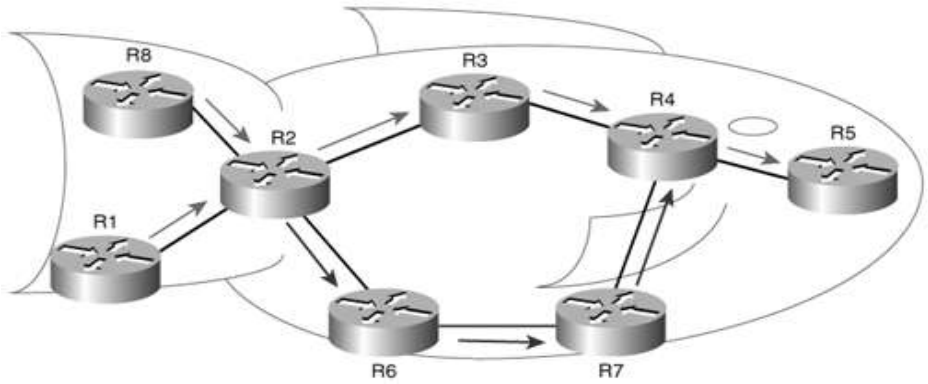


Figura 3. Problema peixe [11]

A engenharia de tráfego do MPLS resolve a limitação do menor caminho utilizada pelo IP. "MPLS [RFC 3031] possibilita novo paradigma de encaminhamento em relação aos convencionais IPv4 e IPv6, permitindo o encaminhamento baseado em critério ao invés do endereço IP de destino" [2]. Assim, como no IP, antes de ser iniciada a transferência de pacotes, é feito um cálculo do melhor caminho. Porém, diferentemente do IP, é feita uma verificação para saber se o caminho escolhido tem capacidade para suportar o envio de novo fluxo. Caso tenha, a largura de banda necessária é reservada para esse novo fluxo, que só

então é enviado. Essa é a grande diferença para o encaminhamento IP. Dessa forma, a sobrecarga do menor caminho é evitada. Conforme a figura 4, em uma rede MPLS, quando um fluxo chega ao roteador A endereçado ao G, o roteador A envia uma mensagem, pelo menor caminho, solicitando aos roteadores desse caminho a reserva da banda necessária para o fluxo. Os roteadores, então, fazem a reserva da banda e informam o valor do rótulo a ser utilizado naquele trecho. Quando a transmissão começa, o roteador lê apenas o rótulo MPLS e troca o rótulo pelo valor que indica o próximo salto. Caso outro fluxo solicite o mesmo caminho e haja banda restante suficiente ele também utilizará o mesmo caminho. Porém, caso a banda restante não seja suficiente para os dois fluxos, será montado um outro caminho para o segundo fluxo.

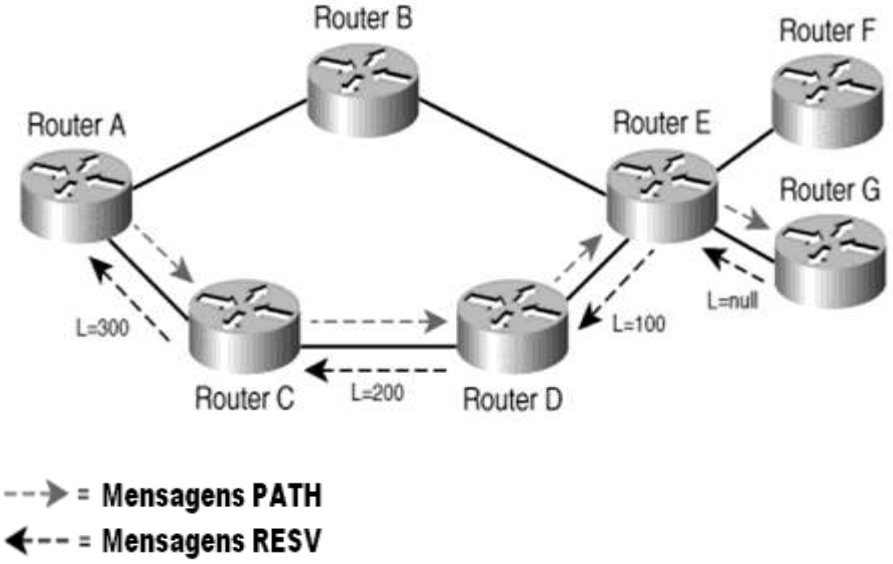


Figura 4. Formação do LSP [11]

Esse caminho, reservado pelo *Resource Reservation Protocol* – RSVP, recebe o nome de *Label Switched Path* – LSP, no MPLS. Se esse caminho for de alta qualidade, a engenharia de tráfego pode vincular a ele uma *Forwarding Equivalence Class* - FEC de alta prioridade. A FEC é uma classe de serviço que corresponde a um conjunto de fluxos com as mesmas características. Essa é outra vantagem do MPLS, o QoS não é baseado somente em políticas de filas mas, também, na qualidade das conexões.

A tecnologia MPLS trás novas possibilidades de tratamento dos diferentes tipos de tráfego de rede. Essa tecnologia, "combina as capacidades de engenharia de tráfego do ATM com a flexibilidade e a diferenciação de classe do serviço IP" [8]. Essa tecnologia trabalha montando caminhos na rede por onde o tráfego será encaminhado. Esses caminhos,

ou *Label Switched Path* **(**LSP), também, podem ser chamados de túneis. Diferentemente do IP, o MPLS permite que a ponta inicial do túnel controle o caminho pelo qual o tráfego passará até o destino final. Isso trás mais possibilidades de encaminhamento de tráfego que o encaminhamento do IP, baseado apenas no destino.

O MPLS, também, permite a reserva de largura de banda quando monta um LSP. Essas "reservas de banda são feitas no plano de controle, o que significa que, se um roteador faz uma reserva para 10Mb e envia 100Mb por esse LSP, a rede tenta oferecer esses 100" [8]. Uma rede MPLS pode, por exemplo, ser configurada para reservar um percentual da largura da banda a uma determinada classe de serviço crítica para uma empresa garantindo assim, a priorização desse tráfego na rede. Essas funções tornam a tecnologia MPLS muito útil para empresas que queiram dar garantias de desempenho dos seus serviços em uma rede. Com essas características, a tecnologia MPLS tende a se tornar bastante poderosa em termos de engenharia de tráfego. Porém, para melhor empregá-la, é necessário que se conheça melhor as características do tráfego das redes.

CAPÍTULO 4. METODOLOGIA PARA ANÁLISE DE REDES MPLS

A medição do tráfego é fundamental para se conhecer o que está acontecendo em uma rede. Sem esse conhecimento não é possível melhorá-la. “Conhecendo o comportamento do tráfego é possível projetar equipamentos melhores, implementar mecanismos de qualidade de serviço e de tarifação” [7]. Vários mecanismos e técnicas foram desenvolvidos para atender à essa necessidade de estudo do tráfego das redes de computadores. O gerenciamento eficiente necessita de uma metodologia para empregar as técnicas adequadas que levem em consideração o objetivo da análise e as características da rede.

Vários estudos são realizados com o objetivo de se aprimorar a forma como são analisadas as redes. Em um desses estudos, IENO JÚNIOR [4] em sua dissertação de mestrado, propõe uma metodologia para a análise de desempenho de redes sem fio do padrão IEEE 802.11. Essa metodologia combina a gerência SNMP com ferramentas de simulação e tem como objetivo melhorar o desempenho e o planejamento da capacidade desse tipo de rede. Essa proposição é composta de quatro etapas onde na primeira é feita a escolha e a dos objetos. Na segunda etapa, são eliminados os dados espúrios e são obtidos os indicadores. Na etapa seguinte, o administrador analisa os dados obtidos. Por último, a solução do possível problema é testada em uma ferramenta de simulação.

Outra metodologia analisada foi a proposta por GIMENEZ [3] para um estudo pragmático visando a monitoração, o controle e o planejamento da capacidade de redes corporativas de computadores. Na primeira fase dessa metodologia é feito um estudo das condições iniciais onde são estabelecidos os padrões de desempenho e a qualidade dos serviços desejados. Na fase seguinte, é feita a coleta e a comparação dos dados com os padrões estabelecidos. Na terceira fase, os possíveis motivos provocadores de problemas são analisados. A simulação é a última fase dessa metodologia e só é executada se forem verificados problemas na fase anterior.

4.1. Passo a passo para a análise do desempenho de redes MPLS

À medida que o uso de redes MPLS vem se ampliando - muitas empresas já contratam esse tipo de tecnologia - aparece a necessidade do conhecimento de seu desempenho. Normalmente, essas redes são grandes e complexas o que aumenta a dificuldade para se obter dados que possibilitem a sua otimização. As redes MPLS possuem, também, muitos recursos de engenharia de tráfego, que podem melhorar a utilização dos recursos instalados. Diante dessa realidade, apresento uma proposta de metodologia de análise para facilitar o entendimento do que está ocorrendo nesse tipo de rede.

4.2. Passo 1 - Definição do objetivo

Algumas definições devem ser feitas antes de ser iniciada a análise de desempenho propriamente dita. Nesse sentido, o primeiro passo é deixar claro qual o objetivo da análise. Por exemplo, se uma empresa precisar saber se seus clientes estão bem atendidos a análise deve ser voltada para o desempenho dos aplicativos. Por outro lado, a direção da empresa pode querer saber como está a relação custo/benefício dessa rede. Já o administrador da rede deve precisar de uma análise mais completa de todos os seus elementos com o objetivo de prevenir falhas e aumentar sua eficiência. Para atingir melhores resultados e verificar a tendência de evolução da rede, pode-se quer saber que tipo de tráfego está circulando e qual a sua distribuição na rede. Empresas que terceirizam esse serviço, normalmente, fazem essa contratação baseada em SLA e, naturalmente, precisam saber se o contrato está sendo cumprido. Neste caso, o objetivo poderia ser uma auditoria. Um outro tipo de análise é voltado para o dimensionamento de novas redes ou de expansão de redes existentes.

4.3. Passo 2 - Identificação do cenário

Nesse segundo passo, é importante que seja identificado a dimensão e a complexidade da rede que será estudada. Esse diagnóstico é importante porque vai influenciar na arquitetura e no padrão do sistema que vai ser montado para essa análise. Dificilmente uma rede MPLS é pequena, mas, se for a análise pode ser feita em um sistema centralizado. A variedade de aplicativos, serviços e equipamentos, também, precisa ser conhecida. Já, para verificar se as prioridades dos serviços estão sendo atendidas, as necessidades de cada serviço e as configurações de QoS, também, precisarão ser levantadas.

4.4. Passo 3 - Definição dos dados a serem coletados

Nesse momento, levando-se em consideração os objetivos da medição e o conhecimento da rede, devem ser definidos que dados são necessários para a efetivação da análise. Por exemplo, para verificar se o cliente da empresa está bem atendido, dados como a disponibilidade e o tempo de resposta são primordiais. Já o administrador, precisa saber, por exemplo, como está a taxa de utilização de um ou de todos os seguimentos da rede. Por outro lado, para a verificação do SLA pode bastar a conferência da largura de banda disponível.

Outra questão a ser definida nesse terceiro passo, é a periodicidade da análise. Essa monitoração poderá ser pontual, freqüente ou permanente, de acordo com o objetivo da pesquisa. A freqüência da coleta dos dados vai depender do interesse da pesquisa. Coletas mais freqüentes são usadas para verificar as demandas de tráfego em um período; ao passo que, as mais longas servem para observar tendências do tráfego. No caso de uma auditoria, por exemplo, a medição poderia ser feita pontualmente ou por um período curto. Porém, do ponto de vista de segurança, a rede não pode ficar sem monitoração em tempo algum.

4.5. Passo 4 - Identificação das fontes dos dados

Nesse passo, são definidos os equipamentos ou aplicativos que vão fornecer os dados necessários para o estudo. Caso a rede não disponha de todas as facilidades necessárias para o atendimento desse requisito, este é um bom momento para rever os objetivos ou prever a aquisição de novos equipamentos ou programas para atender à demanda original. Da mesma forma, a incompatibilidade dos equipamentos existentes com um sistema único de monitoração pode levar à substituição dos equipamentos ou à busca por ferramentas que compatibilizem essa utilização.

No caso específico da tecnologia MPLS, o IETF, desenvolveu em seu *draft-ietf-mpls-te-mib-05.txt*, o que se tornou um padrão da para *MultiProtocol Label Switching Traffic Engineering* MIB (MPLS TE MIB) editado na RFC 3812. A empresa Cisco já adota esse padrão que é compatível com o padrão SNMP. Dessa forma, todos os objetos MPLS TE MIB podem ser acessados pelos agentes SNMP. Os principais benefícios da MPLS TE MIB são a disponibilização de: interface SNMP para a coleta de informações sobre a engenharia de tráfego MPLS; informações sobre os fluxos de tráfego nos túneis MPLS; as rotas definidas para os túneis; informações de como os túneis foram roteados no caso da perda de um link e informações sobre a configuração dos recursos usados em um túnel MPLS [5].

O MPLS TE MIB contem uma grande quantidade de definições de objetos que podem ser lidos pelo gerenciamento SNMP [5]. Algumas das mais importantes tabelas MIB estão listada a seguir:

- mplsTunnelConfigured - Apresenta a quantidade de configurações de túneis definidos neste nó;
- mplsTunnelMaxHops - A quantidade máxima de saltos que um túnel deva utilizar;
- mplsTunnelTable - É uma da principais tabelas, entre outras informações, indica o estado dos túneis;
- mplsTunnelEgressLSRId - Endereço IP de destino do túnel;
- mplsTunnelSetupPrio - Indica as prioridades definidas para o túnel;
- mplsTunnelOperStatus - Indica o estado atual de operação do túnel;
- mplsTunnelResourceMaxRate - Indica a taxa máxima de transmissão
- mplsTunnelResourceMeanRate - Indica a taxa principal de transmissão;
- mplsTunnelResourceMaxBurstSize - Indica o tamanho da rajada;

- mplsTunnelResourceRowStatus - Indica a situação da fila;
- mplsTunnelPerfTable - É a tabela de desempenho do túnel a partir de onde são obtidas a informações da contagem de pacotes e bytes por túnel;
- mplsTunnelPerfPackets - Contador de pacotes;
- mplsTunnelPerfErrors - Contador de erros;
- mplsTunnelPerfBytes - Contador de bytes.

Outro aspecto a ser definido, ainda no passo quatro é se a monitoração da rede vai ser de forma passiva ou ativa. No modo passivo, um elemento da rede é escolhido para ser monitorado. Já na medição ativa, pacotes são enviados com o objetivo específico de verificação do desempenho da rede [2].

A medição passiva, normalmente, é utilizada para verificação de dados estatísticos de QoS de um determinado ponto da rede. Na medição passiva, é feita uma coleta dos dados armazenados no arquivo do *managment information bases* (MIB), de um roteador específico. Na coleta passiva, podem-se obter dados de quantidade de pacotes e de bytes ou tamanho de filas. Essas informações podem trazer quantitativos sobre os fluxos que compõem uma classe, sobre as próprias classes, sobre enfileiramento e descarte de pacotes. As informações obtidas sobre a situação dos circuitos podem ser utilizadas para a identificação de falhas, para a readequação da rede, para o melhoramento das métricas do QoS ou para aplicação de multas previstas no SLA [2].

Diferentemente da passiva, a medição ativa utiliza a inserção de pacotes de teste para obter informações da rede. Esses pacotes são criados exclusivamente para esse fim. É importante ressaltar que os resultados obtidos nesse tipo de teste estão vinculados diretamente ao tipo de pacotes que serão enviados. Portanto, eles devem ter as mesmas características dos aplicativos ou das classes que se quer testar. Por exemplo, o tamanho dos pacotes deve estar de acordo com os pacotes do tráfego normal para evitar diferentes informações sobre o retardo ou sobre perda.

Outra característica importante é a classificação desses pacotes. Nesse sentido, devem ser identificados e classificados da mesma forma que o tráfego original em termos de classes de serviço. Somente assim, alcançarão comportamento e desempenho semelhantes ao tráfego normal da rede. Nesse tipo de medição é possível a obtenção de dados como retardo, variação do retardo, perda de pacotes, largura de banda ou disponibilidade da rede. Pode-se observar, também, o comportamento desse tráfego em relação a cada classe de serviço [2].

Dessa forma, é possível verificar se as prioridades estabelecidas no SLA e no QoS estão efetivamente sendo observadas. Por exemplo, o tempo de resposta pode indicar se a largura da banda está de acordo com o contratado; ou ainda, se os serviços prioritários tiveram menos tempo de resposta que os demais.

4.6. Passo 5 - Definição do sistema a ser utilizado

No momento da escolha do sistema a ser adotado é importante estar atento à confiabilidade, compatibilidade e escalabilidade do padrão proposto. O quesito confiabilidade tem de ser cumprido, principalmente, porque o resultado da análise pode levar à adoção de medidas erradas e a prejuízos. A questão da compatibilidade pode ser resolvida com a adoção de padrões amplamente utilizados como é o caso do SNMP. Já a escalabilidade deve sempre ser prevista, pois, as redes estão em constante crescimento e o sistema deve estar preparado para se adequar a essas ampliações. Nesse sentido, a adoção de sistemas centralizados em redes de grande porte pode indicar um alto grau de risco.

4.7. Passo 6 - Definição de ferramenta de gerenciamento

Para organizar a coleta dos dados e viabilizar a análise das informações obtidas, o administrador da rede necessitará de uma ferramenta. Essa ferramenta deverá permitir a visualização de uma quantidade grande de informações apresentadas em forma de sumários ou gráficos. A definição da ferramenta ou ferramentas depende diretamente de todos os passos anteriores. Existe no mercado uma boa quantidade desses aplicativos que são compatíveis com a tecnologia MPLS. Como, por exemplo: HP Open View; Cisco View; Ethereal; Ntop e Tivoli dentre outras. Existe, também, a possibilidade da adoção de ferramentas proprietárias desenvolvidas internamente na empresa para esse fim.

O gerenciamento de uma rede, também, pode ser feito com a integração de várias ferramentas. Um exemplo típico dessa integração é utilizado para a monitoração da rede MPLS do Banco do Brasil. Aproximadamente a metade dos quinze mil pontos dessa rede é fornecida pela Embratel enquanto a outra metade é provida pela Oi. O gerenciamento da parte da Embratel utiliza a ferramenta Gerencia de Redes de Clientes (GRC), desenvolvida pela própria Embratel. Essa ferramenta usa o protocolo SNMP para buscar dados, atualizados a cada cinco minutos, das interfaces dessa parte da rede. Nessa ferramenta, o administrador

visualiza informações como a disponibilidade dos circuitos, a utilização da memória dos roteadores, tempo de resposta, perda de pacotes, taxa de utilização de banda entre outros. A figura 5 mostra a monitoração, pela ferramenta GRC, do tráfego de saída e de entrada em um circuito de 34 M/bits dessa rede no momento em que um dos circuitos perdeu a conexão.

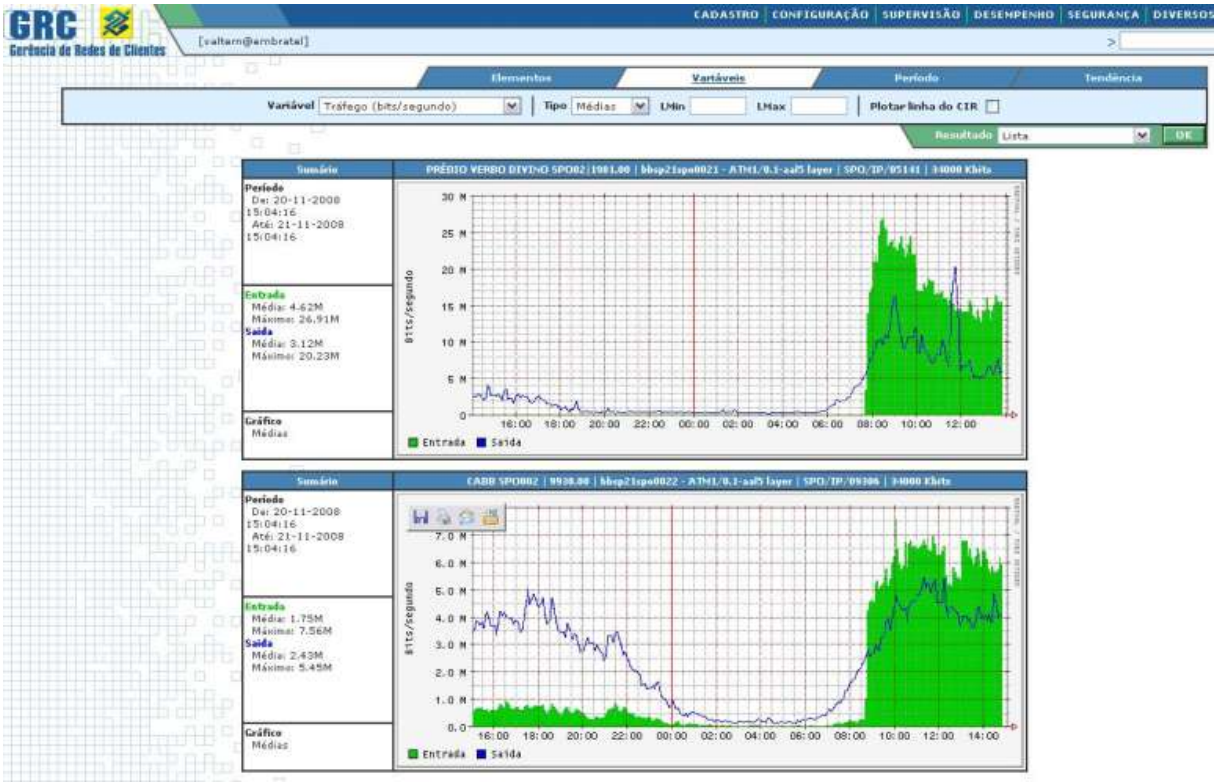


Figura 5. Ferramenta GRC

Já a operadora de telecomunicações Oi, utiliza a ferramenta HP Open View, também baseada no protocolo SNMP, para a coleta de informações semelhantes. Por outro lado, o Banco do Brasil utiliza uma ferramenta própria, o Sistema de Monitoração dos Servidores de Agências (SSA), para a monitoração de 46 (quarenta e seis) eventos de servidores e roteadores das agências e a HP Open View para a monitoração dos ambientes de escritório. Todas essas ferramentas se integram para a abertura automática de bilhetes de incidentes no aplicativo Remedy Action Request System (ARS). Tanto as operadoras como o Banco do Brasil utilizam os resultados obtidos com essas ferramentas para, periodicamente, confrontá-los com as métricas estabelecidas no SLA.

Uma outra forma de estudo do desempenho e capacidade de redes é a utilização de simulação. Foram verificados vários exemplos de utilização de simuladores

Linux como um experimento realizado na Universidade de Virginia, nos Estados Unidos ou, outro realizado pela Universidade de Santa Catarina. Já o Departamento de Engenharia Elétrica do Instituto Indiano de Tecnologia, criou e disponibiliza um simulador chamado *Linux based MPLS Emulator* (LiME). Existem diversas ferramentas simuladoras, compatíveis com o MPLS, disponíveis no mercado como o Network Simulator 2 (NS2) de código aberto, o OPNET Modeler que é uma ferramenta comercial. Já o *Linux based MPLS Emulator* (LiME) e o OpenSimMPLS são ferramentas abertas desenvolvidas especificamente para ambientes MPLS

.

4.8. Passo 7 - Análise dos dados

As ferramentas apontadas no passo 6 podem fornecer ao administrador gráficos e estatísticas que vão subsidiar a decisão sobre as atitudes que precisam ser tomadas. Essas providências podem ser imediatas, no caso do diagnóstico de falhas. Por outro lado, a análise pode indicar medidas de médio ou longo prazo se a solução for uma futura ampliação da rede. Em todos os casos, a análise é necessária para se obter o melhor que os recursos de uma rede podem oferecer.

CAPÍTULO 5. ESTUDO DE CASO

5.1. Passo 1 – Definição do objetivo

Será feita a medição de um protótipo de rede MPLS para verificar se o seu desempenho, em termos de taxa de transmissão, está de acordo com a velocidade do circuito. Nesse sentido, serão testados circuitos na velocidade de 8Kbytes/s e 64Kbytes/s. Também, será verificado se o QoS está sendo cumprido. Para tanto, serão configuradas quatro classes de serviços com prioridades distintas. Nesse caso, o tempo de resposta obtido por cada uma dessas classes indicará se as classes de maior importância tiveram vantagens sobre as demais.

5.2. Passo 2 – Identificação do cenário

Será montado um protótipo de uma rede MPLS com configurações semelhantes às de uma rede real existente, que inicialmente motivou este trabalho. Nessa rede, quatro classes de serviço serão configuradas para testar o QoS. O desempenho será testado em circuitos com duas velocidades diferentes.

A parte física da rede será constituída por dois roteadores de ponta e um roteador central que simulará as funções MPLS, no núcleo da rede. Serão utilizados, também, quatro *modens* nos circuitos que farão a interligação entre os roteadores. Esses circuitos terão velocidades diferentes, possibilitando a análise da resposta à alteração dessa variável.

Para caracterizar o QoS MPLS, os roteadores serão configurados para receber o tráfego IP, classificar em uma das quatro classes de serviço, conforme informação do cabeçalho IP, e encaminhar de acordo com as prioridades de cada classe. Essas quatro classes foram escolhidas de forma a representar serviços típicos existentes em uma rede, conforme abaixo:

- Classe 1 - Missão crítica. Essa classe vai representar os fluxos dos serviços críticos e deverá ser tratada pela rede com o maior nível de prioridade;
- Classe 2 - Gerenciamento. Essa classe representará os fluxos de gerenciamento da rede e deverá receber da rede, um tratamento de prioridade intermediária;
- Classe 3 - Voz. Para essa classe a rede deverá garantir as condições para que o serviço de voz funcione adequadamente. Ou seja, capacidade mínima de banda constante e baixos índices de retardo; e,

- Classe 4 - Padrão. Essa classe representará os demais serviços que trafegarão na rede e não serão tratados com qualquer tipo de prioridade.

Um computador será conectado a um dos roteadores de ponta da rede para servir como cliente. Nesse computador ficarão armazenados os arquivos de 1 e 10 *MBytes*. Outro computador atuará como servidor e será conectado ao roteador da outra ponta. Nesse segundo computador, será instalado o aplicativo que fará o cálculo do desempenho da rede e um banco de dados para armazenar os dados obtidos.

A terceira variável a ser utilizada para a análise da rede é o tamanho dos arquivos que serão remetidos para teste. Serão utilizados arquivos de dois tamanhos, o primeiro com 1 *MBytes* e o segundo com 10 *MBytes*. Esses arquivos terão dados que representarão vários tipos de serviço. Os arquivos ficarão armazenados no computador do cliente e durante o teste serão transferidos para o servidor, passando pela rede MPLS. A topologia da rede foi construída conforme a figura 6.

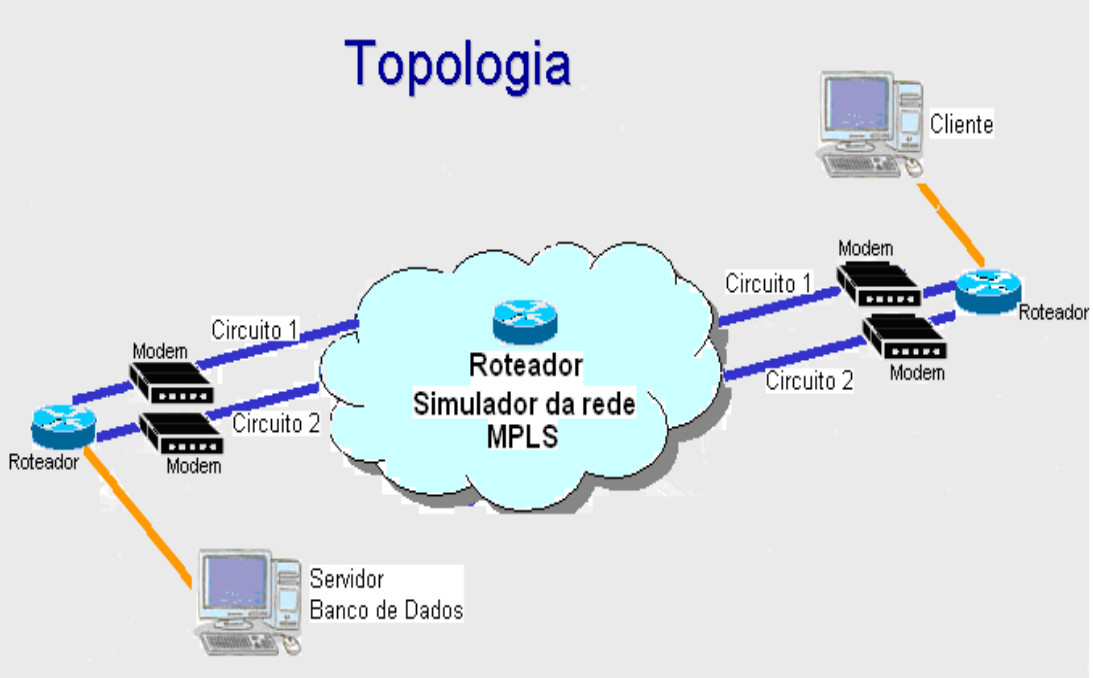


Figura 06. Topologia

O protótipo para a simulação da rede MPLS foi montado com um roteador Cisco modelo 1751, dois roteadores Cisco 1721. Esses roteadores foram conectados em um lado por um circuito com *modens* Digitel modelo MD70. No outro lado, a ligação entre o roteador de ponta o roteador de núcleo foi realizada com um cabo direto com o objetivo de evitar custos desnecessários de instalação e operação dos *modens*.

O roteador 1751 foi configurado para simular, nesta rede, a função de núcleo de rede MPLS. Esse roteador possui 32*Kbytes* de memória NVRAM e 32*Mbytes* de memória flash (leitura e escrita). Os primeiros roteadores que seriam utilizados neste experimento foram descartados porque não possuíam memória suficiente para suportar um sistema operacional compatível com a tecnologia MPLS. Esse roteador possui duas interfaces seriais (sync/async) onde estão conectados os circuitos que fazem a ligação com os outros dois roteadores.

Na configuração desta rede foram criadas quatro classes de serviço. Uma chamada TEMPO_REAL_VOZ que tem prioridade total sobre as demais. Uma segunda classe denominada MISSAO_CRITICA. A terceira recebeu o nome de GERENCIAMENTO. O tráfego que não se enquadra em uma dessas três classes não recebe prioridades na rede e constitui a quarta classe ou classe PADRÃO. Os roteadores das pontas vão enviar os pacotes já rotulados com MPLS para esse roteador do núcleo. Esse roteador vai ler o rótulo MPLS, vai conferir no mapa a qual classe o pacote pertence e o encaminhará de acordo com as informações do rótulo. O trecho de código exposto no Quadro 01 é o mapa de classes que está gravado nos três roteadores.

Quadro 01. Mapa das Classes

```
class-map match-any MISSAO_CRITICA
match ip dscp cs4  af42  af43
class-map match-all TEMPO_REAL_VOZ
match ip dscp cs5  ef
match ip dscp ef
class-map match-all TEMPO_REAL_VC
match ip dscp af41
class-map match-any GERENCIAMENTO
match ip dscp cs3  af31  af32
match ip dscp cs1  af11  af13
```

Uma das características da tecnologia MPLS é a reserva de largura de banda para atingir o QoS. O trecho de código exposto no Quadro 02 contém a política de reserva de banda para cada classe de serviço nesta rede. Essa configuração está presente nos três roteadores.

Quadro 02. Política de Reserva: configuração de roteadores

```
policy-map QPO_64_15_0_25_8_0_0_80B_V1
class TEMPO_REAL_VOZ
priority 15
class MISSAO_CRITICA
bandwidth percent 39
```

```
▪ random-detect dscp-based
▪ class GERENCIAMENTO
▪ bandwidth percent 13
▪ random-detect dscp-based
▪ class class-default
▪ fair-queue
```

Na política adotada, o serviço de voz tem prioridade total sobre todos os outros. Assim, se for necessário, a classe de voz vai usar toda a capacidade de banda passante, independentemente dos outros serviços. No caso de congestionamento, excluindo voz, a classe missão crítica tem 39% de banda garantida, a classe de gerenciamento tem garantia de 13% e a classe padrão entra na fila sem privilégios.

Os testes foram realizados utilizando as taxas de transmissão de 64Kbps (64000 *bits* por segundo) e 512Kbps (512000 *bits* por segundo). Essas configurações foram implementadas nos *modens* e nos roteadores. Nos roteadores, todas as seriais foram atualizadas com o mesmo valor. O código exposto no Quadro 03 contém as configurações das seriais do roteador central.

Quadro 03. Configurações das Seriais do Roteador Central

```
interface Serial0/0
 • description TESTE | 512K | ENLACE FR CLIENTE
 • bandwidth 512
 •  no ip address
 • encapsulation frame-relay IETF!

interface Serial0/0.1 point-to-point
 • description TESTE
 • bandwidth 512
 • ip vrf forwarding TESTE
 • ip address 10.1.1.1 255.255.255.252
 • no ip proxy-arp
```

Os roteadores de ponta têm, entre si, basicamente a mesma configuração. A única diferença entre eles é o no endereçamento de rede. Nesses roteadores, fica a lista de acesso, que é a relação das portas dos serviços que serão vinculados a uma determinada classe. Essa classificação é vinculada ao mapa de classes presente nos três roteadores. O roteador do meio da rede não usa a lista de acesso, usa apenas o mapa para identificar o tráfego. Assim sendo, a lista de acesso não consta da configuração do roteador de meio. O código exposto no Quadro 04 é parte das listas de acesso das três classes, com prioridade, utilizadas nesse experimento. A classe padrão não recebe classificação. Portando, todo o tráfego fora dessa listas é considerado padrão.

Quadro 04. Código de Acesso para Três Classes

```
ip access-list extended GERENCIAMENTO_ACL
    ▪ permit tcp any any eq 62
    ▪ permit tcp any eq 62 any
```

- permit tcp any any eq 1862
- permit tcp any eq 1862 any
- permit udp any any eq 1925
- permit udp any eq 1925 any

ip access-list extended MISSAO_CRITICA_ACL

- permit tcp any any range 19111 19153
- permit tcp any range 19111 19153 any
- permit tcp any any eq 117
- permit tcp any eq 117 any
- permit udp any any eq 56781
- permit udp any eq 56781 any

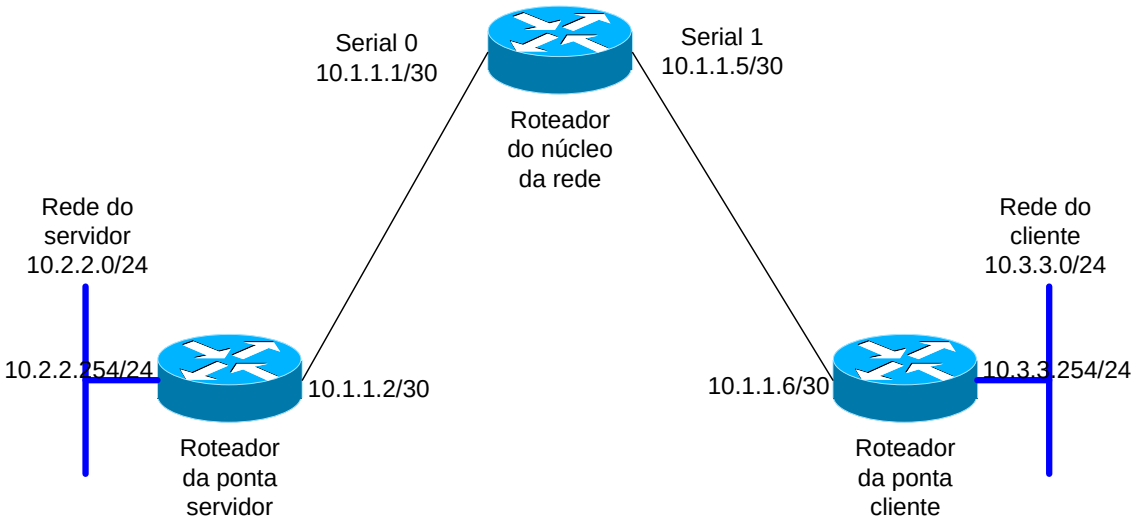
ip access-list extended TEMPO_REAL_VOZ_ACL

- permit tcp any any eq 120
- permit tcp any eq 120 any
- permit udp any range 1133 1177 any range 1133 1177

Os roteadores e *modens*, com as configurações, utilizados no funcionamento deste protótipo de rede MPLS, foram emprestados pela empresa Embratel. Para efeito dos testes o autor deste trabalho fez somente as alterações nos itens que envolvem taxa de transmissão. Mais especificamente, na largura da banda e no *clockrate* dos roteadores e na configuração da quantidade de canais nos modens visando à alteração da velocidade do circuito.

A Figura 07 mostra a topologia da rede de testes. Nessa rede foi incluído um computador que foi utilizado com servidor que recebeu o endereço 10.2.2.2. E, na outra ponta o computador cliente foi configurado com o endereço 10.3.3.3.

Figura 07. Topologia de Rede de Testes



5.3. Passo 3 – Definição dos dados a serem coletados

Será coletado o tempo de resposta de pacotes enviados entre uma estação cliente e um servidor. Serão verificados os dados obtidos com o tráfego de pacotes de tamanhos diferentes e em circuitos de velocidades distintas. O tráfego será classificado em quatro classes de serviços distintas e será averiguado o tempo de resposta para cada uma dessas classes. Essas classes serão reconhecidas através das portas utilizadas.

Os pacotes de teste constituirão o único tráfego existente nessa rede. Portanto, a periodicidade dos testes será determinada pela necessidade repetição de cada situação até ser obtido um padrão de tempo de resposta para cada configuração.

5.4. Passo 4 – Identificação das fontes dos dados

Este trabalho utilizará o mecanismo da medição ativa para analisar o desempenho de um protótipo de rede MPLS com o objetivo de verificar se essa rede está de acordo com o especificado. Neste caso, os arquivos que serão forçados a trafegar na rede terão duas funções. Como nessa rede, originalmente, não há tráfego esses pacotes servirão, primeiramente, para gerar tráfego. Por tanto, deverão ser grandes o suficiente para cumprirem essa missão. Segundo, funcionarão como objetos para a coleta de dados do desempenho dessa rede. Para os testes, será forçada a circulação de pacotes específicos preexistentes, armazenados em arquivos da estação cliente.

5.5. Passo 5 – Definição do sistema a ser utilizado

Como o objetivo é apenas a verificação do tempo de resposta alcançado pelos pacotes nesse protótipo, o sistema desenvolvido é bastante simples. Nesse sistema, será feito apenas a cronometragem do tempo que o pacote leva para se deslocar da estação cliente até o servidor. Devido às características dessa medição, não será utilizado nenhum sistema do tipo SNMP para a requisição de informações dos roteadores do protótipo. A própria ferramenta desenvolvida para esse teste fará o envio dos arquivos e a contagem do tempo de resposta para depois disponibilizar as informações obtidas em uma *interface* gráfica da estação do gerente.

5.6. Passo 6 – Definição de ferramenta

Inicialmente foi pensada a possibilidade de utilização de simuladores para a realização de testes em rede do tipo MPLS. Porém, posteriormente foi feita a opção pela construção de um protótipo de rede. Dessa forma, a opção foi criar uma ferramenta simples que atendesse o objetivo específico de medir o tráfego desse protótipo de rede MPLS.

A ferramenta criada para medir o desempenho de rede envia um arquivo do computador cliente para o servidor, calcula a taxa de transferência, registra esses dados em um banco e mostra um relatório com as informações obtidas para cada classe de serviço. Esse aplicativo foi desenvolvido na linguagem C ++, a página de apresentação é um HTML tratado pelo servidor apache e o banco de dados é MySQL.

O aplicativo transfere um arquivo para o servidor a partir do cliente. Para tanto, é informado à porta a ser enviada, o endereço IP do servidor e a localização do arquivo. No servidor deve ser aberta uma seção com a porta que receberá o arquivo. As Figuras 08, 09 e 10 mostram as telas do cliente e do servidor durante a transferência de um arquivo.

Figura 08. Modo Cliente no início da conexão

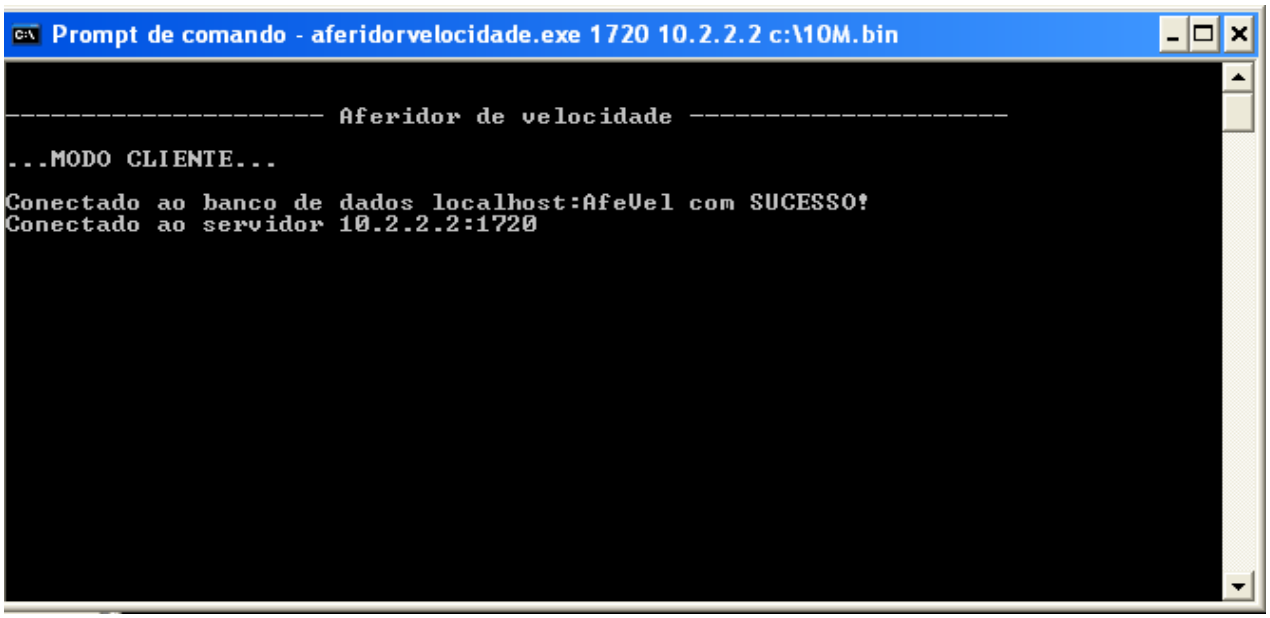


Figura 09. Modo Servidor no início da conexão

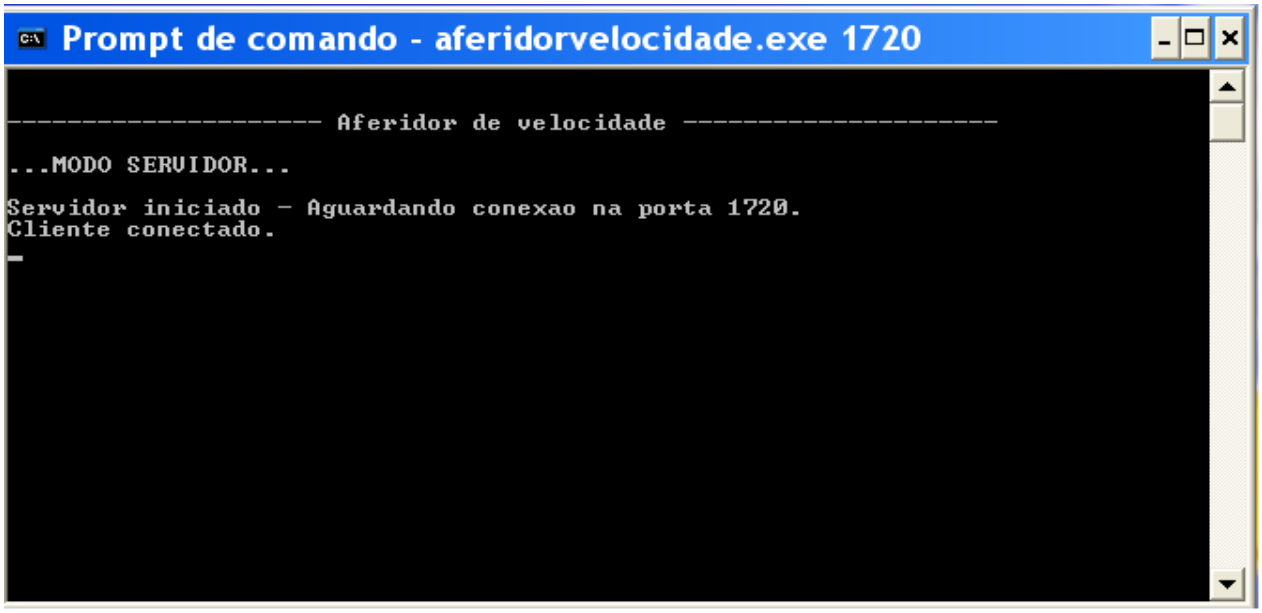
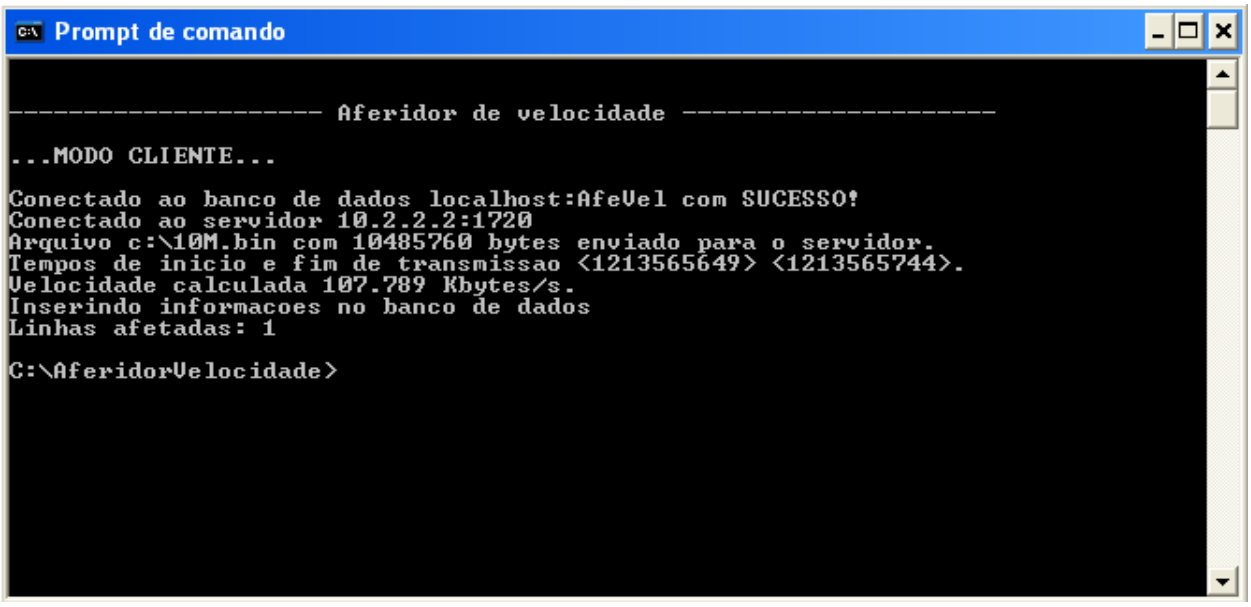


Figura 10. Modo cliente após a conclusão da transferência



Antes da transmissão da mensagem, a ferramenta identifica a porta e determina a classe. Para os testes dessa rede, a classe 1 corresponde à MISSAO_CRITICA e porta 137 está na lista de acesso dessa classe nos roteadores de ponta. Já a porta 1000, também, representa a classe 1, porém, não consta na lista dos roteadores. Dessa forma, o arquivo que for transmitido pela porta 1000 vai ser tratado sem prioridade na rede, ao contrário do que usar a porta 137. Para a classe 2 foram atribuídas as portas 22, pertencente à lista de acesso da classe GERENCIAMENTO e a 1300, fora da lista. A classe 3 representa a classe TEMPO_REAL_VOZ e recebeu as portas 1720, presente na lista de acesso dessa classe e a

1500 e que não está nessa lista. O trecho de código exposto no Quadro 05 mostra como foi feita a classificação para os testes dessa rede.

Quadro 05. Trecho de Código para Classificação de Testes em Rede

```
if (   (atoi(argv[1]) == 1000) ||
            (atoi(argv[1]) == 117))
                        dbClasse = 1;
                else if ((atoi(argv[1]) == 1300) ||
             (atoi(argv[1]) == 62))
                        dbClasse = 2;
                else if ((atoi(argv[1]) == 1500) ||
            (atoi(argv[1]) == 120))
                        dbClasse = 3;
                else if (atoi(argv[1]) == 1800)
          dbClasse = 4;
```

Após a classificação do tráfego, a ferramenta estabelece a conexão com o servidor, transfere o arquivo medindo a velocidade e insere os valores no banco de dados. Para medir a taxa de transmissão em um trecho do circuito (do cliente para o servidor) o aplicativo faz a diferença entre o tempo do início da transmissão e o do término e divide o resultado pelo tamanho do arquivo. A seguir a parte do código que executa essa função, especificada no Quadro 06.

Quadro 06. Código Executador da Função

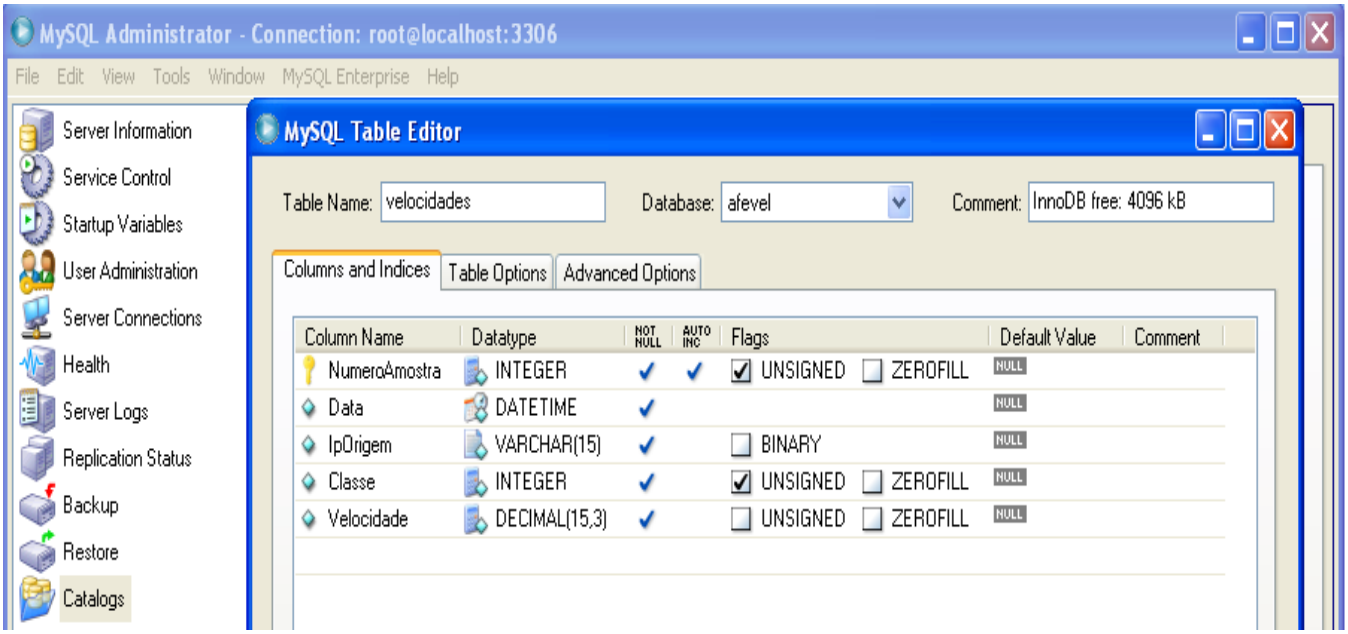
```
        time(&tempoInicio);
            tamanhoEnviado = w.enviaArquivo(fpath);

            printf("Arquivo %s com %ld bytes enviado para o servidor.\n",
fpath, tamanhoEnviado);
            time(&tempoFim);
            tempoDecorrido = difftime(tempoFim, tempoInicio);
            if (tempoDecorrido > 0)
                dbVelUp = (tamanhoEnviado / tempoDecorrido) / 1024;          else
                dbVelUp = tamanhoEnviado / 1024;
            printf("Tempos de inicio e fim de transmissao <%ld> <%ld>.\n", tempoInicio,
tempoFim);
        printf("Velocidade calculada %.3f Kbytes/s.\n", dbVelUp);
```

Os dados obtidos na medição da taxa de transferência são armazenados em um banco de dados. O banco de dados escolhido para esse experimento foi o MySQL 5.0. As principais razões para essa escolha são suas facilidades de instalação e configuração, compatibilidade com o sistema operacional dos computadores utilizados e o fato da versão utilizada ser gratuita. A estrutura criada no banco de dados é bem simples com apenas uma

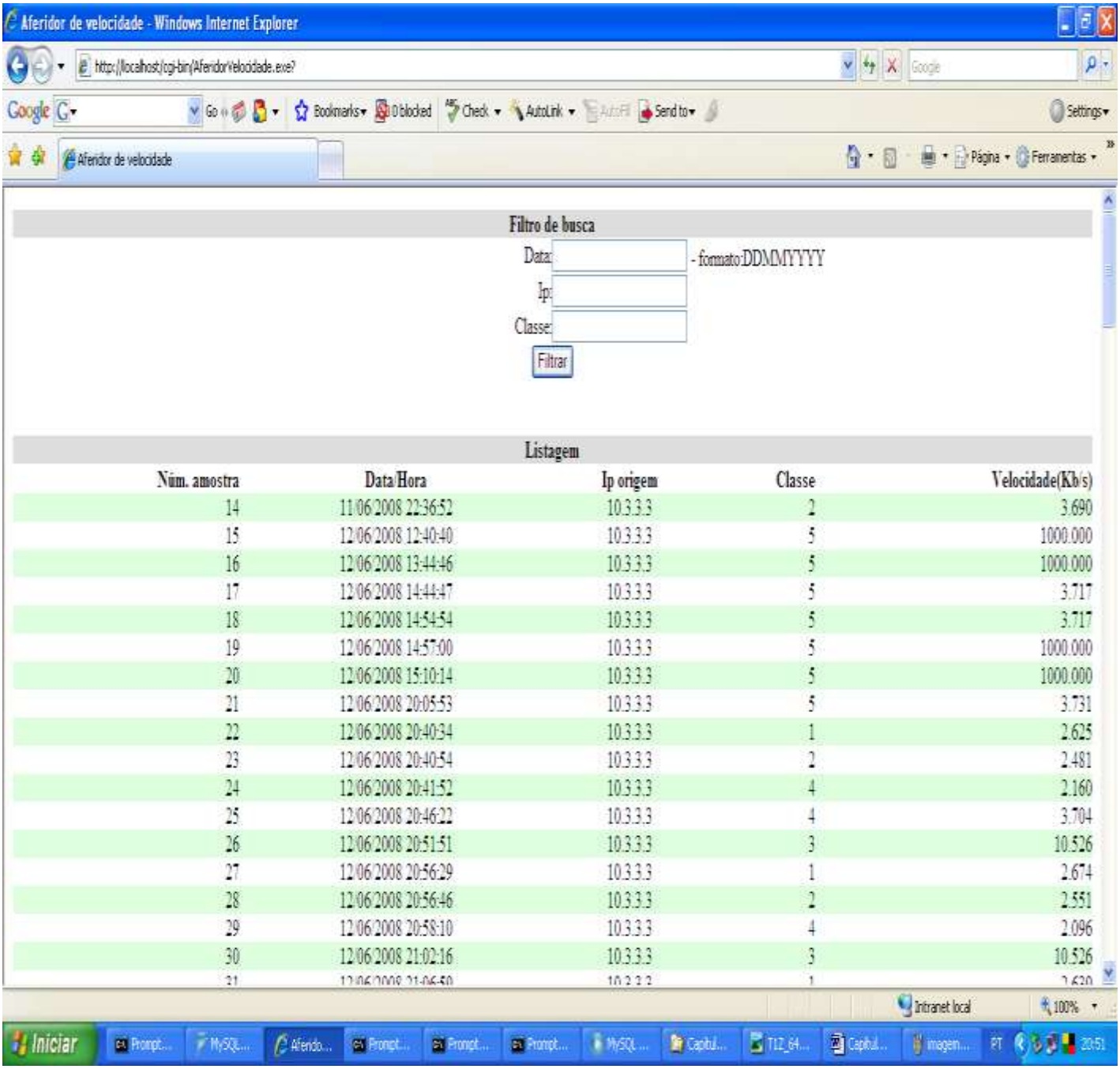
tabela. Essa tabela tem cinco colunas que armazenam os dados que são mostrados na *interface* de pesquisa. A Figura 11 mostra como ficou a organização dessa tabela.

Figura 11. Organização da Tabela



A exibição dos dados para os usuários é feita em uma página HTML acessada por meio do *browser*. O servidor WEB Apache interpreta o executável gerado pelo aplicativo e disponibiliza a página para acesso via *browser*. Nessa página, o usuário pode ter uma visão de todas as medições realizadas com informações da data da medição, IP de origem, a classe de serviço e a taxa de transmissão dos arquivos. O usuário pode, também, fazer a pesquisa dos resultados por data, IP de origem ou classe de serviço. O servidor Apache foi escolhido por ser compatível com a *interface* CGI utilizada no programa. E, também por sua facilidade de instalação e por ser uma ferramenta gratuita. A Figura 12 mostra a tela de interação com o usuário.

Figura 12. Interação com o Usuário



5.7. PASSO 7 – Análise dos Resultados

5.7.1. Resultados Obtidos

Os testes foram realizados utilizando as variáveis taxa de transferência do circuito, tamanho do arquivo e utilização ou não do QoS. Para efeitos didáticos, a unidade de medida utilizada para todos os cálculos foi bytes. Portanto, as taxas de transmissão configuradas para os circuitos, nos testes, foram 8*Kbytes* por segundo e 64*Kbytes* por segundo. Os arquivos utilizados tinham 1024000*bytes* e 10485760*bytes*. Para a caracterização do tráfego com QoS, foram utilizadas portas que constam nas listas de acesso dos roteadores. Já para a verificação do tráfego sem QoS, as portas utilizadas não estão nas listas dos roteadores.

Os testes foram iniciados com a rede configurada para trabalhar com a taxa de transmissão de 8*Kbytes*/s. Com essa configuração, na primeira bateria de testes, foi enviado um arquivo de 1Mbytes por cada uma das portas 117 - Classe Missão Crítica, 62 - Classe Gerenciamento, 120 - Classe Voz e 1800 - Classe Padrão (fora da lista de acesso). O intervalo entre o envio do primeiro arquivo e do último foi de três segundos. Esse mesmo intervalo ocorreu em todos os testes com envio paralelo de arquivos, porém, em nenhum dos casos chegou a 1% do tempo total da transmissão. Dessa forma, a sua interferência na pesquisa pode ser desprezada.

Nessa primeira bateria, a taxa de transmissão máxima do total enviado, 4*Mbytes*, atingiu 8,640*Kbytes*/s. Em relação ao QoS a única classe que se destacou foi a de voz que atingiu a taxa máxima de 8,772*Kbytes*/s. As demais, inclusive missão crítica e padrão, ficaram individualmente em torno de 2,170*Kbytes*/s. Ou seja, nesse teste o QoS não privilegiou a missão crítica em relação ao tráfego de gerenciamento nem, do padrão. Em compensação, a taxa de transmissão ultrapassou, em 8%, o que deveria ser a configuração do circuito. A Tabela 01 mostra as médias de transmissão registradas para cada classe com o tráfego compartilhado.

Tabela 01. Médias de Transmissão Registradas por Classe – Tráfego Compartilhado

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	1*Mbytes*	8Kbytes/s	Sim	2,171 *Kbytes*/s
Voz	1*Mbytes*	8*Kbytes*/s	Sim	8,626 *Kbytes*/s
Gerenciamento	1*Mbytes*	8*Kbytes*/s	Sim	2,168 *Kbytes*/s
Padrão	1*Mbytes*	8*Kbytes*/s	Não	2,166 *Kbytes*/s

Utilizando essa mesma configuração, também, foram feitos testes com o envio de arquivos, separadamente, para a classe missão crítica e a classe de voz. Nesse caso a taxa média da missão crítica ficou em 2,941*Kbytes*/s e a da voz foi de 10,109*Kbytes*/s. Dessa forma, nesse teste, a taxa de transmissão da classe missão crítica atingiu apenas 37% da velocidade do circuito, enquanto a de voz chegou a 126% do especificado para esse circuito.

Na segunda bateria de testes, foi mantido o QoS, a configuração da rede permaneceu com 8*Kbytes*/s e o tamanho do arquivo foi alterado para 10*MBytes*. Mais uma vez os arquivos foram enviados em paralelo e a taxa máxima das quatro classes juntas, totalizando 40*Mbytes*, foi de 8,658*Kbytes*/s. Essa taxa, novamente ficou acima da especificação do circuito. Nesse teste, classe de voz continuou sendo a única diferenciada pelo QoS. Porém, sua taxa de transmissão ultrapassou em mais de dez vezes o que deveria ser a característica do circuito. Em todas as repetições de testes com o uso do QoS e com os pacotes de 10M essa situação se repetiu. O arquivo enviado isoladamente da classe missão crítica teve o desempenho médio de 2,921*Kbytes*/s, mais uma vez ficou abaixo do SLA. Já o de voz teve média de 105,655*Kbytes*/s, treze vezes a mais do que seria a capacidade teórica do circuito. A Tabela 02 mostra as médias obtidas nesse teste.

Tabela 02. Médias Obtidas na 2ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	10*Mbytes*	8*Kbytes*/s	Sim	2,171*Kbytes*/s
Voz	10*Mbytes*	8*Kbytes*/s	Sim	93,347*Kbytes*/s
Gerenciamento	10*Mbytes*	8*Kbytes*/s	Sim	2,166Kbytes/s
Padrão	10*Mbytes*	8*Kbytes*/s	Não	2,167*Kbytes*/s

Em outra seqüência de testes, foi realizada com o circuito configurado para funcionar com a taxa de transmissão de 64*Kbytes*/s. Com essa especificação foram feitos

testes com arquivos de 1M e 10M. Os testes realizados com circuito de 64*Kbytes*, arquivos de 1M e as portas com utilização do QoS mostraram um crescimento de 18% nas taxas de transferência médias em relação ao mesmo teste feito com o circuito de 8*Kbytes*. Essa diferença ficou muito aquém do aumento de oito vezes na taxa do circuito. Nesse teste, o QoS favoreceu ligeiramente a classe gerenciamento que ficou com taxas melhores que a missão crítica. A taxa máxima obtida para o conjunto de quatro arquivos enviados em paralelo nesse teste ficou em 9,112*Kbytes*/s. Esse valor corresponde apenas a 14%, da taxa de 64*Kbytes*/s que o circuito deveria apresentar. O tráfego isolado da classe de voz teve média de 10,582*Kbytes*/s, ficando dentro da mesma faixa de valores dos obtidos nos circuitos de 8*Kbytes*/s. Já a classe missão crítica atingiu uma média de 6,211*Kbytes*/s.

A alteração da alteração da velocidade do circuito para 64*Kbytes*/s não apresentou um aumento do mesmo nível no desempenho da rede. Nesse teste, o desempenho da rede ficou bem abaixo do SLA e o QoS não priorizou a classe missão crítica.A Tabela 03 contém os principais resultados desse teste.

Tabela 03. Médias Obtidas na 3ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	1*Mbytes*	64*Kbytes*/s	Sim	2,529*Kbytes*/s
Voz	1*Mbytes*	64*Kbytes*/s	Sim	9,028*Kbytes*/s
Gerenciamento	1*Mbytes*	64*Kbytes*/s	Sim	2,856*Kbytes*/s
Padrão	1*Mbytes*	64*Kbytes*/s	Não	2,304*Kbytes*/s

Com a alteração do tamanho do arquivo para 10M somente a classe voz teve aumento representativo. As demais classes obtiveram praticamente os mesmos índices mostrados no teste anterior. A taxa máxima do conjunto dos arquivos em paralelo chegou a 9,200*Kbytes*/s, a classe missão crítica, quando enviado um arquivo separadamente, obteve 6191*Kbytes*/s e a classe de voz atingiu 108,926*Kbytes*/s na mesma situação. Mais uma vez o QoS e o SLA ficaram abaixo da expectativa. A Tabela 04 mostra os resultados desse teste.

Tabela 04. Médias Obtidas na 4ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	10*Mbytes*	64*Kbytes*/s	Sim	2,404*Kbytes*/s
Voz	10*Mbytes*	64*Kbytes*/s	Sim	89,537*Kbytes*/s
Gerenciamento	10*Mbytes*	64*Kbytes*/s	Sim	2,484*Kbytes*/s
Padrão	10*Mbytes*	64*Kbytes*/s	Não	2,314*Kbytes*/s

Os testes seguintes foram feitos com o circuito configurado para 8Kbytes/s, arquivos de 1M e as portas 1000 - Classe Missão Crítica, 1300 - Classe Gerenciamento, 1500 - Classe Voz e 1800 - Classe Padrão, todas fora da lista de acesso. Dessa forma, caracterizando um tráfego sem QoS. Novamente, quatro arquivos de 1M foram enviados em paralelo. Os testes mostraram que todas as classes obtiveram desempenho semelhante, em torno de 1,630*Kbytes*/s. Esse desempenho é inferior ao do tráfego com QoS tanto para as classes individualmente, como para o conjunto que reduziu a máxima dos 4*Mbytes* para 6,484*Kbytes*/s. A taxa média obtida pelo envio de uma classe sem QoS individualmente foi de 2,933*Kbytes*/s, ficando na mesma faixa das anteriores. A taxa de transmissão do conjunto do tráfego atingiu 81% do valor nominal do circuito. Já o tráfego individual, de 1*Mbytes*, atingiu apenas 37% do total da banda do circuito. A retirada do QoS, nesse teste, provocou a redução do desempenho da rede. A Tabela 05 mostra os resultados médios obtidos com essa configuração.

Tabela 05. Médias Obtidas na 5ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	1*Mbytes*	8*Kbytes*/s	Não	1,631*Kbytes*/s
Voz	1*Mbytes*	8*Kbytes*/s	Não	1,628*Kbytes*/s
Gerenciamento	1*Mbytes*	8*Kbytes*/s	Não	1,634*Kbytes*/s
Padrão	1*Mbytes*	8*Kbytes*/s	Não	1,632*Kbytes*/s

Para os testes seguintes o tamanho do arquivo foi alterado para 10M. Como mostra a Tabela 06, não houve alteração significativa em relação aos testes feitos com arquivos de 1M. A máxima de transferência do total de 40M foi de 6,502*Kbytes* e o valor para a classe missão crítica isoladamente, sem QoS, foi de 2,916*Kbytes*/s. O resultado desse teste mostrou que sem o QoS os quatro tipos de serviço trafegam na rede sem diferenças no desempenho entre si. Porém, a retirada do QoS reduziu o desempenho geral da rede. Nesse teste, a taxa de transmissão ficou em 81% do valor do circuito, portanto, dentro do esperado em termos de SLA.

Tabela 06. Médias Obtidas na 6ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	10*Mbytes*	8*Kbytes*/s	Não	1,628*Kbytes*/s
Voz	10*Mbytes*	8*Kbytes*/s	Não	1,627*Kbytes*/s
Gerenciamento	10*Mbytes*	8*Kbytes*/s	Não	1,627*Kbytes*/s
Padrão	10*Mbytes*	8*Kbytes*/s	Não	1,627*Kbytes*/s

Os testes feitos com a configuração de 64*Kbytes* para o circuito, arquivos com 1M e sem QoS mostraram um pequeno aumento em relação à situação semelhante com a variável circuito de 8*Kbytes*/s. Esse aumento, de apenas 16%, foi muito pequeno, comparado ao aumento da capacidade do circuito. Nesse teste, o tráfego isolado de um arquivo de 1M atingiu a taxa média de 6,061*Kbytes*/s, apenas 9,47% da taxa do circuito. A máxima do conjunto com 4M foi de 6,932Kbps/s, 10,8% do circuito. Os índices apurados nesse teste demonstram que o circuito de 64*Kbytes*/s não atendeu o SLA. Na Tabela 07 estão os valores desse teste.

Tabela 07. Médias Obtidas na 7ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada
Missão Critica	1*Mbytes*	64*Kbytes*/s	Não	1,889*Kbytes*/s
Voz	1*Mbytes*	64*Kbytes*/s	Não	1,839*Kbytes*/s
Gerenciamento	1*Mbytes*	64*Kbytes*/s	Não	2,354*Kbytes*/s
Padrão	1*Mbytes*	64*Kbytes*/s	Não	1,780*Kbytes*/s

Com a alteração da variável arquivo para 10*Mbytes*, os resultados não foram muito diferentes dos verificados com os arquivos de 1M. A média obtida com somente um arquivo na rede foi de 6,121*Kbytes*/s. E, a máxima taxa com os quatro arquivos em paralelo foi de 6,946*Kbytes*/s. Mais uma vez foi demonstrado que o SLA do circuito de 64*Kbytes*/s não foi atingido. A Tabela 08 apresenta os dados deste teste.

Tabela 08. Médias Obtidas na 8ª Bateria de Testes

Classe	Tamanho do Arquivo	Velocidade do circuito	QoS	Taxa de transmissão média alcançada

Missão Critica	10*Mbytes*	64*Kbytes*/s	Não	1,737*Kbytes*/s
Voz	10*Mbytes*	64*Kbytes*/s	Não	1,822*Kbytes*/s
Gerenciamento	10*Mbytes*	64*Kbytes*/s	Não	1,738*Kbytes*/s
Padrão	10*Mbytes*	64*Kbytes*/s	Não	1,820*Kbytes*/s

5.7.2. Análise dos Dados

A ferramenta construída demonstrou ser capaz de realizar os testes idealizados para a medição do tráfego do protótipo de rede configurado para esse trabalho. Dessa forma, cumpriu seu objetivo acadêmico proposto para esse trabalho. Além disso, esse aplicativo pode sofrer adaptações e melhorias para a medição do tempo de resposta em outras redes. Porém, a maior contribuição dessa ferramenta foi que, por meio dos resultados obtidos nesses testes, ficou evidenciada a necessidade de se conhecer melhor o que, de fato, está ocorrendo em uma rede.

Os dados desses testes mostraram que em algumas situações os valores obtidos para a taxa de transmissão dos arquivos estavam superiores à capacidade do circuito instalado. Como mostra a Tabela 09, isso aconteceu quando se usou o circuito de 8Kbytes e as classes com QoS.

Tabela 09. Médias Obtidas na 9ª Bateria de Testes

Tamanho do Arquivo	QoS	Velocidade do circuito	Taxa de transmissão máxima alcançada	Percentual do circuito instalado
1Mbytes	Sim	8*Kbytes*/s	8,640*Kbytes*/s	108%
10Mbytes	Sim	8*Kbytes*/s	8,658*Kbytes*/s	108%

Porém, em outros casos a taxa máxima de transmissão não atingiu sequer 15% da capacidade teórica da rede. Como mostra a Tabela 10, isso ocorreu quando o circuito deveria ter a capacidade de transmissão de 64*Kbytes*/s.

Tabela 10. Médias Obtidas na 10ª Bateria de Testes

Tamanho do Arquivo	QoS	Velocidade do circuito	Taxa de transmissão máxima alcançada	Percentual do circuito instalado

1*Mbytes*	Sim	64*Kbytes*/s	9,112*Kbytes*	14,2%
10*Mbytes*	Sim	64*Kbytes*/s	9,200*Kbytes*	14,4%
1*Mbytes*	Não	64*Kbytes*/s	6,932*Kbytes*	10,8%
10*Mbytes*	Não	64*Kbytes*/s	6,946*Kbytes*	10,8%

Pelos dados mostrados nas Tabelas 09 e 10 pode-se concluir que, em termos de velocidade do circuito, o SLA foi atendido para o circuito de 8*Kbytes*/s. Mas ficou muito abaixo do esperado para o circuito de 64*Kbytes*/s.

Os dados obtidos evidenciam que essa rede tem melhor desempenho quando utiliza as portas das classes listadas no QoS. A Tabela 11 mostra uma comparação dos dados com QoS e sem QoS.

Tabela 11. Comparação dos dados com QoS e sem QoS

Tamanho do Arquivo	QoS	Velocidade do circuito	Taxa de transmissão máxima alcançada	Percentual do circuito instalado
1*Mbytes*	Sim	8*Kbytes*/s	8,640*Kbytes*/s	108,0%
10*Mbytes*	Sim	8*Kbytes*/s	8,658*Kbytes*/s	108,0%
1*Mbytes*	Não	8*Kbytes*/s	6,484*Kbytes*/s	81,0%
10*Mbytes*	Não	8*Kbytes*/s	6,502*Kbytes*/s	81,3%
1*Mbytes*	Sim	64*Kbytes*/s	9,112*Kbytes*	14,2%
10*Mbytes*	Sim	64*Kbytes*/s	9,200*Kbytes*	14,4%
1*Mbytes*	Não	64*Kbytes*/s	6,932*Kbytes*	10,8%
10*Mbytes*	Não	64*Kbytes*/s	6,946*Kbytes*	10,8%

Por outro lado, ficou claro que o QoS não favoreceu a classe missão crítica. Como mostram as Tabelas de 01 a 04, o desempenho da classe que representa o serviço mais importante de uma organização ficou praticamente igual ao das demais classes de serviço. Já o desempenho observado para a classe voz em relação às outras foi bastante atípico, pode demonstrar que existem limitações impostas pela rede às demais classes.

Com esses testes, ficou provado que é impossível conhecer a rede somente por meio de suas configurações. Normalmente, ao se contratar uma rede deseja-se que essa contratação atenda às necessidades dos serviços de uma empresa e ao SLA contratado. Nesse sentido a metodologia proposta auxilia o administrador a identificar os objetivos, objetos e ferramentas necessárias para o estudo de uma rede MPLS. Definitivamente, só com medição é possível saber o que pode ser melhorado na rede para atender com mais eficiência aos serviços de uma empresa.

CAPÍTULO 6. CONCLUSÃO

É difícil imaginar o mundo hoje sem as redes de comunicação. Em pouco mais de 25 anos o mundo foi tomado por uma teia de redes, e para chegar nesse ponto, houve uma grande evolução na tecnologia da informação, não se podendo negar que dentro dessa área, a evolução das redes cumpre papel fundamental. Essa evolução é fruto de muito estudo e investigação, e, para evoluir foi necessário conhecer. Neste universo de conhecimentos, investigações e evoluções para os profissionais que trabalham com redes, é primordial acompanhar esse desenvolvimento. Nesse sentido, a medição da rede em que trabalha trará conhecimento para subsidiar proposições de melhorias. É nesta direção que o trabalho procurou contribuir para o melhor conhecimento da realidade dessa área.

Com o objetivo de auxiliar os administradores de redes a conhecerem melhor esses ambientes, foi proposta uma metodologia para a análise do desempenho de redes MPLS. Para testar essa metodologia, foi criado um protótipo de rede MPLS. Nessa rede foi configurado um QoS com quatro classes de serviço. Uma ferramenta foi desenvolvida para medir o desempenho dessa rede, de forma que pudesse testar se a rede tratou adequadamente o QoS, e se a taxa de transmissão de dados estava de acordo com as especificações da rede.

Esses testes foram realizados com a transferência de arquivos entre dois computadores nos dois extremos do protótipo. Foram feitos testes com duas velocidades de circuitos diferentes, com a utilização de classes com e sem QoS e com arquivos de 1 e 10 *Mbytes*. Na verificação do QoS, ficou demonstrado que a rede não tratou de forma prioritária o tráfego crítico. Em relação ao desempenho, à taxa de transferência do circuito de menor velocidade ficou acima do especificado e o de maior velocidade atingiu apenas aproximadamente 10% da configuração utilizada. Esses resultados demonstraram que, nesse caso específico, a rede estudada atendeu apenas parcialmente ao SLA especificado, porém, a contribuição mais importante a ser considerada é a comprovação da necessidade desse tipo de estudo.

A evolução na tecnologia da informação é muito rápida, um profissional dessa área não pode parar de se atualizar. Esse trabalho é uma pequena contribuição para a evolução desse conhecimento. Novos estudos devem ser realizados para que se conheça melhor as possibilidades da tecnologia MPLS. Por exemplo, uma boa proposta para a continuidade desse trabalho seria a análise de uma rede MPLS com a utilização de um sistema SNMP para a coleta de informações das interfaces dos dispositivos instalados na rede. Uma outra, alternativa seria a simulação de uma rede MPLS com a utilização de ambientes LINUX.

REFERÊNCIAS

[1] COMER, Douglas E. **Interligação de Redes com TCP/IP**. [trad. Daniel Vieira] 5.ed. Rio de Janeiro: Campus, 2006.

[2] EVANS, John & FILSFILS, Clarence. **Deplying IP and MPLS QoS for Multiservice Networks.** Estados Unidos: Morgan Kaufmann, São Francisco, 2007.

[3] GIMENEZ, Edson Josias C. **Metodologia Pragmática para Avaliação de Desempenho e Planejamento de Capacidade em Redes de Computadores**. Dissertação de Mestrado. Instituto Nacional de Telecomunicações – Inatel. Santa Rita do Sapucaí, MG, 2004.

[4] IENO JÚNIOR, Egídio. **Uma Proposta de Metodologia para Análise de Desempenho de Redes IEEE 802.11 Combinando a Gerência SNMP e Ferramentas de Simulação**. Dissertação de Mestrado. Instituto Nacional de Telecomunicações – Inatel. Santa Rita do Sapucaí, MG, 2003.

[5] **Internetworking Technology Handbook** (Capturado em 10.09.2008, http://www.cisco.com/en/US/docs/internetworking/technology/handbook/ito_doc.html)

[6] LEE, Donald C. **Enhanced IP Services for Cisco Networks**. Estados Unidos: Cisco Press, Indianápolis, 2000.

[7] MORAES, Luís Felipe M. de; VILELA, Guilherme S. **Uma Metodologia de Classificação para Fluxos de Comunicação**. Laboratório de Redes de Alta Velocidade – RAVE. Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ. 24° Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro, RJ, 2006.

[9] **MPLS Traffic Engineering (TE) MIB** (Capturado em 05.10.2008, http://www.cisco.com/en/US/docs/ios/12_2t/12_2t2/feature/guide/te_mib12.html#wp1029860)

[8] OSBORNE, Eric & SIMBA, Ajay. **Engenharia de Tráfego com MPLS**. [trad. Daniel Vieira] Rio de Janeiro: Campus, 2002.

[9] SALLES, Ronaldo Moreira & ROLLA, Vitor Gerra. **Otimização da Função de Roteamento para a Engenharia de Tráfego em Redes IP.** Departamento de Sistemas e Computação. Instituto Militar de Engenharia. 24° Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro, 2006.

[10] VEGESNA, Sriniva. **IP Qualty of Service**. Estados Unidos: Cisco Press, Indianápolis, 2001.

[11] SAYEED, Azhar & MORROW, Monique **Technology Overview: Making the Technology Case for MPLS and Technology Details**. Cisco Press, 2007

APÊNDICE A

Código da ferramenta desenvolvida para medição do protótipo de rede MPLS

```
#include <cstdlib>
#include <iostream>
#include <winsock2.h>
#include <mysql/mysql.h>
#include <stdio.h>
#include <stdlib.h>
#include "wcomm.h"

// Informações para conexão com banco de dados
char *host = "localhost";
char *usuario = "root";
char *senha = "Jackson";
char *database = "AfeVel";

using namespace std;

void runserver(int porta);
double runclient(int porta, char *ip, char *fpath);
void getword(char *palavra, char *linha, char separador);

WComm w;


int main(int argc, char *argv[])
{
	// Variáveis para tabela
	char	sql[512]		= "";
	char	*dbIpOrigem		= "";
	int		dbClasse		= 0;
	double	velocidade;
	int		flag			= 0;

	// Variáveis CGI
	char	*pTamanho,
			*serverSoft;
	int		tamanho,
			i,
			x;
	char	InputBuffer[1024];
	char	fData[20]		= "",
			fIp[15]			= "",
			fClasse[3]		= "",
			fWhere[200]		= "";

	// Define o modo em que será executado (Servidor, Cliente ou CGI)
	if(argc == 2)
	{
		system("CLS");
		printf("\n\n---------------------- Aferidor de velocidade ---------------------- \n\n");
		printf("...MODO SERVIDOR...\n\n");
		runserver(atoi(argv[1]));
	}
```

```
        else if (argc == 4)
        {
                system("CLS");
                printf("\n\n---------------------- Aferidor de velocidade ---------------------- \n\n");
                printf("...MODO CLIENTE...\n\n");


   // Define as classes a partir da porta selecionada no cliente
        if (   (atoi(argv[1]) == 1000) ||
          (atoi(argv[1]) == 137))
                        dbClasse = 1;
                else if ((atoi(argv[1]) == 1300) ||
           (atoi(argv[1]) == 22))
                        dbClasse = 2;
                else if ((atoi(argv[1]) == 1500) ||
          (atoi(argv[1]) == 1720))
                        dbClasse = 3;
                else if (atoi(argv[1]) == 1800)
        dbClasse = 4;
                   else
                        dbClasse = 5;


                // Define as classes a partir da porta selecionada no cliente
        //      if (            (atoi(argv[1]) == 2000)
        //                      ||      (atoi(argv[1]) == 2500)
        //                      ||      (atoi(argv[1]) == 2600)
        //                      ||      (atoi(argv[1]) == 2700) )
        //              dbClasse = 1;
        //      else if (atoi(argv[1]) == 2100)
        //              dbClasse = 2;
        //      else if (atoi(argv[1]) == 2200)
        //              dbClasse = 3;
        //      else if (atoi(argv[1]) == 2300)
        //              dbClasse = 4;
        //      else
        //              dbClasse = 5;

                // Inicializa a conexão
                MYSQL *connDb;
                connDb = mysql_init(NULL);
                if (!mysql_real_connect(connDb, host, usuario, senha, database, 0, NULL, 0))
                {
                        printf("ERRO ao conectar ao banco de dados MySQL: %s\n",
mysql_error(connDb));
                        return 1;
                }
                printf("Conectado ao banco de dados %s:%s com SUCESSO!\n", host,
database);

                // transfere o arquivo medindo a velocidade
                velocidade = runclient(atoi(argv[1]), argv[2], argv[3]);
                dbIpOrigem = w.pegaIp();
```

```
                printf("Inserindo informacoes no banco de dados\n");

                sprintf(sql, "INSERT INTO Velocidades (Data, ipOrigem, Classe,"
                                        " Velocidade) VALUES (NOW(), '%s', %d, %.3f)",
                                        dbIpOrigem, dbClasse, velocidade);

                // efetiva a inserção dos valores
                if (mysql_query(connDb, sql) != 0)
                {
                        printf("ERRO no banco de dados: %s!\n", mysql_error(connDb));
                        return 1;
                }
                printf("Linhas afetadas: %d\n", mysql_affected_rows(connDb));

                // Fecha conexão com o banco de dados
                mysql_close(connDb);

        }
        else
        {
                serverSoft = getenv("SERVER_SOFTWARE");
                if (serverSoft == NULL)
                {
                        system("CLS");
                        printf ("Uso:\n"
                                        "Modo servidor: AferidorVelocidade.exe <porta>\n"
                                        "Modo cliente: AferidorVelocidade.exe <porta> <ip>
<caminho_arquivo>\n"
                                        "Modo CGI: Copiar o executavel para dentro da pasta de
execucao de CGI do Webserver\n\n");
                        return 1;
                }

                // captura os dados enviados pelo navegador
                pTamanho = getenv("CONTENT_LENGTH");
                if (tamanho > sizeof(InputBuffer) - 1)
                        tamanho = sizeof(InputBuffer) - 1;
                i = 0;

                if (pTamanho != NULL)
                        tamanho = atoi(pTamanho);
                else
                        tamanho = 0;

                while (i < tamanho)
                {
                        x = fgetc(stdin);
                        if (x == EOF)
                                break;
                        InputBuffer[i++] = x;
                }
                InputBuffer[i] = '\0';

                // Pega os parâmentros
```

```
                getword(fData, InputBuffer, '=');
                getword(fData, InputBuffer, '&');
                getword(fIp, InputBuffer, '=');
                getword(fIp, InputBuffer, '&');
                getword(fClasse, InputBuffer, '=');
                getword(fClasse, InputBuffer, '&');

                // cabeçalho HTTP
                printf("Content-type: text/html\n");
                puts("\n");
                printf("<html>\n");
                printf("<head>\n");
                printf("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-
8859-2\">\n");
                printf("<title>Aferidor de velocidade</title>\n");
                printf("</head>\n");

                // formulário HTTP
                printf("<form name='main' action=? method='post'>\n");
                printf("<table align='center' width='100%%' cellpadding='0' cellspacing='0'
border='0'>\n");
                printf("        <tr bgcolor='#DDDDDD'>\n");
                printf("                <th colspan='2' align='center'>\n");
                printf("                        Filtro de busca\n");
                printf("                </th>\n");
                printf("        </tr>\n");
                printf("        <tr>\n");
                printf("                <td align='right' width='50%%'>\n");
                printf("                        <label>Data:</label></td><td
width='50%%'><input type='text' name='ip' value="> - formato:DDMMYYYY\n");
                printf("                </td>\n");
                printf("        </tr>\n");
                printf("        <tr>\n");
                printf("                <td align='right' width='50%%'>\n");
                printf("                        <label>Ip:</label></td><td
width='50%%'><input type='text' name='ip' value=">\n");
                printf("                </td>\n");
                printf("        </tr>\n");
                printf("        <tr>\n");
                printf("                <td align='right' width='50%%'>\n");
                printf("                        <label>Classe:</label></td><td
width='50%%'><input type='text' name='classe' value=">\n");
                printf("                </td>\n");
                printf("        </tr>\n");
                printf("        <tr>\n");
                printf("                <td align='center' colspan='2'>\n");
                printf("                        <input type='submit' value='Filtrar'>\n");
                printf("                </td>\n");
                printf("        </tr>\n");
                printf("</table>\n");
                printf("</form>\n");
                printf("<br>\n");

                // listagem HTTP
```

```
		printf("<table align='center' width='100%%' cellpadding='0' cellspacing='0'
border='0'>\n");
		printf("<tr>\n");
		printf("	<th bgcolor='#DDDDDD' colspan='5'
align='center'>Listagem</th>\n");
		printf("<tr>\n");
		printf("<tr>\n");
		printf("	<th align='right'>Núm. amostra</th>\n");
		printf("	<th align='center'>Data/Hora</th>\n");
		printf("	<th align='center'>Ip origem</th>\n");
		printf("	<th align='right'>Classe</th>\n");
		printf("	<th align='right'>Velocidade(Kb/s)</th>\n");
		printf("</tr>\n");

		// Inicializa a conexão
		MYSQL *connDb;
		connDb = mysql_init(NULL);
		mysql_real_connect(connDb, host, usuario, senha, database, 0, NULL, 0);

		// Filtro da busca
		if (strcmp(fData, "") != 0)
			sprintf(fWhere, " AND DATE_FORMAT(Data, '%%d%%m%%Y') =
'%s'", fData);

		if (strcmp(fIp, "") != 0)
			sprintf(fWhere, " AND IpOrigem = '%s'", fIp);

		if (strcmp(fClasse, "") != 0)
			sprintf(fWhere, " AND Classe = '%s'", fClasse);

		// Fazendo a consulta
		sprintf(sql, "SELECT NumeroAmostra, DATE_FORMAT(Data,
'%%d/%%m/%%Y %%H:%%i:%%S'), "
					"IpOrigem, Classe, Velocidade FROM Velocidades
WHERE 1=1 %s", fWhere);

		mysql_query(connDb, sql);
		MYSQL_RES * rs = mysql_store_result(connDb);
		MYSQL_ROW row;

		while (row = mysql_fetch_row(rs)) // Pega o resultado
		{
			// efeito zebra da listagem
			if (flag == 0)
			{
				printf("<tr bgcolor='#DDFFDD'>\n");
				flag = 1;
			}
			else
			{
				printf("<tr bgcolor='#FFFFFF'>\n");
				flag = 0;
			}
```

```
                printf("<td align='right'>%s</td>\n", row[0]);
                printf("<td align='center'>%s</td>\n", row[1]);
                printf("<td align='center'>%s</td>\n", row[2]);
                printf("<td align='right'>%s</td>\n", row[3]);
                printf("<td align='right'>%s</td>\n", row[4]);
                printf("</tr>\n");
            }
            mysql_free_result(rs); // Limpa a memória utilizada
            mysql_close(connDb); // Desconectando

            printf("</table>\n");
            printf("</html>\n");
        }
        return 0;
}


void runserver(int porta)
{
        w.iniciaServidor(porta);
        printf("Servidor iniciado - Aguardando conexao na porta %d.\n", porta);

        while (TRUE) {

            w.aguardaCliente();
            printf("Cliente conectado.\n");

            while(TRUE)
            {
                char rec[50] = "";
                w.recebeDados(rec,32);
                w.enviaDados("OK");

                // recebe o arquivo
                if(strcmp(rec,"EnvioArq")==0)
                {
                    char fname[32] = "";
                    w.recebeArquivo(fname);
                    printf("Arquivo %s recebido.\n", fname);
                }

                // finaliza a conexão
                if(strcmp(rec,"FimCon")==0)
                    break;

                printf("Conexão finalizada.\n");
                printf("Aguardando nova conexao na porta %d.\n\n", porta);
            }

            w.fechaConexao();
        }
}
```

```
double runclient(int porta, char *ip, char *fpath)
{
	char	rec[32]			= "";
	time_t	tempoInicio		= 0,
				tempoFim		= 0;
	double	tempoDecorrido		= 0;
	long	tamanhoEnviado		= 0;
	double	dbVelUp			= 0;


	// Conecta no servidor
	w.conectaServidor(ip, porta);
	printf("Conectado ao servidor %s:%d\n", ip, porta);

	// Envia o arquivo
	w.enviaDados("EnvioArq");
	w.recebeDados(rec,32);

	// Pega o tempo inicio da transmissão e envia o arquivo
	// guardando o tamanhoo enviado
	time(&tempoInicio);
	tamanhoEnviado = w.enviaArquivo(fpath);

	printf("Arquivo %s com %ld bytes enviado para o servidor.\n", fpath,
tamanhoEnviado);

	// Pega o tempo de termino da transmissão e calcula a diferença
	// faz a relação entre o tamanho enviado e tempo decorrido
	time(&tempoFim);
	tempoDecorrido = difftime(tempoFim, tempoInicio);
	if (tempoDecorrido > 0)
		dbVelUp = (tamanhoEnviado / tempoDecorrido) / 1024; // Kbytes por segundo
	else
		dbVelUp = tamanhoEnviado / 1024; // Kbytes por segundo

	printf("Tempos de inicio e fim de transmissao <%ld> <%ld>.\n", tempoInicio,
tempoFim);
	printf("Velocidade calculada %.3f Kbytes/s.\n", dbVelUp);

	// Envia sinal de fechamento de conexão
	w.enviaDados("FimCon");
	w.recebeDados(rec,32);

	return dbVelUp;
}


void getword(char *palavra, char *linha, char separador)
{
	int	x = 0,
		y;

	for (x = 0; ((linha[x]) && (linha[x] != separador)); x++)
		palavra[x] = linha[x];
```

```
	palavra[x] = '\0';
	if (linha[x])
		++x;

	y = 0;

	while ((linha[y++] = linha[x++]));
}



#include "wcomm.h"

WSADATA wsaData;
SOCKET m_socket;
SOCKET m_backup;
sockaddr_in con;
SOCKET AcceptSocket;

WComm::WComm()
{
	// Inicializa o socket do windows
	int iResult = WSAStartup( MAKEWORD(2,2), &wsaData );
	if ( iResult != NO_ERROR )
		printf("Erro no WSAStartup()\n");

	// Cria o socket
	m_socket = socket( AF_INET, SOCK_STREAM, IPPROTO_TCP );

	if ( m_socket == INVALID_SOCKET ) {
		printf( "Error no socket(): %ld\n", WSAGetLastError() );
		WSACleanup();
		return;
	}

	m_backup = m_socket;
}


void WComm::conectaServidor(char *ip,int port)
{
	// Conecta no servidor
	con.sin_family = AF_INET;
	con.sin_addr.s_addr = inet_addr( ip );
	con.sin_port = htons( port );

	if ( connect( m_socket, (SOCKADDR*) &con, sizeof(con) ) == SOCKET_ERROR) {
		printf( "Falha ao conectar no servidor.\n" );
		WSACleanup();
		return;
	}
}
```

```
void WComm::iniciaServidor(int porta)
{
	// Conecta ao servidor
	con.sin_family = AF_INET;
	con.sin_addr.s_addr = inet_addr("0.0.0.0");
	con.sin_port = htons(porta);

	if ( bind( m_socket, (SOCKADDR*) &con, sizeof(con) ) == SOCKET_ERROR) {
		printf( "Falha ao conectar.\n" );
		WSACleanup();
		return;
	}

	// Deixa o socket aguardando
	if ( listen( m_socket, 1 ) == SOCKET_ERROR )
		printf( "Falhar ao deixar socket aguardando conexão.\n");
}


void WComm::aguardaCliente()
{
	AcceptSocket = SOCKET_ERROR;
	while (AcceptSocket == SOCKET_ERROR) {
		AcceptSocket = accept(m_backup, NULL, NULL);
	}
	m_socket = AcceptSocket;
}


int WComm::enviaDados(char *sendbuf)
{
	return send( m_socket, sendbuf, strlen(sendbuf), 0 );
}


int WComm::recebeDados(char *recvbuf, int size)
{
	int sz = recv(m_socket, recvbuf, size, 0);
	recvbuf[sz] = '\0';
	return sz;
}


void WComm::fechaConexao()
{
	closesocket(m_socket);
	m_socket = m_backup;
}


void WComm::recebeArquivo(char *filename)
{
	char rec[50] = "";
```

```
	recv(m_socket, filename, 32, 0);
	send(m_socket, "OK", strlen("OK"), 0);

	FILE *fw = fopen(filename, "wb");

	int recs = recv(m_socket, rec, 32, 0);
	send(m_socket, "OK", strlen("OK"), 0);

	rec[recs]='\0';
	int size = atoi(rec);

	while(size > 0)
	{
		char buffer[1030];

		if(size>=1024)
		{
			recv( m_socket, buffer, 1024, 0 );
			send( m_socket, "OK", strlen("OK"), 0 );
			fwrite(buffer, 1024, 1, fw);
		}
		else
		{
			recv( m_socket, buffer, size, 0 );
			send( m_socket, "OK", strlen("OK"), 0 );
			buffer[size]='\0';
			fwrite(buffer, size, 1, fw);
		}
		size -= 1024;
	}
	fclose(fw);
}


long WComm::enviaArquivo(char *fpath)
{
	long	begin,
			end;
	// Pega somente o nome do arquivo no caminho especificado.
	char filename[50];
	int i = strlen(fpath);

	for(; i>0; i--)
		if(fpath[i-1] == '\\')
			break;

	for(int j = 0;i <= (int) strlen(fpath); i++)
		filename[j++] = fpath[i];
	//////////////////////////////////////

	ifstream myFile (fpath, ios::in|ios::binary|ios::ate);
	int size = (int)myFile.tellg();
```

```
        myFile.seekg (0, ios::beg);
        begin = myFile.tellg();
        myFile.seekg (0, ios::end);
        end = myFile.tellg();

        myFile.close();

        char filesize[10];
        itoa(size,filesize,10);

        send( m_socket, filename, strlen(filename), 0 );
        char rec[32] = "";
        recv( m_socket, rec, 32, 0 );

        send( m_socket, filesize, strlen(filesize), 0 );
        recv( m_socket, rec, 32, 0 );

        FILE *fr = fopen(fpath, "rb");

        while(size > 0)
        {
                char buffer[1030];

                if(size >= 1024)
                {
                        fread(buffer, 1024, 1, fr);
                        send(m_socket, buffer, 1024, 0 );
                        recv(m_socket, rec, 32, 0 );
                }
                else
                {
                        fread(buffer, size, 1, fr);
                        buffer[size]='\0';
                        send(m_socket, buffer, size, 0);
                        recv(m_socket, rec, 32, 0);
                }
                size -= 1024;
        }
        fclose(fr);
        return (end - begin);
}

char *WComm::pegaIp()
{
   char ac[80];
        char *ip;

   if (gethostname(ac, sizeof(ac)) == SOCKET_ERROR) {
      //cerr << "Error " << WSAGetLastError() << " when getting local host name." << endl;
      return "";
   }
   //cout << "Host name is " << ac << "." << endl;

   struct hostent *phe = gethostbyname(ac);
```

```
   if (phe == 0) {
      //cerr << "Yow! Bad host lookup." << endl;
      return "";
   }

   for (int i = 0; phe->h_addr_list[i] != 0; ++i) {
      struct in_addr addr;
      memcpy(&addr, phe->h_addr_list[i], sizeof(struct in_addr));
      ip = inet_ntoa(addr);
      //cout << "Address " << i << ": " << inet_ntoa(addr) << endl;
   }

         WSACleanup();

   return ip;
}



#include "winsock2.h"
#include "iostream.h"
#include "fstream.h"
#include <stdio.h>
#include <conio.h>

class WComm
{

public:
                  WComm();
         void conectaServidor(char*,int);
         void fechaConexao();
         long enviaArquivo(char*);
         void recebeArquivo(char*);
         int enviaDados(char*);
         int recebeDados(char*,int);
         void iniciaServidor(int);
         void aguardaCliente();
         char *pegaIp();

};
```