Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial

---

Dissertação de Mestrado

PLEX MPLS - Análise, projeto e implementação de uma Plataforma para Experimentos com MPLS com suporte a QoS

Autor: Ricardo Sazima

Orientador: Prof. Dr. Maurício Ferreira Magalhães

Banca Examinadora: Prof. Dr. Edmundo Roberto Mauro Madeira
Prof. Dr. Eleri Cardozo
Prof. Dr. Leonardo Souza Mendes

Tese submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos para obtenção do Título de Mestre em Engenharia Elétrica.

15 de outubro de 2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Sazima, Ricardo
Sa98p PLEX MPLS – Análise, projeto e implementação de uma plataforma para experimentos com MPLS com suporte a QoS / Ricardo Sazima. --Campinas, SP: [s.n.], 2004.

Orientador: Maurício Ferreira Magalhães
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Engenharia de tráfego. 2. TCP/IP (Protocolo de rede de computação). 3. Redes de computação – Protocolos. 4. Telecomunicações – Tráfego. 5. Comutação de pacotes (Transmissão de dados). 6. Interconexão de redes (Telecomunicação). I. Magalhães, Maurício Ferreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: PLEX MPLS - Analysis, project and implementation of a platform for experiments with MPLS with QoS support.
Palavras-chave em Inglês: Traffic engineering, TCP/IP (Computer network protocol), Computer network protocols, Telecommunication traffic, Packet switching (data transmission) e Internetworking (Telecommunication)
Área de concentração: Engenharia de Computação e Automação Industrial
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Edmundo Roberto Mauro Madeira, Eleri Cardozo e Leonardo Souza Mendes.
Data da defesa: 15/10/2004

Resumo

Dados a banda de transmissão atualmente disponível, o protocolo (IP) utilizado para transmitir a maior parte de tráfego e a quantidade de tráfego e seus requisitos de QoS, a Engenharia de Tráfego (TE, de *Traffic Engineering*) tornou-se um componente cada vez mais importante nas redes de comunicações. O padrão do IETF para encaminhamento/roteamento entitulado *Multi-Protocol Label Switching* (MPLS) preenche lacunas importantes neste cenário e é peça chave das metodologias de TE mais sofisticadas. Nesta dissertação é apresentado um modelo genérico de TE e uma proposta para uma **Plataforma para Experimentos com MPLS** (PLEX MPLS) que permite que o usuário defina, execute, e analise configurações de Engenharia de Tráfego com MPLS em uma rede IP. Os principais objetivos da PLEX MPLS são:

- Estudo da tecnologia MPLS e suporte a outros trabalhos em andamento no contexto do grupo de estudos em MPLS do DCA
- Experimentos com engenharia de tráfego com MPLS: validação das tecnologias e conceitos relacionados
- Experimentos didáticos em disciplinas de laboratório de redes

Os principais conceitos de TE e MPLS são apresentados como referência, bem como uma descrição do *NIST Switch*, a plataforma MPLS escolhida para este trabalho. A análise, projeto e implementação da PLEX MPLS são apresentados, pois formam parte significativa do trabalho desenvolvido. A PLEX não só utiliza, como também estende as funcionalidades oferecidas pelo *NIST Switch* a fim de oferecer um esquema de TE mais completo e eficiente.

Na fase de **análise**, vemos quais os principais requisitos para a implementação desta plataforma, seguindo uma metodologia de Engenharia de Software. Na fase de **projeto**, vemos as soluções propostas para os problemas identificados na fase de análise e temos uma especificação dos componentes a serem implementados. Na fase seguinte, discutimos a **implementação** das principais características dos componentes da PLEX, justificando as decisões tomadas.

Para validar a implementação da PLEX de acordo com sua proposta, foram realizados alguns **experimentos** em uma rede de testes com tráfego real. A execução destes experimentos é descrita e seus resultados analisados.

Os resultados obtidos assinalam claramente a importância e utilidade de esquemas de TE baseada em MPLS. Uma interessante metodologia para TE, compilada a partir de várias propostas, é apresentada. Finalmente, apontam-se caminhos a seguir em um trabalho futuro de refinamento da PLEX.

Abstract

Given the bandwidth currently available, the protocol (IP) used to transmit most Internet traffic, the quantity of traffic produced and its QoS requirements, Traffic Engineering (TE) has become an increasingly important component of communications networks. IETF's standard for forwarding/routing, which is entitled Multi-Protocol Label Switching (MPLS), presents important solutions in this scenario playing a major role in more sophisticated TE methodologies. This work presents a generic methodology for TE and a proposal for a Platform for Experiments with MPLS (PLEX MPLS) which allows the user to define, execute and analyze Traffic Engineering configurations with MPLS in an IP network. The main goals of PLEX MPLS are:

- Study of the MPLS technology and support of other ongoing works with MPLS in the DCA
- Traffic Engineering experiments with MPLS: validation of the related concepts and technologies
- Support of didactic experiments in academic disciplines

The main concepts of MPLS and TE are presented as reference, as well as a brief description of NIST Switch, the MPLS software chosen for the PLEX MPLS implementation. The analysis, project and implementation of PLEX MPLS are presented, since are significant part of the developed work. PLEX not only uses, but also extends NIST Switch functionalities to offer a more complete and efficient TE scheme.

In the analysis phase the main requirements for the PLEX implementation are specified, following a well-known Software Engineering methodology. The solutions found for the problems identified in the analysis phase are presented in the project phase alongside with a specification of the components that will be implemented. In the next phase, the implementation of PLEX is discussed focusing on the most important characteristics of PLEX components and justifying the implementation decisions.

In order to validate PLEX implementation and its proposal, some experiments were made in a test network with real live traffic. These experiments are described and its results analyzed.

The results obtained clearly indicate the importance and utility of TE schemes based on MPLS. Also an interesting TE methodology compiled from several proposals is presented. Finally, possible improvements and future work on PLEX MPLS are indicated.

Agradecimentos

Agradeço primeiramente a minha família, principalmente meu pai Ivan, minha mãe Marlies e minha irmã Cristina, sem a qual eu não seria o que sou e não teria chegado onde cheguei. Esta dissertação, bem como tudo que alcancei em minha vida só foi possível devido a seu apoio, seja afetivo, psicológico ou financeiro.

Agradeço a meu orientador, Prof. Maurício, pelos bons conselhos, pela amizade e por ter me guiado nesta empreitada, apoiando-me e transmitindo-me ou mostrando onde encontrar respostas nas horas de dúvida. A ajuda do Prof. José Mário na filmagem do segundo experimento também foi muito importante para a conclusão da parte prática do trabalho.

Meus amigos também me ajudaram muito, apoiando-me afetiva e psicologicamente. Quero agradecer ao grupo de trabalho em MPLS do LCA, especialmente a Alex, Leonardo, Siqueira, Vinicius, Tomás e Nicola, pela amizade, apoio e excelente trabalho em equipe. Quero crer que o conjunto dos trabalhos desenvolvidos por esse grupo fez, faz e continuará fazendo diferença.

Reconheço e agradeço também pelo apoio, tanto estrutural quanto financeiro, oferecidos pela Universidade Estadual de Campinas através de seu Departamento de Engenharia de Computação e Automação Industrial e pela CAPES.

Sumário

Lista de figuras

Lista de tabelas

Acrônimos

ACK	Acknowledgement Packet
ALTQ	Alternate Queueing
API	Application Programming Interface
BGP	Border Gateway Protocol
CBQ	Class Based Queuing
CBR	Constraint-Based Routing
CORBA	Common Object Request Broker Architecture
CoS	Class of Service
EGP	Exterior Gateway Protocol
FEC	Forwarding Equivalency Classes (MPLS)
FIFO	First In, First Out
FTN	FEC to NHLFE Map
GUI	Graphical User Interface
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
I/EBGP	Internal/External modes of the Border Gateway Protocol
ISP	Internet Service Provider
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switched Path
LSR	Label Switching Router
MATE	MPLS Adaptive Traffic Engineering
MPLS	Multiprotocol Label Switching
MTU	Maximum Transfer Unit
NHLFE	Next Hop Label Forwarding Entry
NIST	National Institute of Standards and Technology
OMG	Object Management Group
ORB	Object Request Broker
OSI	Open System Interconnection
QoS	Quality of Service
RAPI	RSVP API
RED	Random Early Detection
RFC	Request for Comments
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol-Traffic Engineering (MPLS)
SLA	Service Level Agreement
SQL	Structured Query Language (database query lanquage)
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol
UML	Unified Modeling Language

1 Introdução

A Internet se transforma a cada instante. Sua única característica constante parece ser a eterna necessidade de **mais**. Mais memória e velocidade de processamento nos computadores, mais banda de passagem, estabilidade e confiabilidade nas redes. A complexidade deste cenário aumenta com o fenômeno da **convergência de serviços** atualmente observado. As pessoas desejam transmitir informações em diversos formatos, como texto, áudio, vídeo, muitas vezes de maneira sincronizada! Há dois problemas básicos que podem ser identificados nesta situação:

- Redes de telecomunicação são projetadas para transportar tipos específicos de dados
- As diversas mídias a serem transmitidas possuem diferentes requisitos de comunicação que devem ser respeitados durante a transmissão

As mídias podem ser divididas basicamente em 4 tipos ([RFC2474] e [TG]):

1) Texto: dados textuais, tais como arquivos, email, telnet, …
2) Imagem: arquivos que representam uma imagem estática, como gif ou jpeg
3) Áudio: um fluxo de informações que representa a reprodução de sons, como uma música ou uma conversa telefônica
4) Vídeo: fluxo correspondente à transmissão de vídeo, como um filme ou uma videoconferência (a sincronização entre os participantes é fundamental)

Quanto à natureza do tráfego gerado pelas mídias, pode-se dividi-lo em três classes. Cada classe possui características distintas e é descrita por um conjunto de parâmetros:

1) Rajada (*burst*): alterna períodos com alta taxa de transmissão com outros de inatividade. Os parâmetros que a caracterizam são a taxa de pico, a duração das rajadas, a explosividade (taxa de pico/taxa média) e a distribuição das rajadas no tempo
2) Taxa de bits constante (CBR): como o nome diz, a taxa de transmissão é constante, portanto basta a taxa média para caracterizar esta classe
3) Taxa de bits variável (VBR): apresenta variações na taxa de transmissão, sendo necessários os seguintes parâmetros para caracterizá-la: taxa média, taxa de pico, variância da taxa de transmissão e explosividade

Atualmente, o único serviço oferecido pelas redes de computadores é o melhor-esforço, cujas principais características são: entrega não garantida (perda de pacotes), não há garantia de atraso máximo, grandes variações do atraso, não há banda mínima, não há ordenação intrínseca. Aplicações típicas incluem: ftp, news, mail, www, etc. Dados os requisitos para as mídias que se deseja transmitir atualmente, é necessário que seja oferecido um serviço com características adequadas: maior previsibilidade (reserva de recursos), atraso máximo garantido, pouca variação no atraso, largura de banda mínima, entrega ordenada. Aplicações que fariam proveito deste novo serviço incluem: multimídia, telefonia, tempo real, telemedicina, ...

Outro problema enfrentado atualmente na Internet é de natureza estrutural e se deve ao modelo de roteamento adotado para o protocolo IP. A rota de cada pacote é definida a cada nó baseada em seu destino final e nas métricas das conexões de rede visando sempre ao caminho de menor custo. O grande defeito deste modelo é que em certos cenários todos os pacotes são

enviados pelo mesmo caminho (ou pelo menos com alguns segmentos em comum), o que causa congestionamento da rota (com conseqüente degradação da QoS), além de desperdiçar recursos ociosos em outras rotas.

Vistos os problemas para se conseguir transmitir dados em diferentes mídias sobre uma rede de comunicações baseada em IP (rede de serviços integrados), fica patente a necessidade de se desenvolver mecanismos para prover a QoS de que as aplicações necessitam. Tecnologias de rede como ATM já incluem esses conceitos em seus projetos, no entanto, redes baseadas em IP (como a Internet) foram concebidas sem esta preocupação. Ultimamente, incentivados pela demanda de QoS, estão surgindo diversos conceitos para provê-la na Internet. A capacidade de reservar recursos ao longo do caminho de transmissão de um fluxo é adotada como solução para garantir a QoS fim-a-fim. Os principais conceitos são descritos abaixo:

1) Serviços Integrados/RSVP - IntServ ([RFC1633]): antes de iniciar a transmissão dos dados, as aplicações devem reservar recursos através do RSVP
2) Serviços Diferenciados – DiffServ ([RFC2475]): pacotes são marcados diferentemente para criar classes de serviços, que recebem tratamento diferenciado
3) *Multi-Protocol Label Switching* – MPLS ([RFC3031]): pacotes recebem rótulos ao entrar em um domínio MPLS. Classificação, encaminhamento e QoS dos pacotes são baseados nestes rótulos
4) Engenharia de Tráfego (TE): é um processo de distribuição de tráfego, tal que congestionamentos causados por utilização desigual da rede sejam mais facilmente contornáveis e a utilização dos recursos de rede seja otimizada
5) Roteamento Baseado em Restrições (CBR): computa rotas baseadas em requisitos de QoS, como banda de passagem, atraso, *jitter* (variação), ...

1.1 Engenharia de Tráfego (TE)

A seguir é apresentada uma visão geral de Engenharia de Tráfego (TE), porém um maior entendimento pode ser obtido em [RFC2702] e [RFC3272]. A TE é definida como a área da engenharia de redes que lida com os aspectos de avaliação e otimização da performance de redes. De modo geral, engloba a aplicação de princípios científicos e tecnológicos para a medição, caracterização, modelagem e controle do tráfego da rede. Os principais objetivos de performance podem ser classificados como:

Orientados a recursos: dizem respeito à otimização do uso dos recursos da rede de forma a evitar que partes da rede fiquem sobrecarregadas e congestionadas, enquanto que outros pontos estejam sub-utilizados. Banda é o recurso mais importante das redes atuais, portanto uma função central da TE é gerenciar eficientemente a banda da rede.

Orientados a tráfego: inclui aspectos relacionados à QoS experimentada pelo tráfego. Medidas de performance importantes são: atraso, variação do atraso, perda de pacotes, *throughput*, ...

O ideal é atender aos requisitos de performance orientados a tráfego (atraso, variação do atraso, perda de pacotes e banda) utilizando os recursos de rede de maneira econômica e confiável. Minimizar congestionamentos é um dos principais objetivos de performance (tanto pelo lado do tráfego, como dos recursos). O tipo de congestionamento a ser evitado é aquele de duração maior que congestionamentos transientes resultantes de rajadas de tráfego. Tipicamente congestionamentos ocorrem sob dois cenários:

- Recursos de rede são insuficientes ou inadequados para atender à demanda. Neste caso, pode-se expandir a capacidade da rede, aplicar mecanismos clássicos de controle de congestionamento (regulam a demanda ao limitar o tráfego) ou ambos.
- Tráfego é ineficientemente mapeado aos recursos disponíveis. Neste caso a TE é de extrema utilidade por conseguir mapear o tráfego de maneira diversa ao mapeamento produzido por protocolos de roteamento baseados em SPF e por realizar um melhor balanceamento de carga.

Enquanto é possível resolver o primeiro cenário com medidas relativamente simples, como modificações na arquitetura da rede, utilização de técnicas de controle de congestionamento, ou mesmo com um simples aumento da capacidade, o segundo caso é mais complicado, pois o cerne do problema é o algoritmo utilizado pelos protocolos de roteamento para escolha das rotas. Para solucioná-lo pode-se mapear o tráfego a rotas alternativas e utilizar políticas de balanceamento de carga.

Os opositores da TE argumentam que ela não é necessária porque no futuro haverá banda de passagem em abundância e, portanto, todos os tráfegos experimentarão a QoS de que necessitam. Há 2 problemas com esta colocação: primeiro, mesmo que este cenário se torne realidade, ainda vai demorar (pelo menos 10-20 anos no Brasil); segundo, muitos congestionamentos ocorrem devido à má utilização dos recursos de rede, que é um dos aspectos solucionados com TE, não sendo então necessário aumentar a capacidade da rede. Conclui-se, portanto, que a TE não só é necessária, como também desejável!

Os sistemas para TE podem ser classificados segundo as seguintes características:

***Time-dependent*/*State-dependent*:** *Time-dependent* significa que a TE é realizada a partir de informações históricas baseadas em variações sazonais. *State-dependent* quer dizer que a TE é baseada nas condições atuais da rede e do tráfego.

***On-line*/*Off-line*:** O cálculo de rotas pode ser feito *on-line* ou *off-line*. No caso de algoritmos complexos ou baseados em informação histórica o cálculo é realizado *off-line*. Para a TE se adaptar às condições atuais da rede, o mais indicado é o cálculo *on-line*.

Centralizado/distribuído: Sendo o controle centralizado, uma autoridade central coleta informações de estado da rede, calcula as rotas e informa os roteadores periodicamente. Já no caso distribuído, cada roteador coleta informações sobre o estado da rede e calcula as rotas a ele pertinentes autonomamente.

Informação global/local: O cálculo das rotas pode ser feito baseado em informações de toda a rede ou de apenas uma porção dela. Informação global é apropriada para otimização centralizada e alguns algoritmos distribuídos. Informação local é passível de utilização apenas por algoritmos distribuídos.

Prescritivo/descritivo: TE prescritiva analisa alternativas e recomenda um curso de ação. Já TE descritiva descreve o impacto de diferentes ações na rede sem recomendar nenhuma.

***Open-loop*/*Closed-loop*:** Controle *open-loop* não utiliza informação a respeito do estado atual da rede, ao passo que *closed-loop* a utiliza.

Idealmente, portanto, um bom esquema de TE partiria de informações de tráfego *time-dependent*, da topologia, capacidade dos recursos de rede e outros requisitos, aplicaria um algoritmo CBR centralizado utilizando informação global para instanciar uma configuração prescritiva em que estariam contemplados tanto o roteamento (explícito), quanto o mapeamento de tráfego a rotas, além da QoS a ser reservada para cada rota. Uma vez instanciada esta configuração inicial estática, deveria ser utilizado outro algoritmo distribuído utilizando informação global ou local para lidar com características *state-dependent* e prover tolerância a falhas (confiabilidade) através de rotas sobressalentes e re-roteamento rápido. O algoritmo centralizado deveria ser utilizado periodicamente para levar a rede a um estado padrão.

1.1.1 Dificuldades para se implementar Engenharia de Tráfego

Os principais problemas enfrentados para se implementar um bom esquema de TE são os seguintes:

- Cálculo da alocação ótima de recursos a fluxos: há diversos algoritmos propostos, mas cada um tem seus problemas. Um dos mais eficientes parece ser o de "mínima interferência" ([KAR00])
- Inexistência de mecanismos de mapeamento de tráfego a rotas
- Roteamento baseado no destino: os atuais protocolos de roteamento utilizados em redes IP não implementam roteamento explícito, uma peça fundamental da TE
- Inexistência de mecanismo padrão para garantia de QoS (*IntServ*, *DiffServ*)
- Consistência inter-domínios: devido à autonomia dos domínios, não há como garantir a consistência das políticas de TE entre domínios, bem como a QoS experimentada por tráfego que percorra mais de 1 domínio

1.2 Multi-Protocol Label Switching (MPLS)

A seguir é apresentada uma visão geral do *Multi-Protocol Label Switching* (MPLS), porém um maior entendimento pode ser obtido em [RFC3031].

Existe atualmente um grande interesse no *Multi-Protocol Label Switching* (MPLS) por ser uma tecnologia que possibilita a implementação de TE em redes de grande escala. O MPLS propõe uma separação funcional entre a computação de rotas em uma rede e o processo de encaminhamento. MPLS é, no sentido amplo do conceito, o uso de estruturas da camada 2 do modelo OSI, tais como cabeçalhos *Ethernet* (estendidos) ou identificadores de circuito virtual (VC) e caminho virtual (VP) no ATM, para propósitos de nível mais alto, como roteamento e provisão de QoS. É importante notar que apesar do MPLS não abranger a questão da QoS originalmente, LSPs (Label Switched Paths) podem ser associados a requisitos de QoS que são atendidos através de mecanismos de outros padrões, tais como *DiffServ* (veja a plataforma NIST *Switch* adiante que utiliza o ALTQ).

Roteadores de ingresso (*Label Edge Routers* – LERs) de um domínio MPLS classificam os pacotes em classes de equivalência de encaminhamento (*Forwarding Equivalence Classes* - FECs) baseados em vários fatores, incluindo informações do cabeçalho do pacote (endereço de origem e destino, classe de serviço, ...) e informações locais de roteamento. Um rótulo MPLS é então adicionado ao pacote classificado conforme especificado na tabela FEC-toNHLFE (*Forwarding Equivalence Class to Next-Hop Label Forwarding Entry*, ou simplesmente FTN). Em redes que não sejam ATM ou *Frame Relay*, o rótulo tem 32 bits e é composto pelos seguintes

campos: rótulo com 20 bits, campo experimental com 3 bits (antigo campo CoS), indicador de pilha de rótulos com 1 bit e TTL com 8 bits. Em redes ATM ou *Frame Relay*, o rótulo MPLS consiste de informações codificadas no campo VCI/VPI (DLCI). Após a adição do rótulo MPLS ao pacote, os roteadores MPLS (*Label Switch Routers* - LSRs) utilizam estas informações para tomarem decisões de encaminhamento para o pacote.

Para tomar uma decisão de encaminhamento, o LSR utiliza o rótulo como índice em uma tabela de NHLFEs (Next Hop Label Forwarding Entry), que guarda informações do tipo rótulo/interface de entrada, rótulo/interface de saída, entre outras. O pacote é processado do modo especificado na NHLFE. O rótulo de entrada pode ser substituído por outro e o pacote encaminhado ao próximo LSR. Este processo é muito semelhante ao processo de permuta de rótulos nativo do ATM. Antes do pacote deixar o domínio MPLS, o rótulo é retirado. Um Caminho Comutado por Rótulos (*Label Switched Path* – LSP) é o caminho entre o LER de entrada e o LER de saída. O caminho de um LSP com rota explícita é definido no LER de entrada. Um protocolo de sinalização como *Resource Reservation Setup Protocol* (RSVP) ou *Label Distribution Protocol* (LDP) é utilizado para estabelecer LSPs.

Rótulos são potencialmente úteis para várias finalidades: garantia de espectro de banda e latência, diferenciação de serviço, gerenciamento de tráfego, definição de grupo *multicast*, balanceamento de carga, roteamento mais eficiente.

Diferentemente de algoritmos de roteamento IP, que podem apenas determinar o próximo nó no caminho, um ambiente comutado por rótulos pode criar **rotas explícitas** (ou seja, com os nós do caminho determinados ou parcialmente determinados) através da distribuição apropriada de informações de rótulos. A determinação do caminho só precisa então ser executada uma vez, sendo que nos nós intermediários o rótulo do pacote determina o próximo nó da rota.

Este paradigma de roteamento fornece uma série de vantagens: (1) pode-se gerar e manter rotas individuais e segmentos de rotas como desejado, prevenindo a ocorrência de *loops* ou direcionamento transiente errado de pacotes; (2) através do controle da geração e distribuição de rótulos, pode-se coordenar a reserva de recursos e o priorizações nas filas de modo a garantir a banda de passagem e controlar tempos totais de atraso de pacotes; (3) adicionalmente, já que decisões de roteamento podem ser tomadas em menos pontos e até feitas a priori (inteira ou parcialmente), há espaço para a implementação de algoritmos de roteamento mais complexos que sejam capazes de prover melhor determinação de caminhos e melhorar de maneira geral a utilização da rede (objetivos da TE).

Esta abordagem pode ser resumida do seguinte modo: um rótulo designa um caminho ou segmento de caminho comutado por rótulos numa rede com certas características de QoS. Estes caminhos podem ser pré-alocados ou criados à medida que se fazem necessários (especialmente com o uso de RSVP). O conjunto de caminhos vai em geral formar uma floresta de cobertura do grafo da rede. Então, no resto da rede, os rótulos determinam tanto a rota a ser seguida como as características de manuseio de tráfego utilizadas. Rotas preferíveis são aquelas com um conjunto de segmentos menor e, conseqüentemente, menor pilha de rótulos (idealmente um único).

Determinar boas rotas é um processo em dois passos: a alocação a priori de segmentos úteis e a agregação sob demanda destes em conjuntos apropriados. Do ponto de vista da teoria de grafos, as questões relevantes tornam-se então como alocar uma floresta de cobertura rotulada e como gerenciá-la para alcançar os objetivos de roteamento desejados. Tipicamente, o número permitido de rótulos seria razoavelmente limitado (na casa dos poucos milhares), permitindo

apenas uma fração do número possível de caminhos de ser especificada. Em ambos os casos, parte-se do princípio de que se possui algum conhecimento do estado do domínio inteiro.

1.2.1 MPLS e Engenharia de Tráfego

Ao se utilizar o MPLS para TE os fluxos de tráfego de mesmo tipo são mapeados a LSPs, que por sua vez têm recursos da estrutura física da rede subjacente alocados a si. Portanto, para tornar possível a TE no MPLS, os seguintes elementos são necessários:

- Um conjunto de atributos associados aos fluxos de tráfego que especifica suas características comportamentais
- Um conjunto de atributos associados aos recursos de rede que regulam a participação dos mesmos nos LSPs pelos quais os fluxos irão passar
- Um procedimento para especificar quais recursos serão alocados a quais LSPs e quais fluxos serão mapeados a quais LSPs de modo a otimizar a utilização dos recursos da rede e/ou a QoS experimentada pelo tráfego. Este procedimento pode ser manual (administrador da rede) ou automatizado (MATE, CBR)
- Mecanismos que implementem o mapeamento de tráfego a LSPs e garantam a QoS

A TE baseada em MPLS pode ser decomposta em quatro componentes, que se encontram no plano de controle (onde a TE atua), que é desacoplado do plano de encaminhamento.

Gerenciamento de caminhos: relacionado aos aspectos de seleção de rota explícita, instanciação e manutenção de LSPs.

Atribuição de tráfego a LSPs: particionamento do tráfego ingressante (classificação) e alocação aos LSPs existentes. A alocação do tráfego poderia ser automatizada através de alterações no algoritmo SPF utilizado pelos protocolos de roteamento para que sejam consideradas mais características dos caminhos e do tráfego além do custo do caminho. Atributos adicionais podem controlar a alocação quando há mais de um caminho e pode ser realizada distribuição de carga entre LSPs paralelos.

Disseminação da informação de estado da rede: distribuição de informações da topologia e estado da rede a todo o domínio MPLS. Estas informações são usadas para computar rotas para os LSPs, para alocar tráfego a LSPs, para o controle geral da rede, etc. Entre as diferentes possibilidades para distribuição da informação de estado, vale citar: protocolos de gerenciamento de rede (SNMP) e extensões a protocolos de roteamento existentes (OSPF).

Gerenciamento da rede: responsável pelo controle do sistema através da monitoração do estado da rede e da atuação em seus componentes para atingir os objetivos da TE. Suas principais funções são: a) monitoramento de performance e contabilização; b) gerenciamento de configuração; c) gerenciamento de falhas.

A Figura 1 ilustra os componentes nos diversos níveis de uma arquitetura de TE. O nível mais alto da arquitetura é representado pelo algoritmo offline centralizado de TE, que recebe como entrada os diversos requisitos e fornece uma configuração para a rede. Este nível está relacionado ao ciclo de vida de LSPs e opera em uma escala de tempo de horas ou até dias. O nível intermediário é representado por algoritmos *online* que lidam com a dinâmica da rede e devem alterar a configuração da rede baseados nas informações disseminadas pelos protocolos

de roteamento (adaptados a MPLS ou não). Este nível está relacionado à admissão de tráfego de aplicações e otimizações locais, operando em uma escala de tempo de segundos. Finalmente, o nível mais baixo é representado por mecanismos de classificação de tráfego e restauração que implementam micro-ações de controle. Este nível está relacionado ao mapeamento de tráfego a LSPs, balanceamento de carga, restauração de LSPs (devido a falhas) e opera em uma escala de tempo de microssegundos, milissegundos ou segundos.

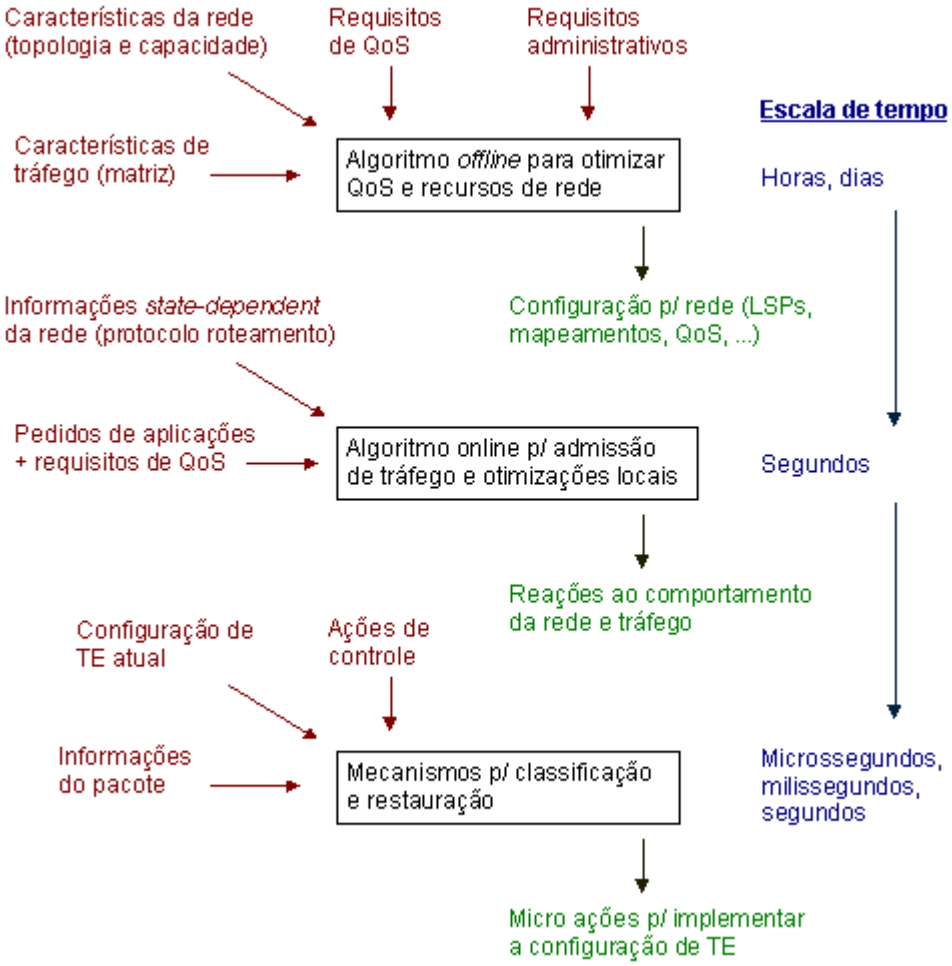


Figura 1: Arquitetura para Engenharia de Tráfego em 3 camadas

1.2.2 Vantagens do MPLS para Engenharia de Tráfego

A utilização de MPLS para TE proporciona os seguintes benefícios:

- MPLS permite esquemas sofisticados de roteamento (como o CBR) baseados na capacidade de estabelecimento de LSPs explicitamente roteados
- Classes de tráfego podem ser mapeadas a LSPs
- Projeto e arquitetura de rede mais simples e baratos do que o modelo *Overlay* (MPOA)
- MPLS permite tanto a agregação quanto a desagregação de tráfego, enquanto que o esquema de encaminhamento tradicional permite apenas agregação
- Integração de métodos automatizados relativamente simples

1.3 O projeto NIST Switch

O NIST ([NIST]) vem desenvolvendo o projeto NIST Switch desde 1998. O NIST *Switch* ([SWITCH]) é um ambiente para testes e pesquisa com MPLS em equipamentos de baixo custo como PCs com placas Ethernet. Mais especificamente, o NIST Switch é um protótipo de um *Label Switch Router* (LSR). O NIST Switch é baseado em *software* de domínio público como FreeBSD ([HIX93]), ALTQ ([ALTQ]) e RSVP ([RSVP]). Inicialmente o protótipo suporta os mecanismos de encaminhamento do MPLS, incluindo suporte para QoS baseado em classes "cientes" de rótulos e extensões simples ao RSVP para distribuição de rótulos e sinalização de QoS.

1.3.1 Arquitetura do NIST Switch

O NIST *Switch* 0.2 foi desenvolvido utilizando ALTQ 2.0 e FreeBSD 3.3. Seus componentes essenciais operam entre as camadas de enlace e de rede. O diagrama mostrado na Figura 2 mostra a arquitetura do NIST *Switch* conforme implementada.

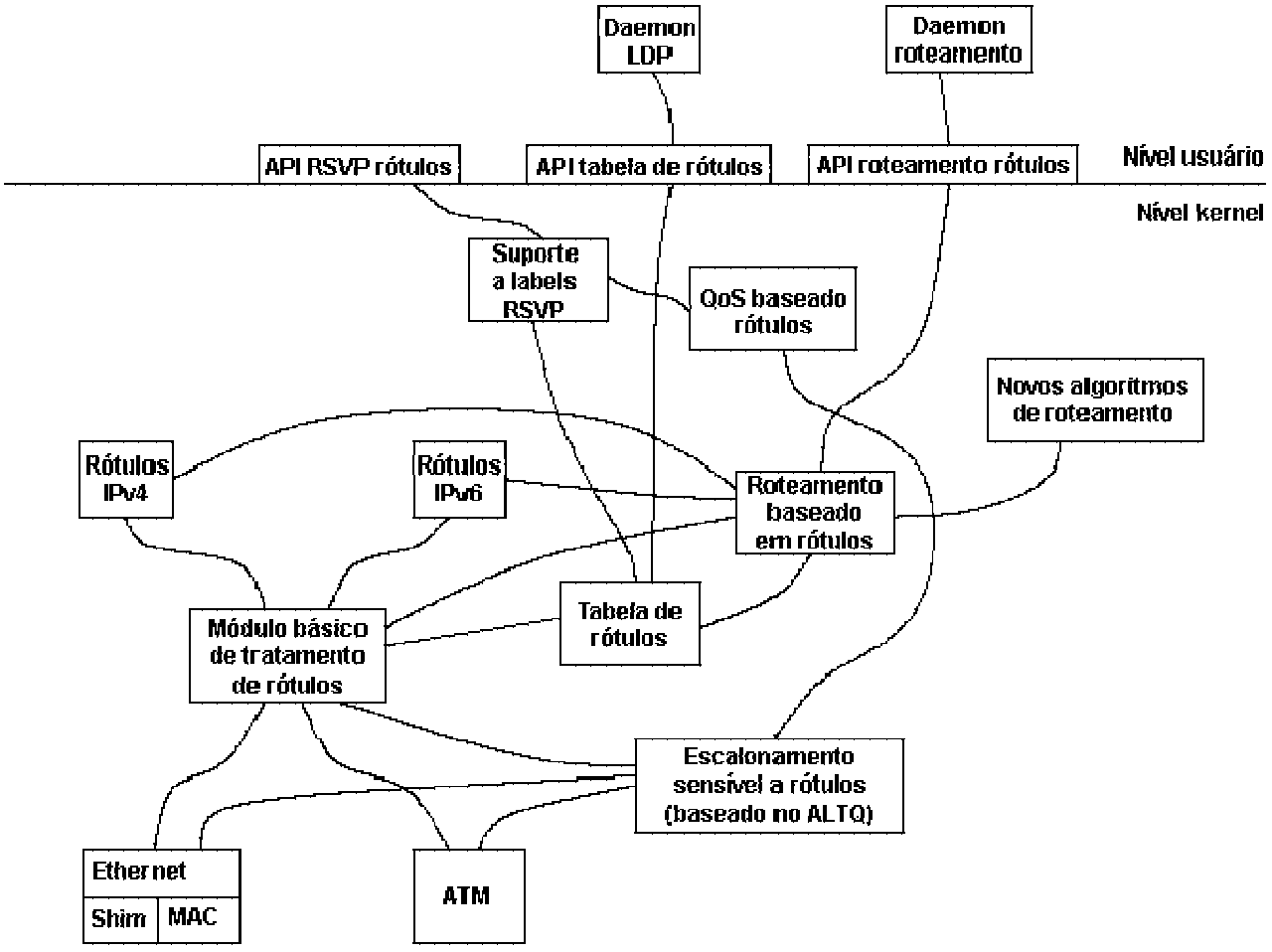


Figura 2: Arquitetura funcional da plataforma NIST Switch

Componentes principais:

Módulo básico de tratamento de rótulos: Cria, interpreta e processa os rótulos ou pilhas de rótulos para os pacotes. Pacotes recebidos podem conter um rótulo ou não e serem encaminhados ou recebidos localmente. Pacotes sendo enviados podem já estar rotulados ou não (caso tenham sido gerados localmente ou recebidos sem rótulo) e podem ser encaminhados rotulados ou não (caso o nó seja um LER de saída). Para pacotes não rotulados, ele determina o rótulo apropriado a ser aplicado (se for o caso) e a interface de saída baseado nos endereços de origem e destino (endereço IP e possivelmente porta UDP/TCP) e classe de serviço. Para pacotes rotulados, ele determina o rótulo apropriado a ser aplicado (se for o caso) baseado no rótulo já existente. Para pacotes sendo recebidos, o rótulo do pacote é usado para determinar como o pacote deve ser tratado (encaminhado ou recebido localmente). Se o pacote recebido deve ser encaminhado, o módulo determina as mudanças necessárias (se existentes) a serem realizadas na pilha de rótulos. O rótulo atual pode ser retirado da pilha, substituído por outro, ou ainda um novo rótulo pode ser colocado no topo da pilha (tunelamento).

Tabela de rótulos: As informações relacionadas aos rótulos são guardadas numa tabela no *kernel* (núcleo), indexada tanto pelos rótulos como por informações de nível mais alto: origem, destino, rota, e informações de serviço. A tabela é projetada para ter acesso de leitura compartilhado (rotinas de encaminhamento de pacotes no nível de interrupção de rede) e acesso de escrita exclusivo (rotinas de atualização da tabela de rótulos em tempo de tarefa, sendo executadas em benefício do RSVP ou protocolos simples de distribuição de rótulos (LDP) ou ainda *daemons* de roteamento baseado em QoS). A tabela é potencialmente grande, com talvez dezenas de milhares de entradas para um domínio com algumas centenas de nós. Indexação rápida é, portanto, de extrema importância, sendo usada uma variante de *dynamic double hashing*. Indexar por rótulo (por exemplo, ao encaminhar um pacote já rotulado) requer apenas uma consulta, por outro lado, indexar pelas informações de nível mais alto (como nos roteadores de ingresso) pode exigir múltiplas consultas para que seja achada a entrada que melhor obedece ao critério de busca.

Tratamento de rótulos no nível de enlace: O código de tratamento de rótulos no nível de enlace escreve e recupera a informação do rótulo no cabeçalho (ampliado) *Ethernet* (ou, num estágio mais avançado, ATM). Para a *Ethernet*, tanto o encapsulamento MAC (*draft-srinivasan-mpls-lans-label-00.txt*) como a codificação *shim* (*draft-rosen-tag-stack-03.txt*) de rótulos são suportados. Qualquer um dos tipos é reconhecido em todas as interfaces de entrada; o tipo usado (se necessário) na interface de saída é configurável por interface. Inicialmente as pilhas de rótulos serão acomodadas excedendo-se a MTU, ou seja, com pacotes maiores que 1500 bytes (as placas de rede escolhidas para o projeto suportam pacotes > 1500 bytes).

Escalonamento baseado em rótulos: Para prover suporte para QoS, são necessários algoritmos de escalonamento que classifiquem e lidem com pacotes de maneira apropriada. Código para isto existe para implementações IP puras (novo código 2.1.1XX para Linux, ALTQ para FreeBSD, sendo que o ALTQ foi modificado de uma maneira tal que possibilita escalonamento baseado em classes de serviço através de rótulos ao invés de cabeçalhos IP). As rotinas CBQ destas implementações foram adaptadas para incluir suporte para um ambiente baseado em rótulos. O código adaptado de escalonamento lida com todos os tipos de tráfego, rotulado e IP, de uma maneira única com um conjunto de filas baseado na classe de serviço à qual pertencem os pacotes.

Suporte do RSVP a rótulos: O RSVP interage com os rótulos basicamente de duas maneiras: (1) como um protocolo sinalizador de QoS, ele distribui pedidos de alocação de recursos para fluxos específicos, que são tipicamente associados a rótulos; (2) adicionalmente, o RSVP serve como protocolo de distribuição dos rótulos (*draft-davie-mpls-rsvp-01.txt*). Esta integração tem a vantagem de que sinalização de QoS e tratamento de rótulos para fluxos estão unificados em um só protocolo. Ambos os tipos de suporte do RSVP a rótulos são implementados.

1.3.2 Modificações no RSVP

1. Roteamento explícito:

A extensão proposta para roteamento explícito torna possível o envio de mensagens RSVP por um caminho diferente daquele que seria utilizado pelo roteamento IP normal. Isto é útil quando um operador de rede quer direcionar tráfego com requisitos de QoS através de certas rotas na rede. Uma vantagem adicional é que se pode especificar onde na rede caminhos comutados por rótulos devem ser estabelecidos e pode-se garantir, através da alocação correta de rotas, que sejam evitadas anomalias como *loops*.

Para possibilitar o roteamento explícito, o RSVP foi estendido com um objeto de roteamento explícito. Este objeto deve ser incluído somente em mensagens *PATH*, já que mensagens *RESV* sempre percorrem o mesmo caminho na direção oposta, não havendo mudanças no manuseio de mensagens *RESV.* Quando um objeto de roteamento explícito está presente em uma mensagem *PATH*, a interface de saída do roteamento padrão é ignorada e a mensagem *PATH* é enviada através da interface que leva ao endereço IP especificado no objeto. A RAPI (RSVP API) também foi modificada para permitir a definição de um caminho explícito através da identificação dos *hosts* (rota restrita) ou das redes (rota solta) que o compõem.

2. Distribuição de rótulos:

Para que o RSVP suporte a distribuição de rótulos, novos objetos RSVP se fazem necessários: o objeto de requisição de rótulo e o objeto de rótulo. O objeto de requisição de rótulo é incluído nas mensagens *PATH* e sinaliza que um rótulo deve ser alocado para esta requisição. No caso de *unicast*, o rótulo é alocado no nó de destino *downstream* e inserido num objeto de rótulo, que é então transportado no sentido *upstream* em uma mensagem *RESV* (alocação de rótulo no egresso). Este rótulo é então utilizado no *host upstream* como rótulo de saída na tabela de rótulos.

No caso de mensagens RSVP *multicast*, a distribuição de rótulos é executada de maneira diferente do caso de mensagens *unicast*. Se um sistema de borda recebe uma mensagem RSVP PATH com um endereço de destino *multicast*, ele inclui um objeto de rótulo em adição ao objeto de requisição de rótulo na mensagem *PATH* (alocação de rótulos no ingresso). O mesmo rótulo é distribuído a todos os roteadores MPLS no grupo *multicast*. Este rótulo é então devolvido nas mensagens *RESV* no sentido *upstream*.

Partindo da suposição de que a troca de rótulos, pelo menos no futuro próximo, será executada apenas em domínios intermediários, no núcleo (core) da rede, os roteadores do NIST *Switch* inserem automaticamente um objeto de requisição de rótulo nas mensagens *PATH*. Todos os objetos relacionados a rótulos são removidos nos pontos de egresso do domínio para que não se propaguem para sistemas que não sejam capazes de lidar com rótulos.

1.4 Proposta da PLEX MPLS

A plataforma para experiências com MPLS – **PLEX MPLS** – será composta de diversos componentes de *software* integrados. Permitirá que o usuário defina, execute, e analise configurações de Engenharia de Tráfego com MPLS em uma rede IP. Serão realizados **experimentos de validação em redes com tráfego real**, o que permite obter resultados mais realísticos (em comparação com simulação e emulação). Os principais objetivos da PLEX MPLS são:

- Estudo da tecnologia MPLS e suporte a outros trabalhos em andamento no contexto do grupo de estudos em MPLS do DCA. Mais especificamente, a implementação da PLEX fornece funcionalidades utilizadas no *broker* de políticas de TE ([SIQUEIRA]) desenvolvido em um dos trabalhos do grupo (NIST_interface, adição de mapeamentos, máscaras).
- Experimentos com engenharia de tráfego com MPLS: validação das tecnologias e conceitos relacionados
- Experimentos didáticos em disciplinas de laboratório de redes

Para atingir estes objetivos, a PLEX MPLS deve permitir ao usuário especificar rotas explícitas para LSPs, as características do tráfego e como este deve ser enviado através da rede (por exemplo, através de que LSPs). Adicionalmente, deve ser capaz de gerar os padrões de tráfego especificados e coletar as estatísticas relevantes ao experimento. Outro requisito para a PLEX MPLS é a transparência de implementação, ou seja, a troca de um de seus componentes de *software* por outro equivalente não deve gerar grandes alterações no código da plataforma. Esta transparência é atingida através da definição de APIs entre os componentes. Somente a implementação da interface para o componente de *software* específico deve ser afetada.

1.4.1 Leiaute da rede de testes

Todos os nós da rede serão baseados em PCs com até quatro NICs e interconectados através de um conjunto de *hubs* e *switches* Ethernet, bem como algumas conexões ponto-a-ponto utilizando cabos UTP do tipo *cross-connect*. Os nós fora do domínio MPLS serão responsáveis pela geração/recepção do tráfego eventualmente especificado no experimento. O domínio MPLS terá 4 pontos de entrada (LERs) e alguns nós internos (LSRs). LERs terão ao menos duas conexões a outros roteadores dentro do domínio MPLS e LSRs podem ter de duas a quatro conexões com outros roteadores. As conexões entre roteadores do domínio MPLS não são estáticas, sendo reconfiguradas baseado nas necessidades do experimento.

A PLEX MPLS será implementada e testada em uma rede com o leiaute lógico mostrado na Figura 3.

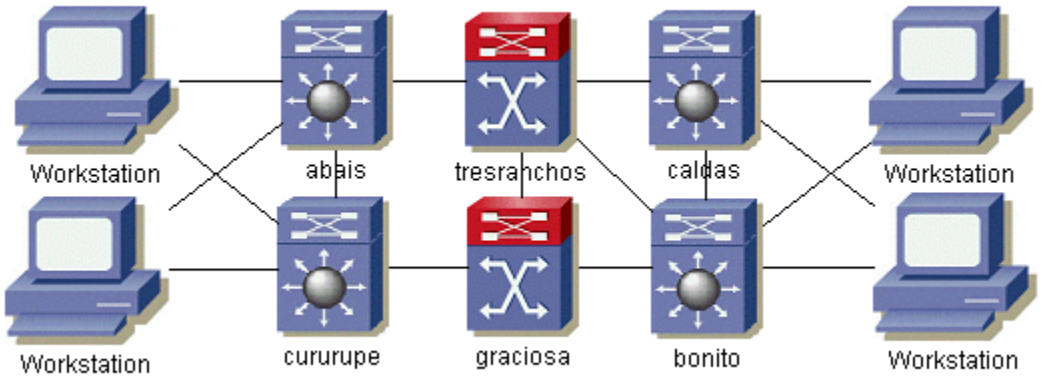


Figura 3: Leiaute da rede de testes

1.4.2 Componentes funcionais

Os principais elementos funcionais da **PLEX MPLS** são descritos a seguir juntamente com uma sugestão de implementação de código aberto quando existente:

MPLS: implementação MPLS capaz de estabelecer LSPs explicitamente roteados e mapear tráfego a esses LSPs conforme definido pelo usuário. Sugere-se o uso do pacote NIST *Switch*, que altera o kernel do sistema operacional FreeBSD para implementar o MPLS, bem como utiliza uma versão modificada do RSVP (seguindo a especificação do RSVP-TE, [RFC3209]) para distribuição de rótulos e reserva de recursos e uma versão modificada do ALTQ para implementar a garantia da QoS.

Geração de tráfego: o *software* para geração de tráfego deve ser capaz de gerar/receber tráfego de acordo com as características especificadas pelo usuário (diferentes protocolos, taxas de transmissão, distribuições de probabilidade, tempos de início/fim) e também coletar e registrar estatísticas a respeito deste tráfego. O software sugerido para geração de tráfego é o tg ([TG]), que é capaz de gerar tráfego TCP e UDP, altas taxas de transmissão, tem uma linguagem de *script* flexível para programar fluxos no decorrer do experimento e admite várias distribuições de probabilidade tanto para o intervalo entre pacotes, bem como para o tamanho dos mesmos.

Gerente PLEX: mostra em uma GUI a topologia da rede (lida de um arquivo de configuração) e permite que o usuário especifique LSPs explicitamente roteados com parâmetros de QoS associados. O usuário também pode definir o tráfego a ser gerado para o experimento (origem/destino, padrão de tráfego, etc.). É responsabilidade do gerente comunicar os agentes PLEX que tarefas devem executar. Alguns agentes PLEX são responsáveis por estabelecer os LSPs (agentes MPLS), outros pela geração/recepção de tráfego (agentes de tráfego). O módulo TE instancia políticas de TE nos LERs de entrada. O gerente também é responsável por gerenciar os LSPs de acordo com as ações do usuário, como criação/alteração/finalização de LSPs.

Agente MPLS: recebe do gerente as informações a respeito dos LSPs a serem estabelecidos. Interage com a implementação MPLS (NIST *Switch*) para estabelecer/finalizar LSPs (inclui uma biblioteca para se comunicar com o protocolo de sinalização do MPLS, por exemplo

RSVP-TE) e para mapear padrões de tráfego aos LSPs estabelecidos (chamadas de sistema).

Agente de Tráfego: recebe do gerente as informações a respeito do tráfego a ser gerado/recebido e interage com o software de geração de tráfego (tg) para efetivamente gerar/receber o tráfego e para coletar as estatísticas relevantes.

Módulo TE: permite particionar o tráfego entrante nos LERs e mapeá-lo aos LSPs estabelecidos de acordo com algum algoritmo ou política de engenharia do tráfego pré-definidos. Alguns exemplos são balanceamento de carga; priorização de tráfego através de seu mapeamento a LSPs com melhores recursos; ou até mesmo uma implementação de MATE ([MATE]).

Comunicação gerente-agente: o gerente invoca remotamente os métodos disponibilizados pelos agentes através de CORBA.

Considerando a arquitetura para TE em 3 camadas apresentada na Figura 1, os componentes da PLEX situam-se tanto na primeira quanto na terceira camadas. O Gerente PLEX lida com a configuração *offline* de Engenharia de Tráfego a ser aplicada à rede, realizada manualmente pelo administrador da mesma. Na camada mais baixa, encontram-se os componentes MPLS, ALTQ e Módulo TE. Estes são responsáveis por micro ações realizadas sobre o tráfego para implementar a configuração de TE especificada no Gerente.

1.4.3 Relação entre nós de rede e componentes funcionais

Os componentes funcionais estarão distribuídos pelos diferentes tipos de nós da rede:

- *Host*: executa o Agente de Tráfego e software de geração de tráfego
- LER: executa o Agente MPLS e o *software* MPLS, além dos componentes do *host*
- LSR: executa o *software* MPLS

O Gerente PLEX pode ser executado em qualquer nó da rede, inclusive fora do domínio MPLS, bastando ter a infra-estrutura CORBA instalada. Ele deve ser executado manualmente pelo usuário. Os outros componentes funcionais podem ser executados manualmente ou inicializados automaticamente por algum *script* durante o *boot.*

1.4.4 PLEX MPLS: alterações e extensões ao NIST Switch

As funcionalidades básicas fornecidas pelo NIST Switch, apesar de muito interessantes, não permitem que se implemente esquemas mais sofisticados de Engenharia de Tráfego. Em virtude desta limitação, a PLEX MPLS possuirá um módulo de TE que oferecerá as seguintes funcionalidades adicionais:

1. **Máscaras para agregação de tráfego:** O mecanismo de classificação do NIST Switch utiliza-se das informações de protocolo e de endereço de origem e destino para capturar nos LERs pacotes que não possuem label. As máscaras possibilitam agregação de tráfego, pois permitem ignorar parte(s) da informação de classificação para mapear os pacotes entrantes aos LSPs.
2. **Mapeamento de múltiplos fluxos de tráfego a LSPs:** Para permitir que diferentes fluxos de tráfego sejam transportados por um LSP, será implementado um mecanismo que

permite que haja múltiplos mapeamentos de tráfego a 1 LSP (com ou sem a agregação proporcionada pelas máscaras). Um mapeamento corresponde a uma FEC. Os mapeamentos permitem um grau ainda maior de agregação, bem como maior granularidade e controle sobre a agregação de tráfegos.

3. **Instanciação de LSPs paralelos para balanceamento de carga:** Nativamente o NIST Switch não permite a instanciação de LSPs paralelos, pois há colisão na tabela de labels devido à coincidência dos LERs de entrada e saída entre estes LSPs. A instanciação de LSPs paralelos permite tanto o balanceamento de carga, quanto a pré-instanciação de um LSP de backup para o caso de falhas.
4. **Balanceamento de carga:** Esquemas mais sofisticados de TE demandam algoritmos de balanceamento de carga para melhor aproveitamento da infra-estrutura de rede. O balanceamento de carga se torna mais útil à medida que há algoritmos mais inteligentes para a escolha do LSP a ser utilizado para transmissão de dados em determinado momento. O algoritmo básico para balanceamento de carga baseia-se em um round-robin proporcional a um fator especificado quando da criação dos LSPs relativo ao número de pacotes transmitido por cada LSP.

1.4.5 Panorama do Grupo de Trabalho em MPLS

Para melhor entender a proposta da PLEX MPLS, convém conhecer o contexto em que ela surgiu, o Grupo de Estudos em MPLS do DCA. O grupo era formado por mestrandos e doutorandos em Engenharia Elétrica e seu foco de interesse era o MPLS. Deste modo, procurou-se realizar trabalhos complementares em torno do tema em comum. A sinergia alcançada foi realmente proveitosa, sendo que alguns trabalhos se aproveitaram dos resultados uns dos outros e tiveram diversos pontos em comum. A seguir lista-se os trabalhos mais intimamente ligados à PLEX MPLS:

- Uma Arquitetura de Políticas Para Gerência de Redes MPLS ([SIQUEIRA])
- NSPLEX-Plataforma de Validação de Algoritmos de Engenharia de Tráfego em MPLS ([LIMA])
- MPLS-DS: Uma Plataforma para Validação de Políticas no Contexto das Redes MPLS/DiffServ ([GUILLEN])
- Desenvolvimento de um Bandwith Broker para a Arquitetura de Serviços Diferenciados ([COSTA])
- Algoritmos de Balanceamento de Carga para Tráfego Tipo Melhor Esforço em Redes IP/MPLS ([CAVALCA])

1.5 Síntese

Neste capítulo são apresentadas as necessidades que justificam a demanda por Engenharia de Tráfego no atual cenário de convergência das telecomunicações. É realizada também uma revisão dos principais conceitos de Engenharia de Tráfego e de MPLS. Em seguida há a descrição do projeto NIST Switch. Finalmente, é apresentada a proposta da PLEX MPLS e seus componentes funcionais.

2 Análise

Os principais aspectos da fase de **análise** são tratados neste capítulo. Especificamente, a análise de requisitos para a interface com o usuário do **Gerente** é realizada seguindo as técnicas descritas em [HIX93]. Os requisitos dos agentes também são analisados. Finalmente, alguns aspectos da comunicação entre os componentes da PLEX são evidenciados.

2.1 Gerente

O método discutido em [HIX93] para análise de sistemas foi parcialmente aplicado a fim de orientar o projeto da interface com o usuário e para compreender melhor os requisitos do **Gerente**. É importante observar que este método é excelente não apenas para análise/projeto de interfaces com o usuário, mas também para evidenciar os requisitos, restrições e funções do sistema como um todo. As atividades desenvolvidas são:

Análise de necessidade: determina se um sistema novo é, de fato, necessário e que necessidades básicas do usuário e do mercado ele satisfará. Se o sistema novo não puder substituir e melhorar sistemas existentes ou a maneira tradicional de lidar com o problema (por exemplo, manualmente), então não vale a pena desenvolvê-lo.

Análise de usuário: é uma caracterização do usuário (potencial) do sistema. Deve abranger desde o conhecimento técnico e habilidades necessárias ou desejáveis para operar o sistema até instrução geral. Esta atividade é de grande importância para o projeto da interface tanto para adaptá-la às limitações dos usuários quanto para aumentar sua eficiência. Idealmente, realiza-se levantamentos com usuários reais.

Análise de tarefas: é considerada a atividade mais importante do processo porque seu resultado é uma hierarquia de tarefas que define como os usuários usarão o sistema. As tarefas são descritas do ponto de vista do usuário, começando com a tarefa de nível mais elevado (utilizar o sistema). As tarefas são sucessivamente decompostas até chegar em tarefas atômicas (indivisíveis).

Análise funcional: descreve as funções internas do sistema que devem ser projetadas e implementadas para suportar as tarefas identificadas para o usuário.

2.1.1 Análise de necessidade

Os principais objetivos da PLEX MPLS são: (i) instanciar uma configuração de TE estática definida manualmente pelo administrador da rede através da interface gráfica do Gerente; (ii) funcionar em uma rede pequena e barata de alguns computadores interconectados por *switches* ou *hubs*; (ii) permitir que estudantes e profissionais que trabalham com MPLS configurem e executem experimentos de TE com MPLS usando pacotes de dados reais (diferentemente de simulação e emulação).

Para a execução da PLEX MPLS, supõe-se que tanto o *hardware* (gateways, roteadores, topologia, etc), quanto o *software* necessário (sistema operacional, plataforma MPLS, componentes RSVP e ALTQ, etc) estejam em conformidade com as especificações da PLEX.

Como a plataforma será utilizada por técnicos especializados, oferece recursos poderosos:

- Estabelecimento de LSPs (rota, tipo, requisitos de QoS, ...)
- Definição do padrão de tráfego a ser gerado (protocolo, taxas, distribuição probabilística, momento de início / final, ...)
- Mapeamento do tráfego aos LSPs (qual tráfego será encaminhado por qual(is) LSP(s))
- Execução de políticas de Engenharia de Tráfego (módulo TE)

2.1.2 Análise de usuário

Somente uma classe de usuários está sendo considerada para o sistema proposto: profissionais de informática que lidam com projeto e operação de redes. A Tabela 1 é uma suposição de características prováveis baseada em usuários potenciais do sistema.

Características	**Usuário (Engenheiro)**
Experiência computacional [1]	Usuário freqüente
Hardware típico	x86
Software típico	Variável
Experiência com sistemas similares	Não
Nível educacional	Universitário
Classe social	Média
Familiaridade com termos e conceitos [2]	Sim
Experiência com experimentos com redes (ou simulações)	Média/grande
Línguas estrangeiras	Inglês
Tempo para experimentos (h/dia)	2

Tabela 1: Características prováveis do usuário

Das características de usuário acima podemos concluir que a PLEX MPLS deve ser capaz de apresentar ao usuário a maior quantidade de informações possível e oferecer recursos poderosos (que são geralmente úteis, mas complexos), porque o usuário está em um nível avançado. É importante observar que as informações que estarão sempre visíveis devem ser de alto nível. Se o usuário quiser ver os detalhes de algum componente (por exemplo, um LSP), ele deve realizar alguma ação (como selecionar o LSP) para visualizar esta informação.

[1] a) O usuário novato é aquele que nunca usou um computador ou tem muito pouca experiência; b) O intermitente é aquele que não usa regularmente (portanto não tem muita experiência), mas tem maior facilidade para aprender; c) O usuário freqüente tem grande experiência no uso de computadores.
[2] Relacionados a MPLS, experimentos com redes, estatísticas, ...

2.1.3 Análise de tarefas

As tarefas são descritas primeiramente em linguagem natural. Após esta descrição é apresentado um diagrama de tarefas (simplificado). Cada nó representa uma tarefa na hierarquia. As setas que conectam os nós representam especializações. Ao contrário da maioria dos diagramas de estado, este não possui um ponto de saída. Isto ocorre quando o usuário finaliza a aplicação. As tarefas do engenheiro de tráfego são descritas abaixo.

Após inicializar a PLEX MPLS (o serviço CORBA e os agentes já devem ter sido inicializados, manualmente ou automaticamente durante o *boot* dos nós de rede), o usuário pode: definir nova configuração de TE; carregar/salvar uma configuração; visualizar os detalhes da configuração ou alterar a configuração. Os seguintes itens da configuração de TE podem ter suas propriedades definidas e/ou visualizadas: nós, LSRs, LERs, conexões, geradores do tráfego, LSPs (rota e parâmetros), política de TE e mapeamentos de tráfego.

A definição ou alteração de uma configuração de TE consiste nas seguintes tarefas: incluir/excluir LSPs, definir a rota dos LSPs; configurar os parâmetros dos LSPs (tipo, requisitos de QoS, ...); configurar a geração de tráfego (protocolo, taxas, momentos de início/final, ...) e definir mapeamentos de tráfego (a LSPs). Também é possível configurar o balanceamento de tráfego (definido por uma FEC) através de um conjunto de LSPs paralelos. As tarefas descritas acima podem ser repetidas até que a configuração desejada seja alcançada. Então o usuário pode realmente instanciar a configuração. A qualquer momento o usuário pode finalizar a PLEX MPLS sendo a configuração de TE, caso instanciada, automaticamente desfeita nos elementos da rede. Após a instanciação da configuração de TE, o usuário pode também retornar às atividades no nível 3.

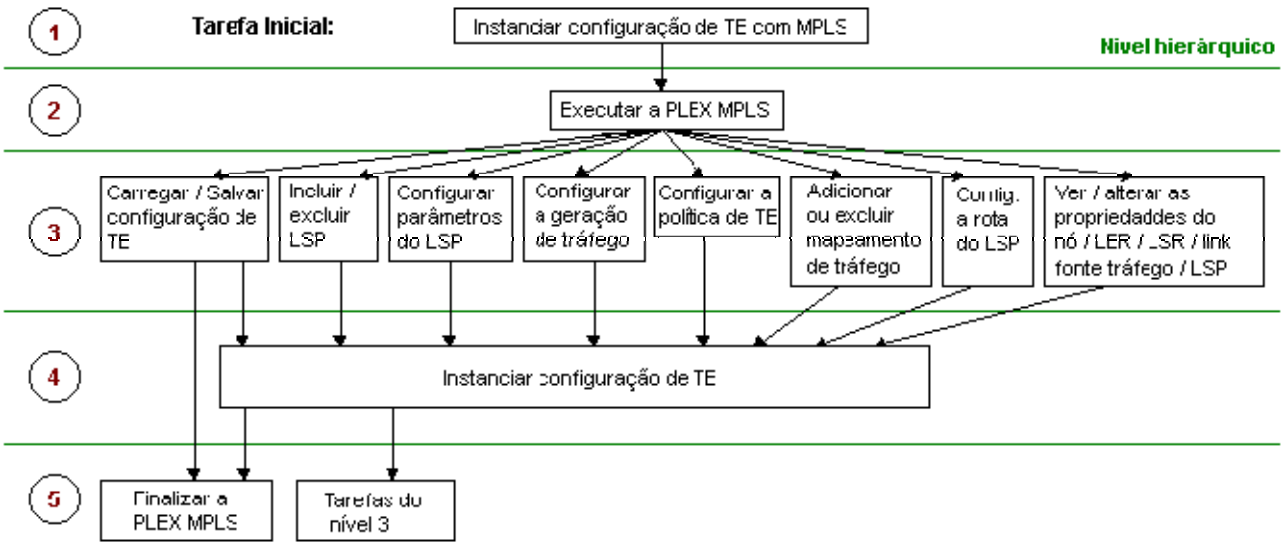


Figura 4: Diagrama de tarefas para o uso da PLEX MPLS

2.1.4 Análise funcional

As principais funcionalidades que a PLEX MPLS deve possuir são:

- Ler/escrever em um arquivo o estado atual do gerente (topologia de rede, configuração do experimento e resultados)
- Validação deste arquivo de configuração (topologia de rede e configuração do experimento)
- Aceitar entrada de dados do usuário (configuração do experimento)
- Capacidade de apresentar a configuração do experimento ao usuário (parte visualmente, parte textualmente)
- Comunicação entre os vários componentes da PLEX MPLS (protocolo de comunicações)
- Estabelecimento de LSPs com rota e os parâmetros selecionados pelo usuário
- Geração, transmissão e recepção de tráfego
- Mapeamento de tráfego a LSPs
- Coleta de estatísticas
- Execução de políticas de TE
- Capacidade de adaptar a configuração da rede (LSPs, geradores do tráfego...) depois de instanciada se o usuário alterar a configuração no gerente e instanciá-la novamente

2.2 Agentes

Além do gerente, a PLEX é composta também por dois agentes. Isto se deve à diversidade das funções a ser realizadas por estes agentes e também para alcançar maior modularização do *software*. Se houvesse somente um agente, seu código seria complexo demais. Além disso, todas as funcionalidades seriam desnecessariamente replicadas em todos os nós da rede. Como cada nó tem um papel específico, os agentes não precisam ser instanciados em todos os nós. Isto permite também um desempenho melhor porque em alguns nós haverá dois agentes executando as tarefas que um agente mais complexo teria que executar sozinho.

Entretanto, há algumas características comuns aos agentes. Ambos devem ser capazes de: (i) receber instruções do gerente; (ii) processar estas instruções; (iii) executar suas tarefas; (iv) informar ao gerente os resultados ou erros ocorridos; (v) registrar suas ações através de estruturas de dados internas.

A informação de estado que os agentes têm que manter representa a configuração do experimento relativa àquele agente. Por exemplo, o **agente MPLS** no LER X tem que registrar toda a informação relacionada aos LSPs que o gerente o instruiu a estabelecer.

Os agentes incorporaram código escrito em C (RSVP, chamadas de sistema, geração de tráfego, etc). A integração com código legado poderia ser realizada através de técnicas de intercomunicação de processos (IPC), mas isto resultaria em código ainda mais complexo. Além disso, os agentes devem ser orientados a objeto (é o paradigma da PLEX MPLS) e compatíveis com CORBA (veja adiante). Isto conduz naturalmente à escolha de C++ como linguagem de programação para os agentes (a escolha de C++ como linguagem de programação seria normalmente feita na fase do projeto, mas esta escolha foi realizada como uma restrição na fase de análise porque é a única compatível com o código legado em C e com CORBA).

2.3 Comunicação entre os componentes da PLEX MPLS

A comunicação entre os componentes do sistema é essencial na PLEX MPLS, especialmente entre o gerente e os agentes. Os principais requisitos são: (i) o gerente deve instruir os agentes sobre o que fazer; (ii) os agentes devem informar ao gerente os resultados de suas ações; (iii) os agentes têm que transmitir instruções e receber os resultados para/de outros componentes de *software* (MPLS e *software* para geração de tráfego). Este último tópico será tratado através de chamadas de sistema e da inclusão de bibliotecas.

A comunicação entre o gerente e os agentes exige a integração de aplicações escritas em linguagens de programação diferentes, tais como JAVA e C++. Um modo de realizar esta integração seria transmitir dados em um formato comum, tal como caracteres com 8 bits através de um *socket* TCP/IP. A desvantagem desta solução é que seria necessário projetar e implementar um protocolo de mensagens, mapear as várias estruturas de dados para este formato comum e implementar o *marshaling* (serialização) e *unmarshaling* dos tipos de dados.

É exatamente neste ponto que *CORBA* e sua linguagem de definição de interfaces (IDL) se encaixam. A IDL fornece um formato comum para representar dados que pode ser utilizado por aplicações que tenham mapeamentos para a IDL. A função do CORBA é fazer com que a invocação de um método em outras aplicações remotas seja transparente para a aplicação, bem como a passagem dos parâmetros e o retorno dos resultados. CORBA executa a serialização/desserialização de estruturas de dados, obtém referências a objetos remotos (serviço de nomes) e oferece várias outras funções convenientes (tais como os serviços de eventos e persistência de objetos).

Dada a grande simplificação no projeto da PLEX MPLS introduzida pela utilização de CORBA esta também será considerada uma restrição no projeto do sistema. Abaixo é apresentado um diagrama esquemático da arquitetura de comunicação para os componentes da PLEX MPLS.

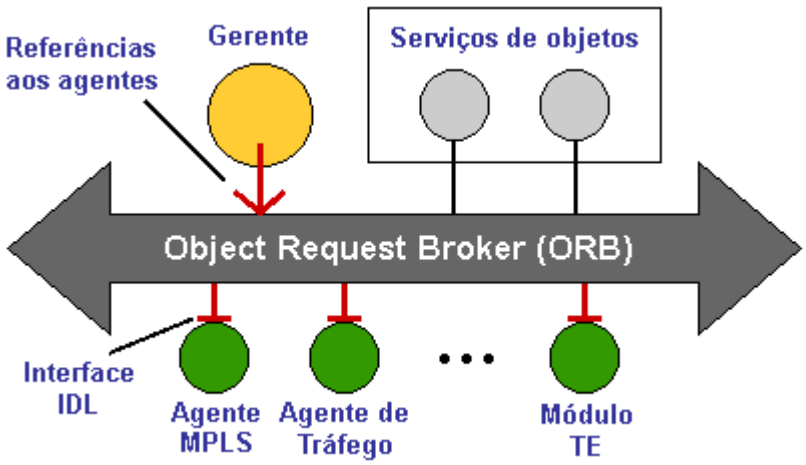


Figura 5: Arquitetura de comunicação para os componentes da PLEX MPLS

2.4 Síntese

Neste capítulo são apresentados os principais aspectos da fase de **análise**. A análise de necessidade racionaliza a real necessidade da aplicação, a de usuário busca categorizar o usuário típico do sistema, a de tarefas organiza as atividades pretendidas para o software e a funcional especifica as funcionalidades do sistema não relacionadas à interface com o usuário. Os requisitos dos agentes também são analisados. Finalmente, alguns aspectos da comunicação entre os componentes da PLEX são evidenciados.

3 Projeto

Neste capítulo, o projeto da PLEX MPLS é discutido. Primeiramente, a escolha de uma metodologia e ferramenta específicas é justificada. Em seguida, o projeto do gerente é apresentado juntamente com suas principais classes e o modelo de dados para a configuração do experimento. Finalmente, o projeto genérico dos agentes é explicado e o diagrama de classes do agente MPLS é apresentado.

3.1 Considerações gerais

À medida que a complexidade do desenvolvimento de *software* aumenta, torna-se cada vez mais importante especificar, visualizar e documentar o projeto, criando desse modo uma especificação que permita aos desenvolvedores compreender corretamente o sistema como um todo. A **modelagem visual** ([RUMBA2]) é uma maneira de pensar sobre problemas usando modelos organizados em torno de idéias do mundo real. A modelagem visual fomenta a melhor compreensão dos requisitos, projetos mais claros, sistemas mais fáceis de manter e ajuda a elaborar a documentação.

Três elementos são necessários para projetar e visualizar um sistema de software com êxito - uma notação (para padronizar a comunicação), uma ferramenta (para documentar eficientemente o projeto) e um processo (como usar a notação). Os mais populares e influentes de tais elementos atualmente são exatamente os utilizados neste projeto, respectivamente: a linguagem de modelagem unificada (UML) [RUMBA2], a ferramenta *Rational Rose* e o processo conhecido como *Rational Objectory Process*. UML e o processo supracitado são a convergência de três das metodologias e notações mais populares para modelagem visual e orientada a objetos, criadas por alguns dos "pais" da orientação a objetos: OMT [RUMBA1] (por Rumbaugh), Booch (por Booch) e OOSE (por Jacobson). Assim, parece ser uma escolha natural.

3.2 Modelo UML

A UML e seu processo são compostos por múltiplos diagramas e atividades, tais como: diagramas de caso, diagramas da classe, cenários, diagramas de seqüência, diagramas de colaboração, diagramas de estado, etc. Não seria prático nem proveitoso apresentar todos aqui. Somente os diagramas de classe serão apresentados juntamente com uma discussão de como o *software* é organizado.

PLEX MPLS é baseada em uma aplicação **gerente** escrita em JAVA, dois **agentes** escritos em C++ e alterações no *kernel* do sistema operacional (FreeBSD). O gerente deve ter uma interface gráfica com o usuário (GUI) através da qual pode-se visualizar a topologia da rede e definir uma configuração de TE e adicionalmente especificar fluxos de tráfego para a realização de um experimento. O gerente é responsável por transmitir esta configuração aos agentes e gerenciar sua execução. É responsabilidade dos agentes instanciar e executar a configuração informada pelo gerente. Temos abaixo uma visão de alto nível da arquitetura da PLEX MPLS quanto aos elementos que a compõem.

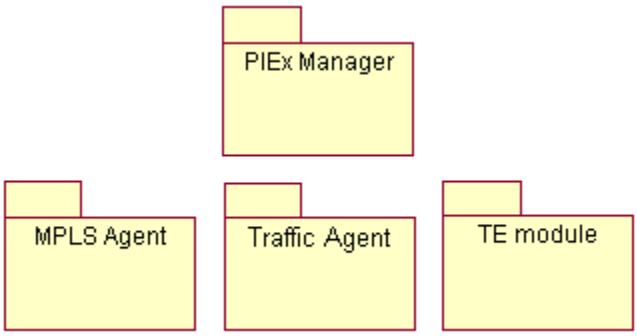


Figura 6: Visão de alto nível do projeto da PLEX MPLS

É importante observar que todas as estruturas de dados e parâmetros de métodos foram projetados para representar informações genéricas a respeito de MPLS, tráfego, estatísticas e assim por diante, de modo que a PLEX MPLS não seja restringida a ferramentas específicas. No entanto, como a implementação resultante utiliza softwares específicos para algumas funcionalidades, a especificação atual inclui a informação necessária às ferramentas utilizadas, como o **NIST Switch MPLS** e o **tg** (gerador do tráfego).

3.2.1 Gerente

O gerente é composto pela classe do gerente (***Manager***) e por diversas classes agrupadas em seis pacotes. O pacote ***graph*** (grafo) contém as classes que representam a topologia da rede (nós e conexões) e uma estrutura de dados para representar conjuntos abstratos de informação descritos em um formato chamado GML[3] (*Graph Modeling Language*). O pacote ***gui*** (interface gráfica com o usuário) contém todas as classes criadas para a interface com o usuário. Em ***algorithm*** (algoritmo) há classes para dispor automaticamente os nós de rede em um leiaute apropriado. Em ***config*** estão as classes necessárias para representar as informações de configuração do experimento. Algumas classes básicas para trabalhar com gráficos estão agrupadas no pacote ***util***. Finalmente, em ***plex***, estão as classes necessárias para a implementação da comunicação via CORBA. Os pacotes do gerente estão representados na Figura 7.

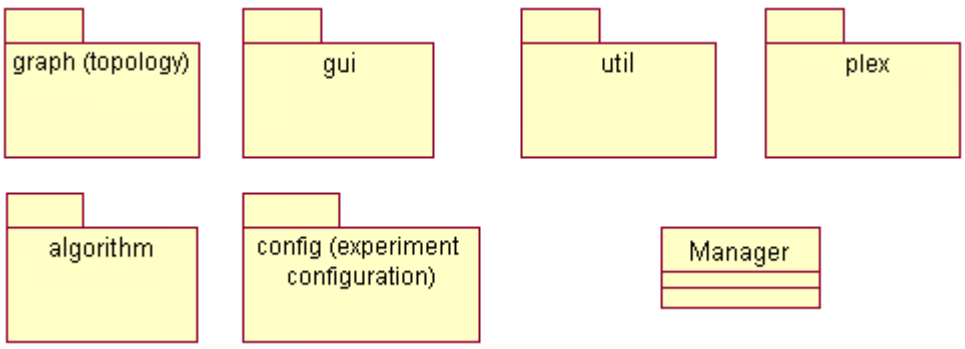


Figura 7: Pacotes que compõem o design do gerente

[3] http://www.infosun.fmi.uni-passau.de/Graphlet/GML/

O modelo conceitual proposto para representar a topologia da rede e a configuração do experimento é composto basicamente por nós, conexões e classes para guardar dados relativos à configuração, conforme mostrado na Figura 8. Há três tipos de nós: **Nodes** (nós de rede simples ou *hosts*), que não entendem MPLS, **LERs** (*Label Edge Routers*) e **LSRs** (*Label Switch Routers*), sendo os dois últimos derivados dos nós simples. Cada nó pode ter múltiplas **conexões** para outros nós, mas cada conexão conecta somente dois nós (conexões ponto a ponto). Um nó pode ter um gerador de tráfego associado a ele cuja configuração é armazenada na classe ***TrafficConfig***. Um **LSP** é composto por *2 a n* nós, sendo que os nós das extremidades devem ser LERs (distintos). O padrão de tráfego (FEC) que deve ser transmitido através do LSP é armazenado na classe ***MappingConfig***. As configurações de políticas genéricas são armazenadas em um objeto da classe ***GenPolConfig***. Finalmente, a configuração de todo o experimento é armazenada em estruturas de dados de um objeto da classe ***Config***. A implementação real da topologia e da configuração do experimento difere pouco do modelo em função de pequenas otimizações.

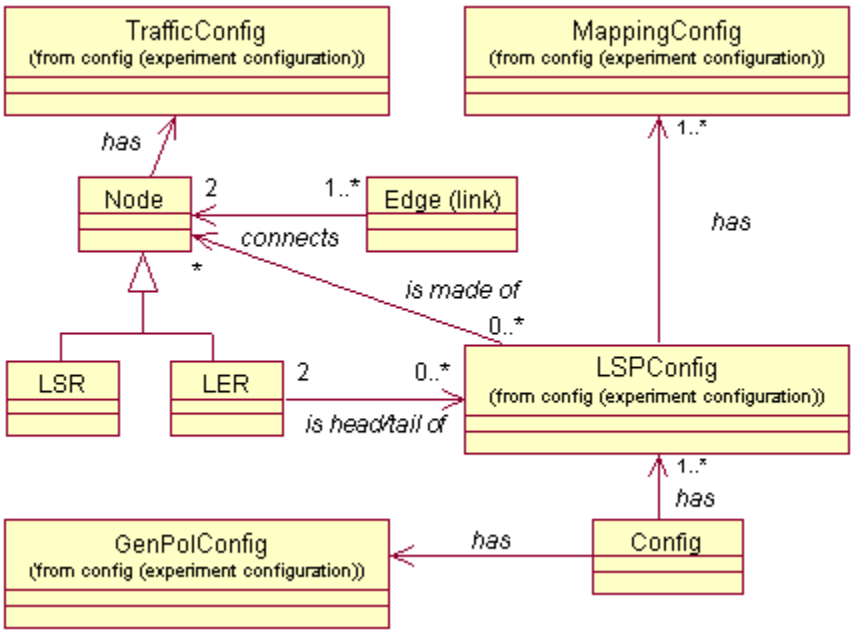


Figura 8: Design do modelo de topologia e configuração

3.2.2 Agente genérico

As principais funções dos agentes da PLEX MPLS são:

- Representar localmente as configurações de TE independentes de implementação transmitidas pelo Gerente
- Instanciar/executar as configurações locais ao comando do Gerente
- Oferecer transparência em relação à implementação local dos serviços via uma interface genérica (para que possam ser usadas outras implementações sem grandes esforços de implementação)

Dados os objetivos acima, o projeto dos agentes tenta incorporar ao máximo características de generalidade e transparência. Todos os agentes têm a mesma estrutura geral:

- **Envoltório (*wrapper*) CORBA** para oferecer seus métodos às aplicações remotas (o gerente) através de uma interface genérica;
- **Classe principal** do agente para gerenciar as informações de configuração;
- **Interface para o código específico** que implementa a funcionalidade do agente;
- **Classes** para guardar as **informações de configuração** do experimento independente da implementação local.

3.2.3 Agente MPLS

As principais classes do agente MPLS são mostradas na Figura 9. **AgentMPLS** é a classe que implementa o envoltório CORBA, oferecendo os métodos do agente a aplicações externas. ***AgentMPLS_impl*** é a classe principal e gerencia as informações de configuração enviadas pelo Gerente. ***NIST_interface*** é a interface para o *software* específico que implementa o MPLS, e provê um nível de indireção oferecendo métodos genéricos para estabelecer LSPs e adicionar mapeamentos de tráfego a LSPs. Estes métodos são mapeados aos mecanismos do *NIST Switch*. Se for desejado utilizar outra implementação de MPLS (por exemplo, o MPLS para Linux desenvolvido pelo grupo de estudos em MPLS do DCA) é necessário apenas instanciar outro componente de "*interface*" em vez de *NIST_interface*. As classes restantes são utilizadas para representar localmente as informações de configuração. ***List*** e ***node*** (nó da lista ligada) são usadas para formar listas ligadas de objetos das seguintes classes: ***lsp***, ***map*** e ***ER_hop***, que contêm a informação dos LSPs, mapeamentos de tráfego e os *hops* (nós) de uma rota explícita, respectivamente.

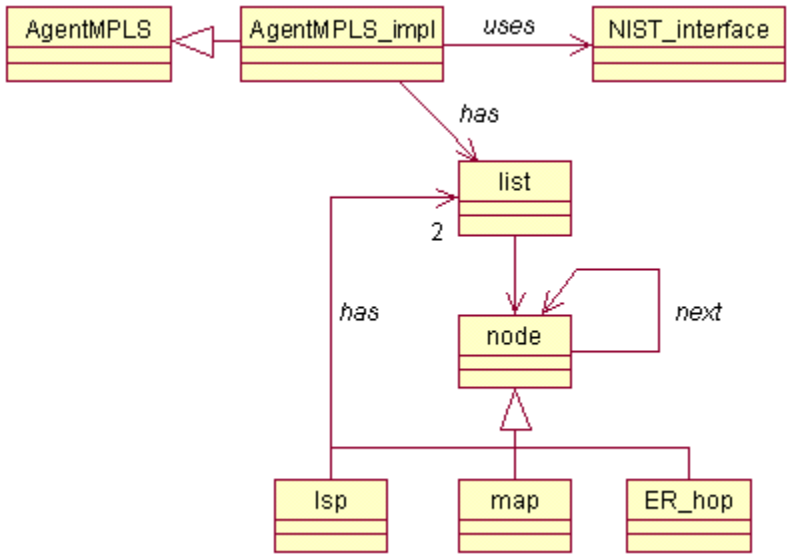


Figura 9: Classes para o agente MPLS

As operações genéricas oferecidas através interface CORBA são as seguintes:

- long create_lsp(lsp_args ARGMS) – instancia um LSP

- long finish_lsp(lsp_args ARGMS) – destrói um LSP
- long add_map(map_args ARGMS) – adiciona um mapeamento a um LSP
- long del_map(map_args ARGMS) – exclui um mapeamento de um LSP
- void callRESV(lsp_args ARGMS) – confirma a instanciação de um LSP em seu LER final
- string print_table() – retorna a tabela de labels do nó

3.2.4 Agente de Tráfego

As principais classes do agente de Tráfego são mostradas na Figura 10. ***AgentTraffic*** é a classe que implementa o envoltório CORBA, oferecendo os métodos do agente a aplicações externas. ***AgentTraffic_impl*** é a classe principal e gerencia as informações de configuração enviadas pelo Gerente. ***tg_interface*** é a interface para o *software* específico que implementa a geração de tráfego, e provê um nível de indireção oferecendo métodos genéricos para configurar o gerador e gerar o tráfego. Estes métodos são mapeados aos mecanismos do *tg*. Se for desejado utilizar outro gerador de tráfego (por exemplo, o *Netperf* ou o ttcp) é necessário apenas instanciar outro componente de "*interface*" (ex: *Netperf_interface*) em vez de *tg_interface*. A classe ***tg*** é derivada de ***node*** para possibilitar a instanciação de múltiplos geradores de tráfego (que são armazenados em uma lista ligada). A classe ***cmd_line_args*** é utilizada para representar localmente as informações de configuração (*script* de geração de tráfego).

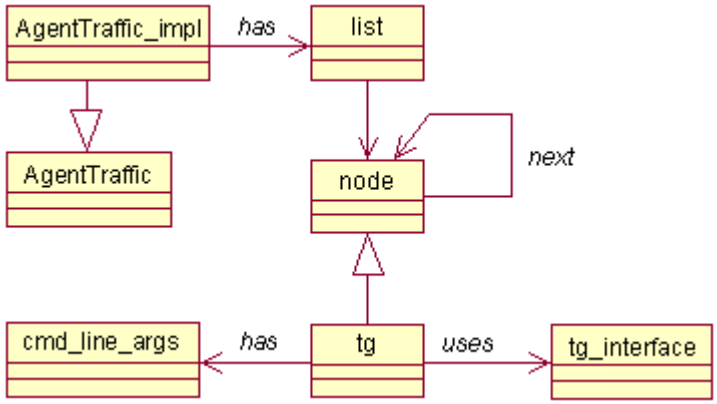


Figura 10: Classes para o agente de tráfego

As operações genéricas oferecidas através interface CORBA são as seguintes:

- long config(long tgid, const char* conf) – recebe a configuração para o gerador de tráfego
- void start(long tgid, long argc, const char* argv) – inicia a geração de tráfego
- long stop(long tgid) – finaliza a geração de tráfego
- string get_results(long tgid) – retorna os resultados da geração de tráfego

Obs.: Um arquivo de log da geração de tráfego é criado, mas os dados são armazenados em um formato proprietário ao qual não foi possível ter acesso, portanto a função de retorno dos resultados – *get_results*() – apesar de ter sido planejada, não foi implementada. O arquivo de log é analisado com uma ferramenta que acompanha o gerador de tráfego.

3.2.5 Módulo TE

Conforme enunciado na seção 1.4.4, a PLEX MPLS estende o NIST Switch oferecendo as seguintes funcionalidades adicionais:

- Máscaras para agregação de tráfego
- Mapeamento de múltiplos fluxos de tráfego a LSPs
- Instanciação de LSPs paralelos para balanceamento de carga
- Implementação de um algoritmo para balanceamento de carga proporcional ao fator de cada LSP (todos os fatores para um conjunto de LSPs paralelos devem totalizar 100%)

Estas funcionalidades, em conjunto com o mecanismo para estabelecimento de LSP oferecido pelo RSVP, assemelham-se muito com o conceito de *Soft Permanent Connection* [MCGUIRE]. Este conceito surgiu mais recentemente com as pesquisas para uso de MPLS em redes óticas.

3.2.5.1 Agregação de tráfego e mapeamento de fluxos a LSPs

O mecanismo de classificação do NIST *Switch* utiliza-se das informações de protocolo e de endereço de origem e destino para capturar nos LERs pacotes que não possuem label. Como essas informações constituem-se somente dos **endereços IP dos nós de rede inicial e final do LSP** (definido pelo RSVP no momento do estabelecimento do LSP), além de portas de origem e destino e protocolo de transmissão informados na configuração do LSP no Manager, a gama de tráfegos que podem ser transmitidos pelos LSPs fica muito limitada. Um exemplo típico é que somente os pacotes originados em 120.1.40.7:3030, destinados a 120.1.40.2:4000 no protocolo 21 serão transmitidos através de determinado LSP. Para contornar essa limitação, o NIST *Switch* utiliza máscaras para aumentar a abrangência do mecanismo de classificação.

As máscaras possibilitam agregação de tráfego, pois permitem ignorar parte(s) da informação de classificação para mapear os pacotes entrantes aos LSPs. Em tese, seria possível ignorar totalmente as informações, fazendo com que todos os pacotes fossem mapeados a um LSP (agregação máxima). Como esse mecanismo de agregação não estava funcionando, o grupo de estudos em MPLS do DCA implementou-o corretamente alterando o código do NIST *Switch*. Pode-se agregar tráfego utilizando-se máscaras tanto nos endereços IP, quanto nas portas e no protocolo em qualquer combinação desejada:

- Endereço IP: IP completo, sub-rede ou nada (o valor 0 no endereço IP representa uma sub-rede). Ex.: 192.168.0.0 representa qualquer endereço IP da sub-rede 192.168.
- Porta: o valor 0 corresponde a qualquer porta
- Protocolo: o valor 0 corresponde a qualquer protocolo

Adicionalmente, como parte dos trabalhos da PLEX, foi implementado um mecanismo que permite que haja múltiplos mapeamentos de tráfego a 1 LSP (com ou sem a agregação proporcionada pelas máscaras). Um mapeamento corresponde a uma FEC. Os mapeamentos permitem um grau ainda maior de agregação, bem como maior granularidade e controle sobre a agregação de tráfegos. Para uma explicação mais detalhada do mecanismo de funcionamento dos mapeamentos, veja o item 4.5.1. Deste modo é possível direcionar diferentes fluxos de tráfego através de um LSP em comum e posteriormente alterar essa agregação sem ter que finalizar o LSP. Portanto, constam da tabela de labels, além das informações dos LSPs

estabelecidos, as informações dos mapeamentos de tráfego adicionados para mapear diferentes FECs aos LSPs.

3.2.5.2 LSPs paralelos e balanceamento de carga

A instanciação de LSPs paralelos permite tanto o balanceamento de carga, quanto a pré-instanciação de um LSP de *backup* para o caso de falhas. Para tanto, é necessário um mecanismo para acomodar múltiplas entradas com informações quase iguais na tabela de labels.

O balanceamento de carga se torna mais útil à medida que há algoritmos mais inteligentes para a escolha do LSP a ser utilizado para transmissão de dados em determinado momento. O **algoritmo básico** para balanceamento de carga baseia-se em um *round-robin* proporcional a um fator especificado quando da criação dos LSPs relativo ao número de pacotes transmitido por cada LSP. Por exemplo, pode-se instanciar 3 LSPs com fatores de transmissão de pacotes de 30% pelo LSP1, 20% pelo LSP2 e 50% pelo LSP3. Um exemplo mais sofisticado seria a implementação do MATE ([MATE]).

Para implementar as funcionalidades supracitadas é necessário efetuar alterações no *kernel*, mais especificamente, na estrutura da tabela de rótulos, nas funções de adição e remoção de rótulos e na função de encaminhamento de pacotes.

A seleção do LSP pelo qual um pacote sem rótulo será encaminhado é realizada na função de transmissão do pacote IP – `ip_output()` – através da invocação da função `lt_find_next_dest()`, uma das alterações introduzidas no kernel do FreeBSD pelo NIST *Switch*. A seleção de um dos LSPs paralelos pode ser realizada tanto na primeira função quanto na segunda.

Caso seja realizada em `ip_output()`, é necessário invocar `lt_find_next_dest()` até que todos os LSPs paralelos (relacionados ao mesmo fluxo de tráfego) sejam retornados por essa função ara então aplicar o algoritmo de seleção. Caso o tratamento seja realizado em `lt_find_next_dest()`, sendo então já retornado o LSP selecionado, o algoritmo de seleção já teria acesso local aos LSPs paralelos. As vantagens desta segunda abordagem são evitar múltiplas invocações da função `lt_find_next_dest()`, o que se reflete na performance, e que essa funcionalidade seria transparente ao mecanismo atual de encaminhamento de pacotes, exigindo alterações mínimas em `ip_output()`. Devido a estes motivos, a segunda abordagem será utilizada na implementação.

Adicionalmente, é necessária e extensão da tabela de rótulos em 1 campo, que armazenará o fator de balanceamento de tráfego de cada LSP. As estatísticas de transmissão de pacotes por cada LSP não deverão ser afetadas, pois serão realizadas normalmente e registradas na estrutura retornada por `lt_find_next_dest()`, existindo uma estrutura para cada LSP paralelo.

3.3 Integração via CORBA

A principal preocupação no projeto da integração via CORBA é a generalidade das funções a serem oferecidas pelos agentes, de modo que a integração seja independente de implementação. Para tanto, as informações transmitidas nas chamadas CORBA devem representar informações genéricas relativas a MPLS e geração de tráfego, o que é refletido nos parâmetros das funções:

LSP	Tipo	Tipo
id	int	ID do LSP
src	string	IP nó origem
dst	string	IP nó destino
sport	int	Porta origem
dport	int	Porta destino
isTunnel	boolean	Tipo de LSP (túnel ou não)
inclLabelReqObj	boolean	Incluir objeto requisição rótulo
prot	int	Protocolo
num_ER_hops	int	Quantidade nós intermediários
rate	int	Taxa média (token bucket)
depth	int	Profundidade do balde
peak	int	Taxa de pico
exroute	ERH_seq	Lista de nós intermediários
label_in	int	Rótulo de entrada
label_out	int	Rótulo de saída
nxt_hop	string	IP do próximo nó
out_if	string	Interface de saída

ERH	Tipo	Função
flag	boolean	Tipo de rota (strict/loose)
hop	string	IP do nó (X.X.X.X)
prefix	int	Máscara (subrede - 8 a 32)

Tabela 2: Parâmetros para funções relativas a LSPs

Mapping	Tipo	Função
id	int	ID do mapeamento
lsp_id	int	ID do LSP
label_in	int	Rótulo de entrada
label_out	int	Rótulo de saída
src	string	IP nó origem
dst	string	IP nó destino
sport	int	Porta origem
dport	int	Porta destino
prot	int	Protocolo
nxt_hop	string	IP do próximo nó
out_if	string	Interface de saída
bal	int	Fator de balanceamento de carga

Tabela 3: Parâmetros para funções relativas a mapeamentos

Traffic	Tipo	Função
tgid	int	ID do gerador de tráfego
conf	string	Informações de configuração
argv	string	Argumentos adicionais p/ gerador

Tabela 4: Parâmetros para funções relativas a geração de tráfego

3.4 Síntese

Neste capítulo são apresentados os principais aspectos da fase de **projeto**, que visa a produzir uma solução conceitual para o problema. É apresentada a modelagem em UML do software e seus componentes (Gerente, Agente MPLS, Agente de Tráfego e Módulo TE), bem como uma descrição do modelo. São explicados também os mecanismos de funcionamento das alterações realizadas no NIST Switch para incrementar suas funcionalidades e permitir esquemas mais sofisticados de TE:

- Agregação de Tráfego
- Mapeamento de fluxos a LSPs
- Balanceamento de carga com LSPs paralelos

4 Implementação

Neste capítulo são discutidos aspectos da implementação da PLEX MPLS. Primeiramente, é oferecida uma visão geral do ambiente de implementação. A seguir, é apresentada a implementação do gerente, bem como algumas de suas telas. Também são discutidas as implementações do agente MPLS e do agente de tráfego. Finalmente, a implementação do módulo TE é apresentada.

4.1 Visão geral

PLEX MPLS é composta por vários elementos integrados: gerente (implementado em JAVA), agentes (C/C++), *software* MPLS (NIST *Switch*, em C), gerador de tráfego (tg, em C) e módulo TE (alterações no kernel, em C). O NIST *Switch* executa no sistema operacional FreeBSD 3.3. Conforme explicado na seção 2.2 (Agentes), a linguagem C++ foi escolhida para os agentes por prover integração com código legado em C e por possibilitar o mapeamento da IDL da arquitetura CORBA. CORBA é utilizada para integrar este sistema distribuído e multi-linguagem. Os compiladores utilizados foram gcc e g++.

4.2 Gerente

Conforme discutido na seção 2.1.3 (Análise de tarefas), o gerente deve possibilitar ao usuário visualizar a topologia da rede, definir uma configuração de TE, definir fluxos de tráfego para um experimento, instanciar a configuração, gerar o tráfego e salvar/carregar as configurações para/de um arquivo. A Figura 11 mostra a tela principal do gerente onde a topologia da rede e informações associadas são visualizadas.

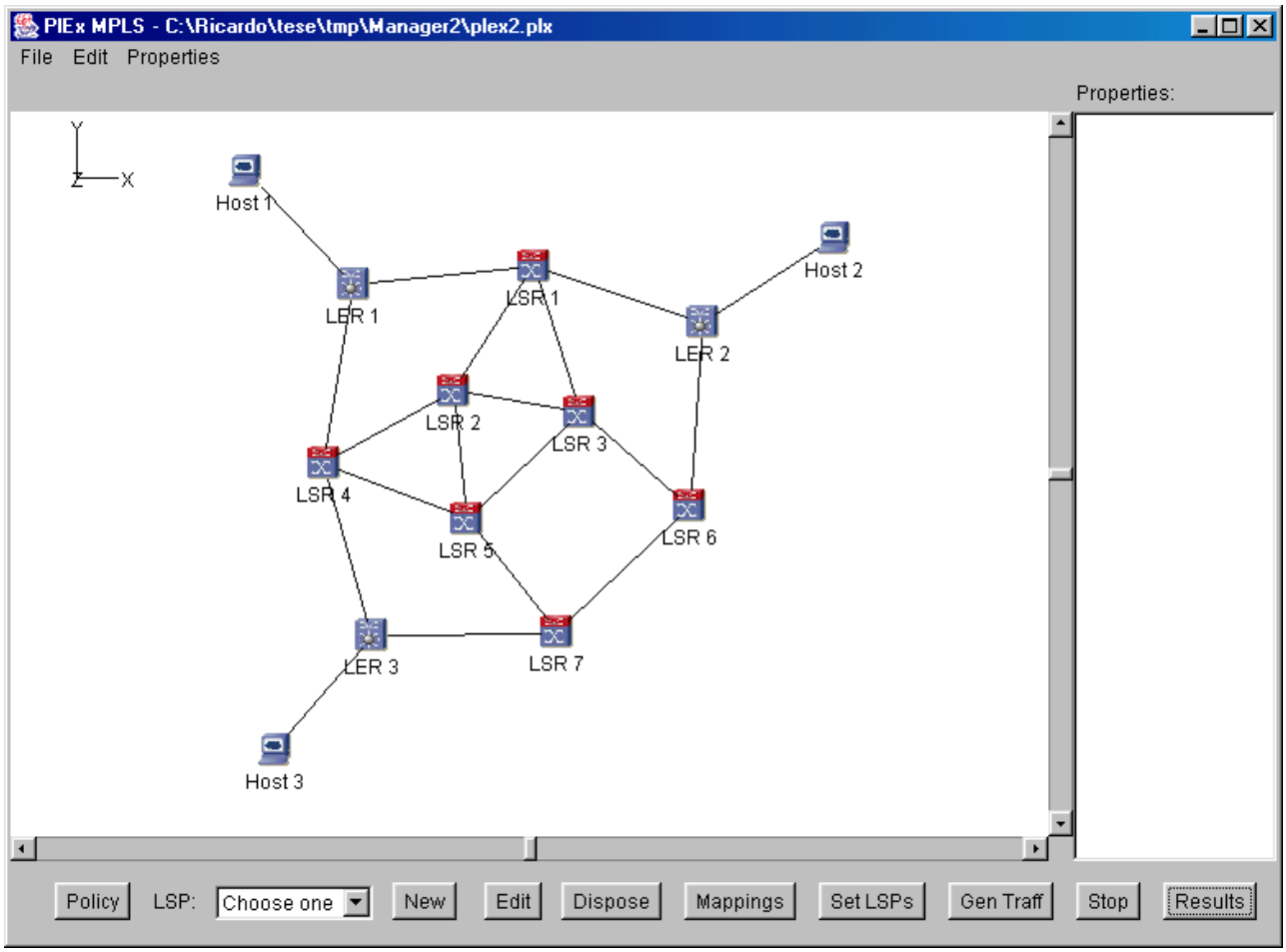


Figura 11: Janela principal da interface gráfica do gerente da PLEX MPLS

O botão ***Policy*** dá acesso a um diálogo em que é possível definir políticas de engenharia de tráfego, mas a única política implementada foi relativa à destruição de LSPs – se um LSP instanciado cuja configuração foi alterada deve ser destruído antes ou depois (*default*) do LSP com a nova configuração ser instanciado. Ao clicar em ***New***, o usuário tem acesso à caixa de diálogo para gerenciamento de LSPs. Devem ser informados: nome do LSP (identificador), número do protocolo (do tráfego que passará por ele), se o LSP será do tipo túnel (conceito do NIST *Switch*), número das portas de entrada e saída (portas em que o tráfego é gerado/recebido), LER de entrada, rota explícita e LER de saída. O ALTQ utiliza um algoritmo do tipo *token bucket* para o controle da banda utilizada, portanto deve-se informar a banda requerida, a taxa de pico e a profundidade do balde. Durante o processo de definição da rota explícita, o caminho selecionado e as próximas opções possíveis são indicados nas caixas de diálogo e também destacados com outras cores na janela principal (o caminho escolhido em verde e as possíveis opções de link em amarelo). Há um algoritmo que calcula a rota e não permite a configuração de LSPs cíclicos. A Figura 12 mostra a caixa de diálogo para gerenciamento de LSPs. Ao clicar em ***Edit***, o usuário tem acesso a uma caixa de diálogo muito parecida com a apresentada na Figura 12, onde é possível alterar as configurações do LSP selecionado, inclusive sua rota.

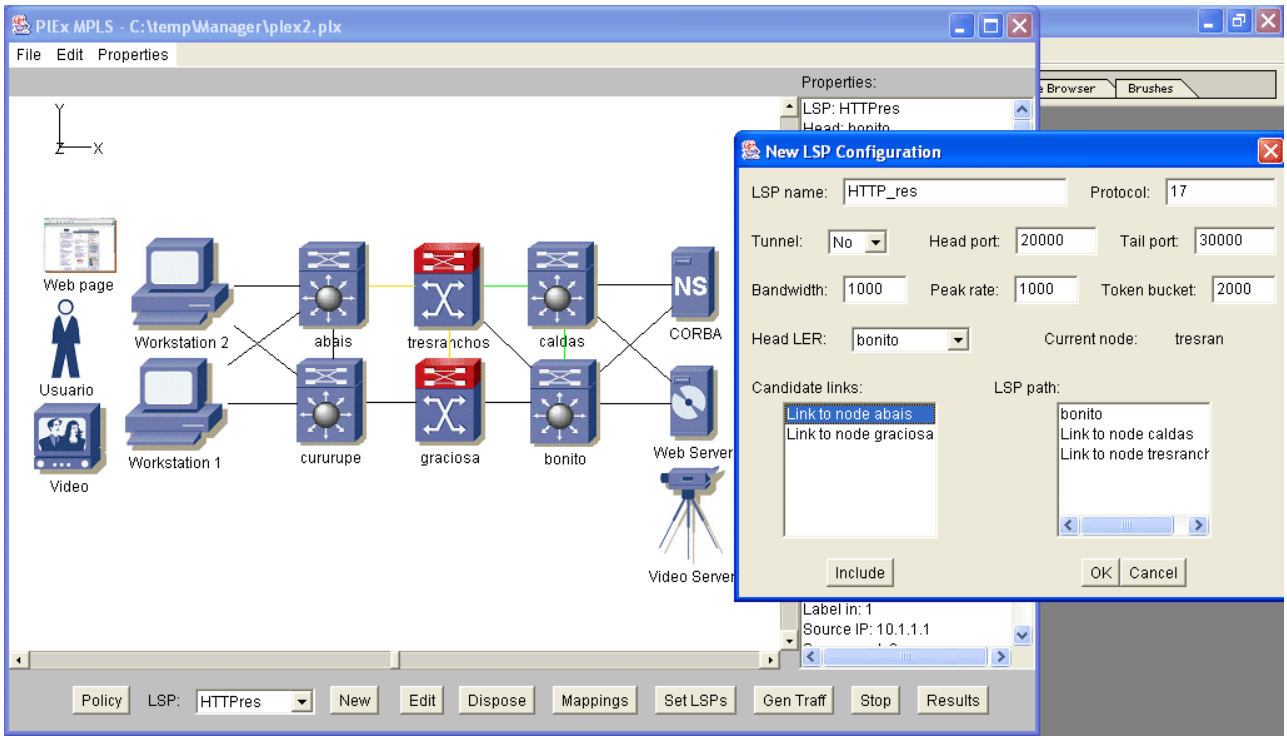


Figura 12: Diálogo para criação de um LSP com a indicação da rota sendo escolhida

A cada LSP é possível associar diversos **mapeamentos de tráfego** (botão ***Mappings***). A caixa de diálogo para gerenciamento de mapeamentos é mostrada na Figura 13. O usuário deve informar: nome do mapeamento (identificador); número do protocolo; portas e endereços IP de origem e destino; um número inteiro para o campo CoS do pacote e o percentual do tráfego a ser mapeado para o LSP em questão (caso existam LSP paralelos para balanceamento de carga). O botão ***Dispose*** permite apagar mapeamentos.

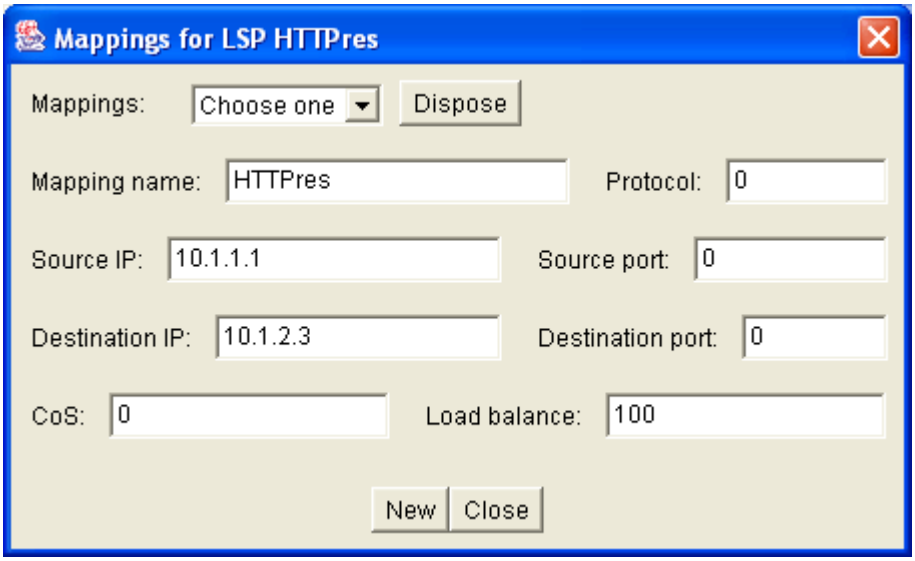


Figura 13: Diálogo para definir os mapeamentos de tráfego a 1 LSP

Retornando à janela principal do gerente, o botão ***Dispose*** permite apagar o LSP selecionado. Ao clicar em ***SetLSPs*** os LSPs configurados são instanciados na rede através de chamadas CORBA aos agentes MPLS. O botão ***Gen Traff*** ativa a geração de tráfego através de chamadas CORBA aos agentes MPLS. O botão ***Stop*** finaliza o experimento. Quando se finaliza o gerente, seja através do menu File ou do botão do Windows (canto superior direito da janela), todas as configurações do experimento porventura instanciadas na rede são desfeitas.

O lado direito da janela principal apresenta um painel onde se pode visualizar informações a respeito do elemento selecionado (caso haja algum), seja um nó de rede, uma conexão ou um LSP. A Figura 14 mostra as informações exibidas para um nó e para um LSP.

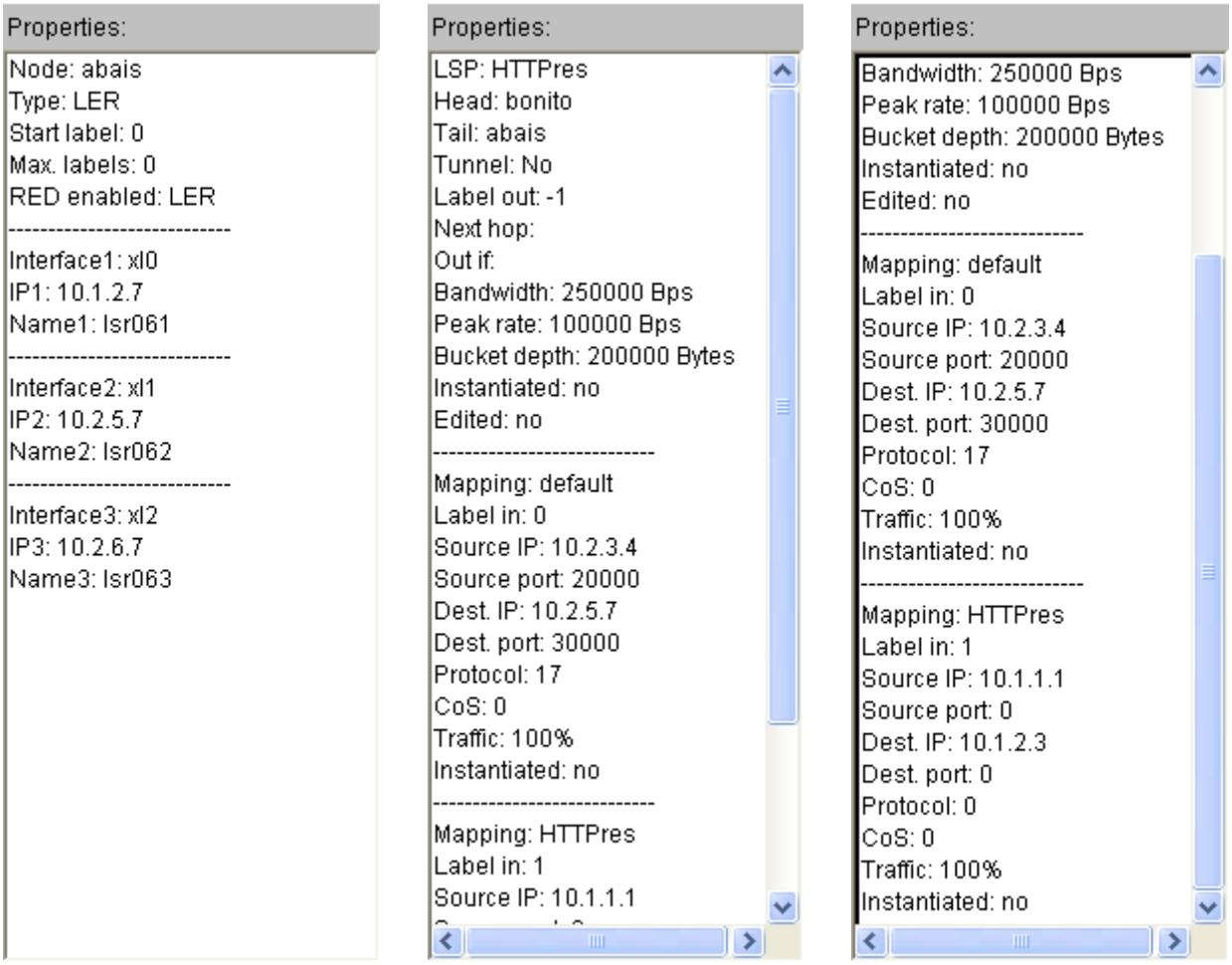


Figura 14: Painel de informações da janela principal do gerente

Um duplo clique nos elementos da topologia da rede (nós e conexões) abre uma caixa de diálogo em que é possível configurar algumas de suas propriedades. A Figura 15 mostra a caixa de diálogo, onde é possível configurar informações do nó, como nome, a faixa de rótulos MPLS, o algoritmo de descarte utilizado e o script para o gerador de tráfego a ser executado no nó.

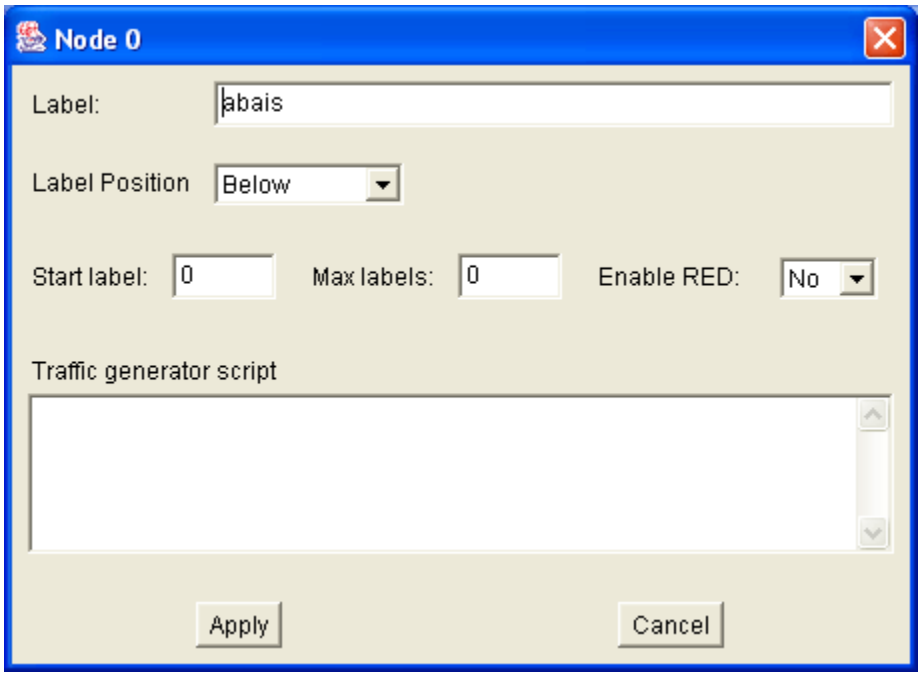


Figura 15: Caixa de diálogo de configuração das propriedades do nó de rede

4.3 Agente MPLS

Os agentes MPLS são executados em todos os roteadores do domínio MPLS (tanto LSRs, quanto LERs). O agente foi implementado em C++ incorporando código legado em C relativo às funcionalidades providas pelo RSVP-TE e pelo MPLS. O agente oferece métodos públicos mapeados para IDL e disponíveis ao gerente para instanciação da configuração definida.

4.3.1 Arquitetura do agente

Como se pode observar na Figura 16, o agente possui um *wrapper* em C++ que contém todas as funcionalidades referentes a CORBA. Este wrapper é composto de 3 classes geradas automaticamente pelo "*ORBacus IDL-to-C++ Translator*" a partir da API denifida em IDL para o agente. A classe **AgentMPLS_impl** contém os métodos definidos na API implementados em C++, que são invocados quando algum método do agente é invocado via CORBA.

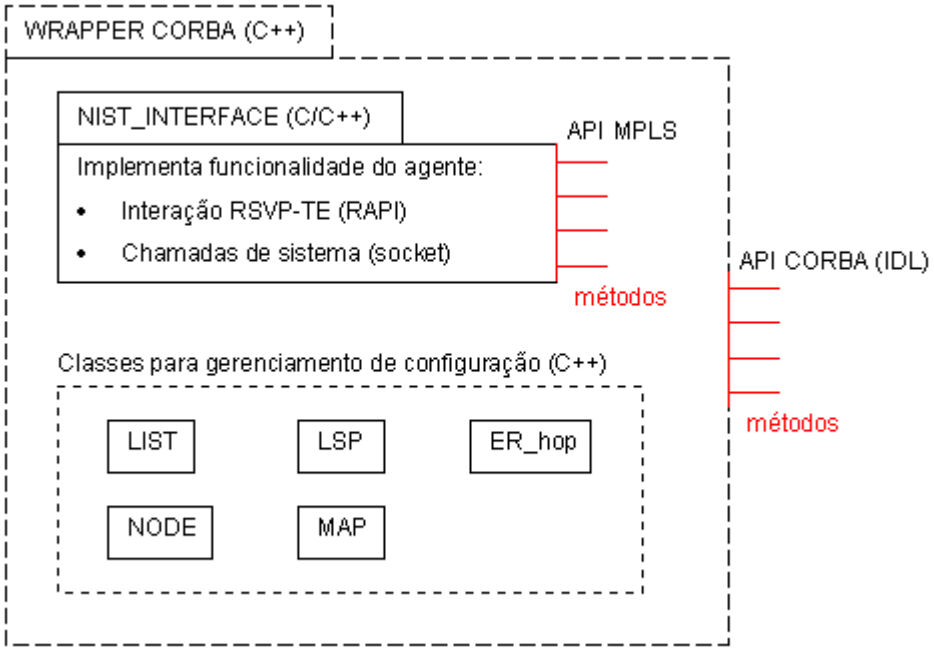


Figura 16: Arquitetura do agente MPLS

Para atender ao requisito de transparência em relação à implementação MPLS utilizada, todas as funcionalidades relativas ao MPLS e dependentes da implementação ficam encapsuladas em uma classe denominada **NIST_*interface***, que é instanciada pela classe AgentMPLS_impl. A classe NIST_*interface* oferece métodos com assinaturas genéricas para MPLS (assim como o *wrapper* e a API em IDL), sendo que a implementação específica para o NIST *Switch* se encontra dentro destes métodos.

Adicionalmente, a configuração de TE definida no gerente relativa àquele agente (nó de rede) é armazenada em objetos das classes: **list**, **lsp**, **map** e **ER_hop** (sendo as 3 últimas derivadas de **node**). Estas classes guardam as configurações relativas à lista de LSPs originados no LER em que o agente se encontra e a lista de mapeamentos de tráfego associada a cada LSP. Cada LSP pode ter uma rota explícita composta de uma lista de ER_hops.

4.3.2 RSVP: distribuição de rótulos e reserva de recursos

O protocolo utilizado para distribuição de rótulos no NIST *Switch* é uma versão alterada do RSVP denominada **RSVP-TE**. Um melhor entendimento do protocolo RSVP pode ser obtido em [RFC3209]. O RSVP-TE é capaz de distribuir os rótulos necessários ao estabelecimento de um LSP, bem como fazê-lo ao longo de uma rota explícita, daí a extensão TE. Adicionalmente, o RSVP é utilizado para reservar recursos de banda (com parâmetros de *token bucket*) para o LSP ao longo de sua rota, de modo a garantir os requisitos de QoS definidos no gerente. As principais alterações realizadas foram:

- Capacidade de seguir uma rota explícita em vez da rota convencional determinada pelo roteamento. Para isto o RSVP-TE possui um objeto (EXPLICIT_ROUTE) descrevendo esta rota.

- Distribuição de rótulos MPLS (objetos LABEL_REQUEST e LABEL).
- Sinalização (sessão) ocorre entre um par de LERs em vez de um par de *hosts*, o que diminui consideravelmente o estado a ser mantido nos LSRs.

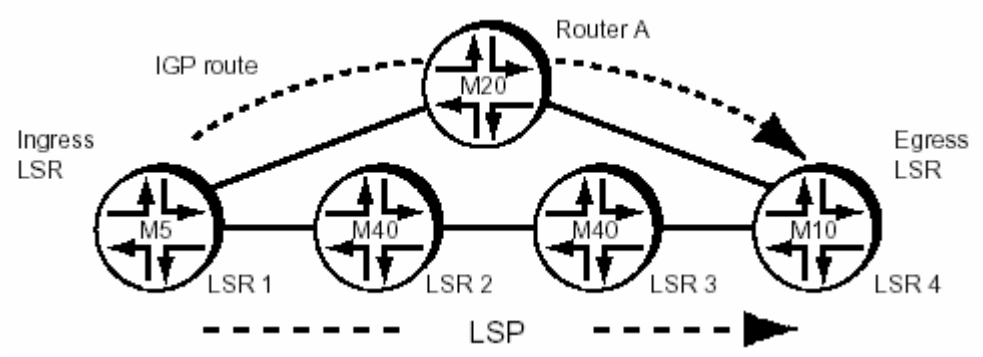


Figura 17: Processo de estabelecimento de um LSP com o RSVP-TE

O processo de estabelecimento de um LSP (de LSR1 para LSR4) pelo RSVP-TE está ilustrado na Figura 17. O tráfego normalmente seria encaminhado pelo *Router A*, mas o LSP é instanciado através dos LSRs 2 e 3.

1. O LSR1 inicia o estabelecimento do LSP enviando uma mensagem *PATH* para o LSR 4 seguindo os procedimentos normais do RSVP. A mensagem contém um objeto EXPLICIT_ROUTE indicando o caminho desejado, um objeto LABEL_REQUEST solicitando um rótulo e opcionalmente outros objetos. São também incluídos objetos do RSVP normal como SENDER_TEMPLATE, que identifica o remetente e SENDER_TSPEC, que caracteriza o tráfego a ser encaminhado pelo LSP.

2. Os LSRs intermediários verificam a possibilidade de estabelecimento do LSP, salvam algumas informações de estado e encaminham a mensagem pela rota explícita.

3. O LSR 4 recebe a mensagem *PATH* e caso tenha condições de estabelecer o LSP, envia uma mensagem *RESV* para o LSR1. Caso contrário, envia uma mensagem de erro revogando o estado salvo nos LSRs anteriores. Um rótulo com valor 0 (LER de egresso) é alocado e um objeto LABEL é incluído na mensagem *RESV*, bem como objetos RECEIVER_TSPEC e RSPEC para reserva dos recursos necessários.

4. Os LSRs intermediários guardam o rótulo recebido do LSR anterior, reservam os recursos necessários, geram um rótulo que esteja livre em sua faixa de rótulos e encaminham a mensagem *RESV* adiante.

5. O LSR1 recebe a mensagem, guarda o rótulo recebido para uso nos pacotes entrantes e reserva os recursos necessários. O LSP está estabelecido.

O AgentMPLS encapsula código do NIST *Switch* e do RSVP-TE na classe NIST_*interface* justamente para realizar essa interação com o RSVP. Ao receber as informações de configuração de um LSP do gerente, o agente no LER de ingresso:

1. Cria um objeto EXPLICIT_ROUTE: *set_route_list*(...)
2. Inicia uma sessão RSVP com o LER de egresso: *rapi_session*(...)

3. Cria um objeto SENDER_TSPEC: *set_tspec*(..)
4. Preenche o objeto SENDER_TEMPLATE
5. Envia a mensagem *PATH*: *rapi_sender*(...)

Ao receber a instrução do gerente, o LER de egresso:

1. Inicia uma sessão RSVP com o LER de ingresso: *rapi_session*(...)
2. Cria um objeto para classificação do tráfego – FILTER_SPEC: *get_filter_rapirecv*(...)
3. Cria um objeto descrevendo a QoS desejada – FLOWSPEC: *get_flowspec*(...)
4. Envia a mensagem *RESV*: *rapi_reserve*(...)

4.4 Agente de Tráfego

Os agentes de tráfego são executados em todos os nós da rede. O agente foi implementado em C++ incorporando código legado em C relativo às funcionalidades providas pelo gerador de tráfego. O agente oferece métodos públicos mapeados para IDL e disponíveis ao gerente para geração do tráfego definido para o experimento.

4.4.1 Arquitetura do agente

Como se pode observar na Figura 18, o agente possui um *wrapper* em C++ que realiza todas as funcionalidades referentes a CORBA. Este *wrapper* é composto de 3 classes geradas automaticamente pelo "ORBacus IDL-to-C++ *Translator*" a partir da API denifida em IDL para o agente. A classe ***AgentTraffic*_impl** contém os métodos definidos na API implementados em C++, que são invocados quando algum método do agente é invocado via CORBA.

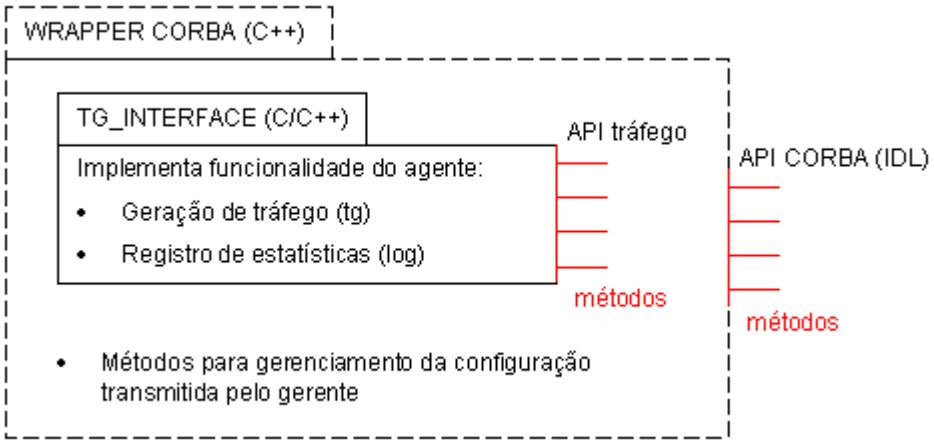


Figura 18: Arquitetura do agente de tráfego

Para atender ao requisito de transparência em relação ao gerador de tráfego utilizado, todas as funcionalidades relativas a ele e dependentes da implementação ficam encapsuladas em uma classe denominada **tg_*interface***, que é instanciada pela classe *AgentTraffic*_impl. A classe

tg_*interface* oferece métodos com assinaturas genéricas para geração de tráfego (assim como o *wrapper* e a API em IDL), sendo que a implementação específica para o tg se encontra dentro destes métodos.

A configuração para o gerador de tráfego transmitida pelo gerente é armazenada em um arquivo, havendo, portanto, transparência na sintaxe do mesmo. As estatísticas geradas pelo gerador de tráfego também são armazenadas em arquivos de log, pois além de terem sintaxe e formato (binário) específico para o tg, geram arquivos muito grandes.

4.4.2 Integração com o gerador de tráfego

O gerador de tráfego escolhido para a PLEX MPLS é o **tg**. Como no agente MPLS, o código em C foi encapsulado por uma classe em C++. O agente recebe a configuração de tráfego do gerente, armazena as configurações e, para efetivamente iniciar a geração de tráfego, cria uma *thread* com o código do tg que executa as configurações armazenadas.

4.5 Módulo TE

O módulo TE está presente em todos os roteadores do domínio MPLS (tanto LSRs, quanto LERs). O módulo foi implementado exclusivamente em C, pois se localiza no kernel do sistema operacional FreeBSD. O módulo constitui-se de algumas alterações no *kernel* para estender as funcionalidades oferecidas pelo NIST *Switch*. As principais funcionalidades oferecidas são:

- Máscaras para agregação de tráfego
- Mapeamentos de múltiplos fluxos a LSPs
- LSPs paralelos
- Algoritmo para balanceamento de carga

4.5.1 Agregação de tráfego e mapeamento de fluxos a LSPs

A agregação de tráfego com o uso de máscaras (ver item 3.2.5.1) foi implementada no *kernel* do FreeBSD (*label_table.c*) e funciona da seguinte maneira:

1. Um pacote sem rótulo no LER de ingresso é comparado com as entradas da tabela de *labels* (IP origem/destino, porta origem/destino, protocolo, CoS)
2. A operação lógica “E” é aplicada entre o pacote e a máscara. O resultado é comparado à entrada da tabela de rótulos. Caso sejam iguais, a entrada é selecionada para o encaminhamento do pacote:

ENTRADA == (PACOTE & MÁSCARA)

Para que uma informação do pacote seja ignorada, é necessário que o respectivo campo seja nulo tanto na máscara quanto na entrada da tabela de rótulos (seja um LSP ou um mapeamento). Há uma lista de máscaras que são utilizadas seqüencialmente nessa comparação. A lista vai da máscara mais restritiva (tudo 1) até a mais abrangente (tudo 0, exceto sub-rede de destino). Todas as entradas da tabela de rótulos são comparadas com a primeira máscara, depois a

segunda e assim por diante até que uma entrada seja selecionada ou que se chegue ao fim das entradas e das máscaras. Logo, seleciona-se a entrada mais exata para o pacote.

A agregação de tráfego com o uso de mapeamentos (ver item 3.2.5.1) também foi implementada no kernel do FreeBSD (label_table.c) e funciona da seguinte maneira:

- Um mapeamento é adicionado à tabela de rótulos numa estrutura de dados igual à utilizada por LSPs verdadeiros, mas com o rótulo de saída do LSP a que o mapeamento se refere.
- Durante a busca por uma entrada para classificar um pacote sem rótulo, caso a entrada correspondente a um mapeamento seja selecionada, é utilizado o rótulo do LSP verdadeiro pelo qual o pacote será encaminhado.

Como não há o envolvimento do RSVP-TE para a adição de um mapeamento à tabela de rótulos, foi necessário o uso de um outro mecanismo. Para tanto, utiliza-se um *socket* especial (PF_*ROUTE*) que origina uma chamada de sistema para efetuar operações com a tabela de rotas convencionais. No tratamento dessa chamada o NIST *Switch* incluiu código para operar a tabela de rótulos, sendo uma das operações a adição de um rótulo.

4.5.2 LSPs paralelos

Como LSPs paralelos possuem as mesmas características quanto a LER de ingresso/egresso, apenas os rótulos que os identificam é diferente, surge o problema de colisão de entradas na tabela de rótulos. Para solucionar este problema, a tabela foi estendida. Foi adicionado um campo na estrutura de dados que representa uma entrada, que é utilizado para construir uma lista ligada de entradas quando há colisão. Quando isso ocorre, a nova entrada é adicionada ao final dessa lista.

```
struct _labelTableEntry *lteBalList; /* Output list for load balancing */
```

4.5.3 Balanceamento de carga

Quando *`lt_find_next_dest()`* é invocada para obter o LSP pelo qual um pacote sem rótulo será encaminhado e a entrada selecionada corresponde a uma lista de LSPs paralelos, aplica-se nesta lista um algoritmo para escolha de qual dos LSPs paralelos será retornado. Este algoritmo foi implementado no *kernel* (*label_table.c*), conforme discutido no item 3.2.5.2. Este algoritmo utiliza um campo que foi adicionado à estrutura que representa o rótulo:

```
u_int32_t   lteBal;         /* Load balancing (% through this LSP) */
```

O código do algoritmo é descrito abaixo:

1. Somar as estatísticas de envio de pacotes dos LSPs paralelos (soma_pac)
2. Selecionar o 1º LSP. Calcular a porcentagem de envio do 1º LSP e atribuí-la à porcentagem selecionada:

   ```
   porc_selec = (pac_LSP / soma_pac) * 100
   ```

3. Calcular a diferença da porcentagem do LSP e o fator de balanceamento de carga (expresso em %)

   ```
   porc_selec = porc_selec - fator_balanc_LSP
   ```

4. Para cada LSP restante da lista:

   4.1 Calcular a porcentagem de envio do LSP e atribuí-la à porcentagem temporária:

   ```
   porc_tmp = (pac_LSP / soma_pac) * 100
   ```

   4.2 Calcular a diferença entre a porcentagem do LSP e o fator de balanceamento de carga do mesmo (expresso em %):

   ```
   porc_tmp = porc_tmp – fator_balanc_LSP
   ```

   4.3 Caso a porcentagem do LSP corrente seja menor que a do LSP selecionado, atribuí-la à porcentagem selecionada e selecionar o LSP corrente:

   ```
   se (porc_tmp < porc_selec) porc_selec = porc_tmp
   ```

5. Retornar o LSP selecionado

Este algoritmo seleciona sempre o LSP cuja porcentagem de envio de pacotes está mais distante de seu fator de balanceamento de carga. Caso mais de um LSP esteja igualmente distante, o primeiro deles na ordem da lista será selecionado.

4.6 Síntese

Neste capítulo é apresentada a implementação da PLEX MPLS, incluindo a interface com o usuário do Gerente; a arquitetura e os componentes de software dos agentes; as características para ET do RSVP-TE; e detalhes da implementação do módulo TE e das alterações realizadas no NIST Switch.

5 Experimentos

Neste capítulo são descritos os experimentos realizados para a validação da PLEX MPLS e apresentados seus resultados. Primeiramente, o cenário e a topologia de rede utilizados são descritos de forma genérica. Em seguida, cada um dos experimentos é descrito em mais detalhes, incluindo: objetivos, cenário específico do experimento, os passos de sua execução e a configuração de seus parâmetros. Após a descrição de cada experimento, os resultados obtidos são apresentados e discutidos.

5.1 Cenário

O cenário que se pretende representar nos experimentos é o *backbone* de um provedor de serviços de telecomunicação. Os computadores utilizados representam *switches* e *gateways* e as conexões ponto-a-ponto entre eles representam as interconexões entre os equipamentos do provedor. A interconexão ponto-a-ponto de apenas alguns computadores entre si (em oposição à topologia totalmente conectada ou *full mesh* proporcionada por um *switch*) foi adotada deliberadamente de forma a modelar o cenário proposto. A topologia genérica da rede utilizada para os experimentos pode ser vista na Figura 19.

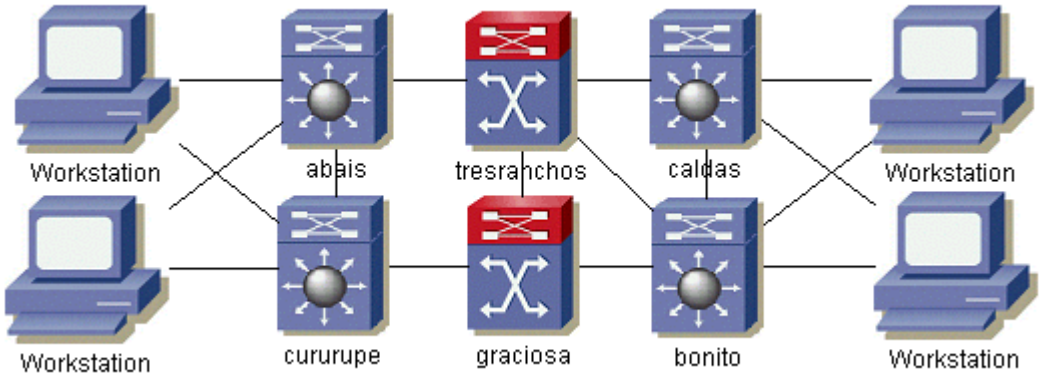


Figura 19: Topologia de rede genérica utilizada para os experimentos

O tráfego utilizado nos experimentos também procura representar fluxos de tráfegos presentes em um *backbone*. São utilizados tráfegos de aplicações multimídia, tráfego HTTP e tráfego de fundo gerado por geradores de tráfego. É importante ressaltar que esses tráfegos são **pacotes de rede reais** e não uma simulação ou emulação. Como geralmente ocorre, o caminho desses tráfegos coincide em diversas conexões, ocasionando uma disputa por recursos computacionais em que geralmente os tráfegos mais frágeis (ex.: multimídia) são severamente afetados. Utiliza-se então a tecnologia MPLS para aplicar esquemas de Engenharia de Tráfego que controlam esta disputa de modo a garantir uma utilização justa dos recursos, eliminar a subutilização de certos recursos enquanto outros estão sobrecarregados e priorizar a transmissão de tráfegos considerados mais importantes. A PLEX MPLS é utilizada para configurar estes esquemas de Engenharia de Tráfego e para implementar a execução dinâmica destes experimentos.

5.2 Experimento 1

Foram realizados dois experimentos para validar a proposta e a implementação da PLEX MPLS. O primeiro visa a validar tanto a proposta, quanto a implementação da PLEX, através da execução de um experimento simples de Engenharia de Tráfego. Os resultados obtidos deste experimento são quantitativos e embasam de forma irrefutável as conclusões a que se chegou. Os resultados são apresentados a seguir na forma de descrições textuais, tabelas, gráficos e *screenshots*.

5.2.1 PLEX como PLataforma para EXperimentos com MPLS para Engenharia de Tráfego

Este experimento visa a demonstrar a capacidade da PLEX MPLS em executar experimentos didáticos simples com MPLS para realizar Engenharia de Tráfego:

- Circuito virtual com MPLS (TE)
- Balanceamento de carga entre 2 LSPs paralelos
- Reserva de banda para garantir QoS do tráfego prioritário

5.2.1.1 Cenário específico

A arquitetura da rede de testes para o experimento 1 é constituída de um *backbone* com 6 LSRs e 2 hosts conforme apresentado na Figura 20. Há 3 pontos de entrada/saída de tráfego no backbone através dos LERs: abais, bonito e cururupe.

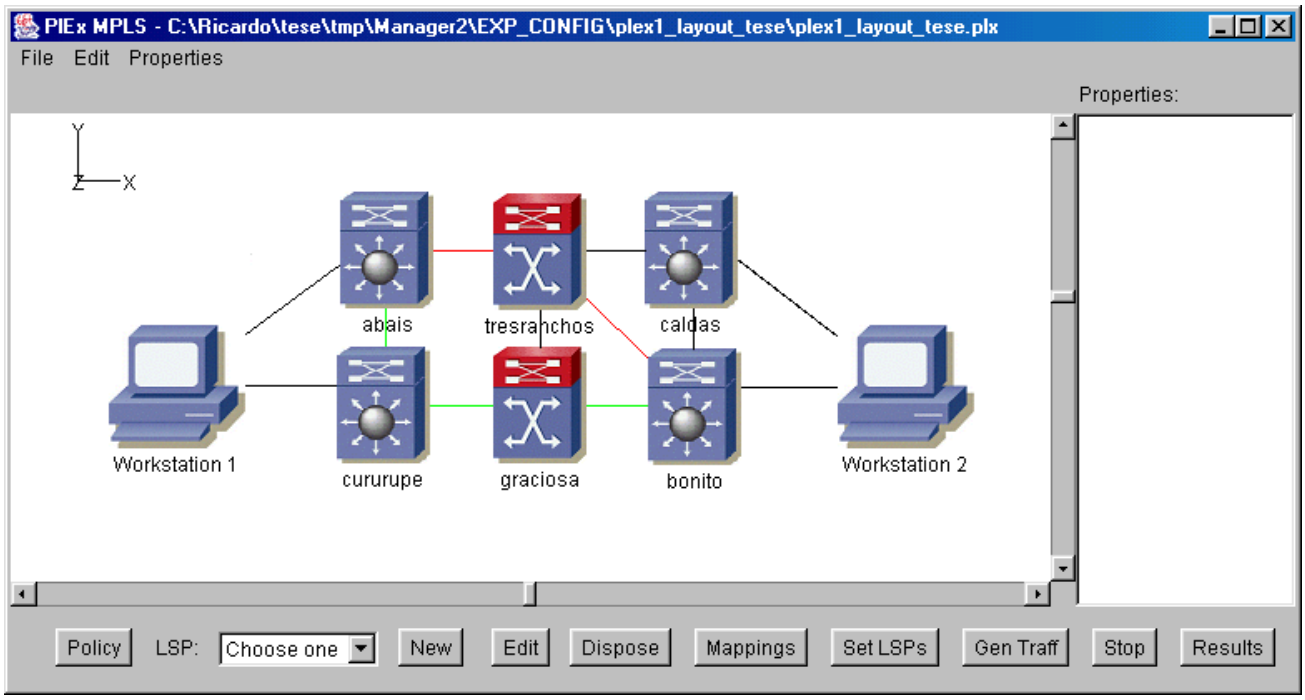


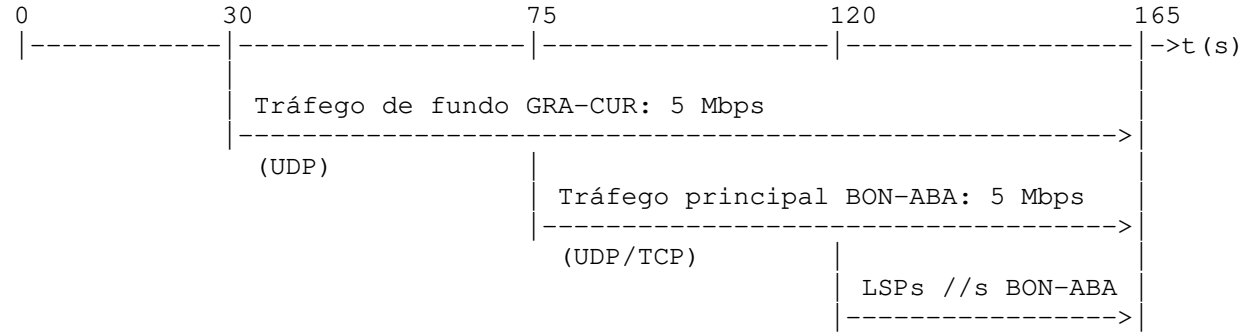
Figura 20: Topologia de rede utilizada para o experimento 1

A rota convencional para tráfego entre em bonito e abais é: bonito → graciosa → cururupe → abais. Será gerado 1 fluxo de tráfego prioritário (UDP/TCP) originado em bonito destinado a abais (que pode ser considerado como um fluxo agregado de alguma sub-rede acessível a partir de bonito para alguma sub-rede após abais). Será também gerado 1 fluxo de tráfego de fundo (UDP) originado em graciosa e destinado a cururupe (que pode ser visto como tráfego de fundo originado/destinado em sub-redes distintas das sub-redes do fluxo prioritário, mas que tem um segmento em comum com aquele fluxo, competindo deste modo pelos recursos da rede). Como a soma dos fluxos será maior que a banda disponível, os fluxos sofrerão uma degradação da QoS experimentada.

Serão então criados 2 LSPs paralelos entre bonito e abais, conforme apresentado na Figura 20 – LSP 1 em vermelho (passando por tresranchos) e LSP2 em verde (passando por graciosa e cururupe), demonstrando assim o primeiro ponto deste experimento. O tráfego prioritário será balanceado (50% a 50%) entre os LSPs (segundo ponto). Por fim, os LSPs terão reserva de banda, de modo a assegurar a QoS e proteger o fluxo prioritário do tráfego de fundo (terceiro ponto).

5.2.1.2 Seqüência de execução

O experimento seguirá a seqüência de eventos mostrada no esquema abaixo. Após 30 s do início do experimento a geração do tráfego de fundo será iniciada e durará até o final do experimento. No instante 75 s o tráfego prioritário será iniciado, também até o final do experimento. A partir deste momento os tráfegos estarão competindo pelos recursos e deve ser notado um impacto negativo na QoS experimentada pelos fluxos. No instante 120 s, serão instanciados os LSPs com reserva de recursos e o tráfego prioritário será mapeado para estes LSPs, garantindo assim a QoS de que necessita.



5.2.1.3 Configuração dos experimentos

O experimento 1 foi executado várias vezes, alterando-se parâmetros como protocolo de transmissão e banda do tráfego prioritário. Os principais parâmetros do experimento 1 são:

- Tráfego de fundo: 5 Mbps
- Tráfego principal: **X** Mbps (variável de acordo com a execução do experimento)
- Banda disponível para melhor-esforço: 5 Mbps
- Garantia de banda dos LSPs: 4 Mbps
- Velocidade das interfaces: 10 Mbps
- Configuração do ALTQ: root (10 Mbps) / default (4,8 Mbps) / ctl (0,2 Mbps) / mpls (5 Mbps)

Tráfego de fundo (UDP):

Gerador (graciosa)	Receptor (cururupe)
on 15 udp 10.2.1.2.30000 at 2 setup at 30 arrival exponential 0.0022676 length constant 1420 seed 321423 time 135	on 15 udp 10.2.1.2.30000 server at 2 wait

Tabela 5: Scripts de configuração do gerador de tráfego para o tráfego de fundo

Na Tabela 5 estão relacionados os scripts de configuração do gerador de tráfego para o tráfego de fundo. Basicamente é configurado um fluxo UDP da máquina graciosa para cururupe (10.2.1.2) na porta 30000. Este fluxo tem início 30 segundos após o início da execução do script, tem característica exponencial e os pacotes com 1420 bytes são transmitidos com um intervalo de 0,0022676 segundos. A última linha inicializa um gerador de números aleatórios e estipula a duração do fluxo em 135 segundos.

Tráfego principal (UDP / TCP):

Gerador (bonito)	Receptor (abais)
on 15 <UDP / TCP> 10.2.6.7.30000 at 2 setup at 75 arrival exponential INTERVALO length constant 1420 seed 321423 time 90	on 15 <UDP / TCP> 10.2.6.7.30000 server at 2 wait

Tabela 6: Scripts de configuração do erador de tráfego para o tráfego prioritário

Na Tabela 6 estão relacionados os scripts de configuração do gerador de tráfego para o tráfego prioritário. Basicamente é configurado um fluxo UDP ou TCP da máquina bonito para abais (10.2.6.7) na porta 30000. Este fluxo tem início 75 segundos após o início da execução do script, tem característica exponencial e os pacotes com 1420 bytes são transmitidos com um intervalo variável dependendo da execução (veja Tabela 7). A última linha inicia um gerador de números aleatórios e estipula a duração do fluxo em 90 segundos.

Parâmetros das execuções:

Na Tabela 7 estão relacionados os parâmetros do tráfego prioritário para as diferentes execuções do experimento.

Execução	Protocolo	Banda - X (Mbps)	Intervalo entre pacotes
1	UDP	5	0.0022676
3	UDP	6	0.0018904
5	UDP	7	0.0016207
7	TCP	6	0.0018904
9	TCP	7	0.0016207

Tabela 7: Parâmetros do tráfego prioritário nas execuções do experimento

5.2.2 Resultados

A seguir são apresentados os resultados para as diferentes execuções do experimento. Primeiramente, pode-se ver as estatísticas do tráfego prioritário transmitido pelos LSPs através da ferramenta "*print_table*", que exibe as informações da tabela de labels, entre elas:

- LabelIn: label de entrada;
- DestAddr: endereço de destino (IP + porta) dos pacotes;
- SrcAddr / SPort: endereço de origem (IP + porta);
- Proto: protocolo de transmissão;
- LabelOut: label de saída;
- Iface: interface de rede pela qual os pacotes serão transmitidos;
- Stat_Out: número de pacotes transmitidos pelo LSP;
- Load_bal: fator de balanceamento de carga (%).

Depois são apresentados os gráficos de banda transmitida e descartada para os tráfegos prioritário e de fundo extraídos do arquivo de log dos geradores de tráfego.

5.2.2.1 Execução 5 – Tráfego prioritário UDP

Os resultados da execução 5 do experimento 1, que corresponde a tráfego prioritário utilizando o protocolo UDP com uma banda nominal de 7 Mbps podem ser conferidos abaixo.

root@bonito>print_table

LabelIn	DestAddr	DPort	SrcAddr	SPort	Proto	LabelOut	Iface	Stat_Out	Load_bal
41500	10.2.5.7	30000	10.2.8.4	20000	17	41500	xl 3	0	100
30000	10.2.6.7	30000	10.2.2.4	20000	17	30000	xl 1	0	100
2	10.2.6.7	30000	0.0.0.0	0	0	41500	xl 3	13420	50
0	10.2.6.7	30000	0.0.0.0	0	0	30000	xl 1	13419	50

3,39 Mbps por LSP ➔ 6,78 Mbps no total

Conforme descrito no item 4.5.1, a tabela de *labels* apresenta tanto as informações relativas aos LSPs, quanto aos mapeamentos. É interessante notar que os pacotes transmitidos são registrados no mapeamento em que foram classificados, não no LSP em que efetivamente foram transmitidos. A seguir são apresentados os gráficos extraídos dos arquivos de log dos geradores de tráfego:

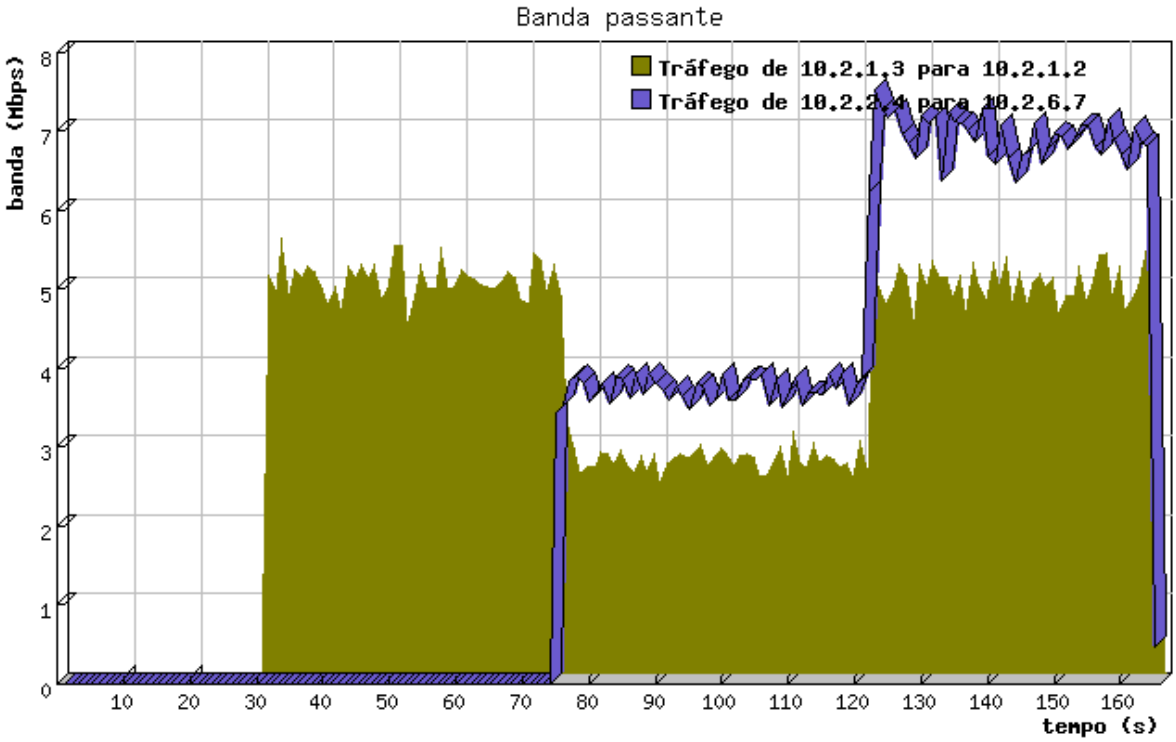


Figura 21: Banda passante (Mbps) X tempo (s)

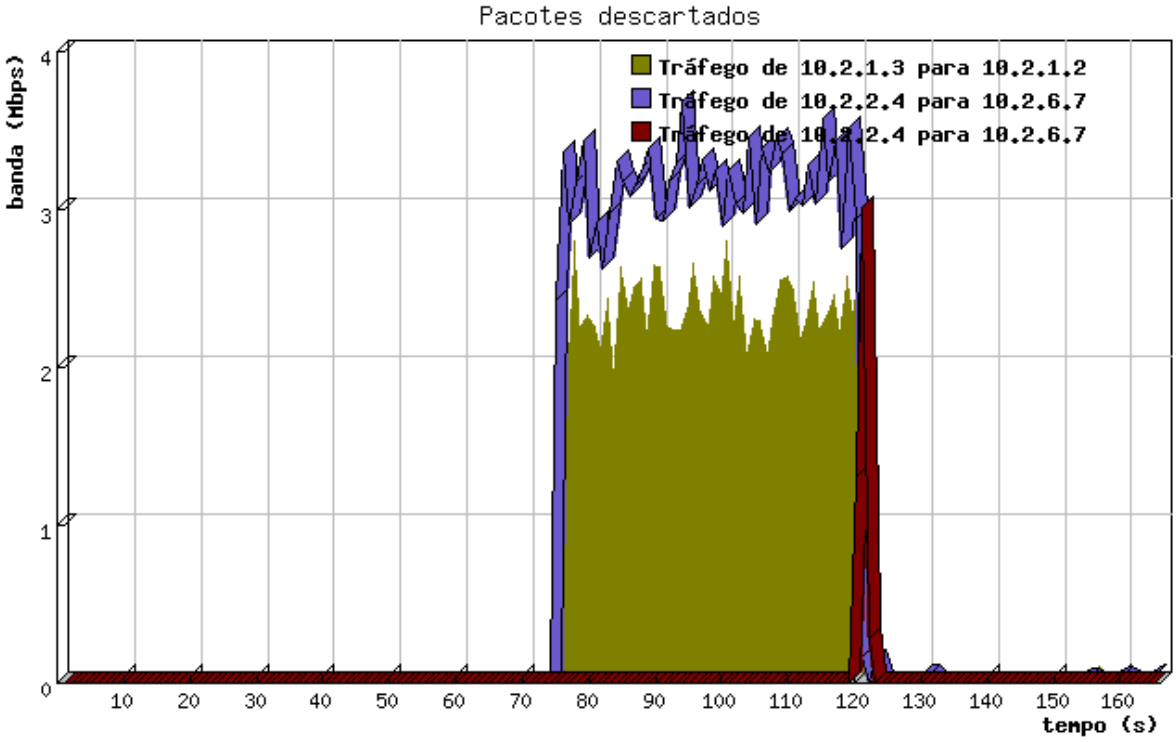


Figura 22: Distribuição da perda de pacotes X tempo (s)

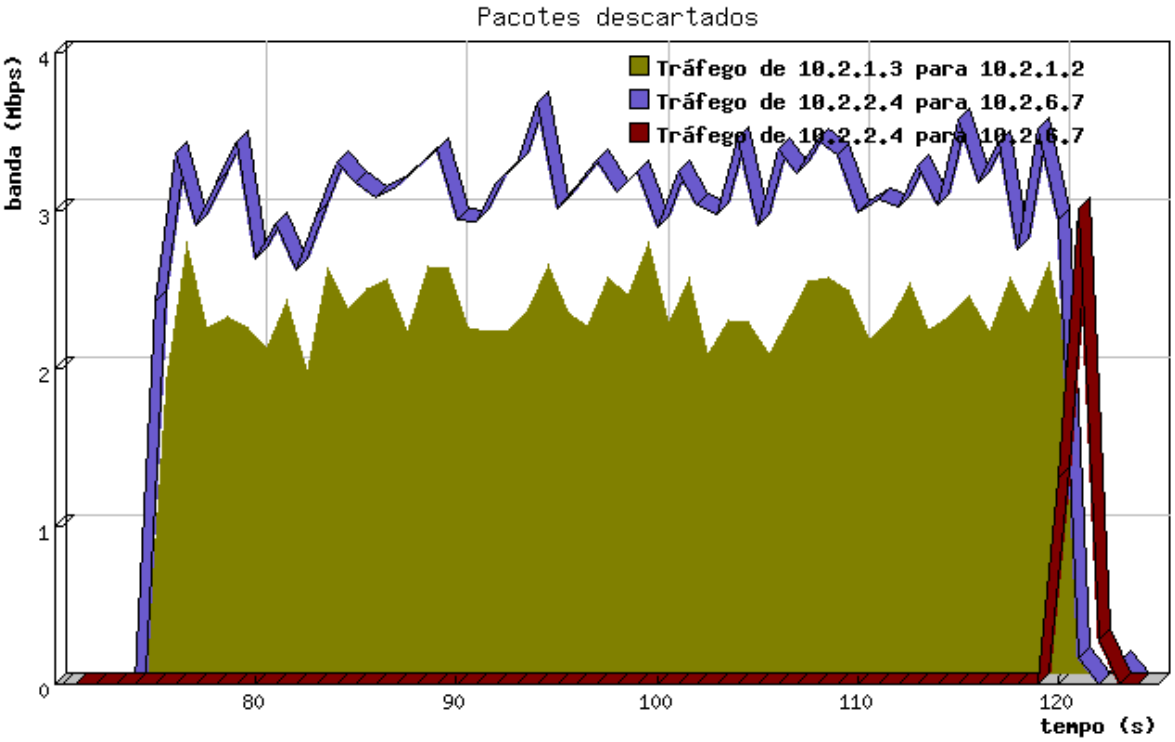


Figura 23: Detalhe da distribuição da perda de pacotes X tempo (s)

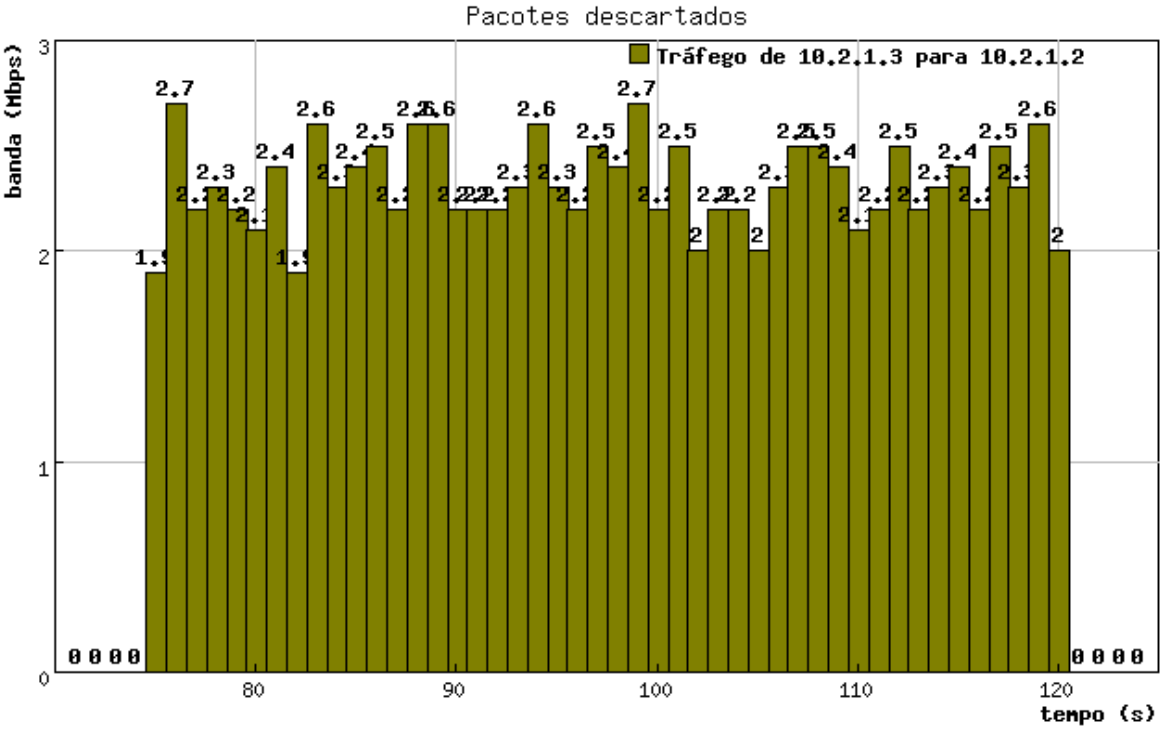


Figura 24: Detalhe da distribuição da perda de pacotes X tempo (s) para o tráfego de fundo

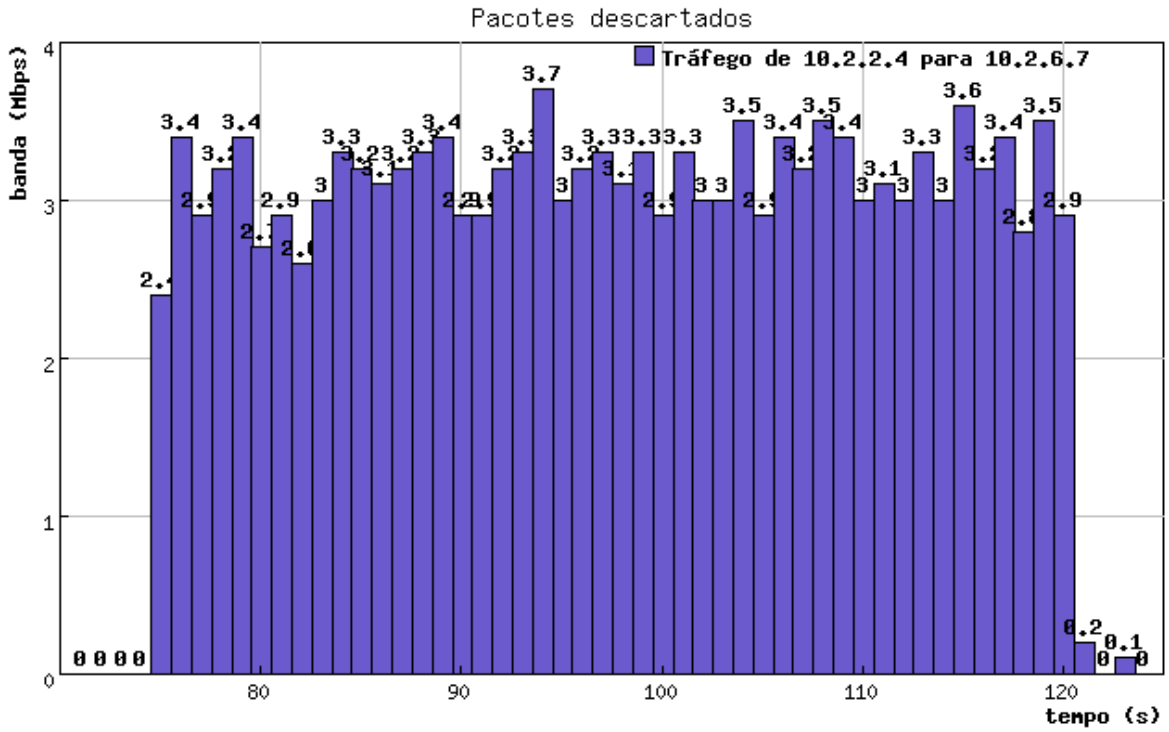


Figura 25: Detalhe da distribuição da perda de pacotes X tempo (s) para o tráfego prioritário

A Figura 21 mostra como se comportou a banda passante dos tráfegos prioritário (azul) e de fundo (amarelo oliva). Conforme descrito na configuração do experimento, o tráfego de fundo foi iniciado aos 30 s com 5 Mbps. Aos 75 s o tráfego prioritário foi iniciado com 7 Mbps e pode-se notar a competição entre os tráfegos por banda de transmissão. Como as *interfaces* de rede estão configuradas para 10 Mbps (nominais, na realidade alcançam entre 7 e 8 Mbps), há descarte de pacotes de ambos os tráfegos. Conforme esperado, a partir dos 120 s, quando os LSPs são estabelecidos, os tráfegos conseguem obter banda de transmissão suficiente para alcançar a banda passante configurada nos *scripts* e o descarte de pacotes cessa. Na Figura 22 e na Figura 23 pode-se ver o comportamento do descarte de pacotes. O tráfego prioritário (azul) exibe um descarte maior de pacotes porque sua taxa de transmissão também é maior. O descarte de pacotes em vermelho nas figuras também corresponde ao tráfego prioritário, porém não é causado pela sobrecarga das conexões de rede. Ele deve-se ao fato de que, durante o estabelecimento do LSP, o mapeamento de tráfego foi criado antes do RSVP finalizar a instanciação do LSP, portanto os pacotes capturados pelo mapeamento eram encaminhados através de um LSP não existente, causando assim o seu descarte. Finalmente, na Figura 24 e na Figura 25 pode-se ver detalhes do descarte de pacotes para os tráfegos de fundo e prioritário respectivamente.

5.2.2.2 Execução 7 – Tráfego prioritário TCP

Os resultados da execução 7 do experimento 1, que corresponde a tráfego prioritário utilizando o protocolo TCP com uma banda nominal de 6 Mbps podem ser conferidos abaixo.

root@bonito>print_table

LabelIn	DestAddr	DPort	SrcAddr	SPort	Proto	LabelOut	Iface	Stat_Out	Load_bal
41500	10.2.5.7	30000	10.2.8.4	20000	17	41500	xl 3	0	100
30000	10.2.6.7	30000	10.2.2.4	20000	17	30000	xl 1	0	100
2	10.2.6.7	30000	0.0.0.0	0	0	41500	xl 3	13165	50
0	10.2.6.7	30000	0.0.0.0	0	0	30000	xl 1	13164	50

3,09 Mbps por LSP ➔ 6,19 Mbps no total

A seguir são apresentados os gráficos extraídos dos arquivos de log dos geradores de tráfego:

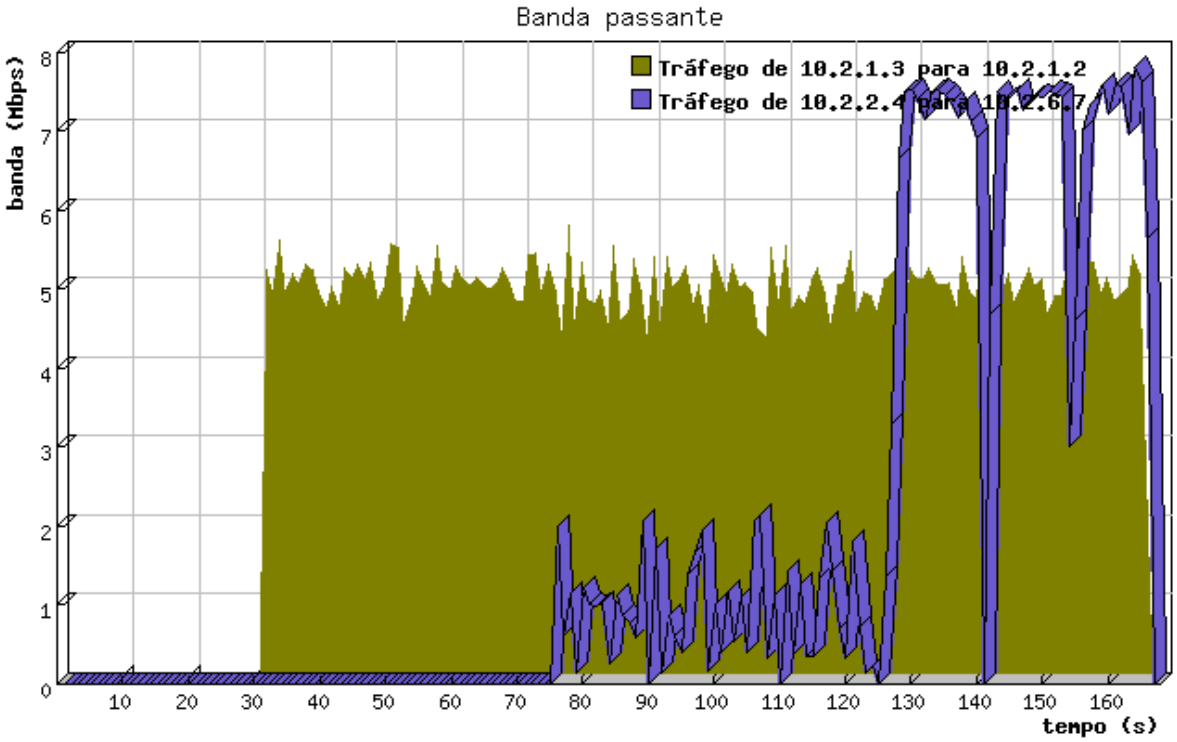


Figura 26: Banda passante (Mbps) X tempo (s)

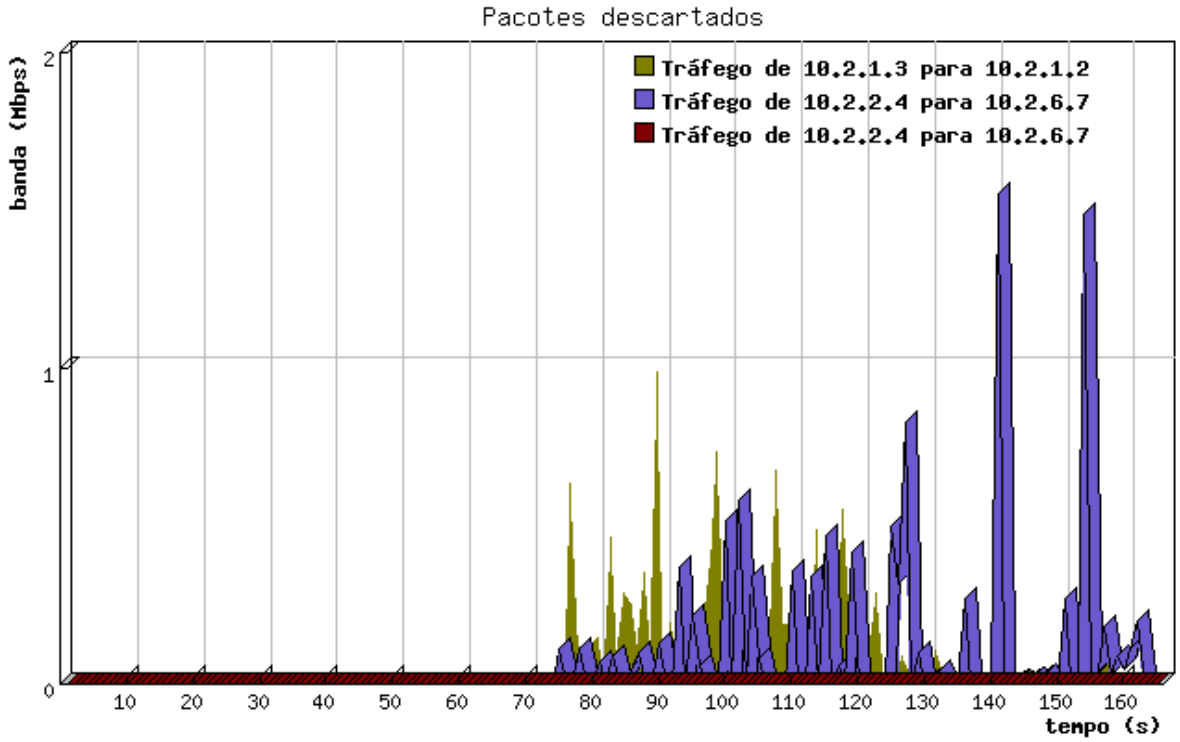


Figura 27: Distribuição da perda de pacotes X tempo (s)

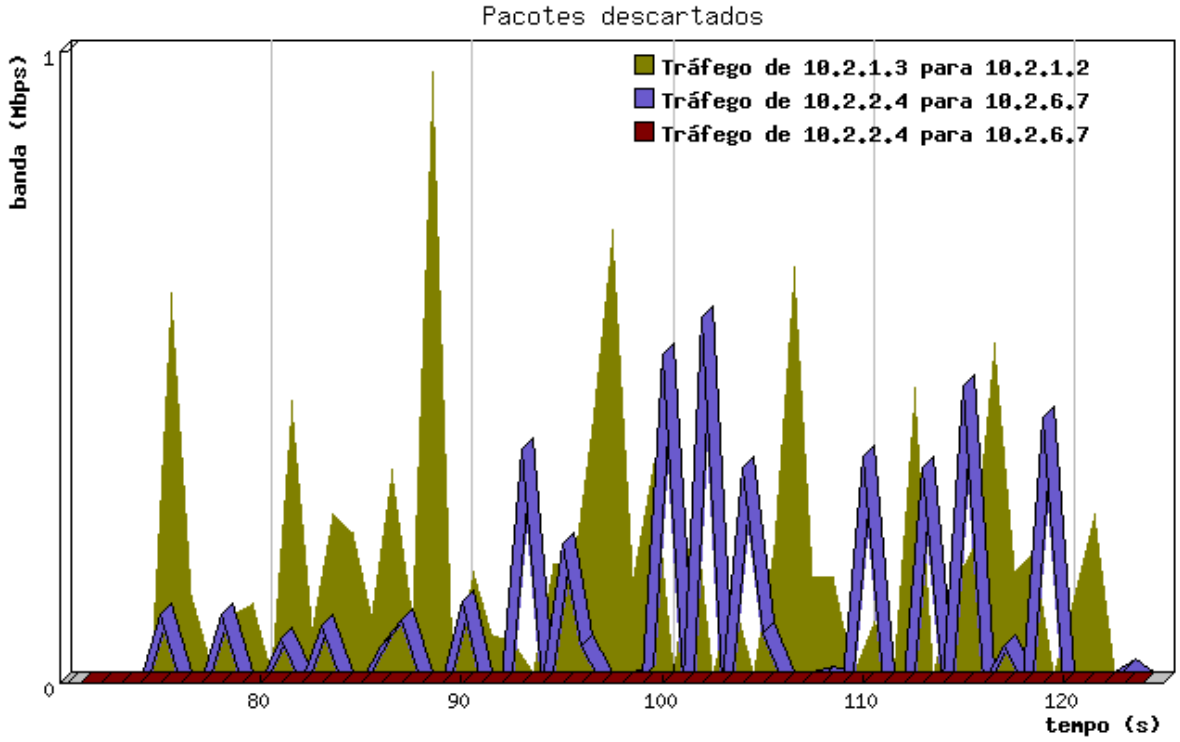


Figura 28: Detalhe da distribuição da perda de pacotes X tempo (s)

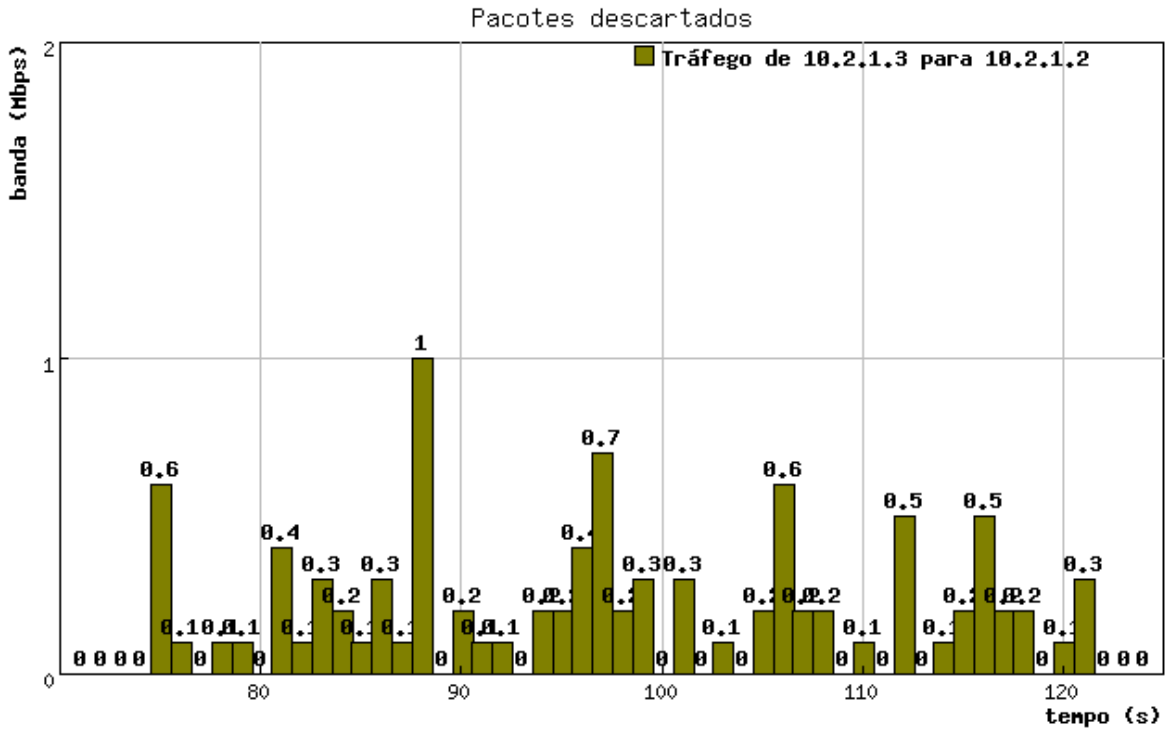


Figura 29: Detalhe da distribuição da perda de pacotes X tempo (s) para o tráfego de fundo

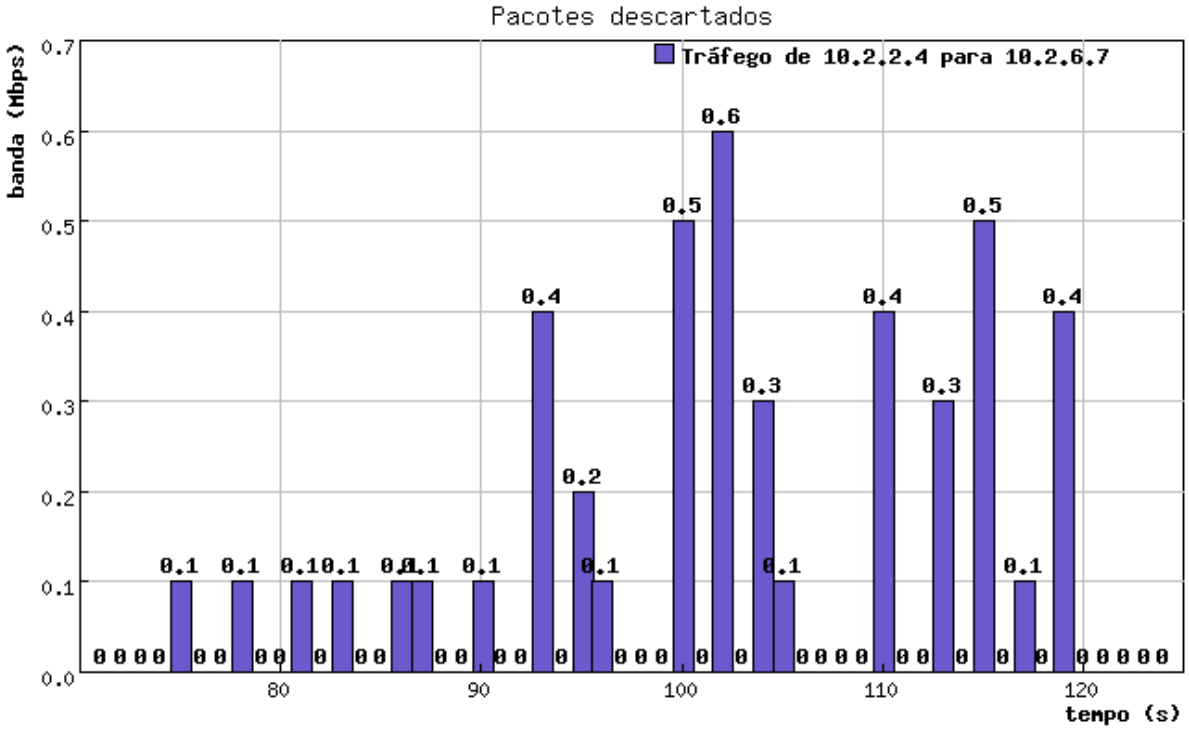


Figura 30: Detalhe da distribuição da perda de pacotes X tempo (s) para o tráfego prioritário

Os gráficos para a execução 7 do experimento 1 são análogos aos da execução 5, porém nitidamente o comportamento dos tráfegos é distinto, pois o tráfego prioritário utiliza o protocolo TCP. Devido às características do TCP, pode-se notar algumas diferenças interessantes:

- O tráfego de fundo (UDP) quase não é afetado pela concorrência com o tráfego prioritário, enquanto este é severamente afetado.
- O tráfego prioritário exibe uma grande variação na banda passante antes do estabelecimento dos LSPs.
- Após o estabelecimento dos LSPs, o tráfego prioritário apresenta uma taxa de transmissão bastante estável (aproximadamente 7,0 Mbps). A média é superior ao valor configurado devido à retransmissão dos pacotes descartados, em adição aos pacotes sendo gerados. As eventuais recaídas são devido ao mecanismo de recuo, intrínseco ao TCP.
- O descarte de pacotes do tráfego prioritário não corresponde à diferença entre a configuração utilizada no script do gerador e a banda efetivamente transmitida, pois o TCP efetua a retransmissão dos pacotes descartados.

5.3 Experimento 2

O segundo experimento realizado pretende validar a implementação da PLEX e ressaltar a aplicabilidade do MPLS como solução efetiva para Engenharia de Tráfego através de um cenário mais complexo e realista. Os resultados obtidos deste experimento são principalmente qualitativos, mas também permitem afirmar de modo enfático as propriedades do MPLS para TE. Os resultados do experimento são apresentados a seguir na forma de descrições textuais, tabelas, gráficos e *screenshots*.

5.3.1 PLEX demonstrando MPLS como solução efetiva para TE

Este experimento visa a demonstrar a capacidade e a viabilidade da tecnologia MPLS para esquemas de Engenharia de tráfego, bem como validar a implementação da PLEX MPLS em um cenário mais complicado que o do primeiro experimento. Os principais conceitos observáveis neste experimento são:

- Otimização da utilização dos recursos de rede
- Garantia de QoS para tráfegos prioritários
- Limitação dos recursos utilizados por qualquer tipo de tráfego de modo a não consumir mais recursos do que lhes é permitido

5.3.1.1 Cenário específico

Enquanto o experimento 1 trazia resultados quantitativos bastante expressivos, este experimento também demonstra o efeito do uso de MPLS para Engenharia de Tráfego de uma forma qualitativa: a qualidade de exibição de um vídeo transmitido através da rede durante o andamento do experimento. A reprodução do vídeo foi filmada com uma câmera profissional que possibilita a percepção das variações em sua qualidade durante o experimento.

A arquitetura da rede de testes para o experimento 2 é constituída de um backbone com 6 LSRs e 4 hosts, conforme apresentado na Figura 31. Há 4 pontos de entrada/saída de tráfego no backbone através dos LERs: abais, bonito, caldas e cururupe.

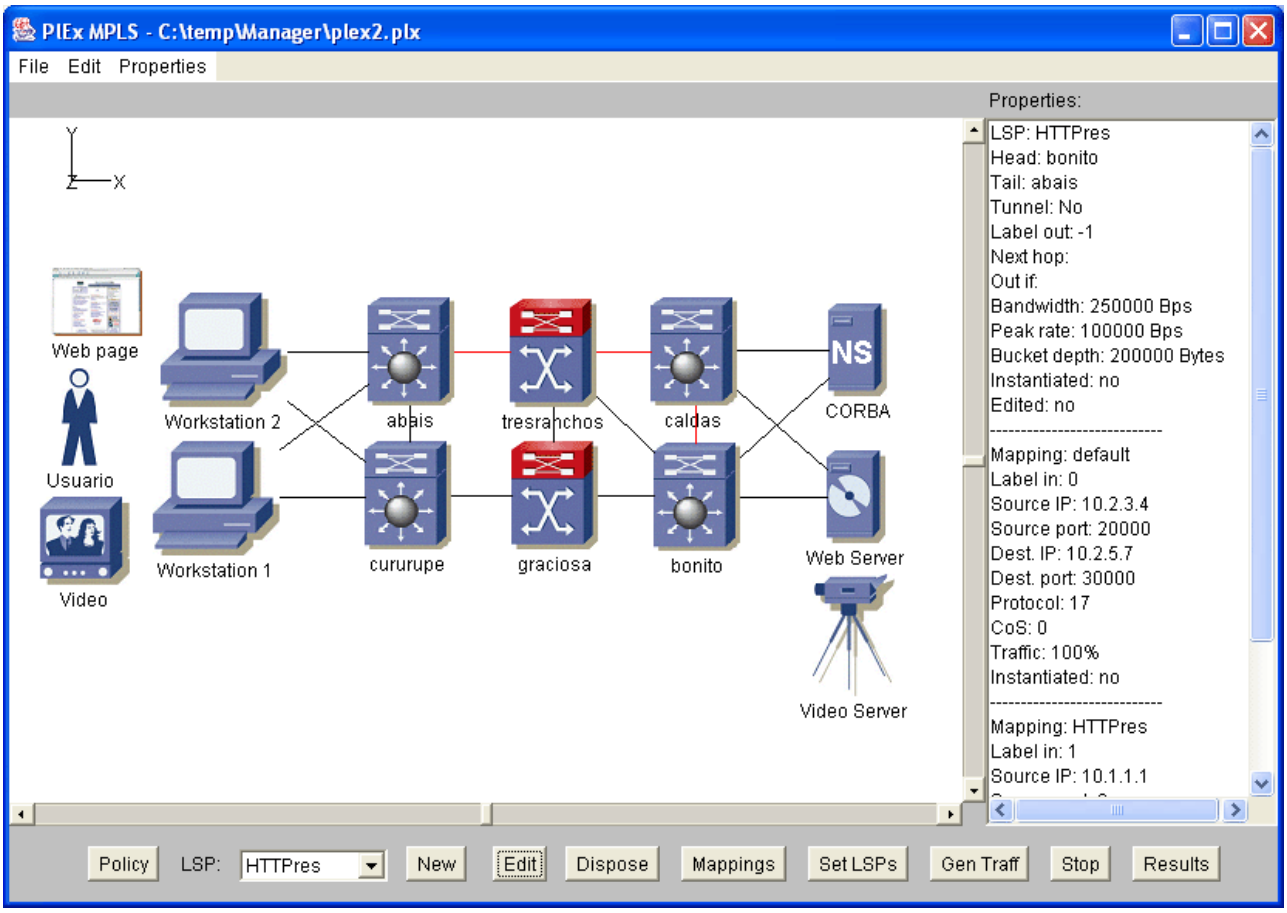


Figura 31: Arquitetura da rede de testes para o experimento 2

Os hosts têm as seguintes funções:

- aruana: servidor de vídeo
- mococa: servidor HTTP
- pontanegra: cliente de vídeo
- sucuri: cliente HTTP

A rota convencional para o tráfego entre os pares cliente/servidor é: bonito → graciosa → cururupe. Inicialmente, tem-se apenas o tráfego HTTP. Quando o tráfego de vídeo é iniciado, nota-se uma deterioração na QoS experimentada pelo HTTP. Em seguida, é iniciado o tráfego de fundo, que deteriora os dois tráfegos anteriores. O tráfego de fundo compartilha apenas 1 conexão com os outros tráfegos (entre graciosa e cururupe), mas este recurso, por estar sobrecarregado, afeta a QoS ofertada aos tráfegos HTTP e de vídeo.

Serão então criados os seguintes LSPs:

1. HTTP request: abais → tresranchos → caldas
2. HTTP response: bonito → caldas → tresranchos → abais
3. Vídeo response: bonito → graciosa → cururupe

O tráfego de vídeo (servidor → cliente) será transportado pelo LSP **3** pois, por consumir menos banda que o tráfego HTTP, permitirá que mais banda seja destinada ao tráfego de fundo. O tráfego HTTP será transmitido através dos LSPs **1** e **2** (um para os pacotes de *request* e outro, na direção contrária, para os pacotes de *response*), e será limitado a 2 Mbps de modo a não prejudicar muito eventuais tráfegos que já existam nessa rota. Assim os conceitos deste experimento são demonstrados: os tráfegos prioritários têm a QoS garantida, o tráfego HTTP tem sua utilização de recursos limitada e a utilização dos recursos de rede é otimizada.

5.3.1.2 Seqüência de execução

O experimento seguirá a seqüência de eventos descrita nos passos abaixo:

1. Tráfego HTTP é iniciado entre os *hosts* sucuri e Mococa até que sua banda de transmissão esteja estabilizada.
2. Transmissão de vídeo é iniciada entre os *hosts* pontanegra e aruana.
3. O impacto da transmissão de vídeo no tráfego HTTP pode ser percebido pela diminuição da banda de transmissão do mesmo.
4. O tráfego de fundo é iniciado simulando uma situação de sobrecarga da rede.
5. O impacto do tráfego de fundo pode ser sentido claramente, tanto no tráfego HTTP (sua banda diminui drasticamente), quanto no vídeo (cuja qualidade deteriora-se até a paralisação da reprodução em casos extremos).
6. São estabelecidos LSPs previamente configurados no PLEX Manager para realizar a Engenharia de Tráfego idealizada.
7. A Engenharia de Tráfego alcança seus objetivos normalizando a situação da rede. Os tráfegos HTTP e de vídeo têm suas transmissões normalizadas. A banda utilizada pelo tráfego de fundo e pelo HTTP é limitada a um teto máximo.
8. A ação dos LSPs estabelecidos pode ser conferida visualizando-se estatísticas do tráfego por eles transmitida de duas maneiras: *print_table* e *cbqstat*.

5.3.1.3 Configuração do experimento

O experimento 2 foi executado após uma sintonia fina em que as bandas do fluxos de tráfego e dos LSPs foram ajustadas para criar a situação de congestionamento desejada e sua posterior normalização através do uso da TE. Abaixo são listadas as características de configuração do experimento:

- Banda do tráfego HTTP: livre (variando durante a execução: 4 Mbps → 3,7 Mbps → 0,3 Mbps → 1,7 Mbps)
- Banda do tráfego de vídeo: 450 Kbps (variando durante a execução: 1,2 Mbps → 450 Kbps → 200 Kbps → 450 Kbps)
- Banda do tráfego de fundo: limitada pela capacidade de transmissão da interface (aproximadamente 8 Mbps)

Os LSPs estabelecidos possuem os seguintes limites de banda:

LSP	Banda
HTTP request	800 Kbps
HTTP response	2 Mbps
Vídeo response	800 Kbps

Tabela 8: Banda dos LSPs criados no experimento 2

Neste experimento, dada a grande carga da rede, foi necessário criar uma proteção para o tráfego de controle e sinalização (protocolos de roteamento, RSVP, chamadas CORBA, etc). Esta proteção foi realizada através da criação de uma classe CBQ com reserva de banda para esse tipo de tráfego. Sem a proteção, o congestionamento afetava esses tráfegos impedindo a correta realização do experimento.

5.3.2 Resultados

Há três principais maneiras de conferir os resultados do experimento 2. A observação qualitativa da reprodução em tempo real do vídeo sendo transmitido, a análise do arquivo de log do cliente HTTP e a análise das estatísticas do tráfego transmitido através dos LSPs.

5.3.2.1 Reprodução do vídeo

A observação da reprodução do vídeo deixa claro que a mesma foi severamente afetada pela sobrecarga da rede. Estes dados estão registrados em vídeo e uma pequena amostra pode ser conferida nas figuras abaixo, que mostram a reprodução antes, durante a sobrecarga da rede e depois de aplicada a Engenharia de Tráfego:

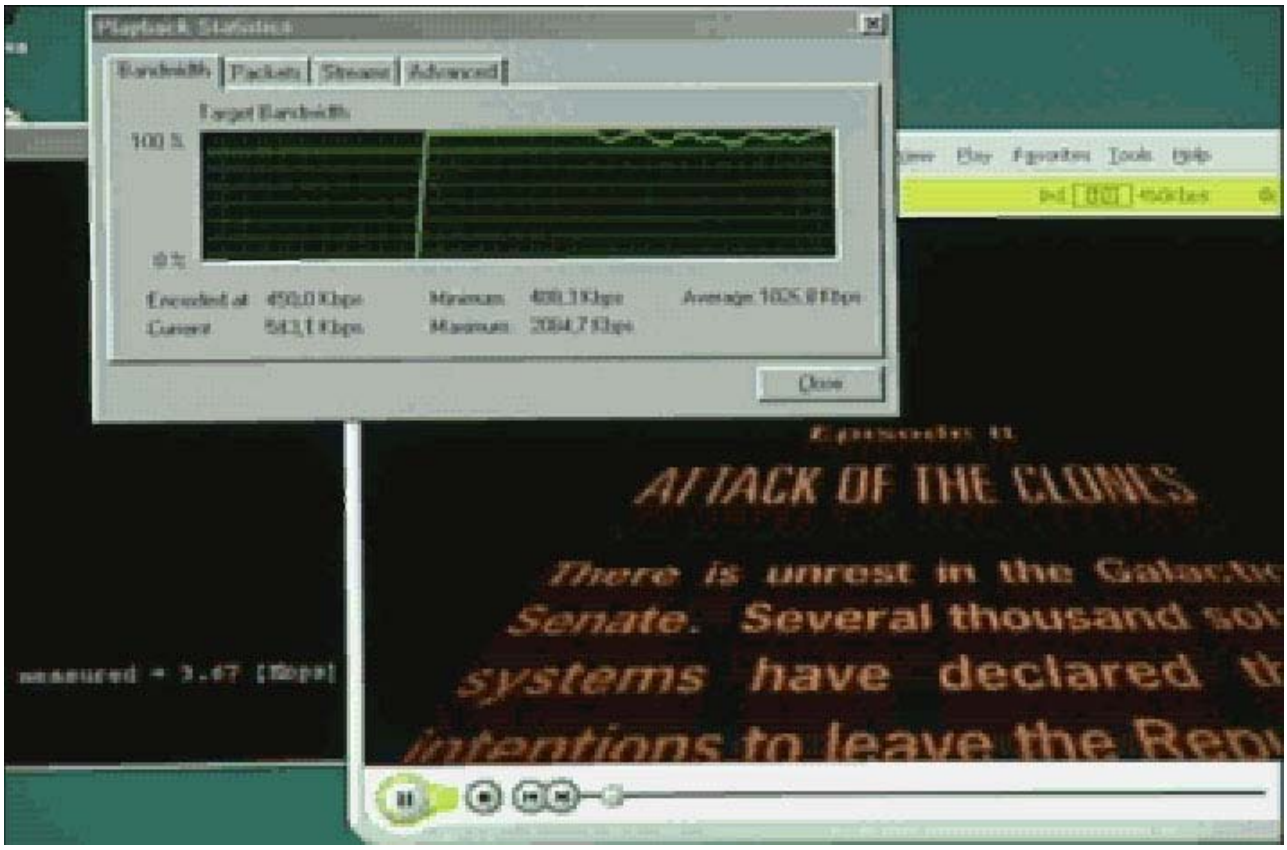


Figura 32: Início da reprodução do vídeo (formação de cachê e HTTP normal)

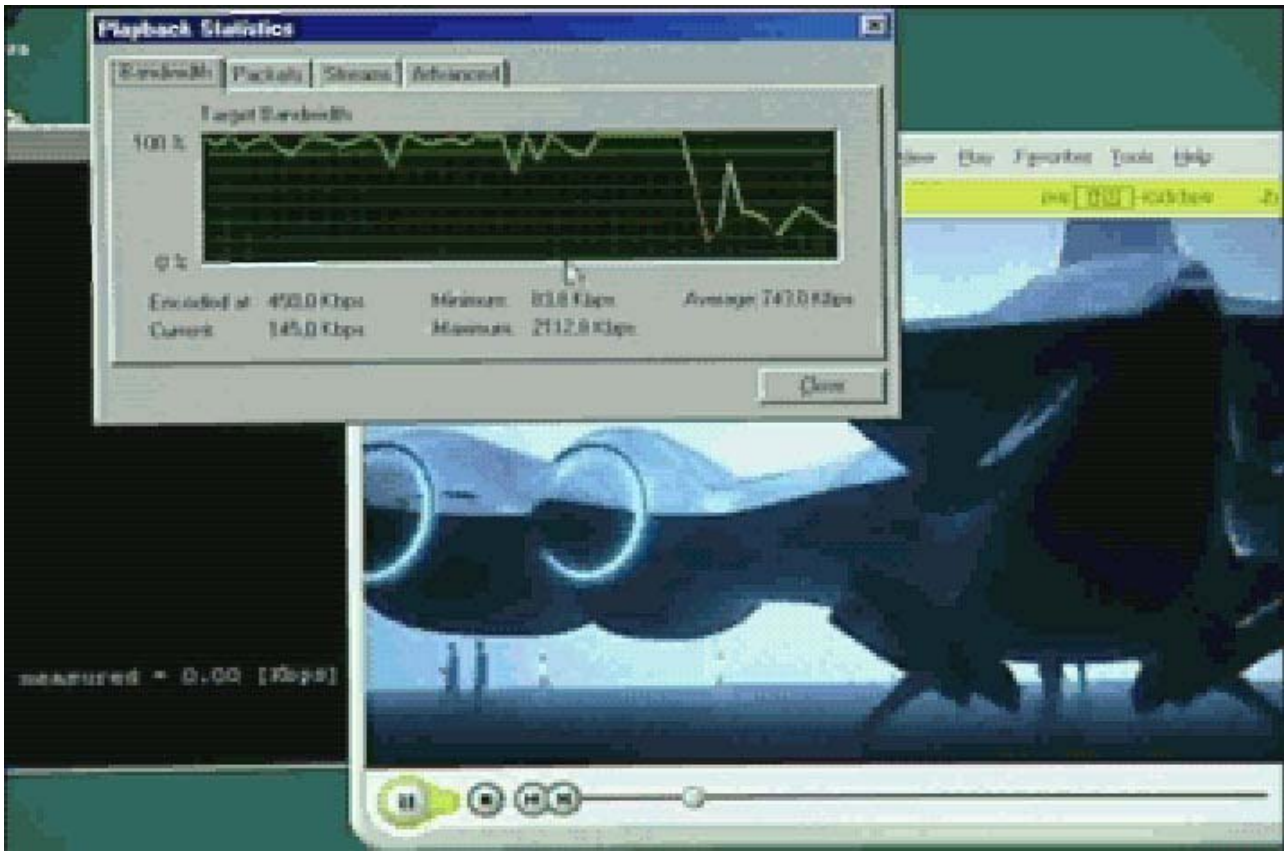


Figura 33: Reprodução do vídeo sob condições normais de rede

Na Figura 32 observa-se o início da transmissão do vídeo, quando é formado o cachê do cliente (a banda pode chegar a 2 Mbps), período em que o tráfego HTTP chega a ser afetado. Na Figura 33 vê-se a reprodução do vídeo em condições normais de rede, em que a banda transmitida varia, mas fica em torno dos 450 Kbps necessários a uma reprodução satisfatória do vídeo por parte do cliente. Nestas condições o tráfego HTTP não é afetado. Já na Figura 34, pode-se ver a reprodução do vídeo durante a sobrecarga da rede, quando o tráfego de fundo compete pela banda de uma das conexões de rede. O vídeo é severamente afetado, o que nota-se tanto pela imagem fragmentada e som metálico (às vezes até interrompidos), quanto pelo gráfico da banda recebida pelo cliente. Durante a sobrecarga da rede o tráfego HTTP é severamente afetado, sendo sua taxa de transmissão reduzida praticamente a zero. Finalmente, na Figura 35 pode-se ver o resultado da aplicação do esquema de Engenharia de Tráfego. O vídeo se recupera, como pode ser visto tanto na qualidade da reprodução, quanto no gráfico de transmissão. O tráfego HTTP também se estabiliza, limitado à banda de seu LSP, mas livre de interferências do vídeo ou do tráfego de fundo.

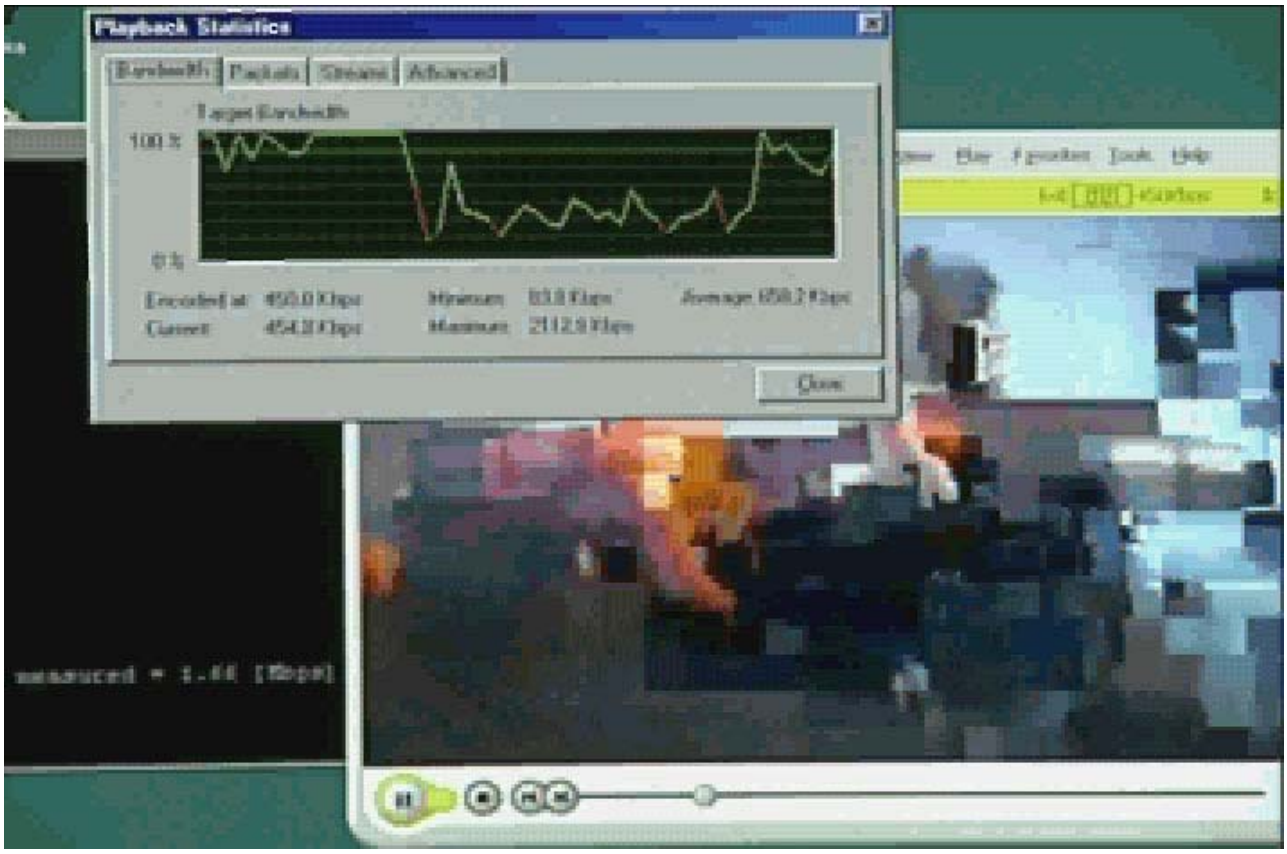


Figura 34: Reprodução do vídeo durante sobrecarga da rede

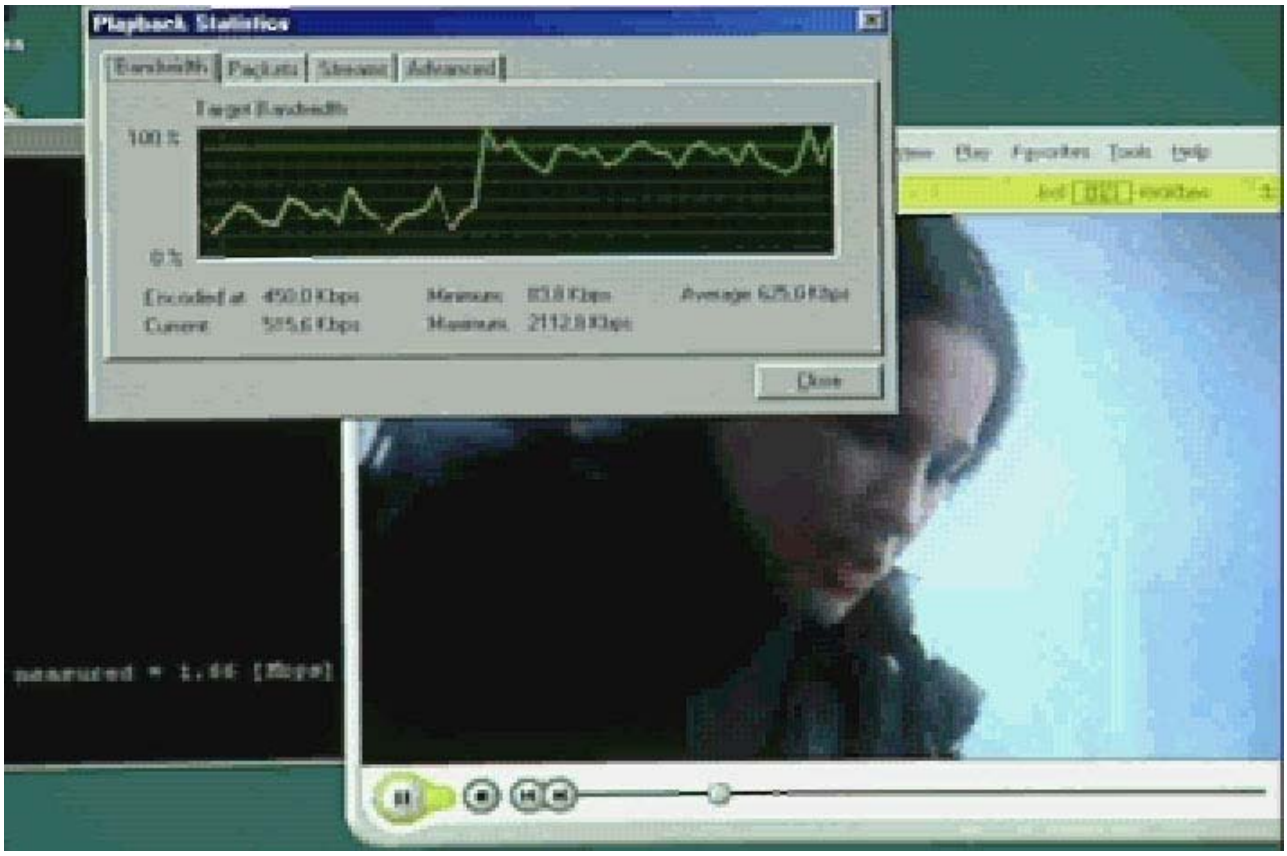


Figura 35: Reprodução do vídeo após a aplicação da Engenharia de Tráfego

5.3.2.2 Log do cliente HTTP

A banda do tráfego HTTP é registrada pelo cliente em um arquivo de log a cada segundo. O gráfico da banda pelo tempo pode ser visto na Figura 36. Nos primeiros 60 segundos, o tráfego HTTP não sofre competição e é limitado pela banda das conexões e pela capacidade do servidor, atingindo uma média de 4 Mbps. O tráfego de vídeo é iniciado neste instante e, após uma certa instabilidade (pois o início da transmissão de vídeo é muito explosivo em função do enchimento do *buffer* do cliente de vídeo), o tráfego HTTP estabiliza-se em torno de 3,70 Mbps. Aproximadamente aos 120 segundos o tráfego de fundo é iniciado e fica clara a fragilidade do tráfego HTTP, que praticamente não consegue mais ser transmitido (basicamente devidos aos mecanismos de controle de congestionamento do TCP e pela necessidade de *acknowledgement*). Quando os LSPs são estabelecidos (180 segundos), nota-se a recuperação do tráfego HTTP. O tráfego HTTP passa da situação de extrema instabilidade e baixíssima taxa de transmissão do período em que sofria interferência dos tráfegos de fundo e de vídeo, para uma taxa de transmissão bastante estável em torno de 1,70 Mbps. Este valor se deve à limitação de banda de 2 Mbps imposta pelo LSP para o qual o tráfego está mapeado (a pequena diferença de 0,30 Mbps deve-se às características do protocolo HTTP, do cliente e do servidor).

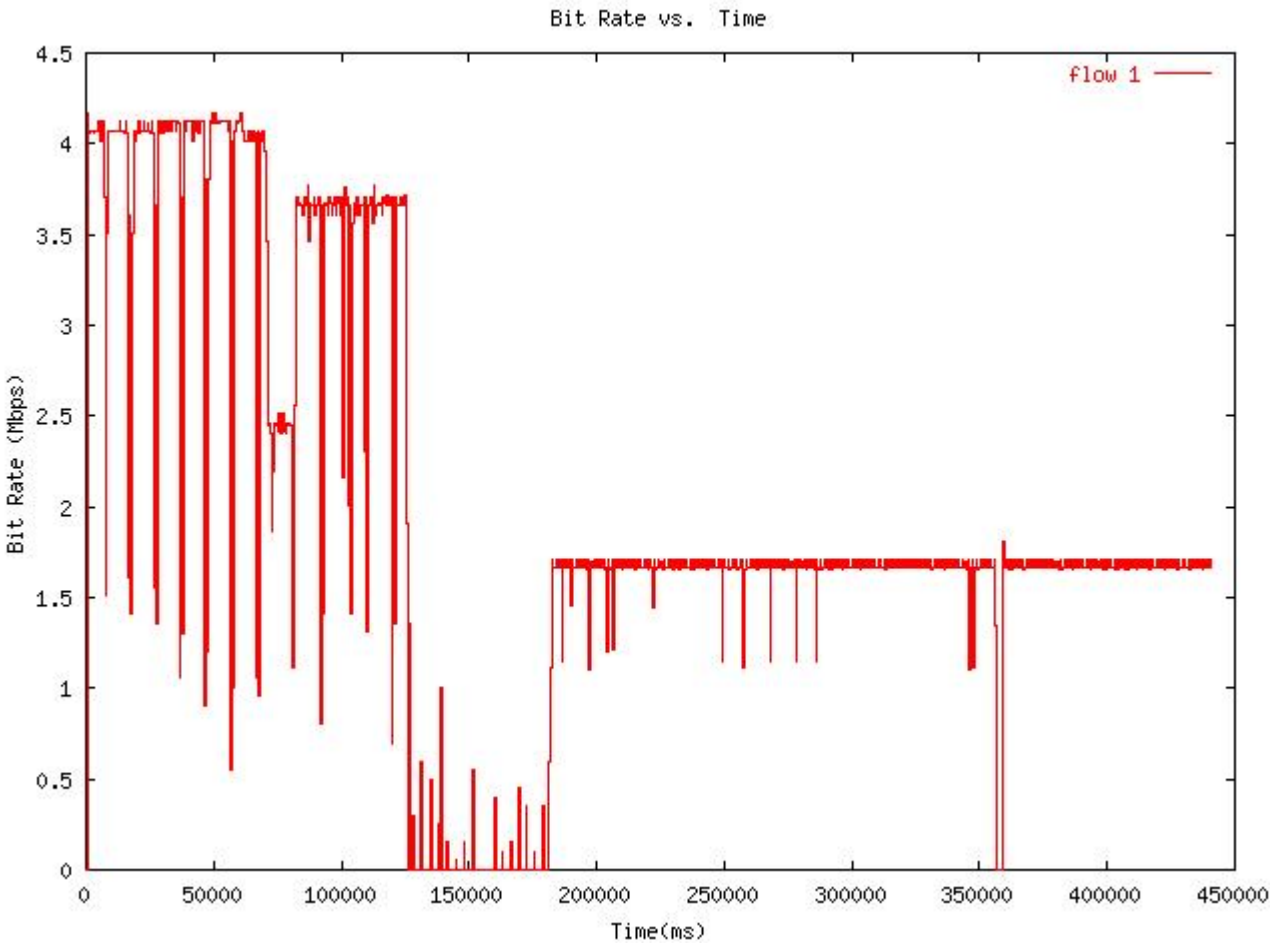


Figura 36: Gráfico da banda do tráfego HTTP a partir do arquivo de log do cliente

5.3.2.3 Estatísticas do tráfego capturado pelos LSPs

Há duas ferramentas para acompanhamento da atuação das tecnologias envolvidas nesta implementação de MPLS. O aplicativo *print_table* mostra as estatísticas de pacotes capturados e transmitidos pelos LSPs em um determinado nó de rede. Ao ser executado em um LER, o mesmo mostra os pacotes que entraram nos LSPs. O aplicativo *cbqstat* mostra as estatísticas do CBQ, que é o escalonador utilizado pelo NIST *Switch*. Uma alteração implementada pelo grupo de estudos em MPLS do DCA possibilitou que essas estatísticas sejam associadas aos LSPs estabelecidos.

Figura 37: Tráfegos prioritários (vídeo e HTTP) sendo transmitidos pelos LSPs.

Na Figura 37 pode-se ver uma tela com três execuções sucessivas do *print_table*. As estatísticas de transmissão pelos LSPs estabelecidos demonstram que os tráfegos HTTP e de vídeo são realmente encaminhados pelos LSPs:

LSP	Banda (pacotes)	Banda (pacotes)	Banda (pacotes)
HTTP response	11462	12464	13455
Vídeo response	1369	1488	1580

Tabela 9: Estatísticas de captura de pacotes pelos LSPs

5.4 Síntese

Neste capítulo são apresentados os experimentos realizados para a validação da PLEX MPLS. O cenário e a topologia de rede utilizados são descritos de forma genérica. Cada um dos experimentos realizados é apresentado em detalhes, incluindo o cenário específico, a seqüência de execução e a configuração dos fluxos de tráfego utilizados. Em seguida são apresentados os resultados obtidos. O primeiro experimento apresenta dados quantitativos dos fluxos de tráfego a da atuação da PLEX MPLS. O segundo mostra de forma qualitativa a atuação da PLEX MPLS.

6 Conclusão

A implementação de uma Plataforma para Experimentos com MPLS que fosse operacional em uma rede com tráfego real permitiu explicitar a importância de MPLS para Engenharia de Tráfego em redes IP. Como os protocolos de roteamento tradicionais não têm inteligência suficiente para realizar esquemas sofisticados de TE, congestionamentos e variações de QoS são comuns e recorrentes, prejudicando aplicações com requisitos específicos de QoS, mesmo havendo recursos de rede sub-utilizados. MPLS fornece a funcionalidade básica para solucionar este problema: a capacidade de encaminhar tráfego (baseado em seu perfil) por rotas explicitamente pré-definidas. Somando-se outras tecnologias, como reserva de recursos e implementação de políticas, é possível realizar esquemas sofisticados de TE.

Mesmo contando apenas com as funcionalidades básicas para Engenharia de Tráfego com MPLS, a PLEX MPLS comprovou a utilidade de MPLS para TE através de 2 experimentos. O primeiro experimento apresenta resultados quantitativos ao solucionar uma situação de congestionamento da rede com a aplicação de uma Engenharia de Tráfego simples que utilizou recursos sub-utilizados da rede para atender à demanda de tráfego. O segundo experimento apresenta resultados qualitativos ao prover a QoS demandada por uma aplicação com requisitos altos de QoS (vídeo).

Adicionalmente, a PLEX MPLS pode ser utilizada para fins didáticos, pois possibilita o estudo de tecnologias essenciais para TE, como MPLS, RSVP-TE, ALTQ, etc. Com a PLEX MPLS é possível constatar o resultado real da aplicação de TE em diversas situações. A implementação da PLEX também permitiu definir um conjunto mínimo de funcionalidades que uma solução para MPLS deve apresentar para ser útil em situações reais:

- Encaminhamento baseado em rótulos
- Roteamento explícito
- Reserva de recursos
- Mecanismos para agregação de tráfego, como máscaras e mapeamentos
- Tolerância a falhas, como re-roteamento rápido e LSPs de *backup*
- Algoritmos para balanceamento de carga entre LSPs paralelos

6.1 Arquitetura e metodologia para TE com MPLS

Tendo como base a experiência de implementação da PLEX MPLS e a realização dos experimentos de validação, foi possível idealizar uma arquitetura e metodologias que exploram ao máximo a potencialidade do MPLS, oferecendo benefícios reais a seus usuários. A arquitetura para Engenharia de Tráfego com MPLS possui os seguintes componentes:

- Software MPLS (NIST *Switch*)
- Protocolo de sinalização e reserva de recursos (RSVP-TE, CR-LDP)
- Software para garantir o uso correto dos recursos e classificar o tráfego (ALTQ/DiffServ)

- Protocolo para distribuição de informações do estado da rede (IGP ou SNMP modificados)
- Algoritmo para cálculo de rotas e reservas ótimas de recursos
- Algoritmo para admissão de novos fluxos e instanciação de LSPs (MIRA, CBR)
- Algoritmo para balanceamento de carga (MATE)
- Sistema integrado de políticas para TE (Broker de políticas + COPS)
- Bandwidth broker para requisição de admissão de fluxos e instanciação de LSPs
- Sistema gerenciador de banco de dados – SGBD (Oracle, MySQL, PostgreSQL)
- Interface (gráfica) com o usuário onde o administrador do sistema pode definir inúmeros parâmetros de configuração (Manager)

Uma vez estando disponível a arquitetura descrita acima, a seguinte metodologia para o uso realmente efetivo de Engenharia de Tráfego com MPLS pode ser empregada:

1. **Coleta de estatísticas de tráfego:** consiste em obter os requisitos de banda dos fluxos de tráfego, especialmente entre os diversos LERs do domínio. O objetivo é montar uma matriz de tráfego fim-a-fim. Para tanto, LSPs sem limite de banda podem ser estabelecidos entre os LERs e o tráfego passante medido. O tráfego é mapeado aos LSPs com base no próximo nó (*next hop*) do IBGP.
2. **Roteamento baseado em restrições *offline*:** deve achar rotas para todos os LSPs necessários de modo a otimizar a utilização dos recursos de rede. Este algoritmo leva em conta requisitos de banda, atributos das conexões, topologia da rede, requisitos políticos, medidas para tolerância a falhas (prioridades, resiliência, LSPs paralelos, etc), etc. Este algoritmo geralmente é executado de modo centralizado e *offline*, pois demanda alto poder computacional e tempo para grandes redes.
3. **Instanciação dos LSPs:** Os LSPs calculados no passo anterior são instanciados na rede com eventuais reservas de recurso que se façam necessárias.
4. **Operação da rede:** durante a operação normal da rede, em que os fluxos de tráfego utilizam os LSPs pré-estabelecidos e com recursos reservados. Os LSPs podem ser modificados, seja pela atuação do sistema de políticas (estabelecimento de LSPs especiais para certos tráfegos ou em certas horas do dia, etc), pela admissão de novos fluxos (estabelecimento de novos LSPs ou alteração em seus requisitos), reotimização, etc. Neste ponto, a coleta de informações do estado da rede é essencial. Nesta fase, também podem ser empregados algoritmos para balanceamento de carga entre LSPs paralelos.
5. **Coleta de estatísticas de utilização:** Para não perturbar o bom funcionamento da rede, agora as estatísticas são coletadas durante o funcionamento normal da rede, evitando assim a fase 1.
6. **Reotimização *offline* periódica:** periodicamente retorna-se à fase 2 (seguida das fases seguintes) para otimizar o uso da rede, pois as eventuais alterações *online* podem levar o sistema a um estado sub-ótimo.

6.2 Trabalho futuro

Dado o limite de recursos disponíveis para a realização deste trabalho, a implementação e conseqüente exploração da PLEX MPLS infelizmente não puderam ser efetuadas em sua plenitude. Trabalhos futuros de refinamento da PLEX MPLS, bem como de pesquisa das tecnologias MPLS e agregadas podem beneficiar-se das seguintes sugestões:

- **Manager:** O gerente da PLEX MPLS pode tornar-se muito mais útil e atraente se dispuser das seguintes funcionalidades: atualização dinâmica do estado da rede em sua GUI e apresentação dos resultados do experimento após sua conclusão
- **Agente TE:** alteração dinâmica da política de balanceamento (cujo algoritmo atual implementa apenas um *round-robin*) para incluir algoritmos mais sofisticados como MATE
- **PLEX MPLS:** capacidade de alteração das características do experimento (LSPs, geração de tráfego, etc) durante sua execução
- **Integração com trabalhos do grupo MPLS de DCA:** os vários trabalhos desenvolvidos pelos outros membros do grupo de estudos em MPLS do DCA renderam resultados (e muitas vezes implementações) muito interessantes e complementares entre si. Uma integração destes trabalhos certamente renderia uma plataforma muito completa para TE com MPLS
- **Escalabilidade:** os trabalhos desenvolvidos não puderam ser testados em grande escala. Logo, um caminho e ser seguido futuramente, seria acompanhar o comportamento destas tecnologias em larga escala.

7 Referências

[ALTQ] Alternate Queueing - http://www.csl.sony.co.jp/person/kjc/kjc/kjc/software.html

[BADAN] T. A. C. BADAN, R. C. M. PRADO, E. N. F. ZAGARI et al. “Uma Implementação do MPLS para Redes Linux”, XIX Simpósio Brasileiro de Redes de Computadores - SBRC, 2001, Florianópolis - SC. Anais do XIX Simpósio Brasileiro de Redes de Computadores - SBRC, 2001.

[CAVALCA] M. D. Cavalcante, “Algoritmos de Balanceamento de Carga para Tráfego Tipo Melhor Esforço em Redes IP/MPLS”, dissertação de mestrado, Universidade Estadual de Campinas, 2001.

[COSTA] L. Costa, “Desenvolvimento de um Bandwith Broker para a Arquitetura de Serviços Diferenciados”, dissertação de mestrado, Universidade Estadual de Campinas, 2001.

[FreeBSD] Sistema operacional FreeBSD - http://www.freebsd.org/

[GUILLEN] A. M. S. Guillen, “MPLS-DS: Uma Plataforma para Validação de Políticas no Contexto das Redes MPLS/DiffServ”, dissertação de mestrado, Universidade Estadual de Campinas, 2001.

[HIX93] D. Hix e H. R. Hartson, "Developing User Interfaces: Ensuring Usability Through Product and Process", John Wiley & Sons, 1993.

[IETF] Internet Engineering Task Force – http://www.ietf.org

[KAR00] K. Kar, M. Kodialam, T.V. Lakshman, "Minimum Interference Routing with Application to MPLS Traffic Engineering", Proc. IEEE INFOCOM, Tel-Aviv, Israel, Março 2000.

[LIMA] T. M. R. Lima, “NSPLEX-Plataforma de Validação de Algoritmos de Engenharia de Tráfego em MPLS”, dissertação de mestrado, Universidade Estadual de Campinas, 2003.

[MATE] I. Widjaja e A. Elwalid, “MATE: MPLS Adaptive Traffic Engineering”, Internet Draft, Outubro 1999.

[MCGUIRE] A. McGuire, S. Mirza e D. Freeland, "Application of Control Plane Technology to Dynamic Configuration Management", IEEE Communications Magazine, Setembro 2001.

[NIST] National Institute of Standards and Technology - http://www.antd.nist.gov/

[RFC1633] R. Braden, D. Clark e S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, Junho 1994.

[RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog e S. Jamin, “Resource Reservation Protocol (RSVP) -- Version 1 Functional Specification”, RFC 2205, Setembro 1997.

[RFC2474] K. Nichols, S. Blake, F. Baker and D. Black, “*Definition of the Differentiated Services*

Field (DS Field) in the IPv4 and IPv6 Headers", RFC2474. Dezembro de 1998.

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang e W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Dezembro 1998.

[RFC2702] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell e J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, Setembro 1999.

[RFC3031] E. Rosen, A. Viswanathan, e R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, Janeiro 2001.

[RFC3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan e G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, Dezembro 2001.

[RFC3272] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja e X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, Maio 2002

[RSVP] Resource Reservation Protocol - http://www.isi.edu/div7/rsvp/

[RUMBA1] J. Rumbaugh, M. Blaha, W. Premerlani,d F. Eddy e W. Lorensen, "Object-Oriented Modeling and Design", Prentice Hall, 1991.

[RUMBA2] J. Rumbaugh, I. Jacobson e G. Booch, "The Unified Modeling Language Reference Manual", Addison Wesley Longman, 1999.

[SWITCH] Projeto NIST Switch - http://www.antd.nist.gov/itg/nistswitch/

[SIQUEIRA] M. Siqueira, "Uma Arquitetura de Políticas Para Gerência de Redes MPLS", dissertação de mestrado, Universidade Estadual de Campinas, 2002.

[SOARES] L. F. G. Soares, G. Lemos e S. Colcher, "Redes de computadores: das LANs, MANs e WANs às redes ATM", Editora Campus, 1995.

[TG] P. E. McKenney, D. Y. Lee, B. A. Denny, "Traffic Generator Software Release Notes", SRI International e 3Com Corporation, Setembro 1998.

[XIAO1] X. Xiao e L. Ni, "Internet QoS: The Big Picture", IEEE Network Magazine, Março 1999.

[XIAO2] X. Xiao, A. Hannan, B. Bailey e L. Ni, "Traffic Engineering with MPLS in the Internet", IEEE Network Magazine, Março 2000.

[ZAGARI] E. N. F. ZAGARI, T. A. C. BADAN, R. C. M. PRADO et al, "Uma Plataforma para Engenharia de Tráfego com Qualidade de Serviço em Redes MPLS", XX Simpósio Brasileiro de Redes de Computadores - SBRC, 2002, Búzios - RJ. Anais do XX Simpósio Brasileiro de Redes de Computadores - SBRC, 2002.