UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM

CIÊNCIA DA COMPUTAÇÃO

Engenharia de Tráfego para Obtenção de QoS na Comunicação entre Tarefas em Grades Computacionais

Guilherme Mundim Torres

São Carlos
Dezembro/2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

T693et

Torres, Guilherme Mundim.
Engenharia de tráfego para obtenção de QoS na comunicação entre tarefas em grades computacionais / Guilherme Mundim Torres. -- São Carlos : UFSCar, 2008.
110 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Redes de comunicação de dados. 2. Engenharia de tráfego. 3. Qualidade de serviço. 4. Grade computacional. 5. Multi-protocol label switching. I. Título.

CDD: 004.62 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Engenharia de Tráfego para Obtenção de QoS na Comunicação entre Tarefas em Grades Computacionais"

GUILHERME MUNDIM TORES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Hélio Crestana Guardia

Prof. Dr. Hélio Crestana Guardia
(Orientador – DC/UFSCar)

Prof. Dr. Luis Carlos Trevelin
(DC/UFSCar)

Sérgio Takeo Kofuji

Prof. Dr. Sérgio Takeo Kofuji
(POLI/USP)

São Carlos
Dezembro/2006

Resumo

O surgimento da computação em grade possibilitou o acesso a recursos distribuídos que podem estar dispersos geograficamente e pertencer a diferentes organizações. O meio mais utilizado para prover acesso a tais recursos é a Internet, uma rede de computadores de alcance mundial baseada na arquitetura TCP/IP.

As grades computacionais fornecem a infra-estrutura necessária à comunicação e ao gerenciamento dos recursos fornecidos por estas organizações, também conhecidas por organizações virtuais (VOs) .

Algumas das aplicações utilizadas nestes ambientes colaborativos podem possuir requisitos mínimos de qualidade de serviço (QoS). Entretanto, o serviço de "melhor esforço" oferecido pela Internet não é capaz de satisfazer tais exigências, sendo preciso utilizar outra forma para se obter garantias em relação à capacidade de tráfego dos canais de comunicação.

Este trabalho de mestrado objetiva aplicar os conceitos de qualidade de serviço de redes para o provimento de qualidade de serviço fim-a-fim nas comunicações entre aplicações para grades computacionais.

Para tanto, investiga o uso da infra-estrutura de comutação provida pelas redes MPLS. Usando mecanismos de determinação de rotas em Engenharia de Tráfego, busca-se prover melhor controle dos fluxos de dados, beneficiando aplicações distribuídas em ambientes de grande dispersão física.

Palavras chaves: Engenharia de tráfego, MPLS, QoS, Grades computacionais

Abstract

The advent of grid computing made possible to access distributed resources, even when they are geographically spread or belong to different organizations. The most used environment for accessing these distributed resources is the Internet, a worldwide computer network based in TCP/IP architecture. Grid computing provides the infrastructure necessary for managing and communicating with the resources offered by different organizations. These organizations are also known as virtual organizations (VO's).

Some of the applications used in these collaborating environments may have minimum requirements by quality of service (QoS). However, the "best effort" service, which is offered by Internet, is not capable to satisfy these QoS requirements. In this case, a different solution is needed, in order to provide guarantees related to the traffic in communication channels.

This master thesis aims to apply the concepts of quality of service for networks in grid computing, providing end-to-end quality of service between grid computing applications. In order to achieve this goal, we investigate the use of commutation infrastructure provided by MPLS networks. Using traffic engineering mechanisms for routes determination, we aim to provide better control of data flows, improving the performance of distributed applications in geographically highly spread environments.

Keywords: Traffic engineering, MPLS, QoS, Grids

Agradecimentos

Agradeço primeiramente a meus pais pelos bons exemplos de vida que contribuíram para minha formação. A minha namorada Camila, pelo apoio nos momentos difíceis e compreensão com relação ao tempo que deixei de passar com ela por estar envolvido com as pesquisas. Agradeço ao meu orientador, Prof. Hélio Crestana Guardia, pelo acompanhamento constante, pelo compartilhamento de seu conhecimento e pela compreensão quando precisei dividir meu tempo entre o mestrado e o trabalho. Aos meus colegas de trabalho, pelo apoio, especialmente ao Luiz Carlos Dotta, responsável pela Seção Técnica de Informática do Instituto de Ciências Matemáticas de Computação da USP, por ter facilitado a conciliação das atividades do trabalho e do mestrado. Ao Erlon, pelo apoio dado na configuração das topologias de testes e codificação da ferramenta proposta por este trabalho. Aos meus colegas de laboratório, pelas sugestões e momentos de descontração que ajudaram a suportar as dificuldades do dia a dia. Por fim, aos meus vizinhos, que contribuíram com cenas hilárias, ajudando a manter o bom humor e o ânimo a cada dia.

Dedico esta conquista a toda minha família.

Lista de Figuras

Lista de tabelas

Acrônimos

AF	*Assured Forwarding*
ASIC	*Application Specific Integrated Circuit*
ATM	*Asynchronous Transfer Mode*
AToM	*Any Transport over MPLS*
BE	*Best Effort*
BGP	*Border Gateway Protocol*
CORBA	*Common Object Resource Broker Architecture*
CRC	*Cyclic Redundancy Check*
DCOM	*Distributed Component Object Model*
DSCP	*Diffservice Code Point*
ECN	*Explicity Congestion Notification*
EF	*Expedited Forwarding*
E-LSP	*EXP-inferred-PSC LSP*
E-LSP	*EXP-inferred-PSC LSP*
EXP	*Experimental*
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo
FEC	*Forwarding Equivalence Class*
FTN	*FEC to NHLFE*
GARA	*Globus Architecture for Reservation and Allocation*
GIIS	*Grid Index Information Service*
GPL	*General Public License*
GRAM	*Grid Resource Allocation Management*
GRIS	*Grid Resource Information Service*
GSH	*Grid Service Handle*
GSI	*Globus Secutiry Infrastructure*

GSR	*Grid Service Reference*
GT	*Globus Toolkit*
IDL	*Interface Definition Language*
IETF	*Internet Engineering Task Force*
ILM	*Incoming Label Map*
IP	*Internet Protocol*
LDAP	*Lightweight Directory Access Protocol*
LDP	*Label Distribution Protocol*
LER	*Label Edge Router*
LGPL	*Library General Public License*
LIB	*Label Information Base*
LIFO	*Last In First Out*
L-LSP	*LABEL-inferred-PSC LSP*
L-LSP	*LABEL-inferred-PSC LSP*
LSP	*Label Switched Path*
LSR	*Label Switching Router*
MDS	*Monitoring and Discovery Service*
MIB	*Management Information Base*
MPLS	*Multi-Protocol Label Switching*
NHLFE	*Next Hop Label Forward Entry*
NM	*Network Meter*
NWS	*Network Weather Service*
OGSA	*Open Grid Services Architetecture*
OGSI	*Open Grid Services Infrastructure*
PBNM	*Policy Based Network Management*
PSC	*PHB Scheduling Class*
PVM	*Parallel Virtual Machine*

QOS	*Quality of Service*
RFC	*Request for Comments*
RFT	*Reliable File Transfer*
RPM	*Red hat Package Manager*
RSL	*Resource Specification Language*
RSVP	*Resource Reservation Protocol*
S	Segundos
SNMP	*Simple Network Management Protocol*
SOAP	*Simple Object Access Protocol*
SSH	*Secure Shell*
TCP	*Transmission Control Protocol*
TE	*Traffic Engineering*
TIDIA	Tecnologia da Informação no Desenvolvimento da Internet avançada
TOS	*Type Of Service*
TTL	*Time To Live*
UDP	*User Datagram Protocol*
UFRGS	Universidade Federal do Rio Grande do Sul
URL	*Uniform Resource Locator*
VC	*Virtual Circuit*
VCI	*Virtual Circuit Identifier*
VPN	*Virtual Private Networks*
WAN	*Wide Area Network*
WSDL	*Web Services Description Language*
WSDM	*Web Services Distribution Management*
WSRF	*Web Service Resource Framework*
XML	*eXtensible Markup Language*

Sumário

1. Introdução

As primeiras tentativas de suprir as necessidades das aplicações que requerem alta capacidade de processamento foram obtidas através de sistemas computacionais centralizados. O aumento da capacidade computacional era obtido principalmente através da construção de computadores de grande porte, também conhecidos por *mainframes*.

Dotados de vários processadores, grande quantidade de memória e alta capacidade de armazenamento em disco, estas máquinas ainda podem ser encontradas em operação. Embora consigam aumentar significativamente o poder computacional quando comparadas a computadores pessoais, possuem um custo muito elevado devido à alta complexidade de sua arquitetura. A necessidade de compartilhar memória entre vários processadores e dissipar grande quantidade de calor são apenas alguns dos fatores que dificultam e oneram seu desenvolvimento [PH1998].

O avanço das redes de computadores aliado ao baixo custo das máquinas destinadas à computação pessoal permitiu o desenvolvimento de sistemas distribuídos. Um exemplo de sistemas desse tipo são os *clusters,* sendo que maiores informações sobre seu funcionamento podem ser obtidas em [Beo2004], projeto de muita importância nesta área.

Os sistemas distribuídos são baseados na interligação de diversos recursos disponíveis através de uma rede local. As máquinas, também chamadas de nós, comunicam-se através da troca de mensagens realizada por bibliotecas destinadas a este fim, como pode ser visto em [GL1995] e [PVM2005].

A principal vantagem dos *clusters* em relação aos *mainframes* está no baixo custo envolvido em sua construção. Além do emprego de *hardware* não proprietário, ainda pode ser utilizado *software* de código aberto, como o Linux.

O crescimento da Internet, juntamente com o desenvolvimento da computação distribuída, possibilitou o surgimento de um novo conceito ainda mais amplo relacionado à execução remota de tarefas. Foram iniciadas pesquisas na área de grades computacionais

(*Grid Computing*), inicialmente visando a interligação de supercomputadores separados por grandes distâncias no globo terrestre [FKT2001].

Com o uso das grades, usuários passam a ter acesso a recursos que podem estar dispersos geograficamente, interligados por um ambiente distribuído e, na maioria das vezes, heterogêneo.

Devido à tecnologia empregada nas redes de longa distância, que oferecem taxa de transmissão normalmente inferiores àquelas existentes em redes locais e *clusters*, as aplicações paralelas que mais se beneficiam das características das grades computacionais são aquelas que não necessitam de muita comunicação entre as partes, conhecidas por aplicações *Bag-of-Tasks*.

Embora a rede possa significar um gargalo também para aplicações que façam uso contínuo de comunicação num ambiente distribuído, esforços nas áreas de qualidade de serviço (QoS) procuram obter diferentes garantias na utilização de rede.

Dependendo do contexto em que for aplicada, a definição de QoS pode adquirir diferentes significados.

Em [ATM FORUM 96], QoS é definida como sendo um conjunto de parâmetros de desempenho relativos a atraso, variação do atraso e taxa de perda de células que podem ser negociados por parte do usuário no estabelecimento de um contrato de tráfego.

Segundo [ISO95], a QoS representa um conjunto de qualidades relacionadas ao comportamento coletivo de um ou mais objetos.

No contexto deste trabalho, o termo QoS está relacionado com a habilidade que a rede tem de garantir que determinados fluxos de dados sejam tratados de forma diferenciada pelos roteadores de uma rede.

Através das informações obtidas com uso de mecanismos de monitoração e previsão de carga, como pode ser visto em [WSH1999], é possível priorizar determinados tipos de tráfego, visando obter qualidade de serviço fim-a-fim. Em alguns casos, agentes SNMP podem ser usados na configuração dinâmica dos elementos de comutação de dados envolvidos.

Nesse sentido, apresentam-se as características relativas ao modelo de Serviços Integrados (Intservice), Serviços Diferenciados (*Diffservice*) e do MPLS (*Multi Protocol Label Swichting*), principais tecnologias existentes para obtenção de qualidade de serviço no nível de rede e enlace.

Em um ambiente de grade computacional, a ação de se distribuir as tarefas entre diferentes nós afim de que sejam processadas é chamada de escalonamento. Um bom escalonamento implica diretamente num menor tempo de execução necessário para se realizar determinada tarefa. Caso um escalonador de tarefas leve em consideração o estado geral de utilização de uma rede a partir de dados provenientes do monitoramento do estado dos enlaces, é possível que ele faça melhor uso dos canais de comunicação, o que resulta em ganho de tempo para processamento das tarefas.

Usando mecanismos de controle de encaminhamento de pacotes na rede, esse trabalho propõe avaliar aspectos de QoS na transmissão de dados como suporte para escalonamento de aplicações com comunicação em grades computacionais.

Para tanto, investiga-se primordialmente o uso de técnicas de Engenharia de Tráfego nas transmissões de dados de diferentes fluxos, como forma de prover Qualidade de Serviço em função da monitoração dos canais de comunicação existentes.

A validação do trabalho realizado será obtida através da inclusão de mecanismos de qualidade de serviço em uma rede ligada ao projeto KyaTera, que visa o estudo, desenvolvimento, demonstração e implantação de uma rede de fibras ópticas para interligar centros de pesquisa de excelência no estado de São Paulo. Esta rede, destinada ao meio acadêmico, será capaz de fornecer alta velocidade às suas aplicações [Kya2005].

O projeto KyaTera pertence ao projeto TIDIA (Tecnologia da Informação no Desenvolvimento da Internet Avançada) [Tid2005], sendo financiado pela FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo).

No próximo capítulo são apresentadas as principais características da computação em grade, sua arquitetura, tipos de serviços existentes, especificações e o atual estado da arte desta tecnologia.

No capítulo 3 são apresentados os principais projetos que buscam obter qualidade de serviço em ambientes de grade, sendo evidenciadas suas principais características e demais aspectos relevantes.

No Capítulo 4 são apresentadas as principais formas utilizadas para se obter qualidade de serviço nas camadas de rede e enlace das redes de computadores.

No Capítulo 5 são abordadas as principais características, princípios de funcionamento e aplicações do protocolo MPLS.

O Capítulo 6 trata da ferramenta proposta por esta dissertação de mestrado, destinada ao fornecimento de Qualidade de Serviço a aplicações voltadas para ambientes de grade computacional.

No Capítulo 7 são demonstrados os resultados obtidos por uma série de experimentos realizados em laboratório. Buscou-se avaliar o desempenho obtido pela utilização do protocolo IP sobre MPLS, bem como sua capacidade de direcionar fluxos de dados por meio de técnicas de Engenharia de Tráfego.

O Capítulo 8 aborda as conclusões e trabalhos futuros.

Por fim, os apêndices trazem informações mais técnicas e demais materiais de apoio que auxiliam no entendimento do trabalho realizado nesta dissertação de mestrado.

2. Computação em Grade

As primeiras grades computacionais surgiram em ambientes acadêmicos e visavam atender às necessidades da comunidade científica. Atualmente, os resultados obtidos com o desenvolvimento dessa tecnologia também estão presentes e atendem a interesses e necessidades comerciais. Por permitir redução de custos, redução do tempo necessário para se realizar uma tarefa, aumento de produtividade e possibilitar compartilhamento de recursos e informações, o futuro da computação em grade é bastante promissor [FKT2001].

A computação em grade é caracterizada pela existência de ambientes colaborativos nos quais usuários tanto disponibilizam quanto fazem uso de recursos e serviços.

Uma grade computacional permite uma utilização mais racional dos recursos computacionais existentes em cada instituição que dela faça parte. Uma de suas principais vantagens está relacionada ao aproveitamento de poder computacional ocioso no processamento de aplicações.

A diferença de fuso-horário entre países separados por grandes distâncias pode ser explorada para exemplificar sua utilização. Empresas situadas no ocidente poderiam processar suas aplicações em instituições situadas no oriente, e vice-versa, já que a maior parte dos usuários de cada localidade faz uso de seus recursos computacionais em momentos diferentes.

Assim como nos sistemas distribuídos, as grades se apóiam sobre uma camada de *software* (*middleware*) responsável por ocultar do usuário a alta complexidade existente em seu interior. O *middleware* é responsável por passar aos usuários da grade a imagem de um sistema único [TS2002].

Do ponto de vista do usuário, não importa, por exemplo, o modelo de sistema operacional que cada máquina está utilizando, o local em que cada máquina está situada e em qual delas sua aplicação será executada. O que interessa é que, após a submissão da aplicação na grade, a resposta chegue em tempo satisfatório. A transparência em relação à complexidade das grades é de fundamental importância para o sucesso e continuidade de seu desenvolvimento.

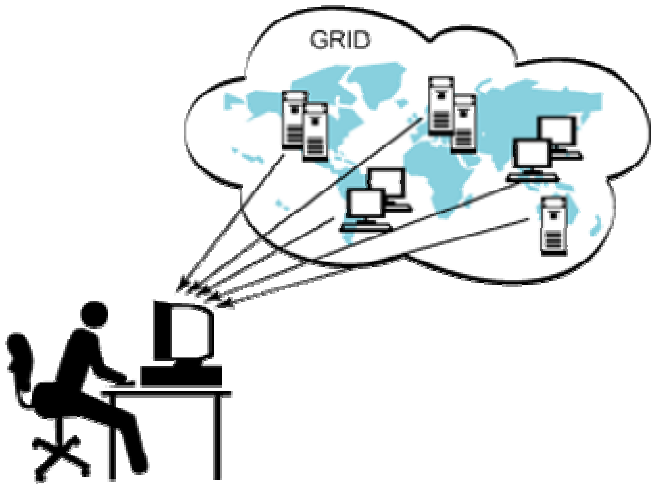

Figura 1: A grade do ponto de vista do usuário [FBD2004].

A Figura 1 ilustra a visão que o usuário tem da grade. Do seu ponto de vista, ele possui um computador virtual de grandes capacidades sobre sua mesa. Em seu interior, um conjunto de padrões e protocolos possibilitam a execução de tarefas em um ambiente heterogêneo e disperso geograficamente.

O termo "*grid*" foi criado a partir dessa transparência, utilizando uma analogia em relação ao termo "*Electrical Power Grid*". No momento em que uma pessoa liga um aparelho elétrico na tomada, ela não precisa ter conhecimento do modo como a força necessária ao seu funcionamento foi gerada. Para o consumidor não faz diferença se foi uma usina nuclear, uma hidrelétrica ou termoelétrica que gerou aquela energia. O consumidor também não tem conhecimento de como ela chega até a sua casa, ou seja, não é necessário conhecer a infra-estrutura de distribuição utilizada pela companhia energética.

As grades funcionam de maneira bem similar ao modelo de distribuição de energia. Ao se conectar à grade, o usuário sabe que dispõe de poder computacional para realizar suas tarefas. A grade é a infraestrutura responsável por prover ao usuário os elementos necessários para que sua aplicação seja executada, da forma mais transparente possível. Maiores detalhes sobre a analogia descrita acima podem ser obtidos em [CB2002].

O dinamismo e a natureza descentralizada desse ambiente são fatores que também dificultam a implementação de mecanismos de segurança, principalmente em relação à

autenticação e ao controle de acesso dos usuários aos recursos. Muitas vezes, os recursos disponibilizados pelas instituições são de grande valia, e por isso não se pode conceber a má utilização dos mesmos. Uma grade deve ser capaz de prover segurança neste meio distribuído, caracterizado por ser escalável e altamente dinâmico.

2.1 Organizações virtuais (VOs)

A partir da interligação entre os múltiplos domínios administrativos que podem constituir uma grade, foi criado o conceito de organizações virtuais (*Virtual Organizations* - VOs). Uma VO se baseia na definição de regras que permitem a obtenção de um compartilhamento de recursos flexível, seguro e coordenado entre conjuntos dinâmicos de indivíduos e instituições [FKT2001].

Uma organização virtual pode ser formada por diferentes instituições que tanto disponibilizam quanto fazem uso de recursos e serviços presentes na grade. Cada instituição pode possuir características particulares relacionadas às políticas e aos meios de controle e segurança. Uma grade deve prover meios que possibilitem a interoperabilidade entre as instituições, sem interferir na individualidade de cada uma delas.

2.2 Aspectos relevantes da computação em grade

Como pode ser visto em [BBL2002], os principais aspectos que podem caracterizar uma grade computacional são:

- **Múltiplos domínios administrativos autônomos:** Os recursos de uma grade computacional se encontram geograficamente distribuídos através de múltiplos domínios administrativos pertencentes a diferentes organizações.
- **Heterogeneidade:** Uma grade computacional pode ser formada por um grande número de recursos envolvendo diversos tipos de tecnologias diferentes.

- **Escalabilidade:** Uma grade deve ser escalável, ou seja, deve ser capaz de expandir de um número pequeno de recursos interconectados até uma grande quantidade deles, preservando os investimentos feitos anteriormente. Em conseqüência disso, aplicações que requeiram um grande número de recursos geograficamente dispersos devem ser capazes de lidar com restrições de latência e capacidade do canal de comunicação.
- **Dinamicidade ou adaptabilidade:** Em uma grade computacional, a possibilidade de falha deve ser considerada uma regra, não exceção. De fato, devido ao grande número de recursos envolvidos em uma computação, a probabilidade de que algum deles apresente comportamento anômalo é alta. Os gerenciadores de recursos ou aplicações devem ser capazes de lidar com isso, utilizando-os de da forma mais eficiente possível.

2.3 Serviços fornecidos pelas grades

Quando um usuário se conecta a uma grade, ele passa a interagir com o ambiente colaborativo fornecido por esta através dos gerenciadores de recursos (*resources brokers)*. Estes são responsáveis pela descoberta, agendamento e processamento das aplicações nos recursos distribuídos [BBL2002].

Do ponto de vista do usuário, uma grade pode prover os seguintes serviços [BBL2002]:

- **Serviços Computacionais:** Estão relacionados ao fornecimento de serviços seguros destinados à execução de tarefas distribuídas nos recursos de forma individual ou coletiva. Também conhecidas por grades computacionais, podem ser exemplificadas por: NASA IPG [JGN1999], World-Wide Grid [Buy2003] e NSF TeraGrid [Ter2005].
- **Serviços de dados:** Fornecem acesso e gerenciamento seguro das informações armazenadas em bancos de dados distribuídos. Também conhecidos por grades de

dados (*data grids*), são muito utilizados no desenvolvimento de novos medicamentos e pela física nuclear de alta energia.

- **Serviços de aplicação:** Estão ligados ao gerenciamento de aplicações, permitindo que programas e bibliotecas sejam acessados remotamente de forma transparente. O recente avanço das tecnologias baseadas em w*eb services* tem conferido grande importância a esses tipos de serviços.
- **Serviços de informação:** São destinados à extração e apresentação de dados, podendo fazer uso de um ou mais serviços computacionais, de dados ou de aplicação.
- **Serviços de conhecimento:** Estão relacionados com a maneira pela qual o conhecimento é adquirido, utilizado, divulgado e mantido, com objetivo de ajudar os usuários a atingir seus objetivos. Aplicações baseadas em mineração de dados são um exemplo desse tipo de serviço.

Na classe dos serviços de aplicação, os portais [Zor2005] têm adquirido cada vez mais importância nos ambientes colaborativos baseados em grades. Sua principal característica é possibilitar acesso seguro e amigável do usuário à grade, sem que este tenha que instalar ou configurar novos aplicativos em seu computador.

2.4 Arquitetura aberta de serviços para grade (OGSA)

A OGSA (*Open Grid Services Architecture*) [Fos2002] é uma proposta de arquitetura padronizada para computação em grade desenvolvida pelo *Global Grid Forum* [GGF2005] que busca definir interfaces de programação, gerenciamento, convenções de nomes e diretórios de forma a aumentar o grau de convergência entre computação em Grade e serviços da w*eb* (w*eb services*).

Além disso, a criação de um padrão aberto é de fundamental importância para as grades computacionais, pois visa permitir a interoperabilidade entre elas.

A primeira implementação dos padrões definidos pela OGSA pôde ser vista no *Globus Toolkit* versão 3. A OGSA adiciona o conceito de serviços às grades computacionais. Estes podem ser acessados tanto por usuários quanto por aplicações, por intermédio de trocas de mensagens.

O protocolo padrão para troca de mensagens é o SOAP (*Simple Object Access Protocol*). Através dele as entidades requisitam serviços aos provedores, utilizando o protocolo HTTP como meio de transporte [Fos2002].

É importante salientar que, embora a arquitetura baseada em serviços *web* tenha sido escolhida como a melhor opção, nem todas suas características satisfaziam as necessidades do OGSA. Assim, seu conceito foi estendido, e quando utilizados em grades computacionais, recebem o nome de serviços de grade (*Grid Services*).

Sucintamente, um serviço de grade é um serviço *web* que respeita um conjunto de convenções (interfaces e comportamento), definindo a forma através da qual um cliente deve interagir com o serviço de grade [TCF2002].

Como a OGSA não especificava muito detalhadamente os serviços de grade, o *Global Grid Forum* decidiu criar um segundo padrão, denominado OGSI (*Open Grid Services Infrastructure*), que fornece uma especificação mais técnica e formal [Sot2003].

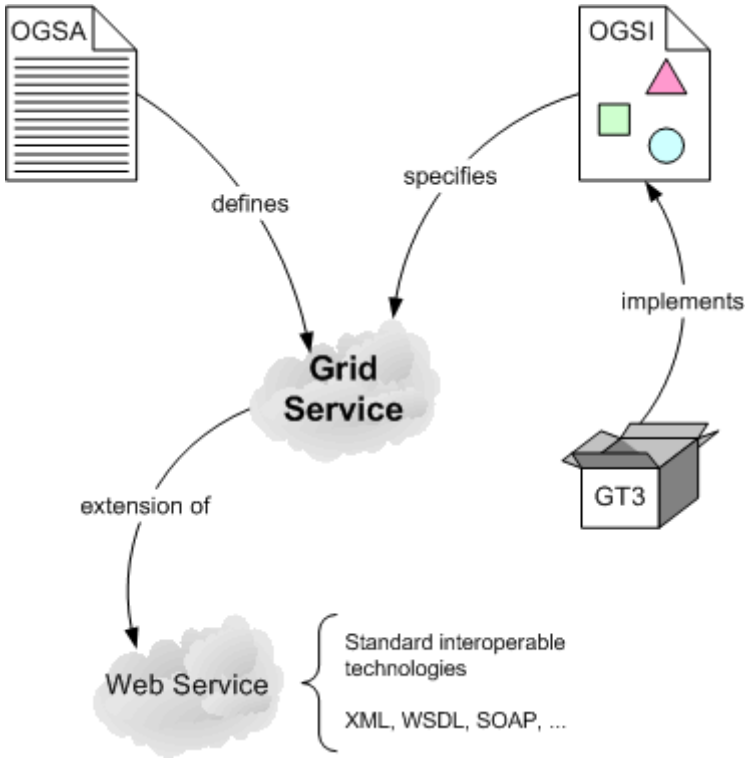

Figura 2: Serviços de grade [Sot2003]

A Figura 2 ilustra a relação existente entre os padrões OGSA e OGSI, bem como a relação existente entre os serviços de grade e os serviços *web*.

Os serviços de grade são descritos usando uma linguagem específica, denominada WSDL (*Web Services Description Language*) [Chr2001]. Um arquivo do tipo WSDL corresponde a um documento XML que contém um conjunto de mensagens no padrão SOAP e especifica a forma como estas mensagens serão trocadas.

Em sua essência, os serviços de grade não diferem muito de outras soluções utilizadas para o acesso a recursos em computação distribuída, como é o caso de CORBA [Cor2004] e DCOM [HK1997]. No caso destes últimos, a troca de mensagens era especificada pela IDL (*Interface Definition Language*), ao invés da WSDL.

O acesso a um serviço de grade é realizado pelo GSH (*Grid Service Handle*). O GSH consiste em um endereço parecido com uma URL comum, responsável por informar a localização do serviço.

Embora o GSH consiga localizar um serviço, é necessário um segundo componente para que o acesso de fato possa se concretizar. Para determinar os métodos que os serviços implementam e a forma como podem ser acessados, é necessário utilizar o GSR (*Grid Service Reference*). Este consiste em um arquivo do tipo WSDL, responsável por descrever os serviços [Sot2003].

Com a evolução da arquitetura de serviços *web*, a especificação OGSI precisou ser refinada. A tendência é que os mecanismos de endereçamento de serviços descritos acima (GSH e GSR) sejam substituídos por um novo mecanismo, o *WS-Addressing*, cujo funcionamento independe da camada de transporte [Kar2004].

A modificação de outras características do OGSI visando acompanhar o desenvolvimento dos serviços *web* culminou no desenvolvimento do WSRF (*Web Service Resource Framework*) [WE2005]. Este utiliza o mecanismo de *WS-Addressing* descrito acima, e já é utilizado pela versão 4 do *Globus Toolkit*.

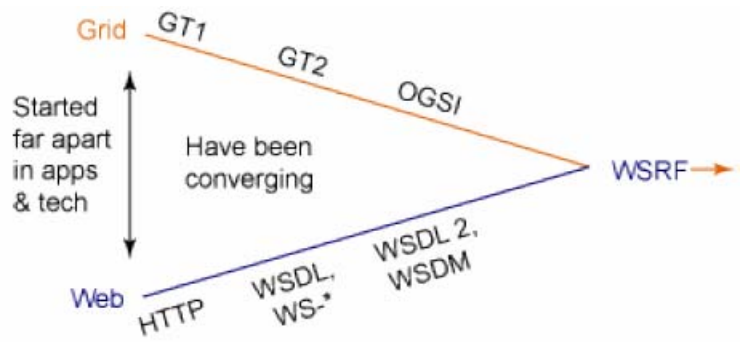

Figura 3: Convergência entre grades computacionais e serviços *Web* [FBD2004]

A Figura 3 ilustra o alto grau de convergência entre as grades computacionais e os serviços *web*.

2.5 Globus

Um *middleware* que tem sido largamente utilizado em ambientes de computação em grade é o *Globus Toolkit* [GT2005]. Ele é composto por um conjunto de serviços e bibliotecas de código livre que oferecem suporte às grades computacionais e suas aplicações [Fos2002].

O *Globus toolkit* permite que pessoas compartilhem poder computacional, dados e outros recursos. Este compartilhamento é realizado de maneira segura, podendo envolver instituições separadas por grandes distâncias sem sacrificar a autonomia local de cada uma delas. Faz parte do *Globus toolkit* um conjunto de serviços e bibliotecas capazes de monitorar, localizar e gerenciar recursos dispersos pela grade.

O grande sucesso obtido por este projeto, somado aos altos investimentos realizados em seu desenvolvimento, faz dele uma base sobre a qual muitas empresas e instituições de pesquisa têm incorporado novas funcionalidades.

2.5.1 Pilares de funcionamento do *Globus toolkit*

O desenvolvimento das grades computacionais depende de alguns serviços essenciais ao seu funcionamento. No *Globus toolkit* podem ser identificados três serviços que podem ser considerados fundamentais. São eles: GRAM, MDS, GRID FTP e GSI. Suas principais características serão vistas a seguir.

2.5.1.1 GRAM (*Grid Resource Allocation Management*)

O GRAM realiza a tarefa de gerenciamento dos recursos fornecidos pela grade. Em suma, este serviço provê suporte à alocação de recursos, submissão e execução de tarefas.

A arquitetura do GRAM utiliza uma linguagem de especificação de recursos (RSL – *Resource Specification Language*) na comunicação entre as aplicações e os escalonadores de recursos (*resource brokers*). Através da RSL [RSL2005] são passadas as informações

necessárias para a execução das tarefas, tais como: entrada padrão a ser utilizada, saída padrão a ser utilizada, máximo de CPU que o processo pode utilizar, dentre outras.

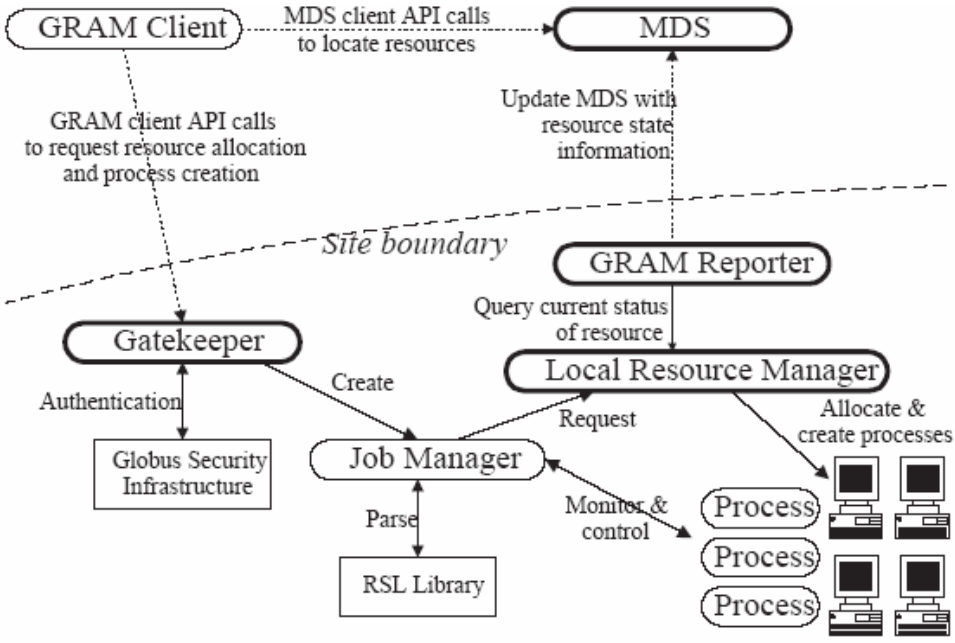

Figura 4: Arquitetura do GRAM [Cza1998]

A Figura 4 ilustra a arquitetura do GRAM, permitindo que sejam identificados seus principais componentes e respectivas funções. O cliente GRAM possibilita que os usuários e aplicações submetam e controlem a execução de tarefas na grade. Ele fornece a abstração necessária para que seus usuários não tenham que se preocupar em definir qual será o escalonador local de recurso (*Local Resource Manager*) a ser utilizado.

O *Gatekeeper* é responsável por autenticar o usuário junto ao serviço de segurança do *Globus toolkit* (GSI). Caso o processo de autenticação seja bem sucedido, é inicializado o gerenciador de tarefas (*job manager*) que, por sua vez, interpreta a requisição feita e em RSL e a repassa a um gerenciador local de recursos para que sejam criados os processos necessários à sua execução.

Desde o início até o término do processo descrito acima o serviço MDS é constantemente atualizado com informações a respeito do status dos recursos.

Como pôde ser visto, ao receber uma requisição de recurso, o GRAM deve decidir entre aceitá-la ou não. Esta decisão é tomada com base tanto na disponibilidade dos recursos quanto na verificação das credenciais fornecidas pelo usuário.

2.5.1.2 MDS (*Monitoring and Discovery Service*)

O MDS [Cza2002] é um serviço de informação, responsável pela disponibilização e descoberta de *status* (dados dinâmicos) e configuração (dados estáticos) dos recursos da grade. Através do MDS são realizadas consultas para descobrir propriedades de computadores, redes e recursos em geral. Ele pode levantar dados como quantidade de processadores disponíveis, largura de banda provida e qual o meio de armazenamento utilizado.

A implementação do MDS se baseia no LDAP (*Lightweight Directory Access Protocol*), um protocolo de serviços de diretório que trabalha na camada de aplicação da pilha de protocolos TCP/IP [HS1995].

O MDS é composto por três componentes principais. São eles:

- **Provedores de informação (*Information Providers*):** Responsáveis pela coleta de informações sobre determinado recurso.
- **GRIS (*Grid Resource Information Service*):** reúne informações locais, como por exemplo: espaço disponível em disco, sistema operacional utilizado, quantidade de memória livre, entre outras.
- **GIIS (*Grid Index Information Service*)**: está situado numa posição mais alta na hierarquia de diretórios provida pelo LDAP, reunindo em um único servidor as informações disponibilizadas por vários GRISs.

2.5.1.3 GRID FTP (*Grid File Transfer Protocol*)

Em ambientes de computação de grade que façam uso de grandes quantidades de dados distribuídos por uma rede de longo alcance, pode ser necessário realizar transferência de informações com bastante freqüência. Para que isso seja possível, é necessário haver um serviço destinado a realizar tais transferências, de forma segura e em altas velocidades.

O GRID FTP destina-se a tratar do problema descrito acima, podendo ser considerado um serviço de gerenciamento de dados. Ao contrário dos serviços de escalonamento de recursos e de informação vistos anteriormente, sua utilização é opcional.

O *Globus toolkit* versão 3 adicionou um novo serviço de transferência de arquivos com o intuito de atender às especificações do OGSA. Apesar da finalidade ser a mesma, o serviço de transferência confiável (RFT – *Reliable File Transfer*) adiciona funcionalidades, principalmente em relação à confiabilidade.

Em suma, seus mecanismos permitem a realização de transferência de dados em ambientes de alto desempenho, de forma segura e confiável. Da mesma forma que nos outros pilares, a segurança é obtida através do GSI. Já a confiabilidade é obtida através da utilização de um banco de dados compatível com JDBC (*Java DataBase Connectivity*), como é o caso do Postgresql. O banco de dados armazena os segmentos de arquivos durante a transferência e permite a continuidade da transação a partir do ponto em que esta tenha sido interrompida, caso haja algum problema.

Maiores detalhes em relação a este serviço podem ser obtidos em [ALL2005].

2.5.1.4 GSI (*Globus Security Infrastructure*)

Uma das características das grades computacionais é a existência de múltiplos domínios administrativos. Para que um usuário da grade possa utilizar os recursos disponíveis por ela, é necessário que ele se autentique junto ao recurso, seja este local ou remoto.

Além disso, dependendo da sua complexidade, o processamento de uma tarefa na grade pode levar um tempo considerável de execução. Durante esse período, pode ser

necessário que o usuário ou a aplicação seja autenticado por diversos recursos mais de uma vez.

Para tratar problemas desse tipo foi criado o GSI. Baseado em padrões de criptografia de chave pública, ele atende a características desejáveis em ambientes de grade, tais como *single sign-on*, delegação e mapeamento de identificações.

O GSI versão 3 foi o primeiro a incorporar conceitos de w*eb services*, facilitando a publicação das políticas de segurança no ambiente de grade e permitindo que as aplicações consigam atender a tais políticas de forma automatizada [Wel2003].

Conforme ilustrado na Figura 5, o GSI representa a base dos pilares anteriormente abordados, sendo responsável por prover uma infra-estrutura de segurança aos serviços da grade de forma transparente.

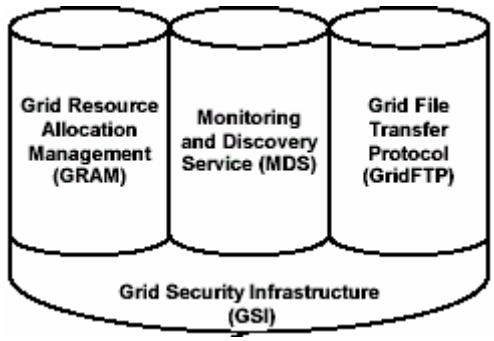

Figura 5: Pilares do *Globus toolkit* [Fer2003]

2.5.2 Globus toolkit 4

A quarta versão do *Globus toolkit* foi a primeira a implementar as características do WSRF.

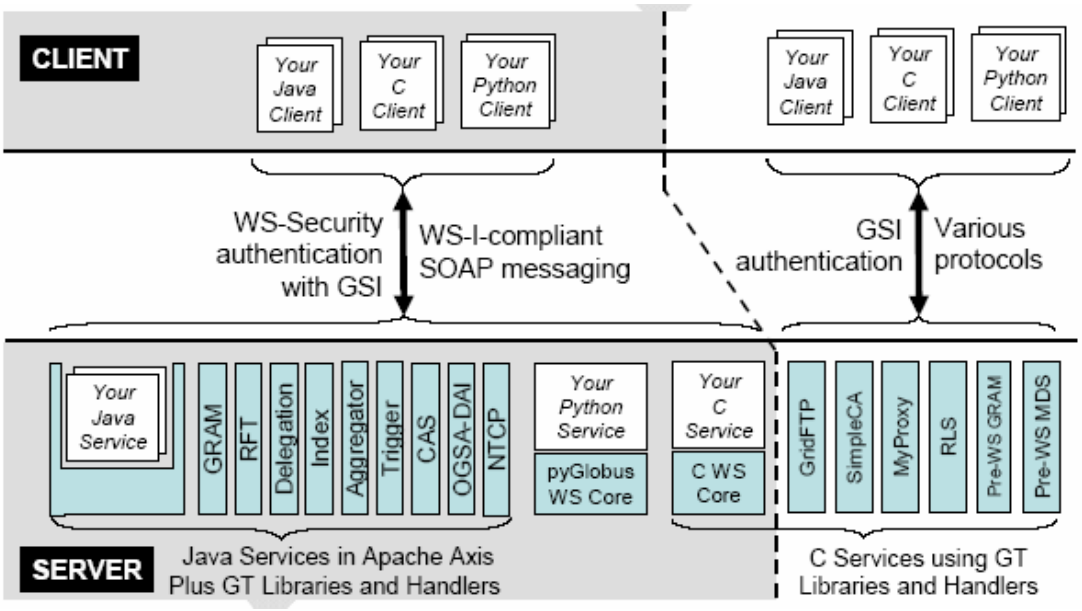

Figura 6: Principais componentes do GT 4 [GT2005]

Na Figura 6 são apresentados os principais componentes fornecidos pelo GT4. Do lado esquerdo da figura, podem ser vistos os serviços que já implementam w*eb services*, como é o caso do GRAM, RFT, serviços de delegação, dentre outros. Como pode ser visto, todos esses componentes utilizam mecanismos de transporte de mensagens e segurança que atendem à *WS-Interoperability (WS-I)*, um consórcio de empresas que objetiva promover a interoperabilidade entre *web services* [Wsi2005].

Do lado direito, estão os serviços que ainda não utilizam w*eb services*, seja por questão de manter a compatibilidade com versões anteriores (GRAM e MDS) ou mesmo porque ainda não foram implementados utilizando esta nova tecnologia (*Grid* FTP, *Simple CA, Myproxy* e RLS).

2.6 Obtendo um melhor escalonamento para aplicações que demandam alto grau de comunicação em grades

Grande parte das iniciativas bem sucedidas de utilização da computação em grade até então estão relacionadas com aplicações que fazem utilização relativamente pequena de comunicação entre processos. É o caso do projeto SETI@Home por exemplo, que busca

evidências de vida extra terrestre. Seu funcionamento é baseado na distribuição de pequenas quantidades de dados para serem processadas independentemente umas das outras [SET2006].

No entanto, o surgimento de aplicações que fazem um uso mais intensivo de comunicação tem criado a necessidade de se maximizar a utilização dos recursos de rede. Podem ser citadas simulações científicas em geral, como análises relativas à viabilidade na prospecção de petróleo, mineração de dados e física de alta energia.

Nestes casos, é conveniente que o serviço responsável pelo escalonamento de tarefas das grades leve em consideração as informações relativas ao estado da rede como um todo. Assim, medidas de taxa de vazão, latência e perda de pacotes passam a ser importantes no processo de decisão adotado pelo escalonador de tarefas.

Além de obter um melhor desempenho, a partir do momento em que é considerada a viabilidade de se utilizar caminhos alternativos entre os nós, também se ganha maior robustez, já que a rede pode determinar o encaminhamento de um fluxo por outra rota no caso de uma falha.

No próximo capítulo serão vistos alguns projetos que deram início às pesquisas que passaram a tratar a rede como um recurso a ser compartilhado entre os nós que compõem uma grade computacional.

3 Qualidade de serviço em grades computacionais

Embora as grades computacionais permitam a criação de ambientes colaborativos capazes de fornecer aos seus usuários um alto poder computacional para executar suas aplicações, dificuldades existentes em relação ao seu desenvolvimento e utilização ainda são fatores que dificultam sua aplicabilidade a problemas do nosso dia-a-dia.

Devido à possibilidade de ser constituída por recursos heterogêneos, o grau de complexidade necessário para conseguir fazer uso deles é bem maior do que o encontrado em um ambiente padronizado.

O grande número de entidades que podem estar envolvidas na constituição de uma grade computacional é outro fator que dificulta sua disseminação. É preciso encontrar formas que possibilitem a interoperabilidade entre os diferentes domínios administrativos, sem impor muitas alterações em seu funcionamento.

Com o intuito de solucionar desafios como os descritos acima, surgiram diversas iniciativas, tanto comerciais quanto acadêmicas.

Neste capítulo, serão apresentados alguns dos sistemas de computação em grade existentes, como o *Globus toolkit* que, sem dúvida, é o projeto de maior importância na área de ferramentas para grades computacionais.

Por fim, serão vistas algumas iniciativas que visam adicionar qualidade de serviço nas transmissões em grades computacionais, focando trabalhos que tratem o problema do ponto de vista das redes de comunicação.

3.1 A importância da rede para uma grade computacional

O desenvolvimento da computação em grade tem alterado a forma como as redes de comunicação são tratadas pelas aplicações em uma grade. No paradigma inicial, até hoje

muito utilizado, as redes de comunicação são tratadas como simples meios de interconexão entre recursos.

Em decorrência da alta velocidade provida pelas redes óticas, a tendência é que as redes adquiram uma importância muito maior, deixando de ser tratadas como simples meios de interconexão de dispositivos e passando a ser consideradas como um recurso a ser compartilhado entre os nós e usuários. Desta forma, características como velocidade e atraso passam a influenciar na decisão sobre o local e o horário que uma aplicação será executada na grade.

Assim como qualquer outro recurso, as conexões de rede passam a fazer parte da arquitetura definida pela OGSA, comportando-se como uma entidade comum, possível de ser descoberta, registrada, monitorada, instanciada, criada e destruída. Além disso, é necessário que exista um gerenciador local de recursos (no caso do Globus, o GRAM é quem faz este papel), responsável por criar e gerenciar o serviço de rede através do uso de MPLS ou de algum outro protocolo de sinalização.

O fato da computação em grade ocorrer em ambientes heterogêneos, administrados independentemente uns dos outros, dificulta em muito o estabelecimento de qualidade de serviço fim-a-fim.

No caso do Globus, as características do GRAM não são capazes de prover a qualidade de serviço descrita acima. Para resolver este problema, foi criado o GARA (*General Purpose Architecture for Reservation and Allocation*).

Aplicações científicas, como as de tele-imersão e simulação que operam em ambientes colaborativos necessitam lidar com prazos fixos de tempo, sendo imprescindível que exista algum tipo de QoS para garantir esta característica. Embora a maioria das grades computacionais funcione sob o princípio do melhor esforço, tratar seus usuários com igual prioridade não é a melhor solução para todas as aplicações.

Sem dúvida, o estabelecimento de QoS nas redes ópticas será de suma importância para o futuro da computação em grade. As requisições geradas por cada aplicação deverão incluir parâmetros que estabeleçam suas necessidades em relação ao serviço de rede.

3.2 Globus *Architecture for Reservation and Allocation* (GARA)

O gerenciador de recursos do Globus (GRAM) não provê funcionalidades de reserva de recursos, o que impede o estabelecimento de níveis pré-estabelecidos de qualidade de serviço às suas aplicações.

Com objetivo de permitir que a alocação e a reserva de recursos pudessem ser feitas de acordo com características relativas a parâmetros de qualidade de serviço, foi desenvolvido o GARA. Como pode ser visto em [Gar2000], o GARA objetivava prover:

- Uma arquitetura flexível de qualidade de serviço a diversos tipos de recursos, incluindo serviços de rede, processadores, agendadores de tarefas e espaço de armazenamento em disco.
- Mecanismos que permitam tanto a reserva prévia quanto imediata de recursos.
- Suporte a computação de alto desempenho, considerando a obtenção de qualidade de serviço mesmo em ambientes complexos em relação ao conjunto de recursos utilizados.

O GARA estende o gerenciador de recursos padrão do Globus através da inclusão de duas novas características. Primeiramente, ele introduz o conceito de "*generic resource object*" que pode abranger fluxos de rede, memória, discos e outras entidades. Em segundo lugar, ele permite a realização de reserva prévia de recursos.

O fato de tratar cada recurso como um objeto permite que o GARA suporte recursos dos mais variados tipos. Cada chamada de criação de um objeto (abstração de um recurso real) retorna um manipulador responsável pelo seu monitoramento e controle.

A criação de objetos no GARA é dividida em duas etapas: reserva e alocação. A primeira fase cria uma sinalização positiva em relação à disponibilidade do recurso, porém, nenhum objeto é criado. Durante a operação de reserva, um manipulador interage com os gerenciadores locais de recursos, com objetivo de garantir que a qualidade de serviço

requerida para utilização do recurso estará disponível desde o momento inicial até final da transação [FKL1999].

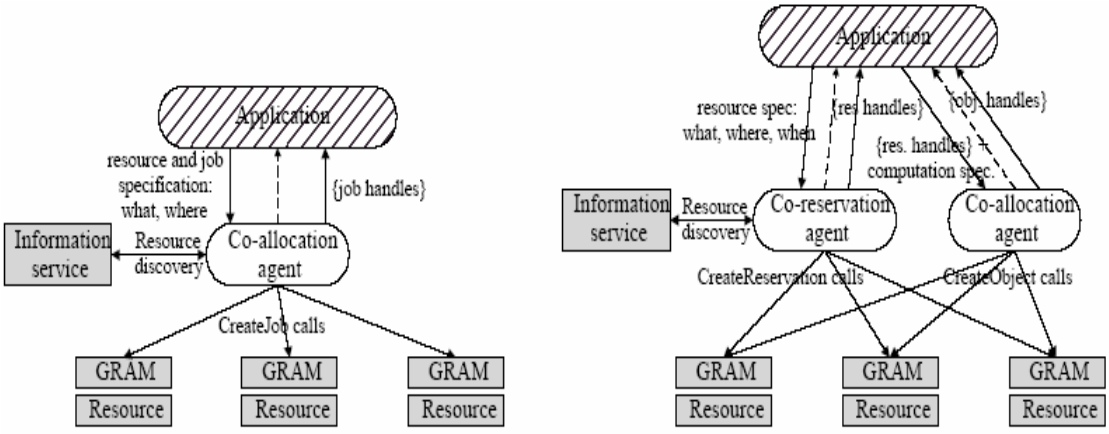

Figura 7: GRAM e GARA [FKL1999]

Como pode ser visto na Figura 7, o GARA introduziu uma nova entidade denominada agente co-reservador de recursos (*co-reservation agent*). De forma semelhante ao agente co-alocador de recursos (*co-allocation agent*) já existente no GRAM, o co-reservador de recursos é responsável pela sondagem dos recursos capazes de satisfazer aos requerimentos de qualidade de serviço fim-a-fim de acordo com as exigências da aplicação. O que os difere é que o co-reservador não chega a alocar o recurso propriamente dito [FKL1999].

3.3 QoSINUS

O QoSINUS é uma proposta que se baseia na construção de um serviço de rede voltado ao fornecimento de qualidade de serviço no nível de rede para grades computacionais. Este serviço permite que aplicações possam alocar recursos de rede dinamicamente, possibilitando que as necessidades individuais dos fluxos de dados gerados pela grade sejam atendidas.

O modelo proposto pelo QoSINUS objetiva fornecer meios para que aplicações executadas em ambientes de grade possam influenciar no comportamento dos dispositivos de rede, oferecendo uma espécie de “qualidade de serviço de melhor esforço” [VPC2004].

Esta denominação se deve ao fato do QoSINUS não estabelecer um contrato rígido a ser atendido (SLA – *Service Level Agreement*) entre aplicações e recursos de rede. Seu funcionamento é baseado no modelo de serviços diferenciados.

Durante sua implementação foi desenvolvida uma API e um serviço de rede, responsáveis pelo controle e estabelecimento de QoS fim-a-fim na grade. Foi utilizado o protocolo XML para definir as especificações de níveis de serviço (SLS – *Service Level Specifications*).

Segundo os autores, a introdução do QoSINUS em um ambiente de grade acarreta um pequeno aumento da complexidade nos pontos de fronteira da rede, não implicando em maiores alterações tanto em seu núcleo quando nas aplicações executadas na grade [VPC2004].

A validação do modelo ocorreu em um ambiente de testes do projeto e-Toile [ETO2002], destinado a avaliar os benefícios que as novas tecnologias de transmissão de dados podem oferecer às aplicações de grade de alto desempenho.

3.4 QAME (*QoS-Aware Management Environment*)

O projeto QAME, desenvolvido pela UFRGS (Universidade Federal do Rio Grande do Sul), está relacionado ao fornecimento de suporte a gerenciamento de qualidade de serviço (QoS – *Quality of Service*) em redes de computadores. Seu principal resultado consistiu na elaboração de uma ferramenta de gerência de redes destinada a auxiliar na implantação, na manutenção e no monitoramento de QoS.

Seu mecanismo de gerenciamento de redes é baseado em políticas (PBNM - *Policy-Based Network Management*), o que permite que os administradores determinem o comportamento do sistema para atingir as expectativas requeridas de QoS em um alto nível de abstração [Slo1994].

```
If {(srcAddress == "143.54.47.242" and srcPort == "*" and
      dstAddress == "143.54.47.17" and dstPort == "80" and
      DSCP == "*" and proto == "TCP" and
      StartTime >= "08/30/2004 00:00:00" and endTime <= "09/01/2004 23:59:59")
{
bandwidth = 10Mbps;
DSCP = 46;
prioriry = 4;
}
```

Figura 8: Exemplo de política de rede [Nei2004]

A Figura 8 ilustra a forma como pode ser representada uma política de rede. Neste exemplo, o tráfego gerado pela máquina cujo endereço IP seja 143.54.47.242 com destino à máquina de endereço IP 143.54.47.17, utilizando qualquer porta de origem "*" com destino à porta http (80) e com qualquer valor no campo DSCP, terá 10Mbps de banda, o campo DS receberá o valor 46 e a prioridade será ajustada para nível 4 [Nei2004].

Com objetivo de adaptar as funcionalidades deste sistema às necessidades das grades computacionais foi criado um módulo específico, acessado via *web*. Este módulo permite que o administrador da grade defina políticas e classes de serviços que posteriormente serão traduzidas em políticas de rede.

O funcionamento da arquitetura do mecanismo de tradução de políticas pode ser dividido em três níveis de abstração. No nível mais alto, são definidas as políticas de gerenciamento da grade. Estas são baseadas nos usuários da grade, recursos fornecidos e necessidades relativas à qualidade de serviço.

No nível intermediário encontram-se as políticas de gerenciamento da rede, geradas à partir de um mecanismo de tradução definido pelo administrador da rede.

No nível mais baixo da arquitetura proposta, as políticas de rede passam a atuar na configuração dos equipamentos, podendo alterar parâmetros relativos ao tamanho da banda requerido pela aplicação, prioridade a ser dada aos fluxos de dados, protocolos e portas utilizadas, definição do campo DSCP relativo às redes de serviços diferenciados entre outros.

As regras utilizadas pelos mecanismos de tradução de políticas utilizam um conjunto de objetos que atendem tanto aos requerimentos das políticas de grade originais quanto às políticas de rede a serem criadas. Tais objetos identificam a origem e o destino de cada troca

de mensagem, bem como o período de tempo em que ela ocorrerá e as exigências de QoS a serem atingidas [Nei2004].

Um ponto relevante neste trabalho se deve ao fato do sistema de rede de comunicação ser tratado como um recurso a ser compartilhado por usuários e serviços. Assim, a linguagem utilizada na definição das políticas trata não somente dos usuários e aplicações, mas também dos equipamentos de rede e parâmetros de QoS a serem introduzidos e verificados nos mesmos.

4 Qualidade de serviço em redes

A simplicidade do protocolo IP é sem dúvida um dos fatores que possibilitaram o sucesso da Internet. O serviço de melhor esforço é baseado no conceito de transporte de datagramas sem qualquer garantia de entrega, ordem de chegada ou atraso. Quem garante a entrega dos pacotes ao destino é um protocolo de nível mais alto, no caso, o TCP.

Para dar suporte a aplicações mais exigentes em relação à utilização da rede foram criadas alternativas, em geral baseadas na introdução de esquemas de regulação nos equipamentos de borda da rede.

Inicialmente foi desenvolvido o conceito de redes de serviços integrados (Intserv). Em função principalmente de sua baixa escalabilidade, sua limitação a redes de pequeno porte acabou por criar a necessidade de novas alternativas.

Atualmente, as duas tecnologias mais utilizadas para estabelecimento de qualidade de serviço em redes de computadores são os serviços diferenciados (Diffserv) e o MPLS (*Multi protocol Label Switching*).

Neste capítulo serão introduzidas as principais características das três alternativas citadas acima.

4.1 Serviços Integrados (Intserv)

Devido ao desenvolvimento de aplicações de tempo real, caracterizadas por serem sensíveis a atrasos na rede, o IETF (*Internet Engineering Task Force*) criou um grupo de trabalho objetivando fornecer qualidade de serviço à Internet. Sua arquitetura é denominada rede de serviços integrados.

O funcionamento do modelo baseia-se em quatro componentes:

- **Escalonador de pacotes:** Sua função está ligada ao estabelecimento de políticas de enfileiramento de pacotes nos roteadores de forma que seja possível atender às prioridades de fluxo.
- **Rotina de controle de admissão:** Determina se um fluxo de dados poderá ou não ser aceito, considerando-se a quantidade de banda disponível.
- **Classificador:** Responsável por classificar os pacotes em diferentes classes, sendo que os pacotes pertencentes à determinada classe receberão o mesmo tratamento.
- **Protocolo de sinalização:** É utilizado o RSVP (*Resource ReSerVation Protocol*) [Bra1997], protocolo que permite que aplicações façam reserva de recursos de banda para seus fluxos de dados junto aos roteadores de uma rede de serviços integrados.

Embora o Intserv consiga fornecer qualidade de serviço a fluxos individuais, alguns problemas acabaram por limitar sua utilização.

O primeiro ponto negativo do modelo de Serviços Integrados era o fato de ele não ser escalável, passando por grande sobrecarga quando utilizado em redes de grande porte. A sua baixa flexibilidade é outro fator que limita sua utilização. O Intserv possui um pequeno número de classes de serviço pré-especificadas, não comportando definições mais qualitativas ou relativas para diferenças entre classes [KR2003].

O Diffservice é a evolução desse modelo, que corresponde às necessidades relativas à flexibilidade e escalabilidade das redes de hoje.

4.2 Serviços Diferenciados (Diffserv)

Com objetivo de resolver as limitações impostas pelo modelo de Serviços Integrados (Intserv) em relação à reserva de recursos por fluxo, foi criado um novo grupo de trabalho no pelo IETF, no ano de 1999, responsável pelo desenvolvimento da arquitetura de serviços diferenciados.

Uma das características do Diffserv está no fato de as operações mais complexas serem realizadas nos roteadores de borda da rede, desafogando o seu interior (núcleo). A partir disso pode-se obter uma maior escalabilidade do que a existente no modelo anterior, de serviços integrados.

Os roteadores de borda da rede são responsáveis pela classificação dos pacotes que chegam até eles. Esta seleção considera dados presentes em seus cabeçalhos, como por exemplo: endereço da máquina de origem e da máquina de destino, porta de origem e porta de destino, dentre outros.

Depois de classificados, uma marca é inserida no campo DS do cabeçalho do pacote. Cada classe de tráfego passa a ser identificada através do seu código DS, o que possibilita a adoção de tratamento diferenciado pelos roteadores situados no núcleo da rede.

O campo DS foi obtido após alteração do nome do campo tipo de serviço (*Type Of Service – TOS*), pertencente ao cabeçalho IPv4, ou seu equivalente classe de tráfego (*Traffic Class*), no IPv6 [NBB1998]. Do total de 8 *bits*, apenas os 6 primeiros formam o sub-campo DSCP (*Differentiated Services Codepoint*), servindo para identificar as classes de tráfego. Os dois *bits* restantes (ECN- *Explicit Congestion Notification*) são utilizados para controle de congestionamento. A Figura 9 ilustra o campo DS.

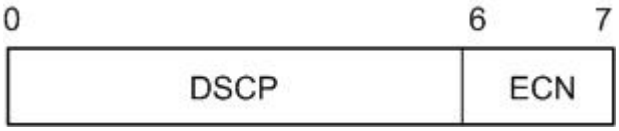

Figura 9: O campo DS

Os roteadores do núcleo da rede são responsáveis pelo encaminhamento dos pacotes, repassando-os ao próximo roteador de acordo com o tratamento especificado por cada classe de tráfego.

Com o uso dos serviços diferenciados é possível definir perfis de tráfego, permitindo obter diferentes comportamentos em relação à taxa com que os pacotes chegam a determinado domínio. Diferentes ações de condicionamento poderão ser aplicadas aos pacotes que estejam dentro ou fora de determinado perfil [BBC1998].

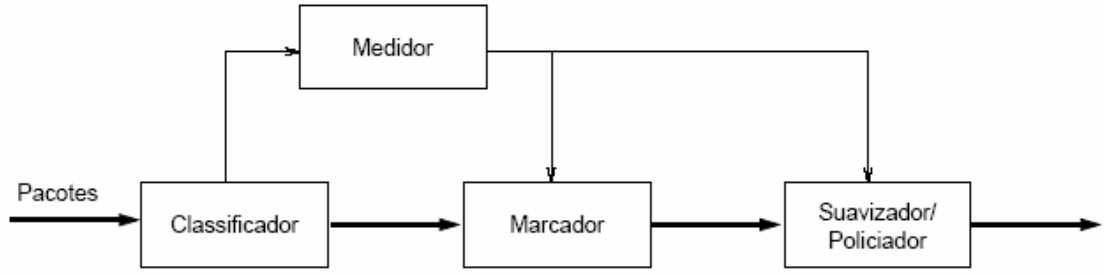

Figura 10: Elementos de um condicionador de tráfego

A Figura 10 ilustra os elementos funcionais de um condicionador de tráfego. Após classificados, um medidor pode opcionalmente ser utilizado para conferir se determinado tráfego satisfaz as condições impostas por seu perfil. O marcador é responsável pela inserção do código DSCP em cada pacote, atribuindo-o a uma classe de agregação. Com base nessas informações, o elemento suavizador/policiador pode agir, impondo um atraso para que o tráfego se comporte dentro dos limites estabelecidos.

5 Multi Protocol Label Switching (MPLS)

Devido ao surgimento de novas aplicações e ao crescimento da Internet, o cenário atual do mercado de redes de computadores necessita de soluções baseadas na convergência de serviços, ou seja, soluções que permitam a coexistência de voz, dados e imagem sobre um canal de comunicação comum.

Embora ainda em fase de desenvolvimento, a tecnologia denominada *Multi Protocol Label Switching* (MPLS) já possui diversos exemplos de aplicação comercial, sendo disponibilizada como um serviço por diversos provedores de comunicação por todo o mundo.

Nas formas tradicionais de encaminhamento de pacotes em uma rede, cada roteador define o próximo salto (*next hop*) através da análise das informações contidas no cabeçalho do pacote e da sua tabela de roteamento local. Esse processo adiciona latência na comunicação, pois o destino a ser tomado por cada pacote é definido a partir da decisão individual de cada roteador pelo qual ele passe. Tal decisão depende da análise de dados contidos no cabeçalho da camada de rede, fator que acaba exigindo grande poder de processamento dos roteadores envolvidos no processo de comunicação, quando o volume de dados a ser transmitido é grande.

O MPLS constitui uma nova forma de encaminhamento de pacotes, que procura ser mais eficiente que os métodos tradicionais, pois combina a velocidade e o desempenho característicos da camada de enlace com a escalabilidade e flexibilidade característicos da camada de rede [NN2001]. Além disso, oferece suporte às redes de serviços diferenciados e permite que sejam utilizadas técnicas de engenharia de tráfego.

5.1.1 Principais características

O MPLS surgiu a partir de especificações determinadas pelo IETF, cujas características fundamentais são [IEC2005]:

- Fornecer mecanismos capazes de gerenciar fluxos de dados com grande granularidade, provenientes de diferentes tipos de equipamentos e aplicações.
- Ser independente dos protocolos utilizados nas camadas 2 e 3 do modelo OSI.
- Permitir o mapeamento de endereços IP para rótulos de tamanho fixo, sendo que o roteamento dos pacotes passa a ser realizado com base nestes rótulos.
- Poder ser utilizado juntamente com protocolos de sinalização e roteamento já existentes, como por exemplo: *Resource Reservation Protocol* (RSVP) e *Open Shortest Path First* (OSPF).

5.1.2 O rótulo do MPLS

Um rótulo MPLS é caracterizado por representar um identificador relativamente curto (20 bits) utilizado no processo de encaminhamento de pacotes MPLS. Comumente, os rótulos possuem um significado local, ou seja, só têm sentido quando ligados a um único enlace de dados.

Os rótulos MPLS são análogos ao par VPI/VCI utilizado nas decisões de encaminhamento em redes ATM [OS2003].

Como pode ser visto na Figura 11, o cabeçalho do MPLS possui 32 *bits*, sendo formado por quatro campos:

- *Label* (20 *bits*): Utilizado para informar a qual *Forwarding Equivalence Class* (FEC) o pacote pertence. O significado de FEC ainda será explicado.
- EXP (3 *bits*): Inicialmente chamado classe de serviço (CoS – *Class of Service*), agora possui caráter experimental. Um dos exemplos de sua utilização é no

mapeamento do campo DSCP proveniente do Diffservice sobre MPLS, objetivando prover funcionalidades relativas à Qualidade de Serviço.

- S (1 *bit*): Fornece suporte ao empilhamento (*Stacking*) hierárquico de rótulos.
- TTL (8 *bits*): Utilizado para informar o tempo de vida (*Time To Live*) do pacote, ou seja, o número de nós MPLS pelos quais o pacote passou até chegar a seu destino, cujo valor é decrementado a cada nó. Segundo [RVC2001], no caso de uma rede IP sobre MPLS estar operando com o protocolo ethernet, o campo TTL é copiado do cabeçalho de camada 3 na entrada do domínio MPLS e decrementado em cada LSR. Na saída do domínio, o valor resultante é copiado de volta para o cabeçalho do pacote IP.

Figura 11: Cabeçalho MPLS [Ros2001]

A Figura 12 ilustra o formato genérico de encapsulamento de um rótulo MPLS. Como pode ser visto, o rótulo é inserido em cada pacote entre o cabeçalho da camada de enlace de dados e o cabeçalho da camada de rede.

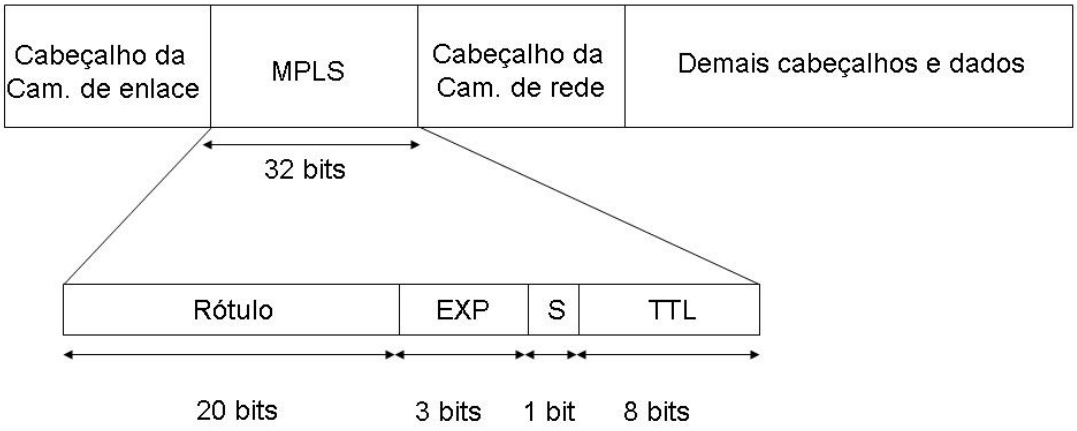

Figura 12: Formato genérico de um rótulo MPLS

Como já foi dito, o MPLS pode operar sobre qualquer protocolo da camada de enlace. Desta forma, uma rede constituída por LSRs (*Label Switching Router*) ATM pode fazer uso do MPLS no plano de controle para trocar informações do VPI/VCI ao invés de usar a sinalização do ATM [OS2003].

No caso de uma rede de LSRs ATM, o roteador de ingresso fica responsável pela segmentação do pacote em células ATM. Este processo consiste em aplicar o valor apropriado do VPI/VCI que foi trocado no plano de controle para que as células possam ser transmitidas. Um LSR ATM se comporta da mesma forma que um *switch* ATM comum, ou seja, ele encaminha cada célula com base no VPI/VCI de entrada e na informação da porta de entrada. O roteador de saída é responsável pela montagem das células em um pacote [OS2003].

A Figura 13 ilustra como é feita a codificação do rótulo MPLS nos campos VPI/VCI de uma célula ATM. Como pode ser visto, no caso das redes ATM, os rótulos são transportados diretamente no cabeçalho da camada de enlace, não sendo necessário inserir um cabeçalho adicional como no caso do encapsulamento genérico.

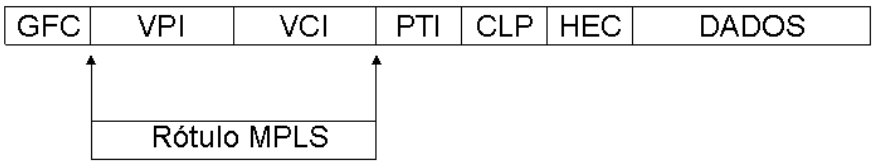

Figura 13: Exemplo de inserção do cabeçalho MPLS para uma rede ATM

Outra tecnologia de rede da camada de enlace que também pode operar em conjunto com o MPLS é o *Frame Relay*. Embora soluções baseadas em *Frame Relay* estejam perdendo espaço para redes VPN-IP, sua utilização ainda é bastante comum, principalmente em sistemas legados. A Figura 14 ilustra como é feito o encapsulamento do cabeçalho MPLS em uma rede *Frame Relay*.

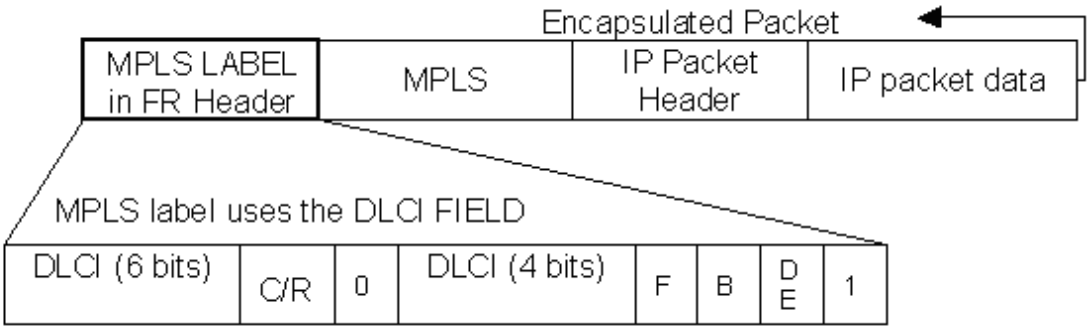

Figura 14: Exemplo de encapsulamento MPLS em rede Frame Relay [Hop2001]

Maiores informações em relação à utilização do MPLS em conjunto com *Frame Relay* podem ser obtidas em [CDM2001].

5.1.3 Empilhamento de rótulos

Existem casos em que pode ser necessário inserir mais de um rótulo em um pacote MPLS. Este procedimento é conhecido como empilhamento de rótulos (*label stacking*), sendo que as operações de inserção e remoção dos rótulos são realizadas no padrão LIFO (*Last In First Out*), ou seja, o último rótulo a entrar é o primeiro a sair [RVC2001].

O empilhamento de rótulos é útil nos casos em que pacotes atravessam diversos domínios MPLS, no tunelamento de canais e também no estabelecimento de redes VPN. Um domínio MPLS deve ser entendido como sendo um conjunto de dispositivos de rede interconectados que fazem uso do protocolo MPLS.

Na Figura 15 pode ser vista uma pilha de rótulos MPLS. Além disso, é possível verificar que a inserção dos rótulos acontece entre os cabeçalhos das camadas dois e três.

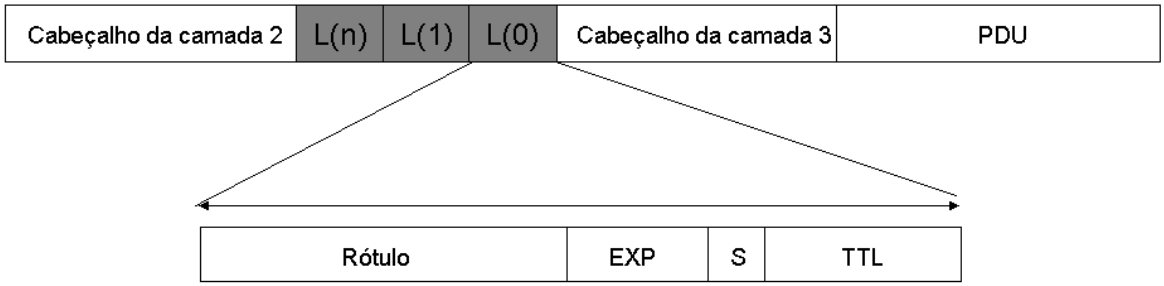

Figura 15: Inserção do cabeçalho MPLS entre camadas dois e três

5.1.4 Funcionamento do MPLS

A técnica de encaminhamento de pacotes do MPLS é baseada na adição de rótulos a eles. Cada rótulo corresponde a um identificador de tamanho fixo inserido no cabeçalho do pacote. Sua utilidade é identificar as FECs (*Forwarding Equivalence Classes*), ou seja, grupos de pacotes que serão comutados de maneira idêntica na rede (seguirão o mesmo caminho, com as mesmas prioridades, etc.)

Quando um pacote entra em um domínio que utiliza MPLS, o roteador de borda (*Label Edge Router* - LER) de ingresso da rede insere um rótulo no seu cabeçalho. Este rótulo liga cada pacote a uma FEC. Os *switches* ou roteadores de núcleo da rede (LSRs) encaminham os pacotes até seu destino final com base somente na informação contida no rótulo, respeitando o tratamento especificado pelas FECs.

Ao chegar ao seu destino final, ou seja, no roteador de egresso do domínio MPLS em questão, o rótulo é removido do cabeçalho. Em seguida, o pacote é repassado ao próximo roteador, que pode ou não estar inserido a um segundo domínio MPLS. A Figura 16 ilustra o processo descrito nos parágrafos acima.

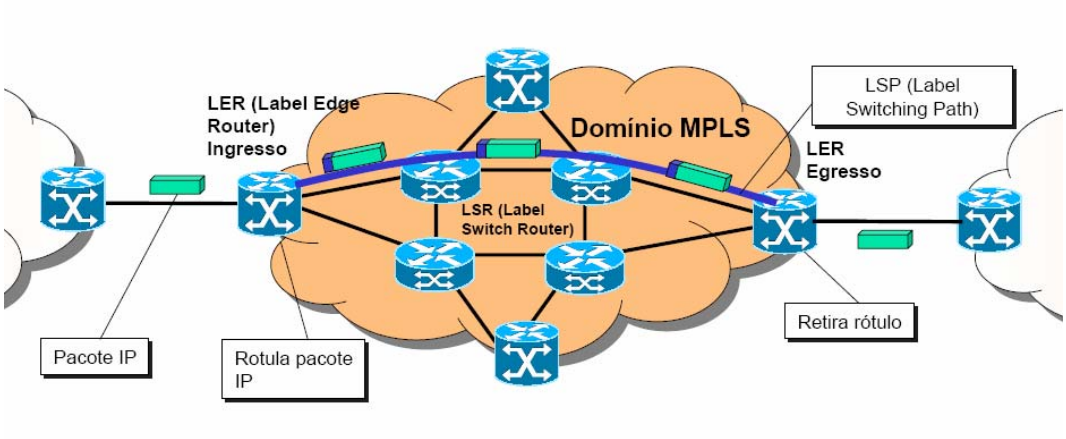

Figura 16: Encaminhamento através de rótulos [ALV2004]

Ainda com relação à Figura 16, também é importante evidenciar que o caminho formado entre o LSR de ingresso e o LSR de egresso é conhecido por *Label Switch Path* (LSP). Cada LSP corresponde a um caminho unidirecional entre roteadores e deve ser estabelecido antes que os pacotes sejam enviados. Por ser unidirecional, sempre que um LSP for estabelecido deve-se considerar a criação de outro LSP responsável pelo encaminhamento do tráfego de retorno do primeiro.

O processo descrito acima traz algumas vantagens em relação ao roteamento de pacotes realizado pelo IP. Uma delas está relacionada à diminuição do tempo de decisão necessário para encaminhar cada pacote. Em uma rede IP, a decisão do caminho a ser tomado por cada pacote depende da consulta a tabelas de roteamento. Estas consultas são realizadas por cada roteador pelo qual o pacote passe.

Já em uma rede MPLS, depois de inseridos os rótulos nos pacotes, estes podem ser encaminhados de forma mais rápida e eficiente, resultando numa diminuição do atraso e simplificação no processo de decisão do caminho a ser tomado. O motivo deste ganho de desempenho no encaminhamento deve-se ao fato do mesmo não depender mais da decisão de cada roteador que estiver no caminho a ser seguido pelos pacotes. Além disso, as decisões de encaminhamento não dependem mais de uma análise dos dados contidos no cabeçalho da camada de rede, sendo realizada a partir das informações contidas nos rótulos dos pacotes.

O MPLS também leva vantagem por não ter que recalcular o *checksum* de cada pacote após o TTL ser decrementado, como ocorre no protocolo IP. No caso do MPLS, ocorre apenas o decremento do TTL.

Outro fator que tem contribuído para o aumento da utilização do protocolo MPLS está ligado às facilidades relativas à priorização de pacotes que ele possui. Como será visto com maiores detalhes, o cabeçalho do protocolo MPLS possui um campo que pode ser utilizado para agrupar pacotes que deverão ser tratados da mesma forma no interior do domínio MPLS. Esta característica permite que o administrador da rede atribua rótulos aos pacotes de acordo com diferentes classes de serviço, facilitando a gerência dos fluxos de tráfego e permitindo o oferecimento de QoS às aplicações.

Conforme pode ser visto na Figura 17, o MPLS atua entre nas camadas 2 e 3 do modelo RM-OSI/ISO, não estando limitado a nenhum protocolo específico nessas camadas [NN2001].

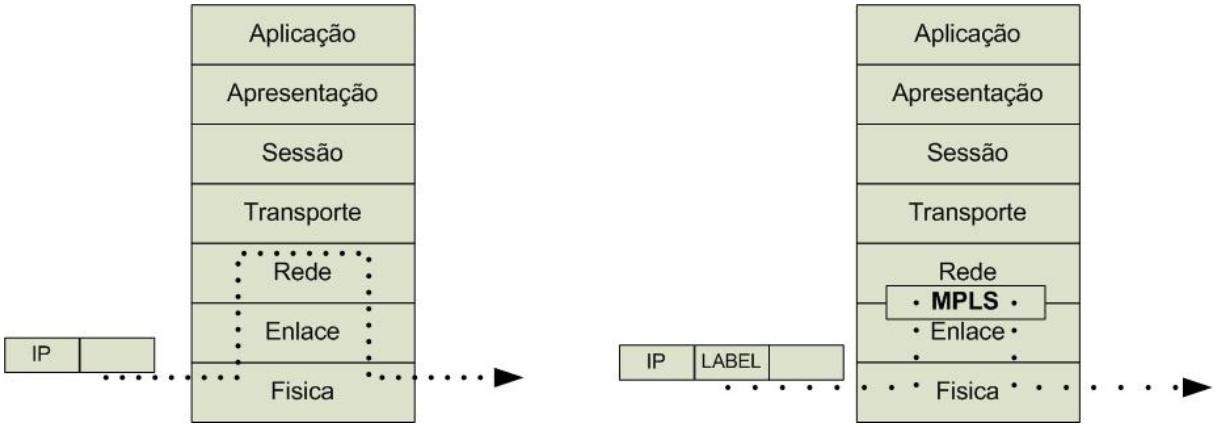

Figura 17: Comparação entre roteamento de pacotes IP e pacotes MPLS

5.1.5 O protocolo de distribuição de rótulos

Antes do estabelecimento de cada LSP e durante sua existência é necessário que os roteadores troquem mensagens entre si com o objetivo de entrarem em concordância mútua em relação aos LSPs existentes e aos rótulos a eles associados. Esta troca de mensagens é realizada por um protocolo específico responsável pela distribuição de rótulos.

A arquitetura MPLS [RVC2001] define um protocolo de distribuição de rótulos como sendo um mecanismo através do qual um LSR informa a outro LSR a respeito dos rótulos utilizados para trocar pacotes entre si.

Uma forma de desenvolver este mecanismo é através da extensão de protocolos já existentes, como é o caso do RSVP-TE [BER2001] e do MPLS-BGP [DKV2001].

Além do RSVP-TE e do MPLS-BGP também foi desenvolvido um novo protocolo, destinado exclusivamente à distribuição de rótulos. É o caso do LDP (*Label Distribution Protocol*) [And2001]. Através do LDP dois ou mais LSRs conseguem estabelecer LSPs (*Label Switched Paths*). Um LSP se assemelha a um túnel através do qual a comunicação entre os roteadores utiliza o protocolo MPLS.

O LDP também é responsável pela associação entre FEC ´s (*Forwarding Equivalent Classes)* e LSP´s.

5.1.6 Elementos da comutação através de rótulos

Para que um pacote seja comutado em função do seu rótulo pelos roteadores que compõem uma rede MPLS é necessário consultar algumas tabelas. Neste tópico serão mostradas as principais características de cada uma delas.

5.1.6.1 NHLFE (*Next Hop Label Forwarding Entry*)

Esta tabela contém informações relativas à ação a ser tomada na pilha de rótulos de cada pacote, sendo responsável por definir [RVC2001]:

- O próximo passo do pacote, ou seja, por qual interface ele irá sair.
- A operação a ser tomada na pilha de rótulos dos pacotes (inserção, substituição ou retirada de rótulos).
- Qualquer outra informação necessária ao tratamento dos pacotes.

A Tabela 1 representa um exemplo de NHLFE. Nela estão ilustradas duas operações muito comuns de serem realizadas na pilha de rótulos dos pacotes. A primeira entrada da tabela NHLFE direciona os pacotes para a interface de IP 10.0.0.6 e empilha um novo rótulo

(2200) na pilha de rótulos dos mesmos. A segunda entrada, identificada por 0x00000002, remove o rótulo (*pop*) que estiver no topo da pilha de rótulos.

Caso a operação de remoção de rótulo descrita no parágrafo anterior fosse realizada no último rótulo da pilha de rótulos, o pacote poderia passar a ser encaminhado de acordo com o seu endereço IP. Isto caracterizaria a saída do mesmo de um domínio MPLS.

Tabela 1: Exemplo de tabela NHLHFE

NHLFE	Próxima interface	Operação
0x00000001	10.0.0.6	Empilha rótulo 2200 (*push*)
0x00000002	10.0.0.3	Remove rótulo do topo da pilha (*pop)*

5.1.6.2 ILM (*Incoming Label Mapping*)

A ILM é uma tabela que só existe nos roteadores de núcleo e de egresso de um domínio MPLS, pois só atua sobre pacotes que já possuem rótulo. Sua função é mapear cada rótulo a uma NHLFE.

A Tabela 2 possui alguns exemplos de entradas ILM. A primeira linha da tabela liga os pacotes entrantes que possuírem o rótulo 2200 à NHLFE identificada por 0x00000001. De forma similar, a segunda linha liga os rótulos 1200 de entrada à NHLFE 0x00000002.

Tabela 2: Mapeamento entre rótulos e NHLFEs

Rótulo de entrada	NHLFE
2200	0x00000001
1200	0x00000002

5.1.6.3 FTN(*FEC to NHLFE*)

Quando um roteador de ingresso recebe um pacote que ainda não possui rótulo, é utilizada a tabela FTN (FEC *to* NHLFE), que mapeia cada FEC a um conjunto de NHLFEs. Ao contrário da tabela ILM, a tabela FTN só é implementada nos roteadores de ingresso na borda de um domínio MPLS.

A Tabela 3 ilustra um exemplo de mapeamento entre FECs e NHLFEs. Como pode ser visto, os pacotes destinados à rede 10.0.1.0 serão tratados pela NHLFE identificada por 0x00000002 e aqueles destinados à rede 10.0.2.0 serão tratados pela NHLFE 0x00000003.

Tabela 3: Mapeamento entre FEC e NHLFE

FEC	NHLFE
10.0.1.0	0x00000002
10.0.2.0	0x00000003

5.1.7 Suporte a serviços diferenciados

O MPLS e o Diffservice possuem diversas características comuns. Uma delas está relacionada com as operações de maior complexidade, como é o caso da classificação dos pacotes, que ocorrem na borda da rede. As duas tecnologias agrupam os fluxos de tráfego que serão tratados da mesma forma em classes, sendo que estas recebem diferentes denominações: FECs no caso do MPLS e PHBs no caso do Diffservice.

Tanto no Diffservice quanto no MPLS são utilizadas seqüências curtas de números na identificação dos pacotes. No primeiro, o campo DSCP determina qual será o tratamento dado a cada pacote pelos roteadores de núcleo do domínio, ou seja, com base no DSCP são definidas ordens de priorização de fluxos junto a mecanismos responsáveis pelo enfileiramento e descarte.

Já no caso do MPLS, o rótulo inserido nos pacotes pelos roteadores de borda do domínio são utilizados na definição do caminho (LSP) a ser seguido por eles. Associando-se os rótulos às diferentes possibilidades de caminhos é possível adotar técnicas de Engenharia de Tráfego.

Devido a estas características, as duas tecnologias devem ser vistas como complementares, podendo muitas vezes serem adotadas em conjunto.

O IETF definiu duas formas para se realizar o mapeamento do Diffservice sobre o MPLS [Fau2005]. São elas:

- E-LSP (EXP-*inferred*-PSC LSP): define o PHB através do mapeamento do campo DSCP do cabeçalho IP no campo EXP do cabeçalho MPLS. Este método é utilizado no caso de redes *ethernet* e demais protocolos da camada de enlace que dependem da inserção do cabeçalho MPLS. Como o campo EXP possui 3 bits, é possível que o provedor de serviços ofereça 8 classes de diferenciação de serviços para cada LSP, sendo que cada uma define um determinado tratamento a ser dado aos pacotes com base no campo EXP.
- L-LSP (LABEL-*inferred*-PSC LSP): neste modo de operação os PHB são definidos utilizando-se os rótulos. Desta forma, é criado um LSP para cada classe de serviço.

Até o momento, não existe um padrão de mapeamento entre o campo DSCP e o campo EXP definido pelo IETF. Usualmente, fabricantes de equipamentos têm copiado os 3 bits mais significativos do campo DSCP para o EXP, como é o caso da CISCO [CIS2006].

Conforme pode ser visto em [CBB2006] há uma proposta em fase de desenvolvimento no sentido de padronizar o mapeamento entre o campo DSCP e o EXP. A tabela X foi retirada desta proposta e ilustra as classes de tráfego mais comuns: BE (Best Efort), AF (Assured Forwarding) e EF (Expedited Forwarding) sendo mapeadas para as 8 possíveis classes suportadas pelos 3 bits que compõem o campo EXP.

Tabela 4: Exemplo de mapeamento entre DSCP e EXP

DSCP	EXP	DSCP	EXP
BE-000000	0	AF32-011100	3
AF11-001010	1	AF33-011110	3
AF12-001100	1	AF41-100010	4
AF13-001110	1	AF42-100100	4
AF21-010010	2	AF43-100110	4
AF22-010100	2	EF-101110	5
AF23-010110	2	Controle de tráfego-110000	6
AF31-011010	3	Reservado-111000	7

5.1.8 Principais aplicações do MPLS

O MPLS pode ser aplicado a diversos tipos de cenários. Os principais exemplos de sua utilização serão vistos a seguir.

5.1.8.1 Configuração de Redes Privadas Virtuais (VPNs)

De acordo com [Gle2000], as VPNs podem ser vistas como uma emulação de uma rede privada de longo alcance (WAN) que opera sobre uma rede real (IP, ATM, Frame Relay, MPLS, etc). Elas permitem, por exemplo, que empresas separadas por longas distâncias interliguem suas redes de forma segura e a custo relativamente baixo utilizando para isso a estrutura da Internet ou alguma outra forma de conexão dedicada.

Inicialmente, as VPNs eram configuradas sobre circuitos ATM e Frame-Relay. Por ser um protocolo que oferece suporte a QoS e que possibilita a adoção de técnicas de Engenharia de tráfego, o MPLS tem sido considerado como sucessor das tecnologias de comutação por circuito citadas acima.

Além do suporte a QOS, o uso do MPLS na criação de VPNs garante um isolamento completo do tráfego a partir do momento em que são criadas tabelas de rótulos exclusivas para cada VPN [RR2006].

Segundo [RR2006], mesmo sem utilizar criptografia de dados, as VPNs baseadas no IP sobre MPLS conseguem atingir um nível de segurança equivalente ao obtido quando são utilizadas outras técnicas de camada 2 no *backbone*, como é o caso do Frame Relay. Isto significa que caso não haja erro de configuração, é impossível que sistemas participantes de uma VPN ganhem acesso à outra.

5.1.8.2 Engenharia de tráfego

Numa rede IP, a principal maneira que pode ser utilizada para se controlar o caminho a ser seguido pelos pacotes se dá por meio da alteração de custo dos enlaces. Não é possível exercer um controle que leve em consideração a origem do tráfego, já que o roteamento é realizado com base no endereço de destino dos pacotes [OS2003].

Embora seja possível realizar Engenharia de Tráfego por meio da alteração do custo dos enlaces, esta prática se torna proibitiva no caso de redes de grande porte, já que seria impraticável de ser realizada do ponto de vista técnico [OS2003].

A aplicação do protocolo MPLS para se obter Engenharia de Tráfego (MPLS TE) se baseia na criação de LSPs que serão utilizados para o encaminhamento dos fluxos de dados. Estes LSPs podem ser vistos como túneis e se assemelham aos circuitos virtuais (VCs – Virtual Circuits) das redes ATM.

Quando comparado ao encaminhamento realizado pelo protocolo IP com base no endereço de destino dos pacotes, o MPLS se mostra muito mais flexível, já que o roteador de entrada do túnel pode utilizar diversos outros campos do cabeçalho IP para decidir para qual LSP encaminhar os pacotes.

5.1.8.3 Infra-estrutura de rede multi-serviço

Também conhecido como AToM (*Any Transport over* MPLS), esta tecnologia é muito utilizada por provedores de serviços na migração para tecnologia MPLS. Sua arquitetura permite que um backbone IP/MPLS forneça transporte a diferentes protocolos de camada 2, tais como ATM, Frame Relay, Ethernet, PPP [Car2002].

6 Ferramenta Proposta

O roteamento IP tradicional determina o caminho a ser seguido por um pacote através da verificação do endereço de destino do mesmo e opta sempre pela rota de menor custo. Embora a simplicidade do IP tenha sido um dos principais fatores que facilitaram a sua disseminação, o que se observa na prática é que alguns enlaces ficam sobrecarregados enquanto outros permanecem ociosos.

Com a utilização das técnicas de engenharia de tráfego, problemas deste tipo podem ser minimizados, resultando num melhor aproveitamento dos ativos de rede e minimizando gastos com a contratação de novos canais de comunicação.

Considerando-se que a Internet deve ser utilizada para interligar diversas grades computacionais espalhadas ao redor do globo terrestre, surge a necessidade de tratar os fluxos de dados criados entre elas. Em um primeiro momento, os fluxos gerados entre grades computacionais podem parecer bem maiores que aqueles existentes entre aplicações comuns da Internet. No entanto, quando comparados com a totalidade do tráfego que flui por uma rede dessa magnitude, chega-se a conclusão que o tráfego entre grades computacionais representa uma pequena porcentagem do total de sua utilização.

À medida que a utilização das grades computacionais for aumentando, passa a ser necessário tratar seus fluxos de forma diferenciada. Além de evitar que eles prejudiquem ou sejam prejudicados por fluxos de outras origens, é possível também fazer um uso mais eficiente dos recursos disponibilizados pela rede, permitindo que sejam oferecidas garantias de QoS às aplicações.

Nesta seção será apresentada uma ferramenta de QoS que utiliza técnicas de Engenharia de Tráfego com o objetivo de fornecer QoS aos canais de comunicação estabelecidos em um ambiente de grade computacional.

Espera-se que sua utilização resulte em ganho de desempenho por meio da otimização no direcionamento dos fluxos de dados através dos LSPs que oferecerem melhores condições de comportar os mesmos.

6.1 Motivação e objetivos

O surgimento do MPLS permitiu que o caminho a ser seguido pelos pacotes fosse definido na sua origem. Esta característica faz dele um protocolo mais apropriado para ser utilizado em aplicações ligadas à Engenharia de Tráfego e Qualidade de Serviço, já que o estabelecimento prévio de rotas explícitas facilita a tarefa atribuir aos fluxos os caminhos capazes de atender às solicitações de garantia de serviço das aplicações.

Além disso, pelo fato do MPLS ser um protocolo que possibilita que a comutação de pacotes seja realizada sobre a camada de enlace, em muitos casos sua utilização já poderia ser justificada apenas devido ao aumento de desempenho esperado.

Dadas suas vantagens, incluindo aspectos relativos à existência de uma versão de código livre que permite o estudo de suas características em computadores equipados com sistema operacional Linux, este trabalho procurou direcionar esforços na pesquisa e aplicação do protocolo MPLS.

Desta forma, foram realizadas simulações em laboratório que culminaram no desenvolvimento de uma ferramenta que tem como objetivo prover QoS a aplicações destinadas a grades computacionais. Para isso, optou-se por fazer uso de técnicas de Engenharia de Tráfego em redes MPLS, tirando proveito de sua escalabilidade e realizando roteamento baseado na origem e nas capacidades de comunicação de cada segmento da rede.

6.2 Princípios de funcionamento

Antes de entrar em maiores detalhes em relação ao mecanismo de funcionamento da ferramenta proposta por esta dissertação de mestrado, é necessário ilustrar um ambiente no qual a mesma poderia ser utilizada.

Por meio da análise da Figura 18, podem ser vistos os principais itens que compõem um domínio MPLS: LERs, LSRs e LSPs. No caso, o LER1 representa o roteador de entrada do domínio MPLS, responsável pela inserção dos rótulos nos pacotes. O LER2 representa o roteador de saída do domínio e pode tanto retirar quanto empilhar rótulos, caso exista um segundo domínio que também utiliza MPLS. Já os LSRs realizam a comutação dos pacotes, utilizando apenas as informações contidas nos rótulos para decidir o caminho a ser tomado pelos mesmos.

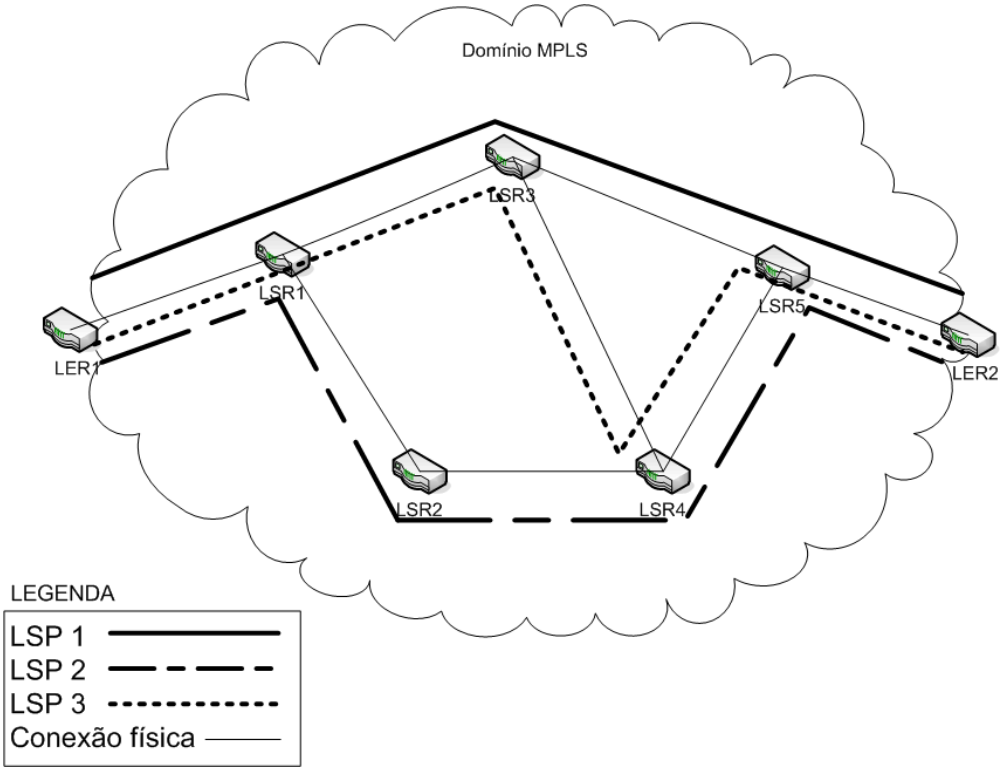

Figura 18: Configuração dos LSPs sobre a rede física

Ainda com relação à Figura 18, é importante visualizar a existência de 3 LSPs, ou seja, existem três possibilidades para que os pacotes provenientes do LER1 alcancem o LER2.

O fato de possuir uma rede que faça uso do protocolo MPLS não é suficiente para que diferentes fluxos sejam tratados com diferentes prioridades em seu interior. É preciso que exista um *middleware* responsável pela tradução das requisições de QoS feitas pelas aplicações em reservas efetivas de recursos. O levantamento das principais funcionalidades que este *middleware* precisa oferecer culminou na elaboração de uma ferramenta que atua

ativamente na configuração dos equipamentos, buscando direcionar os fluxos de dados através dos LSPs mais apropriados.

Supondo a existência de múltiplos caminhos entre o roteador de entrada e o roteador de saída do domínio MPLS, a ferramenta proposta atua como um mecanismo de seleção de LSPs, buscando utilizar aquele que estiver menos congestionado e que seja capaz de atender aos requisitos de QoS das aplicações.

A Figura 19 ilustra a seleção (em negrito) de uma das possíveis rotas existentes entre uma grade cujo acesso à Internet é intermediado pelo LER1 e outra que está ligada à Internet pelo LER2.

.

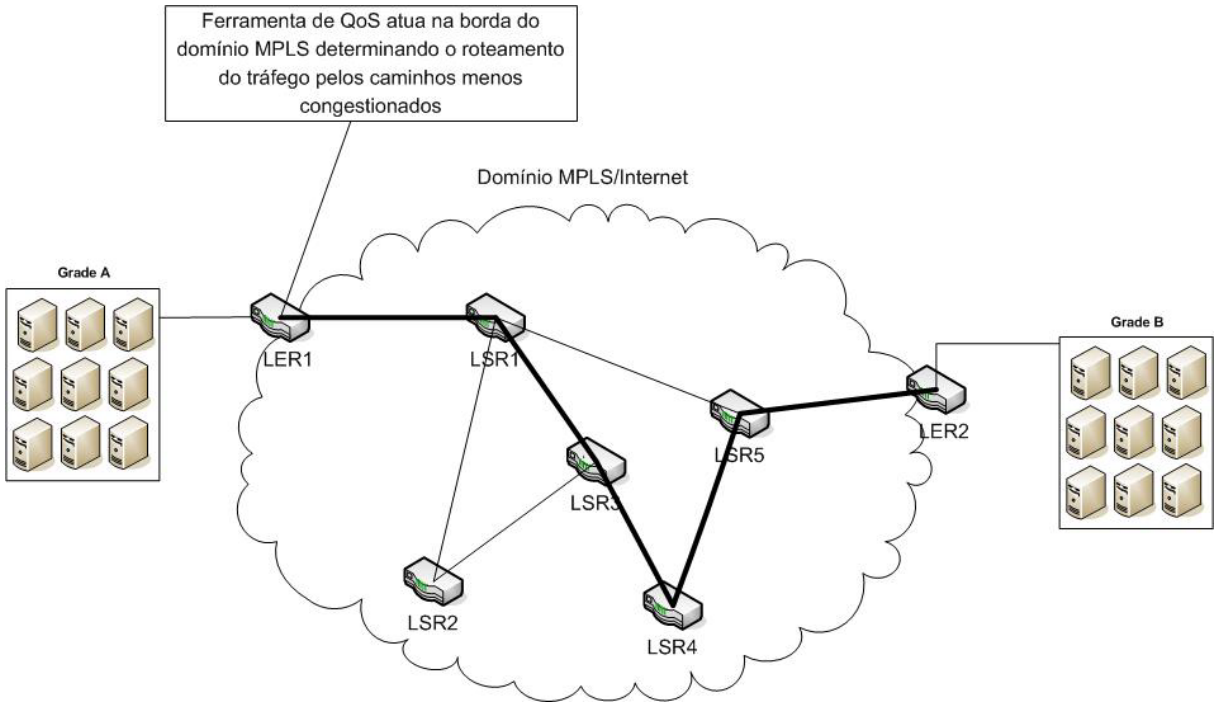

Figura 19: Mecanismo de seleção de rota dentro de um domínio MPLS

O uso do roteamento IP tradicional não permitiria que o processo descrito acima fosse realizado de forma satisfatória, já que é caracterizado pela seleção de rotas de menor custo. Como na maioria das vezes o custo é calculado levando-se em consideração o número de nós

que fazem parte do caminho a ser percorrido, determinadas rotas acabam congestionadas ao mesmo tempo em que outras permanecem ociosas.

Da mesma forma, a utilização de protocolos de propagação e aprendizagem de rotas, como OSPF, não seria capaz de adequar o encaminhamento de pacotes da maneira desejada. Embora ajuste as rotas em função do estado da comunicação, os caminhos ajustados valem para todos os pacotes, falhando ao priorizar fluxos distintos.

Em um domínio MPLS os pacotes partem do roteador de entrada com uma rota pré-definida, ou seja, os LSRs não influenciam no caminho a ser seguido e apenas encaminham os pacotes de acordo com o rótulo inserido no LER de entrada. Em virtude disso, optou-se por uma implementação centralizada ao invés de adotar uma abordagem distribuída. Assim, o LER de entrada do domínio MPLS foi escolhido como sendo o local mais adequado para se executar a aplicação que gerencia as informações e toma a decisão de qual LSP será utilizado pelos pacotes.

Outra tarefa que também fica a cargo do roteador de entrada é a de classificar os pacotes que irão adentrar no domínio MPLS. Esta classificação é realizada por um filtro de pacotes que irá determinar a qual FEC estes serão associados. Na prática, associar um pacote a uma FEC corresponde ao ato de rotular o pacote. Para isso é preciso analisar o conteúdo de informações contidas no cabeçalho dos mesmos, tais como endereço de origem, endereço de destino, protocolo, bits do campo TOS, bits do campo DSCP, etc.

É importante salientar que, em um domínio MPLS, apenas os roteadores de borda são capazes de analisar os pacotes na camada de nível 3 e por isso a determinação da FEC a que cada pacote pertence deve necessariamente ser feita pelo LER de entrada. Os LSRs de núcleo do domínio devem ser vistos como roteadores MPLS “puros”, ou seja, comutam pacotes considerando apenas as informações contidas no cabeçalho MPLS.

6.2.1 Configuração dos LSPs

Antes que a ferramenta possa auxiliar no direcionamento do tráfego é necessário que sejam configurados LSPs buscando utilizar o maior número possível de caminhos entre o roteador de entrada e o roteador de saída do domínio MPLS.

Embora possa parecer tecnicamente inviável mapear a totalidade de nós de uma grade inseridos em um domínio MPLS, segundo [TMK2006], os fabricantes de equipamentos de rede têm trabalhado no sentido de melhorar o desempenho de seus roteadores por meio do aumento das suas capacidades de processamento e memória. O objetivo é que eles sejam capazes de armazenar previamente todas as possibilidades de caminho existentes entre os nós de uma grade. Além disso, têm sido adicionadas a eles funcionalidades que permitem o gerenciamento inteligente de caminhos ociosos.

Idealmente, o uso de um protocolo de distribuição de rótulos poderia simplificar significativamente o processo de configuração dos LSPs. Caso fosse utilizado o LDP, a alocação e a troca de rótulos entre nós vizinhos ocorreriam de forma automatizada. Isto implicaria diretamente na diminuição da carga imposta ao administrador da rede para configurá-la.

Pelo fato de não existir uma implementação do protocolo LDP para o *mpls-for-linux*, o desenvolvimento da ferramenta proposta incluiu um mecanismo mais simples destinado à atribuição de rótulos aos LSPs no momento em que eles são criados. Para isso, foi utilizada uma abordagem centralizada que faz uso de um banco de dados situado no LER de entrada do domínio MPLS. Espera-se que logo que o protocolo LDP seja disponibilizado, o mesmo possa ser incluído na ferramenta como um módulo, substituindo o mecanismo criado até o momento. Ao utilizar um protocolo já consolidado, a ferramenta poderá interoperar com outras soluções com maior facilidade.

O primeiro passo do processo de configuração dos LSPs consiste na coleta de informações dos nós que irão compor o domínio MPLS. Cabe ao administrador da rede registrar no banco de dados as informações relativas a cada nó, tais como: nome (identificador), função (LSR ou LER), endereços das interfaces de rede disponíveis no próprio nó, endereços das interfaces de redes adjacentes (pertencentes aos nós vizinhos) e IP de configuração (utilizado para o acesso inicial necessário para o recebimento de comandos de configuração).

Depois de registrados os nós, ocorre o processo de geração de todos os caminhos possíveis que saem do LER de entrada e levam até o LER de saída do domínio MPLS. Considerando-se que as informações inseridas pelo administrador da rede no banco de dados estejam corretas, é criado um grafo que representa todas as possibilidades de interconexão

entre os nós da rede. Em seguida, um algoritmo recursivo percorre todos os nós que formam cada caminho. Ao final, os caminhos encontrados são armazenados no banco de dados.

Para cada possibilidade de caminho encontrado pela ferramenta, será criado um LSP correspondente. Depois de gerados os scripts de configuração, a ferramenta de QoS os envia para cada nó via SSH (*Secure Shell*). Futuramente, espera-se que sejam desenvolvidos processos residentes (*daemons*) em cada nó responsáveis pelo recebimento e execução de mensagens que resultarão na configuração dos mesmos. Desta forma, elimina-se a necessidade de se trocar as chaves dos nós antes de se começar seu processo de configuração.

Assim, o processo de configuração de cada LSP consiste na:

- Escolha dos rótulos que irão identificar as FECs que trafegarão pelos LSPs formados no interior do domínio MPLS.
- Configuração do nó responsável pela inserção dos rótulos nos pacotes no momento em que estes entram no domínio MPLS (LER de entrada).
- Configuração dos nós intermediários (LSRs), responsáveis pela operação de troca de rótulos (*label swapping*) , ou seja, realizam o repasse dos pacotes.
- Configuração do nó de saída do domínio MPLS (LER de saída), responsável pela retirada do rótulo e posterior entrega do pacote a seu destino final.

Depois de configurados os nós, é preciso determinar as capacidades e monitorar o estado de comunicação de cada enlace. Na próxima seção serão mostrados maiores detalhes relativos ao monitoramento do estado dos LSPs.

Uma segunda abordagem para a manutenção dos rótulos e LSPs consiste no uso do protocolo de gerenciamento de rede, o SNMP. Sua utilização em ativos de rede depende do suporte às MIBs por parte dos fabricantes. Estas podem ser proprietárias ou definidas pelo IETF através de RFCs, ou seja, podem conter tanto objetos proprietários quanto padronizados.

O IETF, através do RFC3815, define algumas bases de dados de informações de gerenciamento (MIBs) para objetos MPLS. No total são definidos quatro módulos: MPLS-

LDP-STD-MIB, MPLS-LDP-GENERIC-STD-MIB, MPLS-LDP-ATM-STD-MIB e MPLS-LDP-GENERIC-STD-MIB.

Supondo a existência de agentes SNMP com suporte às MIBS MPLS nos LSRs/LERs, os dados relativos ao estabelecimento de sessões LDP poderiam ser armazenados em objetos da tabela mplsLdpEntityTable. Os objetos listados nesta tabela permitem que LERs e LSRs iniciem ou respondam a requisições de estabelecimento de sessões LDP [CSL2004].

A Tabela 5 apresenta alguns dos objetos que constituem a tabela mplsLdpEntityTable. Seu conteúdo completo pode ser obtido em [CSL2004].

Tabela 5: Objetos da tabela mplsLdpEntityTable

Objeto	Descrição
mplsLdpEntityLdpId	Fornece o identificador do LDP
mplsLdpEntityIndex	É um índice para a tabela.
mplsLdpEntityProtocolVersion	Indica o número da versão do protocolo LDP a ser utilizado na mensagem de inicialização da sessão.
mplsLdpEntityAdminStatus	Indica o estado administrativo desta entidade LDP. Se seu estado for "habilitado", significa que esta entidade irá tentar estabelecer uma nova seção com um roteador adjacente. Se seu estado for "desabilitado", toda a informação com relação ao roteador em questão deve ser apagada.
mplsLdpEntityOperStatus	Indica o estado operacional desta entidade LDP, podendo ser: desconhecido, habilitado ou desabilitado.
mplsLdpEntityTcpPort	Indica o número da porta TCP utilizada pelo protocolo LDP. Comumente, é utilizada a

	porta padrão: 646.
mplsLdpEntityUdpDscPort	Indica a porta UDP de procura. Comumente é utilizada porta padrão: 646.

Em [CSL2004] podem ser encontradas maiores informações relativas a outros objetos gerenciáveis destinados à manutenção dos rótulos pelo LDP utilizando-se o protocolo SNMP.

Por enquanto, o *mpls-for-linux* ainda não suporta o gerenciamento via SNMP, sendo esta uma funcionalidade já prevista no desenvolvimento do mesmo.

6.2.2 Monitoramento do estado e seleção dos LSPs a serem utilizados

Depois de configuradas todas as possibilidades de rotas entre o roteador de entrada e de saída do domínio MPLS é preciso dar início ao monitoramento dos LSPs. Esta tarefa é realizada por um módulo de monitoramento baseado no NWS (*Network Weather Service*) [NWS2006]. O NWS é um *software* largamente utilizado em grades computacionais e realiza o monitoramento e previsão de carga de diversos tipos de recursos, inclusive de rede.

Sua integração com a ferramenta proposta por esta dissertação se dá por meio de uma API de programação fornecida com o próprio NWS, desenvolvida em linguagem C.

Para que o NWS funcione adequadamente, é preciso que sejam executados um processo servidor de nomes (realiza o registro dos endereços IP e portas de cada *host* que está sendo monitorado) e um servidor de memória (armazena os dados obtidos pelas medições realizadas). Neste trabalho, optou-se por executar estes processos no LER de entrada do domínio, responsável pela maior parte do gerenciamento da rede.

Além do servidor de nomes e de memória, também devem ser executados os sensores, responsáveis pela realização das medições dos estados dos recursos disponíveis. A partir deles são obtidos dados relativos ao estado dos LSPs, ou seja, pode-se monitorar os estados dos enlaces de comunicação. Além das informações relativas à rede, o NWS também possui funcionalidades que permitem medir a utilização de processador e a utilização de disco [WSH1999].

A Figura 20 ilustra o processo de monitoramento de LSPs de um domínio MPLS. Nela podem ser vistos o processo servidor de nomes (NS), o processo servidor de memória (MS) e os sensores (S, S1, S2 e S3).

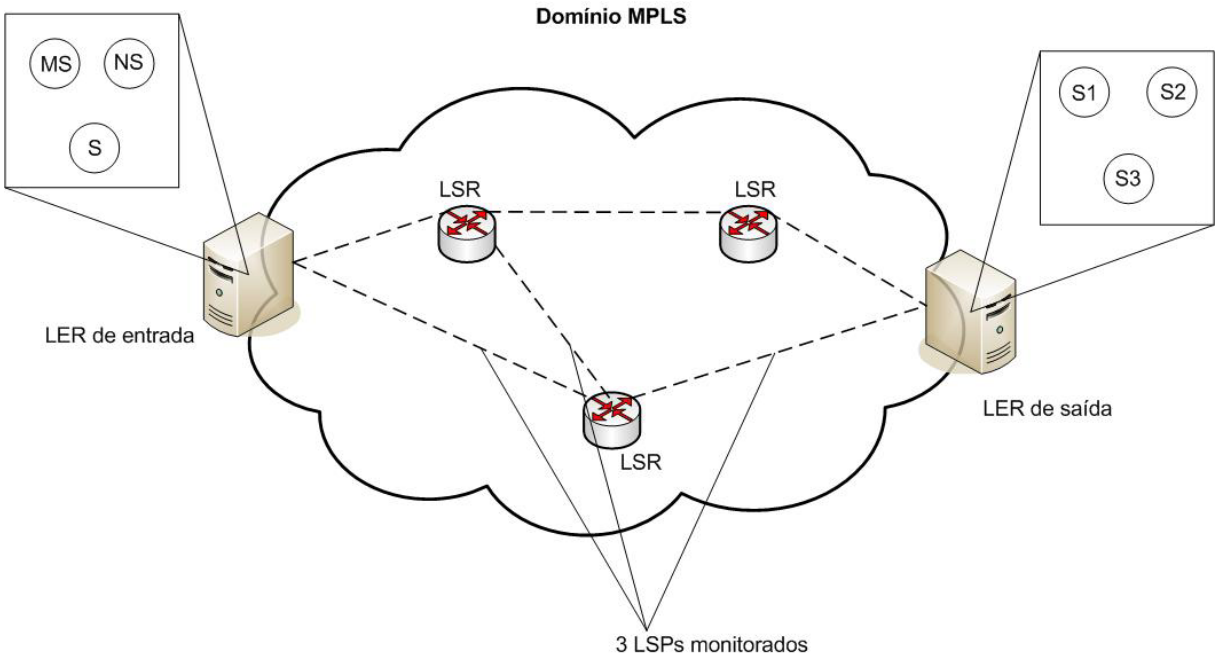

Figura 20: Exemplo de monitoramento de 3 LSPs

Para monitorar o estado de vários LSPs foi preciso executar um processo sensor tanto no LER de entrada quanto no LER de saída para cada LSP existente entre eles. Como é possível determinar qual porta deve ser utilizada para que os diferentes sensores troquem informações, não há problemas em se executar diversas instâncias da mesma aplicação.

Neste trabalho, foram utilizadas 3 métricas para se estimar o estado de utilização de cada LSP: largura de banda disponível, latência média e disponibilidade.

Com relação aos métodos de monitoramento de recursos de rede utilizados, houve uma preocupação com o caráter intrusivo dos mesmos. Uma medição intrusiva é aquela em que pacotes são inseridos na rede com o objetivo de levantar o estado da mesma. Segundo [FH1999], quanto maior o nível de detalhamento desejado a respeito do estado da rede, também maior será o tráfego inserido para obtenção das informações, o que também acarreta imprecisão nos dados coletados.

Nos testes realizados, foi observado que os sensores de rede do NWS interferem muito pouco na carga da rede. Em intervalos regulares são estabelecidas conexões entre pares de *hosts* em que os sensores estão em execução. Tanto o tamanho dos *buffers* de envio e recepção quando o tamanho das mensagens a serem transferidas podem ser passados como parâmetro aos processos executados pelo NWS. No entanto, segundo [WSH1999], observou-se empiricamente que mensagens de 64K bytes e 32K bytes de *buffer* para *socket* permitem obter resultados que retratam fielmente o estado de utilização da rede. Desta forma, optou-se por utilizar os valores padrão dos parâmetros acima citados.

Os resultados obtidos com as medições juntamente com outras informações relevantes para o estabelecimento de LSPs são armazenados em um banco de dados situado no LER de entrada do domínio. Por ser de código livre, foi utilizado o Mysql [Mys2006].

Com base nas informações coletadas, é criada uma matriz de roteamento, localizada no LER de ingresso do domínio MPLS. Esta matriz deve ser mantida em constante atualização, buscando refletir precisamente o verdadeiro estado de cada LSP.

A Tabela 6 ilustra o conteúdo de uma matriz de roteamento. A partir da análise dos dados dessa matriz são tomadas as decisões relativas a qual LSP será utilizado para o encaminhamento dos fluxos de dados. Como pode ser visto, cada linha que a constitui traz informações relativas ao status de comunicação de cada um dos LSPs. Optou-se por monitorar os seguintes itens:

- Largura de banda disponível: Representa a parcela da banda total ainda não alocada no canal de comunicação.
- Latência: Corresponde a uma medida do atraso de transmissão que existe entre dois *hosts* de uma rede. Seu valor pode sofrer variações de acordo com o nível de utilização da rede.
- Disponibilidade do canal: Indica se o canal está ou não em operação.

Tabela 6: Matriz de roteamento

LSP	Banda Disp.	Latência	Disponível
1	8.3 Mbps	8ms	Sim
2	98 Mbps	3ms	Sim
3	65.3 Mbps	5ms	Não

Por enquanto, cabe à aplicação optar pela utilização do caminho que oferecer maior quantidade de banda disponível ou menor latência. No futuro, pretende-se criar um mecanismo com lógica mais elaborada e que seja capaz de optar pelo caminho que oferecer melhores condições de trafegabilidade de forma mais inteligente.

Definida a rota a ser tomada, a ferramenta passa a associar os pacotes às suas respectivas FECs utilizando um filtro de pacotes existente no LER de entrada para direcioná-los aos devidos LSPs.

Uma funcionalidade do NWS que não foi explorada é seu mecanismo de previsão de carga dos recursos de rede. A partir do histórico de utilização dos recursos ele prevê qual será a carga existente em determinado recurso em um momento futuro. Este é um recurso bastante útil para se efetuar reservas prévias de recursos, porém, suas informações só teriam utilidade caso houvesse uma melhor integração entre a ferramenta de QoS e o escalonador de processos da grade computacional.

O monitoramento constante dos estados dos *links* visa garantir que as informações relativas aos estados dos mesmos estejam sempre atualizadas. Com base nessas informações, prioriza-se o estabelecimento da comunicação por meio do direcionamento do tráfego pelos LSPs que apresentam melhores taxas no envio de dados, menores atrasos e maior disponibilidade.

Visando estabelecer o caminho a ser percorrido pelos fluxos de dados, os *daemons* da ferramenta de QoS também disparam nos nós intermediários comandos responsáveis pela interconexão de LSPs. Esta interligação é conhecida como *cross-connecting*. A interconexão de um ou mais LSPs forma verdadeiros túneis através dos quais os pacotes passam a ser comutados com base apenas nos seus rótulos. Isto implica diretamente num menor tempo de

latência no processo de comunicação, já que, depois de inseridos no túnel, passa a ser desnecessária a consulta a tabelas de rotas para definir o caminho a ser tomado pelos pacotes.

6.3 Indo além da Engenharia de Tráfego

Do ponto de vista das aplicações que fazem uso da rede gerenciada pela ferramenta proposta, a definição dos parâmetros de QoS a serem atendidos (vazão, atraso, perda de pacotes, disponibilidade, etc) podem ser vistos como contratos de nível de serviço, mais conhecidos como *Service Level Agreements* (SLAs).

Um SLA é composto por um conjunto de parâmetros relacionados principalmente com o desempenho e a disponibilidade da rede. Para atender às exigências das aplicações, a rede deve ser capaz de prover no mínimo o que for acordado através do SLA, caso contrário não será possível proceder a execução das mesmas.

Muitas vezes, a definição inicial de roteamento tomada para cada fluxo de pacotes acaba tornando-se inadequada devido à possibilidade de congestionamento em um determinado LSP da rede. Em alguns casos, o simples remanejamento do tráfego para outro LSP pode não ser suficiente para atender aos requisitos de QoS das aplicações. Assim, passa a ser necessário executar a marcação dos pacotes de forma que eles passem a ser tratados pelos roteadores com diferentes níveis de prioridade.

A partir do kernel 2.2 o Linux passou a contar com um novo conjunto de ferramentas capazes de executar funções de roteamento, filtragem e classificação de pacotes [HUB2002]. Suas funcionalidades podem ser exploradas através do comando "ip", que controla configurações relativas ao IPv4 e ao IPv6. Além disso, um outro comando, o "tc" (*traffic controller*) permite que seja realizado um controle mais minucioso do tráfego.

Através da adição de regras com o comando iptables é possível executar a marcação de pacotes para que posteriormente eles sejam tratados pelo conjunto de ferramentas que compõem o iproute2 [HUB2002] de acordo com diferentes níveis de prioridade pré-estabelecidos pelo administrador da rede.

O tc é um comando do Linux que pode ser utilizado para criar e associar filas às interfaces de rede. Atuando em conjunto com o iptables, ele permite que o tráfego seja

filtrado para que posteriormente ocorra o mapeamento das FECs nos LSPs previamente configurados.

Considerando que um determinado domínio MPLS pode estar inserido dentro de um domínio *diffservice*, é necessário utilizar mecanismos de mapeamento entre o campo DSCP do diffservice e o EXP do MPLS. Esta tarefa é realizada através do comando ds2exp, presente no pacote *mpls-for-linux*.

Neste ponto, é interessante lembrar que, na classificação de pacotes MPLS, só é possível realizar comparações com o campo EXP mais externo da pilha de rótulos. No caso dos pacotes IP, eram várias as possibilidades de comparação: endereço IP de destino, endereço IP de origem, valores DSCP, etc. [OS2003]

Depois de realizada a conversão na interface de entrada de cada LER, cabe às interfaces de saída a função de policiamento do tráfego, ocorrendo nova conversão através do comando exp2tc.

7 Experimentos realizados

Diversos experimentos foram realizados visando comparar o desempenho de uma rede que utiliza as técnicas de roteamento tradicional do IP com o de uma rede baseada no protocolo IP sobre MPLS. Procurou-se também validar a ferramenta proposta, verificando sua aplicação.

Devido ao fato da ferramenta estar em fase final de desenvolvimento no momento em que foram realizados os testes abordados por este capítulo, algumas configurações foram realizadas manualmente por meio da execução de scripts diretamente nos *hosts*.

As experiências obtidas na configuração do ambiente de testes mostraram que detalhes inerentes a cada sistema operacional e à correta configuração dos dispositivos de *hardware* (principalmente placas de rede e respectivos *drivers*) podem fazer com que os resultados obtidos por uma simulação sejam bem diferentes daqueles obtidos por uma rede real.

Devido à dificuldade em se retratar fielmente todos os comportamentos de uma rede real por meio de uma simulação, foi utilizada uma rede de testes visando aproximar ao máximo a reprodução de um ambiente de produção.

Inicialmente, pretendia-se utilizar placas de rede do tipo Gigabit Ethernet. Foram testados dois modelos de placas instaladas em diferentes máquinas. Também foram compiladas 3 versões do do *kernel* do Linux. Em todas as tentativas os resultados obtidos não se mostraram satisfatórios, variando ao acaso sem que se fosse detectado o motivo do problema. Provavelmente, os *drivers* das placas de rede que não estavam aptos a operar com o MTU tradicional Ethernet (1500 bytes) com suporte para o cabeçalho MPLS.

Foram feitas algumas tentativas de alteração do código dos *drivers,* porém sem sucesso. Como a velocidade nominal da rede não era um fator determinante para a pesquisa, as placas Gigabit foram substituídas por outras, desta vez, compatíveis com o padrão Fast Ethernet. Somente após esta substituição os resultados adquiriram consistência e se aproximaram do que era esperado.

Devido ao fato do laboratório de pesquisa não possuir ativos de rede compatíveis com a tecnologia MPLS, foi necessário configurar os roteadores em computadores comuns utilizando-se uma implementação do protocolo MPLS para Linux.

Antes de serem apresentados os resultados e conclusões dos testes realizados, a próxima seção apresentará uma introdução ao *mpls-for-linux* [LC2005].

7.1 Características do *mpls-for-linux*

Como pode ser visto em [LC2005], o projeto *mpls-for-linux* surgiu a partir da implementação do protocolo de distribuição de rótulos (LDP), seguindo as especificações fornecidas pelo RFC 3036. O início do desenvolvimento ocorreu no ano de 1999. Em 2000 o pacote "linux-mpls-ldp" foi dividido em dois, sendo eles:

- mpls-linux: composto por um plano de comutação mpls a ser inserido no *kernel* do Linux através de um *patch*. Foi disponibilizado sob GPL.
- ldp-portable: composto por uma versão do protocolo LDP adaptada para Linux, disponibilizada sob LGPL.

Desde o final do ano de 2000 o projeto é disponibilizado pelo *SourceForge* [LC2005], um site dedicado à hospedagem de projetos de código livre.

As primeiras versões do *mpls-for-linux* foram desenvolvidas para o *kernel* 2.4.X . O utilitário *mplsadmin* era o responsável pela criação dos LSPs, realizada manualmente a partir da linha de comando. A *interface* entre o ambiente do usuário e o *kernel* adaptado para o mpls utilizava chamadas à função ioctl [CL2003].

As versões do *mpls-for-linux* compatíveis com o *kernel* 2.6.X passaram a utilizar uma nova API de comunicação: o Netlink. Além disso, foi disponibilizado um novo utilitário de configuração dos LSPs, agora chamado mpls[CL2003].

Atualmente, o *mpls-for-linux* encontra-se na versão 1.950, lançada no mês de dezembro do ano de 2005. Pelo fato do desenvolvedor do *mpls-for-linux*, James R. Leu, disponibilizar as versões mais recentes do seu projeto via pacotes rpm destinados à

distribuição Fedora [Fed2006], optou-se também por sua utilização nos experimentos descritos por este capítulo.

7.2 Topologias de testes utilizadas

A validação da ferramenta foi realizada em uma rede constituída por roteadores MPLS baseados em software livre (Linux) com *kernel* modificado para suportar o roteamento baseado em rótulos.

Com o objetivo de obter uma comparação inicial entre uma rede IP e uma rede MPLS, foram configurados 3 roteadores utilizando-se computadores convencionais compostos pelo hardware abaixo:

- Processador: Pentium IV 2.8 GHz
- Memória: 512 MB
- Placa de rede: Intel ethernet 10/100, *chipset* i815

7.2.1 LER – LSR – LER

Embora a configuração de apenas 3 máquinas possa não parecer muito representativa quando comparada a um ambiente real, seu estudo foi bastante útil para a aprendizagem inicial do funcionamento do protocolo MPLS e do utilitário *mpls-for-linux*. Além disso, o experimento descrito nesta seção não visa ilustrar o funcionamento da ferramenta de QoS. O objetivo principal é obter um comparativo inicial da diferença de desempenho existente entre uma rede que utiliza o protocolo IP e outra que utiliza o protocolo IP sobre MPLS.

Uma das dificuldades encontradas na configuração deste ambiente de testes foi a falta de documentação em relação aos comandos disponibilizados pelo utilitário *mpls-for-linux*. Este fator dificulta sua utilização e acaba consumindo bastante tempo do usuário. As informações disponibilizadas em [LC2005] se resumem a exemplos de seqüências de comandos necessários para se configurar pequenas redes. No entanto, em [LC2005] também

existe uma lista de discussão onde podem ser feitas perguntas que muitas vezes são respondidas pelo próprio desenvolvedor do projeto.

No APÊNDICE D foram mostrados detalhadamente todos os passos necessários para se configurar a rede de testes utilizada nesta seção. A partir do que foi descrito é possível entender o funcionamento dos comandos e configurar topologias de redes mais complexas com bastante facilidade.

A Figura 21 ilustra a topologia de testes utilizada nesta seção. As máquinas Júpiter e Urano atuam como LERs, sendo a primeira o LER de entrada do domínio MPLS e a segunda o LER de saída. A máquina denominada Saturno é um LSR, sendo capaz de encaminhar pacotes com base apenas nas informações contidas em seus rótulos. Podemos considerá-lo como um "roteador mpls-puro" já que ele não possui nenhuma entrada em sua tabela de rotas para a rede 10.0.0.0.

A topologia de testes é formada por 2 LSPs. Pelo LSP 1 trafegam os pacotes no sentido da esquerda para direita, sendo o LSP 2 responsável pelo tráfego no sentido contrário.

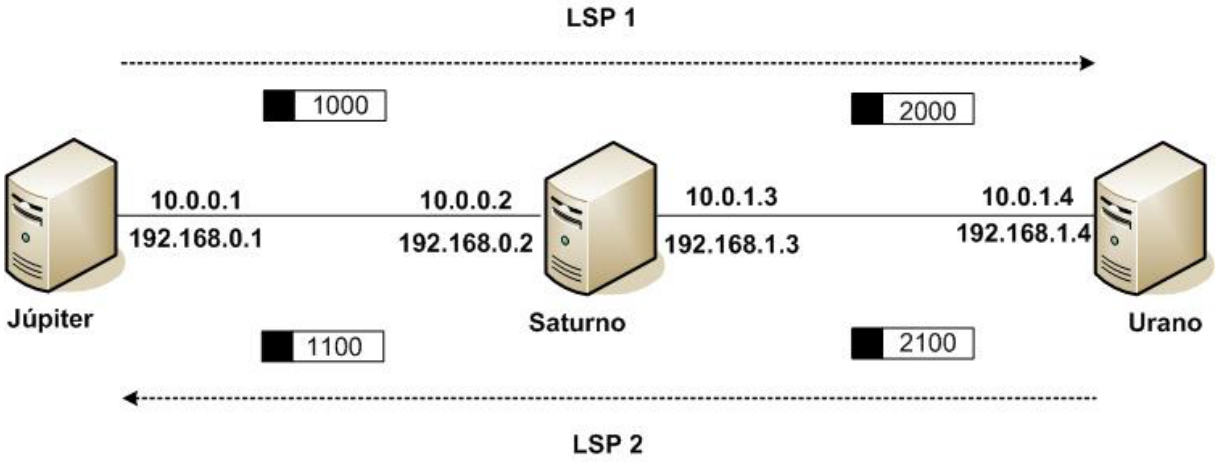

Figura 21: Configuração de uma rede composta por 3 roteadores MPLS

Ainda com relação à Figura 21, é possível visualizar que foram configuradas duas redes diferentes. A rede 10.0.0.0 utiliza o protocolo MPLS. Já a rede 192.168.0.0 é baseada no protocolo IP. Desta forma, o desempenho de ambos os protocolos pode ser comparado sem a necessidade de realizar configurações adicionais entre as baterias de testes.

Procurou-se avaliar se há ganho de desempenho ao se utilizar roteamento IP sobre MPLS ao invés do roteamento IP tradicional. Para isso, foram gerados fluxos de dados variando-se o tamanho do pacotes, facilitando a análise da carga adicional existente no processo de inserção e remoção de rótulos nos pacotes.

Os testes realizados buscaram seguir os aspectos definidos em [RFC2544] e [RFC3511] que definem metodologias destinadas à avaliação de desempenho de equipamentos de rede.

Como as ferramentas de geração e medição de tráfego mais conhecidas não seguem todas as recomendações definidas por estes RFCs, utilizou-se também uma ferramenta desenvolvida pelo próprio departamento de computação da Universidade de São Carlos, chamada NM (*Network Meter*). A maior familiaridade com seu código permitiu que alguns ajustes fossem feitos visando seguir o que determinam os RFCs acima citados.

7.2.1.1 IP x IP sobre MPLS

Este teste foi executado visando à obtenção de um comparativo inicial de desempenho entre uma rede IP e uma rede MPLS. Foram utilizadas duas ferramentas geradoras de tráfego: o NM e o Iperf.

As medições realizadas com a ferramenta NM levaram em consideração os seguintes parâmetros:

- Conforme determinado em [RFC2544], considerando-se que os testes foram realizados em uma rede do tipo Ethernet, devem ser feitas medições variando-se o tamanho dos pacotes. Neste caso, foram utilizados pacotes de: 64, 128, 256, 512, 1024, 1280 e 1518 bytes.
- A quantidade de pacotes enviada para cada tamanho foi de 1.000.000.
- Cada medição foi repetida 5 vezes. Considerou-se como resultado final a média aritmética simples dos valores obtidos. O desvio padrão também foi calculado para cada medição.

Com relação ao NM é preciso salientar que diferentemente de outras ferramentas de avaliação de redes, como o Iperf e o Netperf, o programa NM mede o *throughput* obtido pela aplicação considerando os dados lidos diretamente do *socket*. Desta forma, a latência sentida pela aplicação nas transmissões está inclusa nos resultados obtidos.

O resultado pode ser visto na Figura 22 :

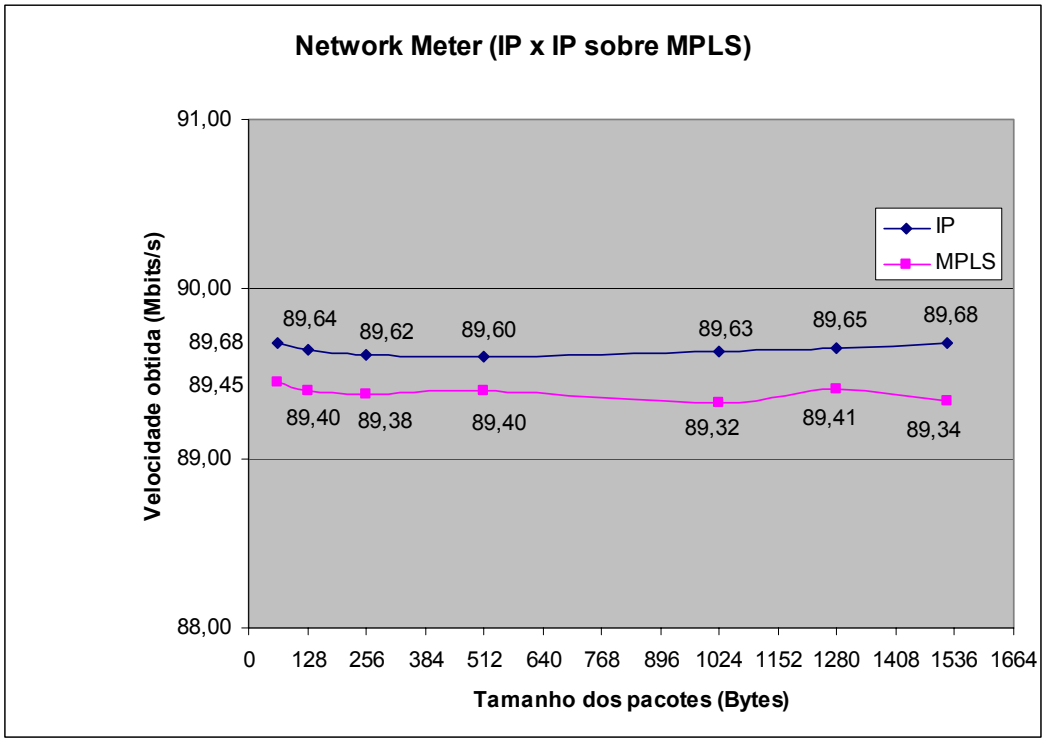

Figura 22: Resultado da medição utilizando o NM

Como pode ser visto, o desempenho obtido com o uso do protocolo IP foi ligeiramente superior que o do MPLS. Quando utilizado o protocolo IP, o desvio padrão (σ) obtido foi de 0,03. Já no caso do protocolo IP sobre MPLS, obteve-se um desvio padrão (σ) de 0,04.

A segunda medição foi feita com o Iperf. Ao contrário do NM, ele não permite que o tamanho dos pacotes seja variado e nem que o número deles seja escolhido. Assim, foi estipulado um tempo de 30s para cada teste. Da mesma forma, o resultado obtido foi proveniente da média aritmética simples de 5 medições consecutivas, com desvio padrão (σ) de 0,16 para o IP e 0,11 para o IP sobre MPLS. O resultado obtido pode ser visualizado na Figura 24:

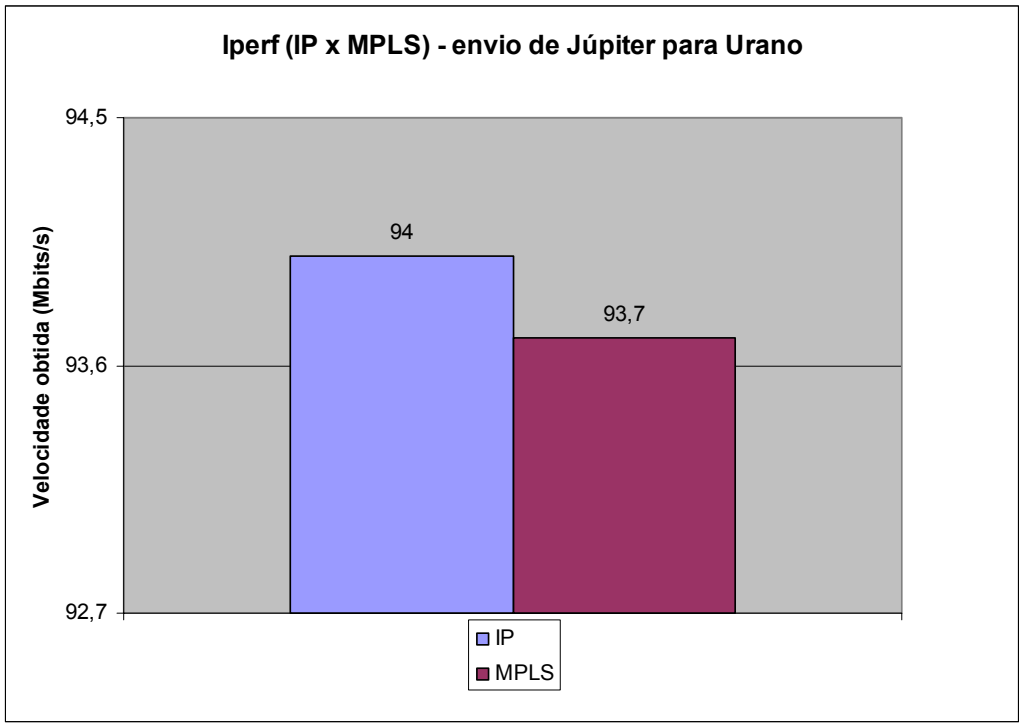

Figura 23: Resultado da medição utilizando o Iperf

7.2.1.2 Conclusões preliminares

Os resultados obtidos mostram que, neste cenário, não houve um ganho de desempenho do roteamento IP sobre MPLS em relação ao roteamento tradicional feito pelo protocolo IP. Devido a maior velocidade que pode ser obtida com o encaminhamento realizado pela camada 2, esperava-se um ganho de desempenho nos testes realizados com o protocolo IP sobre MPLS.

Em teoria, o fato de não ter que chegar a analisar um cabeçalho de camada 3 e não depender da decisão individual de cada roteador para definir a rota a ser tomada pelos pacotes deveria ter implicado em taxas mais altas de transmissão entre os roteadores de entrada e saída do domínio MPLS.

Além disso, o protocolo MPLS também leva vantagem por não ter que realizar o cálculo do *checksum* em cada roteador de núcleo do domínio. No caso do protocolo IP, é sabido que, toda vez em que há alguma alteração em algum campo de seu cabeçalho, é preciso recalcular o *checksum* por meio de uma função *hash* denominada CRC (*Cyclic*

Redundancy Check). O valor do *checksum* é calculado e inserido no cabeçalho para que possa ser conferido após a transmissão do pacote. Como o campo TTL é decrementado de uma unidade em cada roteador, sempre há alteração no cabeçalho, o que obriga o cálculo do *checksum*. Este procedimento assegura a manutenção da integridade do cabeçalho e implica em carga de processamento nos roteadores.

Provavelmente, o fato do projeto *mpls-for-linux* ser experimental e seu código não se encontrar no mesmo estágio de desenvolvimento do código do protocolo IP é um fator que contribui para este resultado. Além disso, há de se levar em consideração o trabalho extra realizado pelos LERs para rotular e atribuir os pacotes às suas FECs.

7.2.2 Rede peixe

Para ilustrar o funcionamento da ferramenta de QoS proposta por este trabalho e sua capacidade de direcionar os fluxos de dados foi configurada uma rede de testes que possui mais de uma alternativa de caminho a ser seguido entre a estrada e a saída do domínio MPLS. Sua topologia é ilustrada na Figura 24.

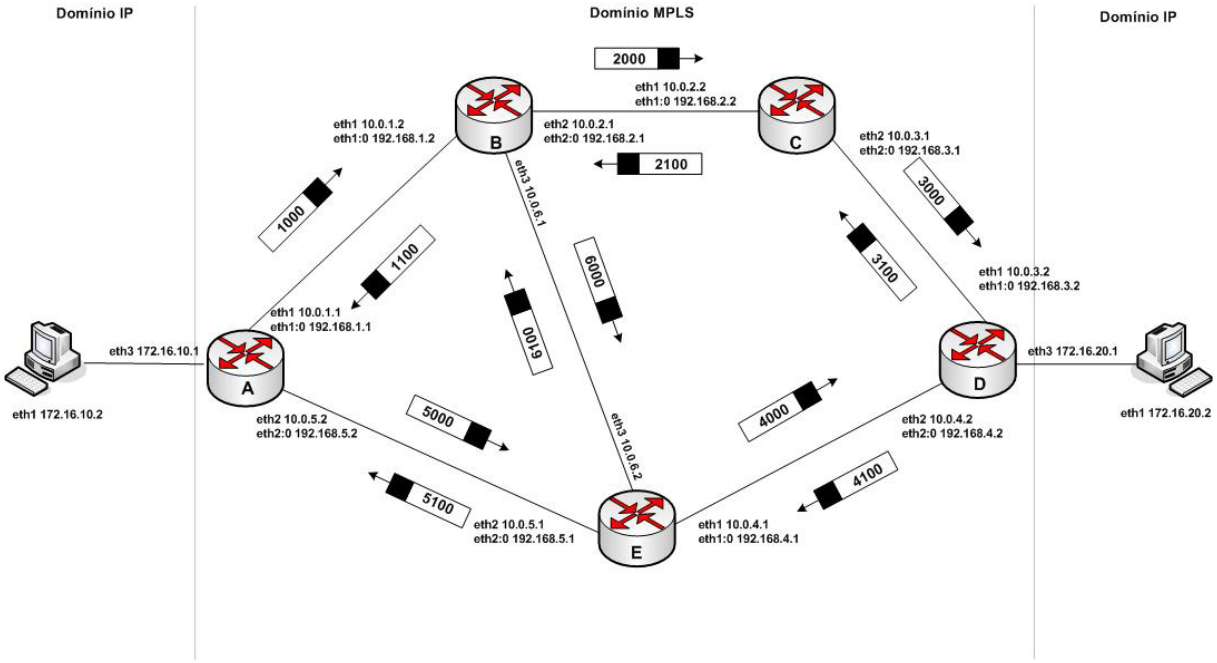

Figura 24: Topologia de testes utilizada

Como pode ser visto, existem 5 roteadores MPLS, sendo 2 LERs (A e D) e 2 LSRs (B,C e E). Todos os testes foram realizados no sentido da esquerda para direita, ou seja, o LER A deve ser considerado como sendo a entrada do domínio MPLS e o LER D deve ser considerado como sendo a saída do domínio MPLS.

Com exceção das conexões entre as redes IP e seus respectivos roteadores de acesso à malha (A e D) que possuem velocidade de 1 Gbps, todos os demais enlaces possuem velocidade de 100 Mbps. Desta forma é possível saturar com maior facilidade os roteadores de núcleo do domínio.

Com relação aos LSPs, a ferramenta de QoS foi capaz de mapear 3 diferentes caminhos entre o LER A e o LER B :

- A > B > C > D
- A > E > D
- A > B > E > D

Propositalmente, os roteadores da rede de testes foram interconectados de forma que houvesse um caminho mais curto que os demais (A > E > D). Esta assimetria permite

comparar melhor as características do roteamento IP tradicional com a capacidade do IP sobre MPLS em direcionar os fluxos levando em consideração outros fatores além do endereço de destino dos mesmos.

7.2.2.1 Roteamento IP tradicional X IP sobre MPLS com 2 fluxos TCP

Nesta seção será ilustrado um cenário em que o roteamento tradicional utilizando o protocolo IP não seria capaz de atender aos fluxos de dados de forma otimizada. A Figura 25 ilustra a mesma topologia de rede abordada na seção anterior. A rede 172.16.10.0 originará fluxos de dados destinados à rede 172.16.20.0. Supondo a utilização do protocolo IP e que a métrica de custo de rotas utilize um algoritmo do tipo vetor-distância, a rota escolhida seria: AED. Desta forma, foram adicionadas entradas de rotas manualmente em cada um dos roteadores A, E e D de forma a fazer com que o tráfego do protocolo IP seguisse pela parte inferior da rede.

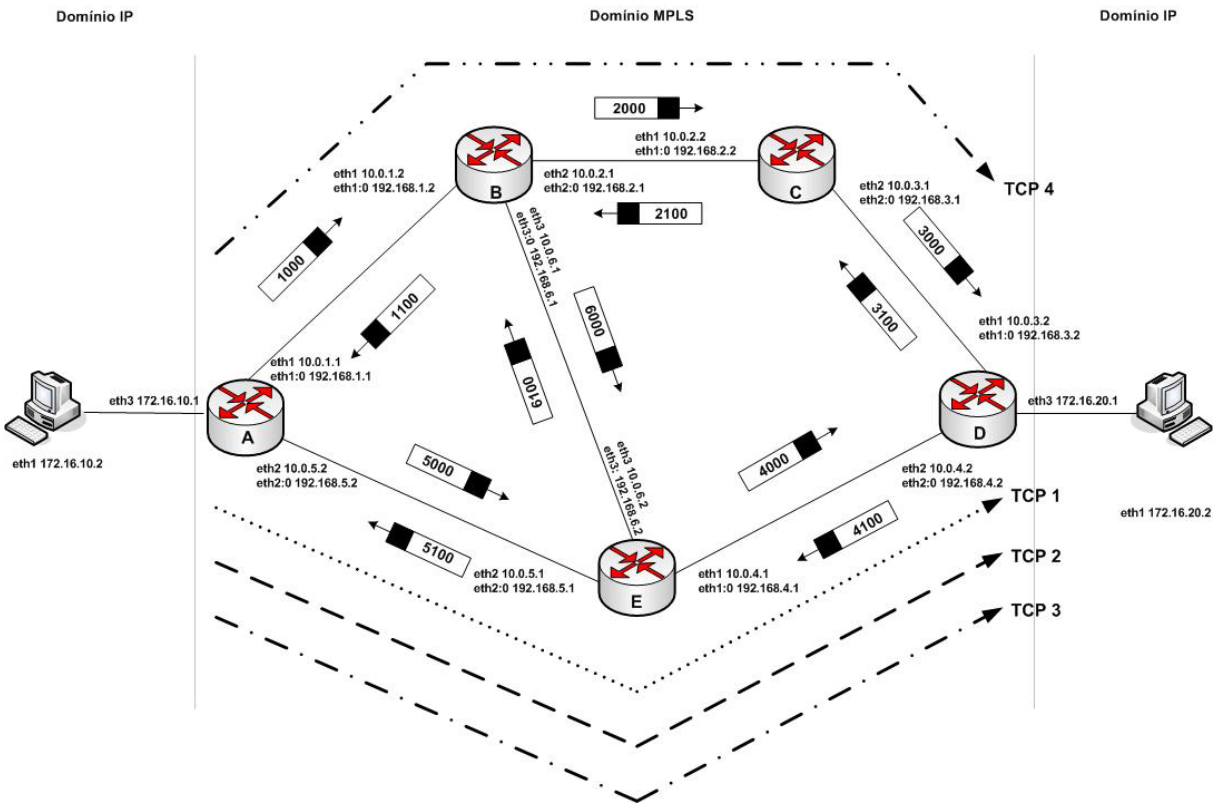

Figura 25: IP x MPLS com 2 fluxos TCP

Para comprovar os ganhos que podem ser obtidos por meio da utilização da ferramenta proposta foram realizados dois testes, sendo que em cada um deles foram gerados 2 fluxos de dados de cada vez.

Primeiramente foi utilizado o programa Iperf para gerar 2 fluxos de dados (TCP1 e TCP2) entre as redes 172.16.10.0 e 172.16.20.0 sem fazer uso do protocolo MPLS.

Num segundo momento, utilizou-se a ferramenta de QoS para criar 2 LSPs também entre as redes 172.16.10.0 e 172.16.20.0 . O LSP 1 foi formado pelos roteadores da parte superior da rede (ABCD) e recebeu o fluxo indicado pela sigla TCP 4. Já o LSP 2 foi composto pelos roteadores da parte inferior a rede (AED) e teve o fluxo TCP 3 atribuído a ele.

É importante relembrar que cada LSP é unidirecional. A partir de agora, ao mencionar a criação de um LSP, fica implícito que o LSP de retorno também foi configurado. Desta forma é possível que as figuras ilustrativas fiquem menos sobrecarregadas.

Em cada caso foram realizadas 5 medições consecutivas com o tempo de duração de 60s cada uma. A média aritmética simples destas medições foi considerada como resultado final e pode ser visto na Figura 26. O desvio padrão (σ) das medidas identificadas no gráfico da Figura 26 por TCP1, TCP2, TCP3 e TCP4 foram respectivamente: 0,33; 0,14; 0,07 e 0,31.

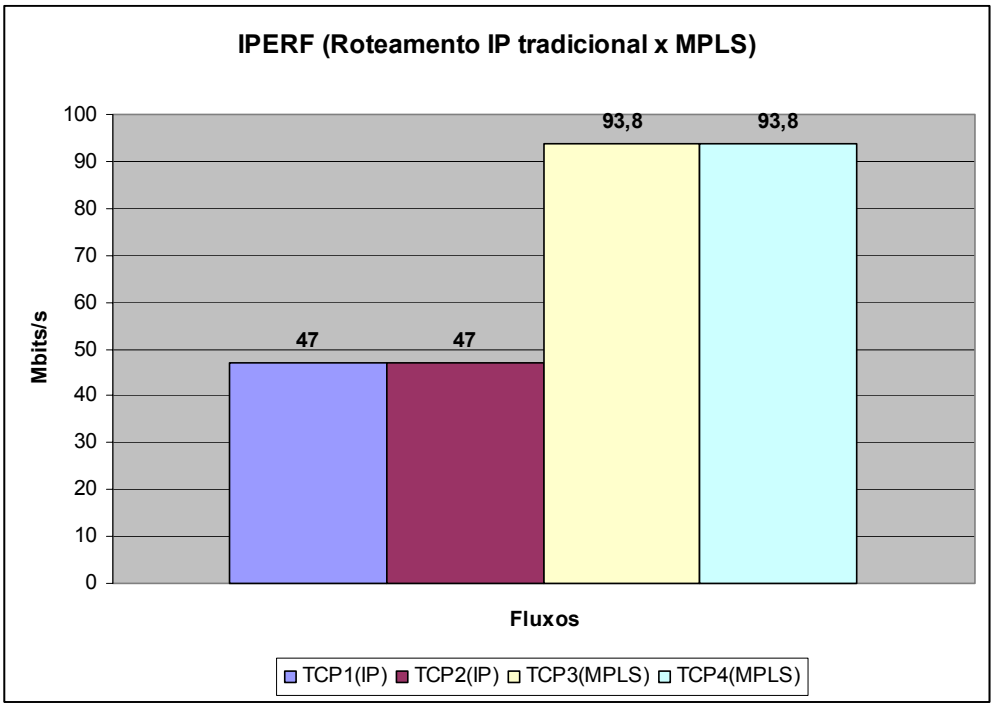

Figura 26: Comparação entre pares de fluxos utilizando protocolo IP ou MPLS

A análise do gráfico mostra que o roteamento realizado apenas com o protocolo IP não seria capaz de otimizar a utilização dos recursos de rede buscando atender às necessidades dos fluxos da melhor maneira possível. Devido ao congestionamento ocorrido no roteador E, o desempenho obtido na transmissão dos fluxos TCP 1 e TCP 2 foi bastante afetado. Ao mesmo tempo em que cada um deles era atendido com cerca de metade da capacidade nominal do canal, uma possível rota alternativa (ABCD) entre a origem e o destino da comunicação permaneceu ociosa.

A utilização da ferramenta proposta em conjunto com o protocolo MPLS permitiu que cada um dos fluxos TCP 3 e TCP 4 atingisse uma velocidade de transmissão bem próxima da capacidade nominal do canal de comunicação. Isto foi possível devido a sua capacidade de detectar o congestionamento ocorrido no enlace inferior da rede por meio dos dados coletados pelos sensores do NWS.

Por meio do tratamento das informações coletadas, a ferramenta atua no LER de entrada, executando um comando que insere uma regra em seu filtro de pacotes e direciona o segundo fluxo de dados ao enlace superior, que estava ocioso.

Idealmente, visando atender às necessidades de uma grade, seria interessante que seu escalonador de recursos fizesse uso das informações relativas aos estados dos enlaces antes de submeter tarefas a serem executadas nos nós. Infelizmente, foi gasto mais tempo que o previsto para se entender o funcionamento do *mpls-for-linux,* o que acabou por sacrificar o desenvolvimento da ferramenta proposta e obrigou a realização de testes antes que ela fosse finalizada.

7.2.2.2 Roteamento IP tradicional X IP sobre MPLS com 1 fluxo TCP e outro UDP

Para ilustrar outro caso no qual a capacidade de uma rede de comunicação pode ser melhor explorada como um todo com a inclusão de um mecanismo de controle de tráfego, foi realizado um novo teste, agora com a geração de tráfego UDP e TCP simultaneamente.

O protocolo TCP possui um mecanismo baseado em janelas que limita a taxa de transmissão em caso de congestionamento da rede. Seu algoritmo se auto-ajusta de acordo com as condições da rede, procurando sempre enviar a maior quantidade possível de dados sem que o buffer de recepção do destinatário seja saturado.

Já no caso do UDP, não há controle de congestionamento. Ele é um protocolo que permite que a informação seja transmitida de forma mais rápida, já que não estabelece sessão, não garante a entrega e nem manutenção da ordem de envio dos pacotes.

Na Figura 27, são mostrados os fluxos e suas respectivas rotas. Primeiramente foram estabelecidos os fluxos TCP 1 e UDP 1 sem que fosse utilizado o protocolo IP sobre MPLS. Num segundo momento, foram estabelecidos os fluxos TCP 2 e UDP 2, sobre os LSP 1 e LSP 2 respectivamente, agora utilizando o protocolo IP sobre MPLS.

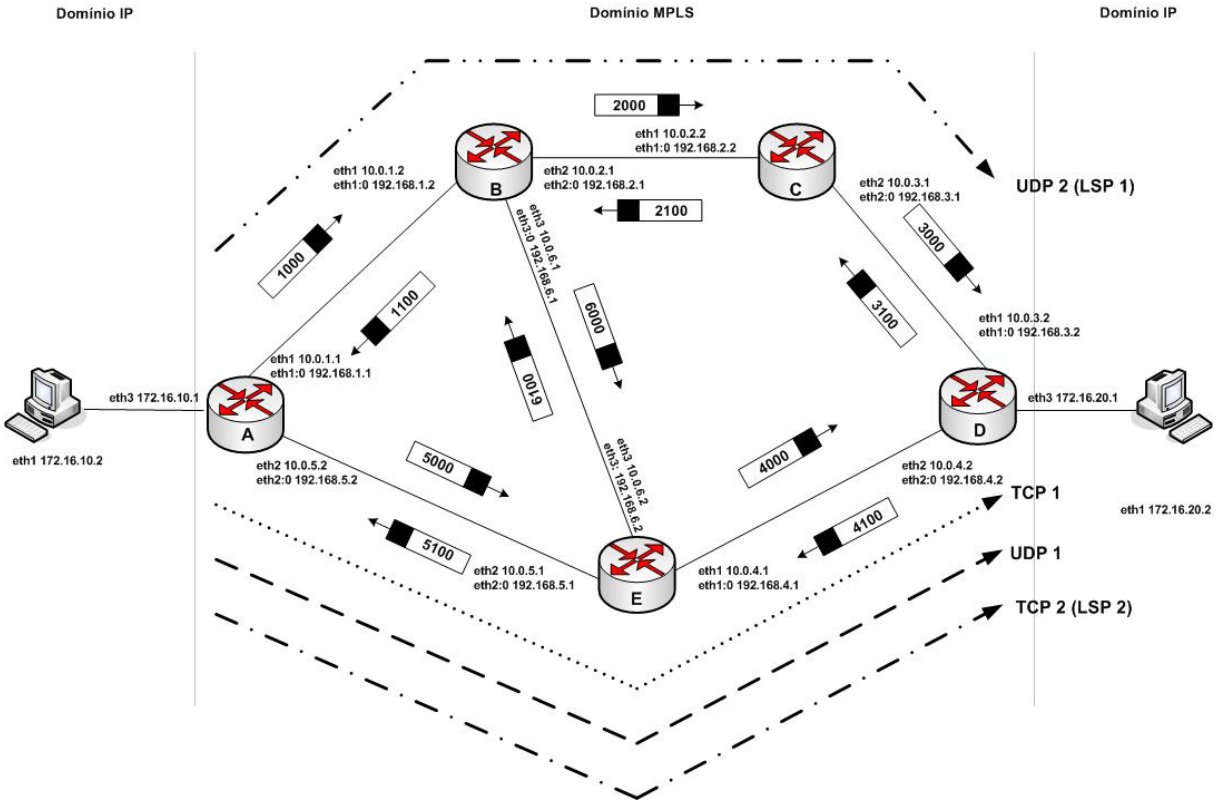

Figura 27: IP x MPLS com 1 fluxo TCP e outro UDP

Da mesma forma que na seção anterior, em cada caso foram realizadas 5 medições consecutivas com o tempo de duração de 60s cada uma. A média aritmética simples destas medições foi considerada como resultado final. A geração de tráfego UDP pelo Iperf foi limitada em 80 Mbits/s, sendo que no caso do TCP não foi imposto nenhum limite além da própria capacidade dos canais de comunicação. O desvio padrão (σ) das medidas identificadas no gráfico da FIGURA por TCP1, UDP1, TCP2 e UDP2 foram respectivamente: 0,25; 0,30; 0,23 e 0,22.

Como pode ser visto na Figura 28, o tráfego simultâneo de dois diferentes protocolos por um mesmo anal de comunicação pode ser muito mais danoso para um deles do que para o outro. Isto mostra que a análise do protocolo que está sendo utilizado em cada enlace de comunicação é importante e deve ser levada em consideração no momento de decisão sobre o caminho a ser utilizado no direcionamento dos fluxos.

No momento em que foi provocado um congestionamento no roteador E com os fluxos TCP 1 e UDP 1 sem a existência de um controle de tráfego, o fluxo que utiliza o

protocolo UDP tendeu a ocupar todo o canal de comunicação. Se não tivesse sido aplicado um limite de 80 Mbps, a parcela de banda alocada pelo protocolo TCP seria ínfima. Isto se deve justamente ao fato do TCP possuir mecanismos de controle de congestionamento.

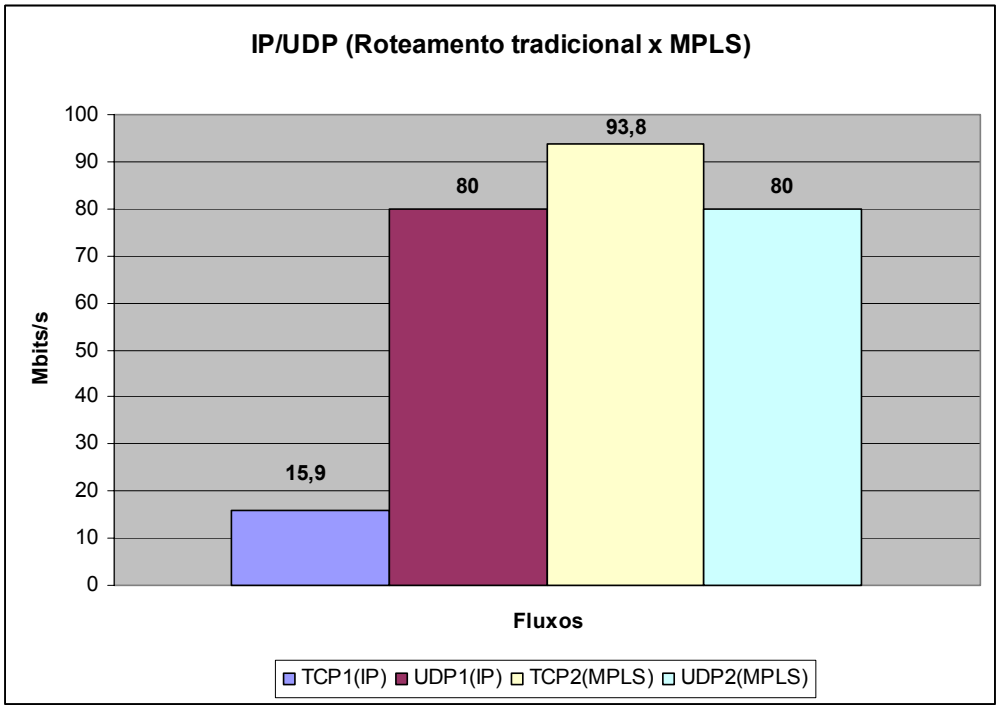

Figura 28: Comparação entre pares de fluxos TCP e UDP utilizando protocolo IP ou MPLS

Da mesma forma que na seção anterior, ao mesmo tempo em que problemas de congestionamento no roteador E prejudicavam o desempenho da rede, parte dela permaneceu ociosa no caso da utilização do roteamento IP tradicional. Assim, na segunda parte deste teste foram criados 2 LSPs. Através do uso da ferramenta proposta, o administrador da rede determinou que o fluxo UDP fosse direcionado para o LSP 1 (parte superior da rede) e o fluxo TCP fosse direcionado para o LSP 2 (parte inferior da rede).

Novamente, um direcionamento mais apropriado dos fluxos foi capaz de melhorar sensivelmente o desempenho da rede, sendo que o fluxo que utilizava protocolo TCP obteve uma taxa de transmissão 6 vezes maior do que quando atribuído a um LSP exclusivo. Esta diferença seria aumentada ainda mais caso a taxa de geração de tráfego UDP não tivesse sido limitada em 80 Mbps.

Embora neste caso a determinação do direcionamento do fluxo tenha sido tomada provisoriamente pelo administrador da rede, este experimento mostra que uma análise mais elaborada das características de cada fluxo pode ser determinante na obtenção de um melhor aproveitamento dos recursos de rede. Assim, espera-se que, quando concluída, a ferramenta de QoS proposta por este trabalho tenha mecanismos mais elaborados de decisão capazes de considerar o protocolo utilizado por cada fluxo, alocando-os de forma a evitar interferências entre os mesmos.

Ainda, deve-se salientar que, em um ambiente real, o número de fluxos, rótulos, fontes geradoras de tráfego e possibilidades de caminhos a serem tomados pode ser maior do que as grandezas envolvidas na topologia de rede considerada nesta seção. A princípio, o MPLS não seria um fator limitante, já que os 20bits destinados aos rótulos em seu cabeçalho seriam suficientes para criar 1.048.576 diferentes identificadores de fluxo. Assim, a maior complexidade ficaria a cargo da ferramenta, que teria que gerenciar um maior número de variáveis antes de determinar a melhor opção de caminho a ser seguido por cada fluxo.

8 Conclusões e trabalhos futuros

Embora um dos principais objetivos da proposta inicial de desenvolvimento do MPLS fosse o de diminuir o tempo de encaminhamento nos roteadores, não foi isto que ficou constatado durante os estudos que culminaram nesta dissertação de mestrado. A princípio, era esperado que o processo de encaminhamento de pacotes via troca de rótulos, por si só, já seria suficiente para obtenção de melhor desempenho frente ao encaminhamento IP tradicional. Isto porque os estudos iniciais levavam a crer que, caso os roteadores intermediários deixassem de participar do processo de definição do caminho a ser seguido por cada pacote, a economia de tempo seria significativa.

Contrariando os resultados esperados após um estudo mais aprofundado do protocolo MPLS, os testes de desempenho realizados mostraram que as redes baseadas no protocolo IP são ligeiramente mais rápidas do que aquelas baseadas na implementação Linux do protocolo MPLS.

Um dos fatores que levaram a esta constatação foi o fato de ter sido utilizada uma implementação experimental do protocolo MPLS para Linux. É de se esperar que o *mpls-for-linux* fique em desvantagem com relação ao tempo gasto no processamento dos pacotes quando comparado ao encaminhamento dado pelo protocolo IP, já que este último possui suporte nativo e seu código é mais otimizado devido ao maior grau de amadurecimento.

Além disso, o surgimento dos ASICs (*Application Specific Integrated Circuit*) fez com que o encaminhamento de pacotes pudesse ser realizado diretamente no *hardware.* Esta característica permite que roteadores de grande porte utilizados nos *backbones* de operadoras de telecomunicação possam encaminhar pacotes IP em *wire-speed*. Com a utilização desta tecnologia, a diferença existente entre se pesquisar um rótulo MPLS de 20 bits ou um cabeçalho IP de 32 bits passa a ser insignificante.

Desta forma, aplicações no campo da Engenharia de Tráfego, implementação de VPNs e integração com redes ópticas passaram a justificar a utilização do MPLS. Nesta dissertação

de mestrado foram exploradas as suas vantagens na realização de Engenharia de Tráfego com objetivo de obter Qualidade de Serviço em grades computacionais.

Conforme foi visto, para se conseguir uma alocação mais eficiente dos recursos de rede foi preciso verificar o estado dos mesmos constantemente, e, além disso, determinar o caminho a ser seguido por cada fluxo à partir de sua origem. Para este fim, o MPLS mostrou ser um protocolo apropriado, permitindo a criação e manutenção de canais por onde os fluxos são dirigidos.

Neste sentido, foi elaborada uma ferramenta de QoS capaz de monitorar o estado dos enlaces e determinar o caminho a ser tomado pelos fluxos, de forma a atendê-los da melhor maneira possível por meio da melhor utilização dos recursos disponibilizados pela rede. Embora sua aplicação tenha sido verificada em uma rede composta por um número relativamente pequeno de nós, acredita-se que seus benefícios também sejam sentidos em redes de grande porte utilizadas por grades computacionais.

A fim de aprimorar e estender as funcionalidades da ferramenta de QoS desenvolvida nessa dissertação de mestrado, podem ser levantadas as seguintes perspectivas de trabalhos futuros:

- Desenvolvimento de uma interface capaz de prover uma maior integração entre o escalalonador de processos de uma grade e a ferramenta de QoS.
- Aprimoramento do mecanismo de decisão que escolhe qual rota deve ser tomada por cada fluxo.
- Adição da funcionalidade de previsão de tráfego já existente no NWS, permitindo que sejam realizadas reservas prévias de recursos de rede.

Referências Bibliográficas

[ALL2005] ALLCOCK, B. et al. *The Globus Striped GridFTP Framework and Server*. 2005. Disponível em: <http://www.globus.org/alliance/publications/papers/gridftp-1.1.pdf>. Acesso em 27/06/2005.

[ALV2004] Alvarez, S. *QoS in MPLS Networks*. CISCO Networkers 2004. Disponível em: < http://www.cisco.com/networkers/nw04/presos/rst.html>

[And2001] ANDERSSON, L. et al. *RFC3036-LDP Specification*. Janeiro 2001. Disponível em: < http://www.faqs.org/rfcs/rfc3036.html>

[BBC1998] BLAKE, S.; BLACK, D.; CARLSON, M.; DAVIES, E.; WANG, Z.; WEISS, W. *RFC 2475-An Architecture for Differentiated Service*. 1998. Disponível em: <http://www.faqs.org/rfcs/rfc2475>

[BBL2002] BAKER, M.; BUYYA, R.; LAFORENZA, D. *Grids and Grid technologies for wide-area distributed computing. In software-practice and experience*. 2002.

[Beo2004] BEOWULF. *The Beowulf Cluster Site.* 2004. Disponível em: <http://www.beowulf.org/> . Acesso em: 30/05/2005.

[BER2001] Berger, L; et al. *RFC3209- RSVP-TE: Extensions to RSVP for LSP Tunnels*. 2001. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc3209.txt>

[Bra1997] BRADEN, R. et al. *RFC2205-Resource ReSerVation Protocol (RSVP)-Version 1 Functional Specification*. Setembro 1997. Disponível em: <http://www.rnp.br/ietf/rfc/rfc2205.txt>

[Buy2003] BUYYA, R. *The World-Wide Grid*. 2003. Disponível em: <http://www.buyya.com/ecogrid/wwg/>. Acesso em 25/05/2005.

[Car2002] CARDOSO, R. *A integração de múltiplos serviços com MPLS*. 2002. Disponível em: < http://www.ciscoredacaovirtual.com/redacao/perfistecnologicos/conectividad.asp?Id=18 >. Acesso em 20/05/05.

[CB2002] CHETTY, M.; BUYYA, R. *Weaving Computational Grids: How Analogous Are They with Electrical Grids?. In Computing in Science & Engineering (CiSE) Magazine*. 2002.

[CBB2006] Chan, K; Babiarz, J.; Baker, F. *Aggrregation of Diffserv Service Classes. Internet-IETF-Draft. 2006.* Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-diffserv-class-aggr-00.txt>

[CDM2001] Conta, A; Doolan, P.; Malis, A. *RFC3034-Use of Label Switching on Frame Relay Networks Specification*. Janeiro 2001. Disponível em: < http://www.faqs.org/rfcs/rfc3034.html>

[Chr2001] CHRISTENSEN, E.; et al. *Web Services Description Language (WSDL) 1.1*. Disponível em: <http://www.w3.org/TR/wsdl> . Acesso em: 27/05/2005.

[CIS2006] CISCO. *Quality of Service for Multi-Protocol Label Switching Networks*. Disponível em: < http://www.cisco.com/en/US/tech/tk828/technologies_q_and_a_item09186a00800a43f5.shtml >. Acesso em: 02/10/2006.

[CL2003] Casellas, R. Leu, J. *MPLS for Linux Developers' Guide.* 2003. Disponível em: <http://perso.enst.fr/~casellas/mpls-linux/index.html>

[Cor2004] CORBA. *Common Object Request Broker Architecture: Core Specification.* 2004. Disponível em: <http://www.omg.org/cgi-bin/apps/doc?formal/04-03-01.pdf>. Aceso em: 06/06/2005.

[CSL2004] CUCCHIARA, J.;SJOSTRAND, H.; LUCIANI, J.; *RFC3815-Definitions of Managed Objects for the Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP).* Junho 2004. Disponível em: < http://www.faqs.org/rfcs/rfc3815.html>

[Cza1998] CZAJKOWSKI, K; et al. *A Resource Management Architecture for Matacomputing Systems*. *In IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing. 1998.*.

[Cza2002] CZAJKOWSKI, K; et al. *Grid Information Services for Distributed Resource Sharing. In 10th IEEE Symposium On High Performance Distributed Computing*. 2001.

[DIG2003] DIGITAL HERMIT. *Kernel Rebuid Guide*. Disponível em: < http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html>

[DKV2001] Das, S. K.; Venkataram, P.; Biswas, J. *MPLS-BGP based LSP setup techniques*. *Proceedings of the 28th Annual IEEE International Conference on Local Computer Netwoks (LNC'03)*. 2003.

[Eto2002] E-TOILE. *E-Toile project homepage*. Disponível em: <http://www.urec.cnrs.fr/etoile/>. Acesso em: 21/06/05.

[Fau2005] Faucheur, F. Le. *RFC 4124 – Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering*. 2005. Disponível em: < http://www.rfc-editor.org/cgi-bin/rfcdoctype.pl?loc=RFC&letsgo=4124&file_format=pdf >

[FBD2004] FERREIRA, L.; BURNETT, C.; DIRKER, J.; et al. *The information grid, Part 1: The infrastructure.* 2004. Disponível em: < http://www-106.ibm.com/developerworks/library/gr-info1/> . Acesso em 01/06/2005.

[Fed2006] Fedora. *Fedora Project, sponsored by Red Hat*. 2006. Disponível em: < http://fedora.redhat.com/>. Acesso em 19/06/2006.

[Fer2003] FERREIRA, LUIS; et al. *Globus Toolkit 3.0 Quick Start. IBM Redpaper.* Setembro 2003. Disponível em: <http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf>

[FH1999] Ferguson, P.; Huston, G. Quality os service: Delivering QoS on the Internet and in Corporate Networks. Wiley Computer Publishing Press. 1999.

[FKL1999] FOSTER, I.; KESSELMAN, C.; LEE, C.; et al. *A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In Proceedings of the International Workshop on Quality of Service*. 1999.

[FKT2001] FOSTER, I.; KESSELMAN, C.; TUECKE, S. *The Anatomy of the Grid : Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications.* 2001.

[Fos2002] FOSTER, I. et al. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Junho 2002.

[Gar2000] GARA. *The GARA project homepage.* 2000. Disponível em: <http://www-fp.mcs.anl.gov/qos/>. Acesso em: 02/06/2005

[GGF2005] GLOBAL GRID FORUM. *The Global Grid Fórum homepage.* 2005. Disponível em: <http://www.ggf.org/> . Acesso em 26/05/2005.

[GL1995] GROPP, W.; LUSK, E. *Dynamic process management in an {MPI} setting. In Proceedings of Seventh IEEE Symposium on Parallel and Distributed Processing.* 1995.

[Gle2000] GLEESON, B; et al. RFC 2764-A FRAMEWORK FOR IP BASED VIRTUAL PRIVATE NETWORKS. 2000. Disponível em: < http://www.rfc-editor.org/rfc/rfc2764.txt>

[GT2005] GLOBUSTOOLKIT. *Project Globus Toolkit homepage*. 2005. Disponível em: <http://www-unix.globus.org/toolkit/>. Acesso em: 30/05/2005.

[HK1997] HORSTMANN, M.; KIRTLAND, M. *DCOM Architecture*. Julho 1997. Disponível em: <http://msdn.microsoft.com/library/en-us/dndcom/html/msdn_dcomarch.asp>. Acesso em: 06/06/2005.

[Hop2001] Hopkins, H. Harman. *Frame Relay + MPLS And How they fit together. White Paper. The Frame Relay Forum*. 2001. Disponível em: < http://www.mfaforum.org/frame/Whitepaper/whitepapers/MPLSwhitepaper .shtml>

[HS1995] HOWES, T. A.; SMITH, M. C. *A Scalable, Deployable Directory Service Framework for the Internet. CITI Technical Report 95-7*. 1995.

[HUB2002] HUBERT, B; et al. *Linux Advanced Routing & Traffic Control HOWTO*. 2002. Disponível em: <http://lartc.org/howto/>. Acesso em 27/06/06.

[IEC2005] IEC. International Engineering Consortium web site. 2005. Disponível em: < http://www.iec.org/online/tutorials/mpls/> . Acesso em: 10/04/2005.

[IPR2006] IPROUTE2. *Iproute2 – LinuxNet.* Disponível em: < http://linux-net.osdl.org/index.php/Iproute2>

[JGN1999] JOHNSTON, W.; GANNON, D.; NITZBERG, B. *Grids as production computing environments: The engineering aspects of NASA's information power grid. In Eighth IEEE International Symposium on High Performance Distributed Computing, Redondo Beach, CA.* Agosto 1999.

[Kar2004] CZAJKOWSKI, K; et al. *From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution.* 2004. Disponível em: <http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf>

[KER2006] KERNEL. *The Linux Kernel Archieves*. Disponível em: < http://www.kernel.org/>

[KR2003] KUROSE J.; ROSS K.; *Redes de Computadores e a Internet. Primeira edição, Editora Addison Wesley.* 2003.

[Kya2005] KYATERA. *KyaTera project homepage.* 2005. Disponível em: <http://www.kyatera.fapesp.br/portal> . Acesso em: 26/05/2005.

[LC2005] LEU, J. R.; CASELLAS, R. *Project MPLS for Linux homepage.* 2005. Disponível em: <http://sourceforge.net/projects/mpls-linux>. Acesso em 18/07/2005.

[Mys2006] MYSQL. Mysql homepage. Acesso em 29/10/06. Disponível em: < http://www.mysql.org/ >

[NBB1998] NICHOLS, K; BLAKE, S.; BAKER, F.; BLACK, D. *RFC 2474-Definition of the Diffetentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* 1998. Disponível em: <http://www.faqs.org/rfcs/rfc2474.html>

[Nei2004] NEISSE, R. et al. *On Translating Grid Requirements to Network Configurations through Policy-Based Management. In fifth IEEE/ACM International Workshop on Grid Computing.* 2004.

[NN2001] NORTEL NETWORKS. *MPLS-An Introduction to multiprotocol label switching. White Paper*. 2001. Disponível em: < http://www.ucer.nurcad.ufsc.br/docs/mplstutorialnortel.pdf>

[NWS2006] NETWORK WEATHER SERVICE. *Network Weather Service homepage*. Acesso em 29/10/06. Disponível em: <http://nws.cs.ucsb.edu/ewiki/>

[OS2002] Osborne, E.; Simha, A. *Engenharia de tráfego com MPLS.* Editora Campus. 2002.

[OS2003] OSBORNE, E.; SIMHA, A. *Engenharia de tráfego com MPLS*. Editora Campus. 2003.

[PH1998] Patterson, D. A.; Hennessy, J. L. Organização e projeto de computadores. 2ª edição. Editora LTC. 1998.

[PI1998] Peixinho, I. *Roteando pela Origem com o Linux*. RNP NewsGeneration, Volume 2, Número 7. Setembro de 1998.

[Pos1981] POSTEL, J. RFC 0792-Internet Protocol. 1981. Disponível em: < http://www.faqs.org/rfcs/rfc791.html >

[PVM2005] PVM. *Parallel Virtual Machine homepage.* 2005. Disponível em: <http://www.csm.ornl.gov/pvm/pvm_home.html>. Acesso em: 30/05/2005.

[Ros2001] Rosen, E; et al. *RFC 3032 - MPLS Label Stack Encoding*. 2001.

[RR2006] Rosen, E.; Rekhter, Y. RFC 4364 – BGP/MPLS IP Virtual Private Networks (VPNs). 2006. Disponível em: <http://www.rfc-editor.org/cgi-bin/rfcdoctype.pl?loc=RFC&letsgo=4364&file_format=pdf>

[RSL2005] *The Globus Resource Specification Language RSL v1.0.* 2005. Disponível em: <http://www-fp.globus.org/gram/rsl%5Fspec1.html>. Acesso em: 16/06/2005

[RVC2001] ROSEN E.; VISWANATHAN A.; CALLON R.; *RFC3031-Multiprotocol Label Switching Architecture.* Janeiro 2001. Disponível em: <http://www.faqs.org/rfcs/rfc3031.html>

[SET2006] SETI@home. SETI@home homepage. 2006. Disponível em: < http://setiathome.berkeley.edu/ > Acesso em 18/10/2006.

[SG2005] Silva, G. *Guia Foca GNU/Linux.* Capítulo 10 – Firewall iptables. Julho de 2005. Disponível em: < http://focalinux.cipsga.org.br/guia/avancado/ch-fw-iptables.htm>

[Slo1994] SLOMAN, M. *Policy driven management for distributed systems. In Journal of Network and Systems Management*. 1994.

[Sot2003] SOTOMAYOR, B.; *The Globus Toolkit 3 Programmer's Tutorial.* 2003. Disponível em: <http://gdp.globus.org/gt3-tutorial/download/progtutorial-pdf_0.4.3.tar.gz>. Acesso em 03/03/2005.

[SZ2005] Suehring, S.; Ziegler, R. *Linux Firewalls. 3rd Edition. Novell Press.* 2005.

[TCF2002] TUECKE, S.; CZAJKOWSKI, K.; FOSTER, I. *Grid Service Specification.* 2002.

[Ter2005] TERAGRID *TeraGrid Project homepage.* 2005. Disponível em: <http://www.teragrid.org/>. Acesso em 24/05/2005.

[Tid2005] TIDIA. *Tidia project homepage*. 2005. Disponível em: < http://www.tidia.fapesp.br/portal> . Acesso em: 26/05/2005.

[TMK2006] Travostino, F.; Mambretti, J.; Karmous-Edwards, G. Grid Networks: Enabling grids with advanced communication technology. John Wiley & Sons Press Ltd. 2006.

[TS2002] Tanenbaum, A. S.; Steen, M. V. *Distributed Systems: Principles and Paradigms. Prentice Hall*. 2002.

[VPC2004] VICAT, P.; PRIMET, B.; CHANUSSOT, F. *End to End Network Quality of Service in Grid environments: the QoSINUS approach. In First Annual International Conference on Broadband Networks*. 2004.

[WE2005] CIRNE, W.; NETO, E. *Grids Computacionais: da Computação de Alto Desempenho a Serviços sob Demanda. Minicurso apresentado no SBRC2005*. 2005.

[Wel2003] WELCH, V. et al. *Security for Grid Services. In 12th International Symposium on High Performance Distributed Coputing (HPDC-12)*. 2003.

[WSH1999] WOLSKI, R.; SPRING, N. T.; HAYES, J. *The Network Weather Service: A Distributed Resource Performance Forecasting Service For Metacomputing. In Journal of Future Generation Computing Systems, Volume 15*. Outubro, 1999.

[Wsi2005] WSI. *WS-I Organization's web site.* 2005. Disponível em: < http://www.ws-i.org >. Acesso em 27/06/2005.

[Zor2005] ZORZATTO, J. A. *Um Sistema WEB para Monitoramento e Controle da Grade ProGrid. Dissertação defendida no programa de mestrado da Universidade Federal de São Carlos*. 2005.

Apêndices

A. Instalação do *mpls-for-linux*

A.1. Via pacotes RPM

Uma das formas de instalação do *mpls-for-linux* consiste na obtenção dos pacotes RPM em [LC2005]. Considerando a última versão disponível, foi feito o *download* dos seguintes arquivos:

- kernel-2.6.15-1.1831_FC4mpls_1.950.i686.rpm
- iptables-ipv6-1.3.0-2_mpls_1.950d.i386.rpm
- iproute-2.6.11-1_mpls_1.950d.i386.rpm

O primeiro pacote é destinado à quarta versão da distribuição Fedora Core. Ele realiza as modificações necessárias no *kernel* do sistema, permitindo que ele passe a comutar pacotes rotulados via protocolo MPLS.

O segundo pacote adiciona à distribuição uma versão modificada do filtro de pacotes iptables, incluído no *kernel* do Linux a partir da versão 2.4. O iptables realiza a filtragem e a manipulação dos pacotes com base nas informações contidas em seus cabeçalhos, como por exemplo: endereços e portas de origem/destino, protocolo, dentre outros. Com a modificação realizada no *kernel* do Linux, o iptables passa a ser capaz de filtrar pacotes também com base em seus rótulos.

O terceiro pacote, iproute-2.6.11-1_mpls_1.950d.i386.rpm, adiciona à distribuição uma versão modificada do comando ip, adaptada ao tratamento de pacotes MPLS. A ferramenta ip possui todas as funcionalidades dos comandos arp, route e ifconfig.

De posse dos arquivos acima, basta realizar a instalação dos mesmos, conforme pode ser visto a seguir:

```
root@jupiter:~# rpm –ivh kernel-2.6.15-1.1831_FC4mpls_1.950.i686.rpm
root@jupiter:~# rpm –ivh iptables-ipv6-1.3.0-2_mpls_1.950d.i386.rpm
root@jupiter:~# rpm –ivh iproute-2.6.11-1_mpls_1.950d.i386.rpm
```

Depois de instalados os pacotes, o sistema está preparado para ser configurado como um roteador MPLS.

Embora a instalação via pacotes RPM seja muito fácil de ser realizada, ela é pouco flexível. Um dos motivos é que nem todas as distribuições Linux possuem suporte nativo a pacotes RPM. Além disso, por ser independente da distribuição, a instalação do *mpls-for-linux* a partir dos seus arquivos fonte permite a remoção do *debug* no seu código, otimizando seu desempenho.

A.2. Via recompilação do *kernel*

Outra forma de se adaptar o *kernel* do Linux ao protocolo MPLS é através da sua recompilação. Para isso, é necessário realizar algumas alterações no seu código fonte antes de recompilá-lo. Estas alterações são adicionadas aplicando-se um *patch* ao código fonte que adiciona o suporte desejado.

Primeiramente é preciso fazer o *download* do código fonte do *kernel* em [KER2006]. Como o *patch* do *mpls-for-linux* é disponibilizado somente para algumas versões específicas do *kernel,* é importante verificar em [LC2005] qual será a versão mais apropriada a ser utilizada.

Neste trabalho foi utilizada a versão 1.950 do *mpls-for-linux* em conjunto com a versão 2.6.15 do *kernel*.

Além do código fonte do *kernel*, é também necessário fazer o *download* do arquivo que contém as alterações a serem realizadas no *kernel* para que ele se torne compatível com o MPLS. Em [LC2005] deve ser obtido o seguinte arquivo:

- mpls-linux-1.950.tar.bz2

Para descompactar o arquivo do *kernel,* deve-se executar o seguinte comando dentro do diretório /usr/src:

root@jupiter:/usr/src# tar xvfz linux-2.6.15.1.tar.bz2

Agora, deve ser criado o *link* simbólico /usr/src/linux para /usr/src/linux-2.6.15.1 :

root@jupiter:/usr/src# ln –s /usr/src/linux-2.6.15.1 linux

Neste ponto, já é possível aplicar o *patch* do *mpls-for-linux* no *kernel*. Para isso, deve-se decompactar o arquivo mpls-linux-1.950.tar.bz2. Neste exemplo, optou-se por descompactá-lo no diretório /usr/src:

root@jupiter:/usr/src#tar xvfj mpls-linux-1.950.tar.bz2

Para proceder com a instalação do *patch,* dentro do diretório /usr/src/linux, executa-se o seguinte comando:

root@jupiter:/usr/src/linux# patch –p1 < ../mpls-linux/patches/linux-kernel.diff

Caso o comando seja bem sucedido, deverá ser exibida na tela uma saída semelhante à que é mostrada na caixa de texto a seguir:

```
patching file include/linux/if_arp.h
patching file include/linux/ipv6_route.h
patching file include/linux/mpls.h
patching file include/linux/netdevice.h
patching file include/linux/netfilter_ipv4/ipt_spec_nh.h
patching file include/linux/netfilter_ipv6/ip6t_spec_nh.h
patching file include/linux/ppp_defs.h
patching file include/linux/rtnetlink.h
patching file include/linux/socket.h
patching file include/net/ip_fib.h
patching file include/net/mpls.h
patching file include/net/spec_nh.h
patching file net/core/dev.c
patching file net/core/Makefile
patching file net/core/spec_nh.c
patching file net/ipv4/fib_semantics.c
patching file net/ipv4/ip_output.c
patching file net/ipv4/Kconfig
patching file net/ipv4/Makefile
patching file net/ipv4/mpls4.c
```

O próximo passo consiste na seleção das funcionalidades que serão adicionadas ao *kernel*. Serão evidenciadas apenas aquelas que estão ligadas ao MPLS. Dentro do diretório /usr/src/linux, deve-se executar o seguinte comando:

root@jupiter:/usr/src/linux/# make menuconfig

Será aberta uma interface na qual deverão ser feitas as seguintes seleções:

```
Code maturity level options  --->
      [*] Prompt for development and/or incomplete code/drivers
      Networking options  --->
        [*] Multi Protocol Label Switching - MPLS
```

Além das anteriores, para os casos em que for necessário fazer uso de mecanismos de filtragem de pacotes via iptables, também deverão ser selecionados os seguintes itens:

```
[*] Network packet filtering (replaces ipchains)
        IP: Netfilter Configuration  --->

      <*> IP tables support (required for filtering/masq/NAT)
         <*> spec_nh target support
```

Feitas as modificações acima, basta sair da interface de configuração e seguir com os comandos de compilação do *kernel* normalmente:

root@jupiter:/usr/src/# make

root@jupiter:/usr/src/# make modules_install

Por se tratar de um procedimento relativamente complexo, antes de se executar o processo de compilação do kernel é desejável que se leia documentos específicos sobre o tema, como [DIG2003].

Uma explicação mais detalhada sobre o processo de compilação do kernel do Linux foge ao escopo deste trabalho.

A.3. Compilação de programas adicionais

Como foi visto na seção que abordou o processo de instalação do *mpls-for-linux* via pacotes rpm, versões modificadas de outros dois aplicativos também foram adicionadas ao sistema. São eles: iproute e iptables.

Para permitir que o processo de instalação aqui descrito seja totalmente independente da distribuição Linux que se deseja utilizar, serão mostrados a seguir os passos necessários para se compilar tanto o iproute quanto o iptables.

A.3.1. Compilação do iproute2

O iproute2 é um aplicativo que fornece o comando ip, utilizado para adição, remoção e alteração da tabela de rotas. Outro utilitário disponibilizado pelo software é o "mpls" que, por sua vez, permite que sejam manipuladas as tabelas NHLFE e ILM.

O código fonte do *software* iproute pode ser obtido em [IPR2006]. Sua compilação e instalação irão disponibilizar o comando "ip", utilizado na adição, modificação e remoção de rotas no sistema.

Primeiramente, deve-se descompactar o arquivo:

root@jupiter:/usr/local/src/# tar xvzf iproute2-2.6.11-050310.tar.gz

A seguir, de forma similar ao que foi feito com o kernel, deve-se aplicar o *patch*:

root@jupiter:/usr/local/src/iproute# patch –p1 < /usr/src/mpls-linux/patches/iproute2.diff

Para compilá-lo, devem ser executados os seguintes comandos:

root@jupiter:/usr/local/src/iproute# ./configure

root@jupiter:/usr/local/src/iproute# make

root@jupiter:/usr/local/src/iproute# make install

A.3.2. Compilação do iptables

Para que seja possível mapear os fluxos de dados da rede nos LSPs é preciso adicionar um *patch* ao iptables. O código fonte do mesmo pode ser obtido em [http://ftp.netfilter.org/pub/iptables/]. Neste trabalho foi utilizada a versão 1.3.0 do mesmo.

Depois de obtido o arquivo, ele deve ser descompactado:

root@jupiter:/usr/local/src/# tar xvjf iptables-1.3.0.tar.bz2

Agora, é preciso aplicar o *patch* que adicionará o suporte ao protocolo MPLS ao iptables:

root@jupiter:/usr/local/src/iptables# patch –p1 < /usr/src/mpls-linux/patches/iptables.diff

Da mesma forma que na seção anterior, os comandos utilizados para compilá-lo são os seguintes:

root@jupiter:/usr/local/src/iptables# ./configure

root@jupiter:/usr/local/src/iptables# make

root@jupiter:/usr/local/src/iptables# make install

A principal diferença entre a versão originada e a modificada do iptables é que esta última possui uma nova ação que pode ser especificada após o parâmetro “-j” na linha de comando. Esta nova funcionalidade pode permitir por exemplo que todo o tráfego gerado em

determinada máquina seja direcionado para um LSP identificado pelo número 0x3 na tabela NHLFE, conforme pode ser visto abaixo:

```
# iptables –A OUTPUT –d 10.0.1.0/24 –j spec_nh –spec_nh 0x8847:0x3
```

Regras desse tipo são essenciais para aplicações destinadas à Engenharia de tráfego e foram utilizadas nos testes realizados neste trabalho.

B Filtrando pacotes com iptables

A partir da versão 2.4.x, o *kernel* do Linux passou a contar com um novo subsistema responsável pela filtragem de pacotes: o netfilter/iptables. O netfilter é um *framework* do *kernel* do Linux e o iptables corresponde a uma ferramenta que faz uso deste *framework,* permitindo que o usuário edite tabelas que serão utilizadas na filtragem de pacotes.

Ao contrário dos seus antecessores (ipfwadm e ipchains), o iptables mantém o estado das conexões que passam por ele, ou seja, ele é capaz de se lembrar se uma conexão de entrada é ou não proveniente de uma requisição interna. Embora a filtragem de pacotes sem armazenamento do estado das conexões consiga obter um melhor desempenho em determinadas situações, a elaboração de regras para redes complexas pode se tornar muito trabalhosa.

B.1. Funcionamento

O mecanismo de funcionamento de um filtro de pacotes se baseia na análise do cabeçalho dos mesmos. Os dados contidos no cabeçalho são comparados com algumas regras pré-estabelecidas que culminarão no aceite, descarte ou encaminhamento para uma outra *chain.*

Cada *chain* pode ser vista como um local onde conjuntos de regras de *firewall* são armazenadas. Já as tabelas são utilizadas para armazenar as *chains* e demais regras que possuírem características comuns [SZ2005].

B.2. Tabelas

O iptables disponibiliza três tabelas, que serão vistas a seguir.

B.2.1. A tabela *filter*

Esta tabela é utilizada caso não seja especificada nenhuma outra. Por padrão, ela contém três cadeias de regras (*chains*):

- INPUT: Consultada para pacotes que chegam ao roteador.
- OUTPUT: Consultada para pacotes que saem do roteador.
- FORWARD: Consultada para pacotes que são redirecionados de uma interface de rede para outra no roteador.

A Figura 29 ilustra o diagrama de fluxo de pacotes da tabela filter. Como pode ser visto, a filtragem dos pacotes é realizada pelo próprio *kernel* do Linux. No momento em que um pacote chega até a interface de entrada do roteador, uma decisão de roteamento é tomada.

Caso o destino do pacote seja o próprio roteador, ele será encaminhado para a *chain* INPUT. Após passar pelas regras da *chain* INPUT, se não houver nenhuma regra de descarte, o pacote será aceito e irá originar um processo local.

Os pacotes destinados a outra interface de rede ou a outra máquina serão encaminhados para a *chain* FORWARD. Para que seja possível rotear o pacote entre duas interfaces de rede é preciso que a função de roteamento seja habilitada no *kernel*. Uma forma de se fazer isso via linha de comando é dada a seguir:

```
# echo “1” >/proc/sys/net/ipv4/ip_forward
```

Assim, com o suporte a roteamento habilitado, o pacote será verificado pelas regras da *chain* FORWARD. Finalmente, se na houver nenhuma regra de descarte, ele será encaminhado para a interface da sua rede de destino.

A terceira e última *chain* que pode ser vista na Figura 29 é a OUTPUT. Sua função é filtrar pacotes originados na própria máquina antes que eles sejam encaminhados para a devida interface de saída.

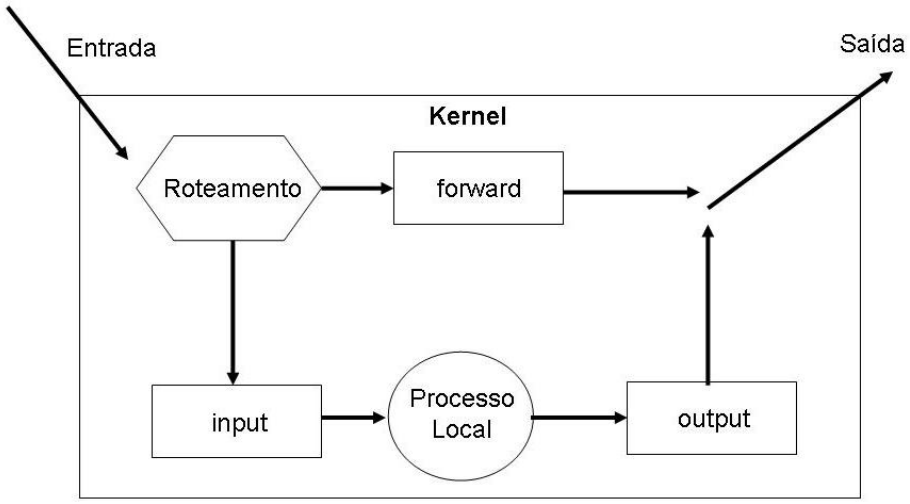

Figura 29: Componentes da tabela *filter*

B.2.2. A tabela *mangle*

A tabela *mangle* é utilizada na especificação de ações especiais no tratamento do tráfego, sendo particularmente útil tanto em funções de manipulação e priorização de tráfego na rede.

Uma de suas principais funções é a marcação de pacotes, facilitando a classificação dos mesmos para que posteriormente seja tomada uma decisão de roteamento ou aplicada determinada política de fila com base nesta marcação.

Outro exemplo de sua utilização pode ser visto no caso das redes de serviços diferenciados onde a tabela *mangle* pode ser utilizada na alteração do campo DSCP dos pacotes, associando-os a outra classe de serviço.

Por padrão, a tabela *mangle* possui cinco chains [SG2005]:

- PREROUTING: Consultada quando os pacotes precisam ser modificador antes de serem enviados para a *chain* INPUT da tabela *filter*.
- POSTROUTING: Consultada quando os pacotes precisam ser modificador antes de serem enviados para a *chain* POSTROUTING da tabela *nat*.
- INPUT: Consultada quando os pacotes precisam ser modificador antes de serem enviados para a *chain* INPUT da tabela *filter*.
- OUTPUT: Consultada quando os pacotes precisam ser modificador antes de serem enviados para a *chain* OUTPUT da tabela *nat*.
- FORWARD: Consultada quando os pacotes precisam ser modificador antes de serem enviados para a *chain* FORWARD da tabela *filter*.

B.2.3. A tabela nat *(Network Address Translation)*

A tabela nat é utilizada para os casos em que é necessário alterar o endereço IP original dos pacotes. Sua utilização mais comum se dá na criação de mecanismos de tradução de endereços IP que permitem que máquinas de uma rede interna possam acessar a Internet. Um outro exemplo de sua utilidade está no redirecionamento de portas, que pode ser explorado na configuração de um *proxy* transparente.

Suas *chains* são [SG2005]:

- PREROUTING: É consultada antes da decisão de roteamento, sendo utilizada para DNAT (*Destination Network Address Translation*) e redirecionamento de portas.
- OUTPUT: É consultada nos casos em que pacotes gerados localmente precisam ser modificados antes de serem roteados.

- POSTROUTING: É consultada após a decisão de roteamento, sendo utilizada na realização de SNAT (*Source Network Address Translation*) e mascaramento de IP.

B.3. Diagrama de funcionamento do *netfilter*

Após a introdução sobre as principais características e aplicações das três tabelas que formam o iptables, é interessante observar o diagrama completo de funcionamento do mesmo. Na Figura 30 são ilustrados os caminhos que um pacote pode tomar durante seu recebimento, processamento e encaminhamento final.

Quando um pacote chega ao *netfilter* através de uma interface de rede, primeiramente ele passa pela *chain* PREROUTING das tabelas mangle e nat onde ele pode ser tratado de acordo com o que foi explicado na seção anterior.

Após deixar a *chain* PREROUTING ocorre uma decisão de roteamento. Caso o destino do pacote seja a própria máquina, ele seguirá pelo caminho da esquerda onde será tratado pela *chain* INPUT das tabelas filter e mangle antes de ser recebido por alguma aplicação local que esteja escutando determinada porta. A saída da aplicação segue para a *chain* OUTPUT das tabelas mangle, nat e filter.

Já os pacotes que não forem destinados à própria máquina seguem o caminho da direita, estando sujeitos às regras da *chain* FORWARD das tabelas filter e mangle. Antes de serem encaminhados para a interface de rede eles ainda passam pela *chain* POSTROUTING das tabelas nat e mangle.

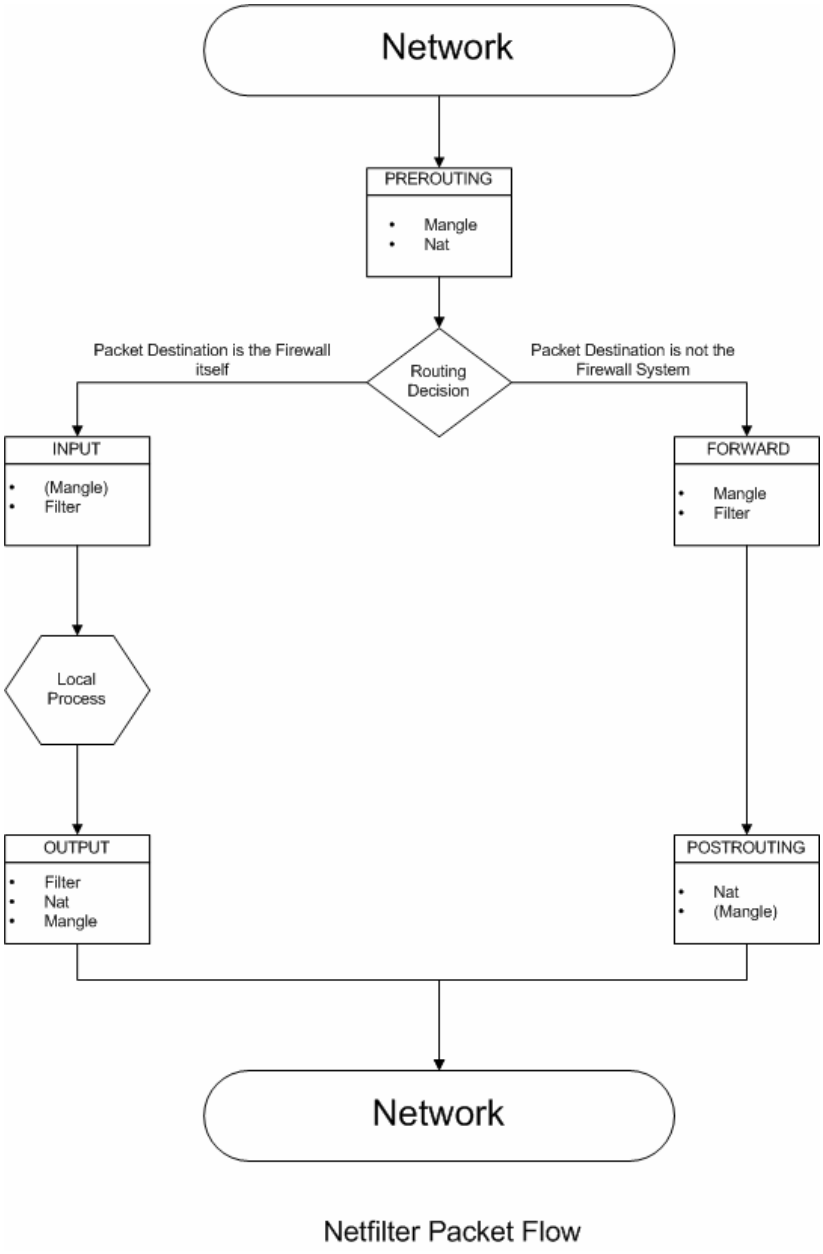

Figura 30: Diagrama de funcionamento do *netfilter* [SZ2005]

B.4. Inserindo regras no iptables

Nesta seção serão mostrados os principais parâmetros que podem ser utilizados para compor as regras no iptables. É importante salientar que o objetivo desta seção é apenas o de

fornecer ao leitor uma introdução sobre este utilitário. Informações mais detalhadas podem ser encontradas em [SZ2005].

A sintaxe do comando iptables é:

iptables [-t tabela] [opções] [cadeia] [parâmetros] –j [ação]

Onde:

- tabela: especifica uma das três possíveis opções de tabela (*filter,nat* ou *mangle*), sendo que se nada for especificado será utilizada a tabela padrão: *filter*.
- opções: especifica a operação a ser realizada na cadeia de regras, sendo as mais usuais:
    - Adicionar uma regra ao final da cadeia de regras: -A.
    - Apagar uma regra da cadeia de regras: -D.
    - Inserir uma regra em determinada posição da cadeia de regras: -I.
    - Listar as regras da cadeia de regras: -L.
- cadeia (*chain*): especifica em que cadeia de regras a regra irá atuar, podendo ser: INPUT, OUTPUT, FORWARD, POSTROUTING e PREROUTING.
- parâmetros: permite identificar em que pacote a regra irá atuar através da especificação de dados contidos em seu cabeçalho, como por exemplo:
    - protocolo: -p <protocolo>
    - endereço de origem: -s <endereço>
    - endereço de destino: -d <endereço>
    - valor do campo dscp, em hexadecimal ou décima: --dscp <valor>
    - valor do campo tipo de serviço (TOS – *type of service*): --tos <valor>
    - procura por "pacotes marcados" que devem ser manipulados de forma diferenciada: --mark <valor>
- ação: especifica o que deverá ser feito com o pacote que possuir parâmetros iguais aos procurados, sendo as mais usuais:

- aceita o pacote: ACCEPT
- descarta o pacote: DROP
- gera log para cada pacote que passar pela regra: LOG

C Princípios de roteamento

Para que seja estabelecida a comunicação entre diferentes redes de computadores é preciso que entre elas exista um equipamento denominado roteador. Um roteador pode ser constituído tanto por um computador comum com duas ou mais interfaces de rede quanto por um equipamento destinado especificamente para a tarefa de rotear pacotes.

Cada roteador define o caminho a ser seguido pelos pacotes que passam através de suas interfaces de rede executando um algoritmo de busca em uma tabela de rotas. A Tabela 7 ilustra um exemplo real de uma tabela de rotas:

Tabela 7: Exemplo de uma tabela de rotas

Destino	Máscara	Gateway	Interface
10.0.2.0	255.255.255.0	10.0.2.1	eth0
10.0.1.0	255.255.255.0	10.0.1.1	eth1
default	0.0.0.0	10.0.0.1	eth2

Como pode ser visto na Tabela 7, o roteador utilizado para ilustrar o exemplo possui três interfaces de rede, ou seja, é capaz de rotear pacotes entre três diferentes redes: 10.0.2.0, 10.0.1.0 e 10.0.0.0. A última linha da tabela é composta pelo endereço "*default"*. Caso o endereço de destino do pacote seja diferente de todas as entradas anteriores, o mesmo será encaminhado pela rota "*default*".

Cada endereço IP é formado pelo endereço da rede a qual o equipamento pertence e o endereço do próprio equipamento dentro dessa rede. A máscara define a divisão entre o endereço de rede e o endereço do equipamento [Pos1981].

No momento em que um pacote chega ao roteador, é realizada uma operação "AND" entre seu endereço de destino e a máscara, resultando assim no endereço de rede ao qual ele

pertence. A seguir é feita uma comparação entre o endereço de rede resultante da operação anterior e a tabela de rotas. Por fim, é feito o envio do pacote pela devida interface de rede.

C.1. Roteando pela origem

Embora a maioria dos roteadores esteja preparada para realizar as operações de roteamento baseadas no endereço de destino dos pacotes, este procedimento pode não ser muito apropriado para algumas aplicações. No caso da Engenharia de Tráfego, a inclusão de uma coluna adicional na tabela de rotas pode ser muito útil e foi abordada em [PI1998].

A Tabela 8 ilustra uma tabela de roteamento com uma coluna denominada Origem. Esta coluna possibilita um número maior de possibilidades para decisão de roteamento, podendo ajudar a resolver problemas de congestionamento de enlaces [PI1998].

Tabela 8: Exemplo de tabela de roteamento com base no endereço de origem

Origem	Destino	Máscara	Gateway	Interface
0.0.0.0	10.0.2.0	255.255.255.0	10.0.2.1	eth0
0.0.0.0	10.0.1.0	255.255.255.0	10.0.1.1	eth1
0.0.0.0	default	0.0.0.0	10.0.0.1	eth2

O roteamento com base na origem é muito similar ao roteamento explícito realizado pelos LERs de entrada de um domínio MPLS. Neste último, a adição de rótulos nos pacotes é realizada a partir da análise de dados dos seus cabeçalhos: endereço/porta de origem/destino, protocolo, etc. Assim, o caminho a ser seguido por cada pacote é definido na entrada do domínio e não depende da decisão individual de cada LSR que o compõe.

No caso do MPLS, o roteamento explícito é primordial tanto para as aplicações de Engenharia de tráfego quanto para o provisionamento de QoS.

D Detalhes de configuração da rede de testes

Nesta seção serão mostrados em detalhes os procedimentos de configuração de uma topologia de testes. Espera-se que as informações aqui contidas possam sirvam de base para a configuração de outras topologias de rede.

Conforme pode ser visto na Figura 31, o roteador de entrada do domínio, Júpiter, é responsável pela inserção dos rótulos MPLS nos pacotes e pela seleção do caminho a ser utilizado. Já o LER de saída, Netuno, realiza a retirada dos rótulos dos pacotes MPLS que passam a seguir seu caminho na rede local (192.168.1.0) utilizando o protocolo IP.

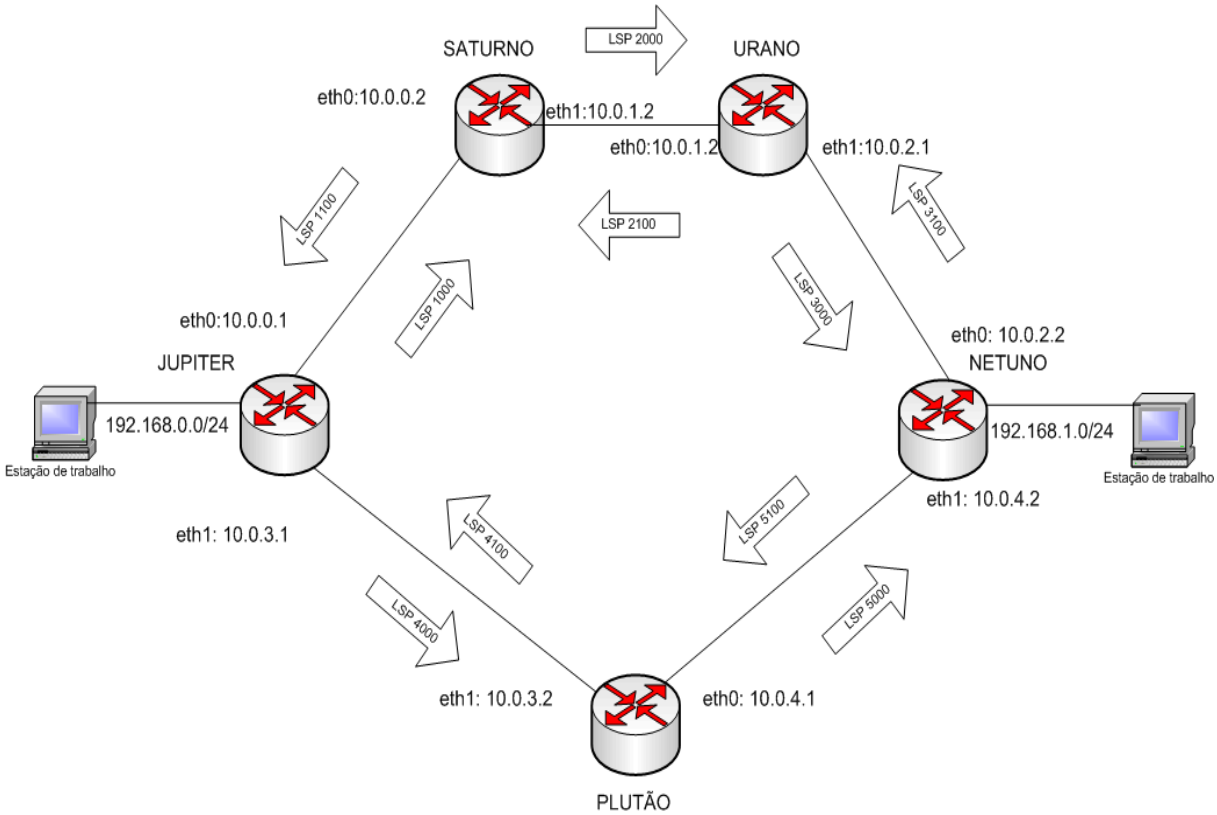

Figura 31: Topologia de testes

A topologia de testes utilizada é particularmente interessante devido ao fato de possuir dois caminhos diferentes entre a entrada e a saída do domínio MPLS, sendo que um deles é composto por dois seguimentos de rede e o outro por três.

Esta configuração é um exemplo clássico de engenharia de tráfego, sendo encontrada em diversos livros que tratam deste assunto, como é o caso de [OS2003].

Considerando que todos os enlaces possuem o mesmo custo, é possível estabelecer comparativos entre o uso do protocolo MPLS e o roteamento feito pelo protocolo IP.

D.1. Configuração da topologia de testes

Para que os LSPs da topologia de testes sejam configurados, é necessária a inserção de diversos comandos no terminal de cada um deles. Nesta seção serão mostrados os passos necessários para o estabelecimento de tais LSPs. A configuração do primeiro roteador trará maiores detalhes do que está sendo realizado. Nos demais roteadores, serão explicados apenas os comandos que ainda não tenham sido explicitados em passos anteriores.

É importante lembrar que os LSPs são unidirecionais, ou seja, devem ser configurados LSPs tanto no sentido de ida quanto de volta do caminho a ser estabelecido.

D.1.1. Configuração do LER de entrada do domínio MPLS (Júpiter)

O comando abaixo cria um número de chave que será utilizado na identificação da tabela NHLFE. A criação das chaves ocorre de forma seqüencial, sendo que o primeiro valor retornado é 0x00000002.

root@jupiter:/root/mpls# mpls nhlfe add key 0

O próximo comando utiliza a chave criada no anterior para adicionar rótulos de valor 1000 aos pacotes que deixarem o LER júpiter pela interface de rede superior (eth0).

root@jupiter:/root/mpls# mpls nhlfe change key 0x2 instructions push gen 1000 nexthop eth0 ipv4 10.0.0.2

Neste momento, é possível verificar o conteúdo da tabela NHLHE, como pode ser visto a seguir:

```
[root@jupiter mpls]# mpls nhlfe show
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 1000 set eth0 ipv4 10.0.0.2  (0 bytes, 0 pkts)
```

É interessante observar o valor do MTU (*Maximum Transmit Unit*), ou seja, Unidade Máxima de Transmissão. O MTU se refere à máxima quantidade de dados que um quadro da camada de enlace é capaz de carregar. Como nem todos os protocolos da camada de enlace podem carregar datagramas do mesmo tamanho, no caso do IP, por exemplo, pode ser necessário fragmentar os dados dos datagramas em pedaços menores [KR2003]. Para isso, pode ser necessário configurar a interface de rede para operar com valores menores de MTU.

Em um roteador IP que interliga diversos enlaces, é possível que cada um deles utilize um protocolo diferente da camada dois. Assim, caso o enlace de saída possua MTU menor que o enlace de entrada, o roteador deve proceder com a fragmentação dos datagramas. Cabe à camada de rede do endereço de destino remontar estes fragmentos antes de entregar os dados para as camadas superiores da pilha de protocolos.

Com relação à saída da tabela NHLFE mostrada nos parágrafos anteriores, tem-se o valor de 1496 para MTU. Os experimentos realizados neste trabalho se deram em redes padrão *ethernet*, capazes de carregar datagramas de 1500 *bytes.* Este valor pode ser visto diretamente nos dados de configuração da própria interface de rede, como mostrado a seguir:

```
[root@jupiter mpls]# ifconfig

eth2      Link encap:Ethernet  HWaddr 00:0C:F1:6E:E5:62
          inet addr:200.18.98.133  Bcast:200.18.98.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:f1ff:fe6e:e562/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:26084 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8167 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2663650 (2.5 MiB)  TX bytes:529300 (516.8 KiB)
          Base address:0xac00 Memory:ff3e0000-ff400000
```

Desta forma, existe uma diferença de quatro bytes entre o valor do MTU reportado pela interface de rede (1500) e o valor reportado pela tabela NHLFE (1496). Estes quatro

bytes são relativos ao cabeçalho MPLS, encapsulado entre a camada de enlace e a camada de rede.

O próximo passo consiste no redirecionamento do tráfego IP que será roteado pelo LER para o LSP de saída, identificado pela chave 0x2 na tabela NHLFE.

root@jupiter:/root/mpls# ip route add 10.0.2.0/24 via 10.0.0.2 mpls 0x2

A conferência da tabela de rotas é vista a seguir:

```
[root@jupiter mpls]# ip route show
10.0.2.0/24 via 10.0.0.2 dev eth0 mpls 0x2
default via 200.18.98.1 dev eth2
```

Desta forma, ao ingressarem no LSP, os pacotes IP receberão o rótulo 1000 identificado pela chave 0x2 da tabela NHLFE. Resumidamente, os três comandos vistos acima definiram um mapeamento do tipo FEC-to-NHLFE.

Neste momento é preciso configurar o caminho de volta dos pacotes dos pacotes destinados ao LER Júpiter, provenientes da parte superior da rede. Isto é feito através da configuração do seu LSP de entrada na interface de rede eth0:

root@jupiter:/root/mpls# mpls labelspace add dev eth0 labelspace 0

root@jupiter:/root/mpls# mpls ilm add label gen 1100 labelspace 0

Para conferir o conteúdo da tabela ILM, basta executar o comando a seguir:

```
[root@jupiter mpls]# mpls ilm show
ILM entry label gen 1100 labelspace 0 proto ipv4
        pop peek  (0 bytes, 0 pkts)
```

Aqui termina a configuração do LSP superior para o LER Jupiter. De forma similar, os próximos comandos irão tratar da configuração do LER Júpiter para que ele atenda a parte inferior da topologia de testes:

root@Jupiter:/root/mpls# mpls nhlfe add key 0

root@Jupiter:/root/mpls# mpls nhlfe change key 0x3 instructions push gen 4000 nexthop eth1 ipv4 10.0.3.2

root@Jupiter:/root/mpls# ip route add 10.0.4.0/24 via 10.0.3.2 mpls 0x3

root@Jupiter:/root/mpls# mpls labelspace add dev eth1 labelspace 1

root@Jupiter:/root/mpls# mpls ilm add label gen 4100 labelspace 1

D.1.2. Configuração do LSR Saturno

A configuração deste LSR é bem parecida com a configuração do LER vista na seção anterior. Assim, será abordada sucintamente.

Abaixo será dada a configuração do LSP de ida no sentido de Júpiter para Netuno:

root@Saturno:/root/mpls# mpls nhlfe add key 0

root@Saturno:/root/mpls# mpls nhlfe change key 0x2 instructions push gen 2000 nexthop eth1 ipv4 10.0.1.2

root@Saturno:/root/mpls# ip route add 10.0.2.0/24 via 10.0.1.2 mpls 0x2

root@Saturno:/root/mpls# mpls labelspace add dev eth1 labelspace 0

root@Saturno:/root/mpls# mpls ilm add label gen 2100 labelspace 0

Agora será dada a configuração do LSP de volta no sentido de Netuno para Jupiter:

root@Saturno:/root/mpls# mpls nhlfe add key 0

root@Saturno:/root/mpls# mpls nhlfe change key 0x3 instructions push gen 1100 nexthop eth1 ipv4 10.0.0.1

root@Saturno:/root/mpls# ip route add 10.0.0.0/24 via 10.0.0.1 mpls 0x3

root@Saturno:/root/mpls# mpls labelspace add dev eth1 labelspace 1

root@Saturno:/root/mpls# mpls ilm add label gen 1000 labelspace 1

Os dois próximos comandos são responsáveis pela interligação dos LSPs criando túneis que comutam os pacotes com base apenas no conteúdo de seus rótulos. Este processo é denominado *cross-connecting*.

root@Saturno:/root/mpls# mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key 0x2

root@Saturno:/root/mpls# mpls xc add ilm_label gen 2100 ilm_labelspace 0 nhlfe_key 0x3

Para conferir o se os comandos relativos ao *cross-connect* foram executados corretamente, basta executar o comando a seguir:

```
[root@jupiter mpls]# mpls xc show
XC entry ilm_label gen 1000 ilm_labelspace 0 nhlfe_key 0x00000002

XC entry ilm_label gen 2100 ilm_labelspace 0 nhlfe_key 0x00000003
```

D.1.3. Configuração do LSR Urano

Segue abaixo a configuração do LSP de ida no sentido de Júpiter para Netuno:

root@Urano:/root/mpls# mpls nhlfe add key 0

root@Urano:/root/mpls# mpls nhlfe change key 0x2 instructions push gen 3000 nexthop eth1 ipv4 10.0.2.2

root@Urano:/root/mpls# ip route add 10.0.2.0/24 via 10.0.2.2 mpls 0x2

root@Urano:/root/mpls# mpls labelspace add dev eth1 labelspace 0

root@Urano:/root/mpls# mpls ilm add label gen 3100 labelspace 0

Segue abaixo a configuração do caminho de volta, ou seja, sentido Netuno para Júpiter:

root@Urano:/root/mpls# mpls nhlfe add key 0

root@Urano:/root/mpls# mpls nhlfe change key 0x3 instructions push gen 2100 nexthop eth0 ipv4 10.0.1.2

root@Urano:/root/mpls# ip route add 10.0.0.0/24 via 10.0.1.2 mpls 0x3

root@Urano:/root/mpls# mpls labelspace add dev eth0 labelspace 0

root@Urano:/root/mpls# mpls ilm add label gen 2000 labelspace 0

root@Urano:/root/mpls# mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key 0x2

root@Urano:/root/mpls# mpls xc add ilm_label gen 3100 ilm_labelspace 0 nhlfe_key 0x3

D.1.4. Configuração do LER Netuno

A seguir o LER Netuno será configurado para tratar dos pacotes relativos à porção superior da topologia de testes:

root@Netuno:/root/mpls# mpls nhlfe add key 0

root@Netuno:/root/mpls# mpls nhlfe change key 0x2 instructions push gen 3100 nexthop eth0 ipv4 10.0.2.1

root@Netuno:/root/mpls# ip route add 10.0.0.0/24 via 10.0.2.1 mpls 0x2

root@Netuno:/root/mpls# mpls labelspace add dev eth0 labelspace 0

root@Netuno:/root/mpls# mpls ilm add label gen 3000 labelspace 0

Os próximos comandos configuram o LER Netuno para que este possa tratar dos pacotes relativos à porção inferior do domínio MPLS.

root@Netuno:/root/mpls# mpls nhlfe add key 0

root@Netuno:/root/mpls# mpls nhlfe change key 0x3 instructions push gen 5100 nexthop eth1 ipv4 10.0.4.1

root@Netuno:/root/mpls# ip route add 10.0.3.0/24 via 10.0.4.1 mpls 0x3

root@Netuno:/root/mpls# mpls labelspace add dev eth1 labelspace 0

root@Netuno:/root/mpls# mpls ilm add label gen 5000 labelspace 0

D.1.5. Configuração do LSR Plutão

Abaixo será dada a configuração do LSP de ida no sentido de Júpiter para Netuno:

root@Plutao:/root/mpls# mpls nhlfe add key 0

root@Plutao:/root/mpls# mpls nhlfe change key 0x2 instructions push gen 5000 nexthop eth0 ipv4 10.0.4.2

root@Plutao:/root/mpls# ip route add 10.0.4.0/24 via 10.0.4.2 mpls 0x2

root@Plutao:/root/mpls# mpls labelspace add dev eth0 labelspace 0

root@Plutao:/root/mpls# mpls ilm add label gen 5100 labelspace 0

Agora será dada a configuração do LSP de volta no sentido de Netuno para Jupiter:

root@Plutao:/root/mpls# mpls nhlfe add key 0

root@Plutao:/root/mpls# mpls nhlfe change key 0x3 instructions push gen 4100 nexthop eth1 ipv4 10.0.3.1

root@Plutao:/root/mpls# ip route add 10.0.3.0/24 via 10.0.3.1 mpls 0x3

root@Plutao:/root/mpls# mpls labelspace add dev eth1 labelspace 0

root@Plutao:/root/mpls# mpls ilm add label gen 4000 labelspace 0

root@Plutao:/root/mpls# mpls xc add ilm_label gen 4000 ilm_labelspace 0 nhlfe_key 0x2

root@Plutao:/root/mpls# mpls xc add ilm_label gen 5100 ilm_labelspace 0 nhlfe_key 0x3