PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CEATEC

ALBERTO LOTITO

ENGENHARIA DE TRÁFEGO ENTRE DOMÍNIOS DE REDES DISTINTAS

CAMPINAS

2007

ALBERTO LOTITO

ENGENHARIA DE TRÁFEGO ENTRE DOMÍNIOS DE REDES DISTINTAS

Dissertação apresentada como exigência para obtenção do Título de Mestre em Engenharia Elétrica, ao Programa de Pós-Graduação na área de concentração Gestão de Redes de Telecomunicações, Pontifícia Universidade Católica de Campinas.

Orientador: Prof. Dr. Marcelo Abbade

PUC-CAMPINAS

2007

Ficha Catalográfica
Elaborada pelo Sistema de Bibliotecas e
Informação - SBI - PUC-Campinas

t004.62 Lotito, Alberto
L883e Engenharia de tráfego entre domínios de redes distintas / Alberto Lotito - Campinas: PUC-Campinas, 2007.
119p.

Orientador: Marcelo Luís Francisco Abbade.
Dissertação (mestrado) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pós-Graduação em Engenharia Elétrica.
Inclui anexos e bibliografia.

1. Redes de computação - Protocolos. 2. TCP/IP (Protocolo de rede de computação). 3. Engenharia de tráfego. 4. Redes de informação. I. Abbade, Marcelo Luís Francisco. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologias. Pós-Graduação em Engenharia Elétrica. III. Título.

20.ed.CDD – t004.62

ALBERTO LOTITO

ENGENHARIA DE TRÁFEGO ENTRE DOMÍNIOS DE REDES DISTINTAS

Dissertação apresentada ao Curso de Mestrado Profissional em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre em Gestão de Redes de Telecomunicações
Área de Concentração: Gestão de Redes e Serviços .
Orientador: Prof. Dr. Marcelo Luís Francisco Abbade

Dissertação defendida e aprovada em 05 de dezembro de 2007 pela Comissão Examinadora constituída dos seguintes professores:

Prof. Dr. Marcelo Luís Francisco Abbade
Orientador da Dissertação e Presidente da Comissão Examinadora
Pontifícia Universidade Católica de Campinas

Prof. Dr. Omar Carvalho Branquinho
Pontifícia Universidade Católica de Campinas

Prof. Dr. Maurício Ferreira Magalhães
Universidade Estadual de Campinas

À minha família,
mãe, pai (em memória),
esposa, irmãos, sobrinhos, afilhados,
cunhados, sogro e sogra,
pois todos de alguma forma ajudaram.

AGRADECIMENTOS

Ao Prof. Dr. Marcelo Luís Francisco Abbade
Orientador e Incentivador de meus trabalhos que teve muita paciência e me orientou nos momentos mais difíceis desse trabalho.

Ao Prof. Dr. Omar Branquinho
Guia e mestre sempre com uma palavra positiva a me incentivar durante o curso.

Ao amigo Fernando Lino
Que me incentivou e ingressou junto comigo no curso de mestrado.

Ao Prof. e amigo Ranieri Marinho de Souza
Que me auxiliou na formatação e editoração desta dissertação.

Aos Profs. Indayara Bertoldi Martins e Luiz Henrique Bonani do Nascimento
Que me auxiliaram com o NS-2 e os Objetos do TCL.

À Lucent Technologies
Que permitiu o uso de seus laboratórios no início deste trabalho.

“Nem tudo que se enfrenta pode ser modificado.
Mas nada pode ser modificado até que seja enfrentado.”

Albert Einstein
(1879-1955)

RESUMO

LOTITO, Alberto. Engenharia de tráfego entre domínios de redes distintas. Dissertação (Mestrado em Gestão de Redes de Telecomunicações) – Pós-Graduação em Engenharia Elétrica, Centro de Ciências Exatas, Ambientais e de Tecnologias. Pontifícia Universidade Católica de Campinas. Campinas, 2007.

Este trabalho tem como objetivos realizar simulações de engenharia de tráfego fim a fim em redes que trabalham com protocolos distintos e avaliar parâmetros usados. Propusemos e testamos parâmetros que servem para que o operador da rede dimensione a atuação de um algoritmo para engenharia de tráfego. Demos especial atenção às redes que trabalham com IP interconectando-se a redes MPLS.

Executando diversas simulações, trabalhamos questões de melhoria de desempenho por meio de avaliação da perda de pacotes nas redes testadas comparando-se os resultados com as tradicionais redes com protocolo OSPF e chegando a melhoria de até 50% se comparado com este.

Este trabalho contribui com melhorias em engenharia de tráfego em redes do mundo real, desenvolvendo aplicação prática desde as tradicionais redes IP até as atuais redes MPLS e também em redes de tecnologias mistas.

Termos de Indexação: MPLS, ATM, TCP-IP, Dijkstra, engenharia de tráfego

ABSTRACT

LOTITO, Alberto. Engenharia de tráfego entre domínios de redes distintas. Dissertação (Mestrado em Gestão de Redes de Telecomunicações) – Pós-Graduação em Engenharia Elétrica, Centro de Ciências Exatas, Ambientais e de Tecnologias. Pontifícia Universidade Católica de Campinas. Campinas, 2007.

This work intends to perform end-to-end traffic engineering through simulations in broadband multiprotocol networks and evaluate used parameters. We've proposed and tested parameters used by the network administrator to dimension the action of an algorithm for Traffic Engineering. We dedicated special attention to IP networks interconnected to MPLS networks.

Through simulations, we've addressed questions of performance optimization, evaluating the network packet losses and compare the results with packet loss in the traditional OSPF network achieving improvements of up to 50%.

This work contributes with improvements in traffic performance for real world networks, developing practical application since the traditional IP networks up to the present MPLS networks and also for mixed technologies.

Index Terms: MPLS, ATM, TCP-IP, Dijkstra, Traffic Engineering

LISTA DE FIGURAS

Lista de Quadros

LISTA DE ABREVIATURAS E SIGLAS

AAL = ATM Adaptation Layer
AS = Autonomous System
ATM = Asynchronous Transfer Mode
BGP = Border Gateway Protocol
CAC = Call Admission Control
CBR = Constant Bit Rate
DLCI = Data Link Channel Identifier
EGP = Exterior Gateway Protocol
FEC = Forwarding Equivalence Class
FIB = Forward Information Base
GCAC = Generic Call Admission Control
IGP = Interior Gateway Protocol
IP = Internet Protocol
IPTV = IP TeleVision
IS-IS = Intermediate Systems to Intermediate Systems
ITU = International Telecommunications Union
ITU-T = ITU-Telecommunications
LAN = Local Area Network
LDP = Label Distribution Protocol
LIB = Label Information Base
LSP = Label Switch Path
LSR = Label Switch Router
MPOA = Multiprotocol Over ATM
MPLS = Multiprotocol Label Switch
MPLS-TE = MPLS – Traffic Engineering
MIB = Management Information Base
NMS = Network Management Station
NS-2 = Network Simulator 2
OSPF = Open Shortest Path First
PNNI = Private Network to Network Interface
PTSE = PNNI Topology State Element

PVC = Private Virtual Circuit
RSVP = Resource reSerVation Protocol
RSVP-TE = RSVP-Traffic Engineering
SLA = Service Level Agreement
SVCC = Soft Permanent Virtual Channel
SPVC = Soft Permanent Virtual Connection
SNMP = Simple Network Management Protocol
TCP-IP = Transmission Control Protocol- Internet Protocol
TTL = Time To Live
UDP = User Datagram Protocol
VCC = Virtual Channel Connection
VCI = Virtual Channel Identifier
VPC = Virtual Path Connection
VPI = Virtual Path Identifier
WAN = Wide Area Network

SUMÁRIO

1 INTRODUÇÃO

O cenário atual das redes de dados mostra um crescimento e popularização da Internet nunca antes imaginado, chegando a muitos milhões de computadores pessoais e empresariais e continua crescendo dia após dia em todo o mundo. Aplicações que antes usavam redes de tecnologia própria como meio de transporte hoje usam a Internet ou outras redes IP proprietárias. Como exemplos, podemos citar o transporte de telefonia sobre IP, de televisão sobre IP (IPTV), vídeo sob demanda, transporte de dados em geral e muitas outras aplicações. Tudo isso tem feito a demanda por redes, serviços e largura de banda crescer exponencialmente. O tráfego da Internet utiliza, basicamente, a pilha de protocolos da arquitetura TCP/IP e, dessa forma, isso tem feito difundir cada vez mais a plataforma de redes TCP/IP instalada.

Sob o ponto de vista de telecomunicações, o problema é interligar as redes remotamente localizadas. Dessa forma, uma das tarefas das empresas de telecomunicações é transportar a informação usando diferentes tecnologias através de suas redes. Para cumprir esse objetivo, surgiram nos últimos anos diversas tecnologias de comutação visando transportar diferentes tipos de tecnologias e em especial, transportar TCP/IP. Uma das primeiras tecnologias *Wide Area Network* (WAN) capaz de transportar diferentes tipos de aplicações foi o *Frame-Relay*, que trabalha atribuindo *Data Link Channel Identifiers* (DLCI) (McDYSAN, 2000, p165-174) às aplicações a serem transportadas, e é capaz de transportar voz, imagens e dados, inclusive TCP/IP. Em seguida veio a tecnologia *Asynchronous Transfer Mode* (ATM) (KYAS; CRAWFORD, 2002, p.33-48) criada pelo ITU-T (*International Telecommunications Union – Telecommunications*) para ser uma rede universal, capaz de transmitir Voz, Dados e Imagens em redes locais ou remotas. Um dos principais pontos da tecnologia ATM é dividir a informação a ser transmitida em pequenas células de tamanho fixo, de 53 bytes, o que permite controlar os fluxos de dados e criar

diferentes qualidades de serviço. A tecnologia ATM atribui canais às células para identificar seus fluxos de dados, isso é feito através dos parâmetros Virtual Path Identifier (VPI) e Virtual Channel Identifier (VCI). Apesar de todas as funcionalidades, de ser capaz de transportar diferentes tecnologias e de fornecer mecanismos excelentes de qualidade de serviço, a tecnologia ATM nunca conseguiu a projeção pretendida. Especialmente devido aos altos custos de seus equipamentos se tornou muito caro instalar e manter uma rede ATM, especialmente em redes locais (*Local Area Network* - LAN), e seu uso se tornou restrito à redes de operadoras de telecomunicações, servindo de transporte para outras tecnologias. Especialmente em redes LAN o TCP/IP se tornou muito mais popular que o ATM. O custo reduzido de equipamentos como roteadores, *hubs*, *switches* e especialmente das placas de rede para computadores PC foi um fator decisivo para que as redes ethernet com TCP/IP prevalecessem sobre as redes ATM nas redes locais.

Outra tecnologia muito usada e que vem se tornando o backbone das redes metropolitanas é a *Multi Protocol Label Switching* (MPLS) com a proposta de tornar mais rápida a comutação dos pacotes nos roteadores da rede. Os roteadores de borda da rede MPLS inserem rótulos aos pacotes IP a fim de identificar os fluxos de dados e em cada roteador encaminhar os pacotes IP baseado nesse rótulo e não no endereço IP, tornando o processo de encaminhamento de pacotes mais rápido, já que faz o encaminhamento como se fosse um *switch* e não um roteador (BLACK, 2002, p.8-11).

O aparecimento de uma nova tecnologia não faz necessariamente com que a tecnologia anterior venha a desaparecer. Muitas dessas tecnologias funcionam simultaneamente em ambientes de operadoras de telecomunicações, sendo necessário manter as redes legadas para atender a base instalada. É o caso das redes Frame-Relay inter-operando com as redes ATM, assim como as redes ATM operando em conjunto com as redes MPLS. É necessário manter as redes antigas e instalar as novas redes mantendo interoperabilidade entre elas.

A tarefa de engenharia de tráfego nessas redes mistas é muito complexa por envolver muitas tecnologias e detalhes da interoperabilidade entre elas.

Em um esquema de tradução de protocolos tradicional, de ATM para MPLS e vice-versa, um ponto fixo de tradução, *switch* ou roteador, é escolhido e configurado sobre algum nó da rede para uma determinada conexão. Problema semelhante ocorre quando uma rede MPLS convive com uma rede IP, onde, usando-se um protocolo *Interior Gateway Protocol* (IGP) como o *Open Shortest Path First* (OSPF), a rede toma as decisões de roteamento e os fluxos de dados seguem o caminho determinado pelo IGP.

Outro ponto que não é tradicionalmente tratado pelos esquemas de roteamento é o fato de que os recursos de rede mudam dinamicamente. Parâmetros como ocupação dos enlaces, perda de pacotes, utilização do *backplane* e carga da CPU não são levados em conta pela maioria dos protocolos de roteamento. Quando o custo de um enlace é computado para criar uma nova conexão, pode ocorrer que os custos não reflitam os recursos sendo usados através do caminho escolhido. Quanto ao uso do *backplane*, se um *switch* ou roteador estiver trabalhando sem nenhum tipo de sobrecarga, ou estiver sob condições severas de tráfego, a conexão não será desviada para outro ponto de tradução, ficando à mercê das condições de tráfego do *switch* para o qual ela foi criada. Da mesma forma, novas conexões que venham a ser criadas estarão sujeitas a passar por pontos da rede que estejam congestionados.

As empresas de telecomunicações têm que conviver com esse cenário no qual os operadores das redes têm que criar e manter conexões de dados entre essas diversas tecnologias. Isso cria dificuldades para o operador, já que, tem que decidir o ponto de ligação entre uma tecnologia e outra, para cada nova conexão. Muitas vezes acabam fazendo o trabalho de engenharia de tráfego apenas instintivamente, distribuindo as conexões entre os roteadores da rede

sem muito critério. Isso é um problema para as operadoras, pois acabam utilizando de forma inadequada os recursos da rede, podendo sobrecarregar ou subutilizar nós e enlaces da rede.

1.1 Contextualização do problema

Alguns trabalhos foram escritos com o propósito de encontrar a melhor rota em redes de dados e a melhor estratégia para aplicá-la. Conte (CONTE, 2000, p.47-68) propôs uma estratégia na qual aplicava Markov ao estado dos enlaces em redes ATM com circuitos permanentes (*Private Virtual Circuit* - PVC) com o objetivo de reduzir o número de conexões bloqueadas. Buriol (BURIOL, 2003) apresentou um trabalho onde pesos são aplicados aos arcos em redes OSPF. Resende *et al* (RESENDE *et al*, 2002) propôs um método de redistribuir tráfego em redes MPLS. Seu método trabalhava dinamicamente criando conexões virtuais e cuidando da ocupação do enlace, a qual mudava o custo do mesmo. Verdi *et al* (VERDI *et al*,2007) propôs um trabalho para prover roteamento com QoS entre domínios usando virtualização. Seu trabalho abstrai detalhes de cada Sistema Autônomo da rede integrado com o BGP.

Em redes que trabalham com mais de um protocolo, como é o caso das redes MPLS conectadas a redes ATM, ou ATM com TCP-IP e também MPLS com TCP-IP, têm-se protocolos bem definidos para engenharia de tráfego dentro do domínio de cada protocolo isoladamente. Por exemplo, em redes MPLS têm-se o *Resource reSerVation Protocol* (RSVP) enquanto que nas redes ATM têm-se o *Private Network-Network Interface* (PNNI). Ambos podem criar dinamicamente conexões de um ponto a outro dentro de seu domínio e restrito a este. O RSVP pode reservar recursos e escolher o melhor caminho, usando RSVP – Traffic Engineering (RSVP-TE), dentro do domínio MPLS. Mas quando reunimos redes de protocolos distintos, a engenharia de tráfego não é realizada fim a fim, ficando restrita ao domínio de cada protocolo. O PNNI pode criar uma conexão dentro do domínio ATM, iniciando em um *switch* ATM e terminando em

outro. Mas quando há interoperabilidade entre ATM e MPLS, a interoperabilidade, no que se refere a engenharia de tráfego, é restrita a mecanismos simples como o *crankback* (MFA-FORUM-18.0.0, 2007) .

1.2 Objetivos e proposta do trabalho

Este trabalho tem por objetivo apresentar e avaliar uma nova técnica de engenharia de tráfego fim a fim em redes de protocolos distintos. Avaliamos o desempenho da técnica proposta por meio de simulações através de medidas de pacotes perdidos na rede proposta e comparamos os valores com as medidas de pacotes perdidos em uma rede simulada com esquema de roteamento tradicional usando OSPF como o protocolo de roteamento IGP e verificando que aplicando essa técnica ocorrem melhorias na distribuição de tráfego.

Realizamos a avaliação da melhoria do desempenho da rede variando dois parâmetros de engenharia de tráfego relativos à ocupação dos enlaces e uso do *backplane* e as possíveis combinações entre esses parâmetros, que são usados para modificação de custos dos enlaces entre os roteadores da rede simulada.

Usando os mecanismos propostos neste trabalho, uma estação de gerência pode escolher automaticamente o melhor caminho levando em conta parâmetros estabelecidos pelo operador da rede, mesmo que o caminho escolhido passe por um ponto de tradução de um protocolo para outro.

A estratégia proposta faz medidas dinâmicas de parâmetros relevantes nos elementos da rede (*switch*es ou roteadores) e muda os custos dos enlaces em uma tabela criada na memória da estação de gerência. É interessante notar que não se mudam os custos nos *switch*es propriamente ditos, mas sim nessa

tabela localizada na estação de gerência. O algoritmo proposto trabalha medindo a utilização dos enlaces conectados aos nós da rede, além de outros parâmetros como uso de CPU, uso da Matriz de Conexões (*backplane*), descarte de células, ou qualquer outro parâmetro que o operador da rede queira usar.

Para o propósito de encontrar o melhor caminho dentro das redes IP e MPLS, decidimos usar o algoritmo de Dijikstra (DIJKSTRA, 1959). Selecionando o ponto de início de uma conexão no domínio MPLS da rede e o ponto de terminação dessa conexão no domínio IP da rede, o algoritmo calcula o melhor caminho para a conexão e configura o ponto de tradução. Nesse ponto é criado o *Interworking*, configurando os lados IP e MPLS da conexão no *switch* de interwork. No domínio MPLS é criado um túnel enquanto que no domínio IP criamos rotas estáticas que fazem com que o tráfego siga o caminho determinado por nossa engenharia de tráfego. O mesmo trabalho pode ser aplicado facilmente a redes ATM, sendo que, no domínio ATM, deve-se criar circuitos PVC (*Private Virtual Circuit*) pelo caminho definido pela engenharia de tráfego.

1.3 Método de pesquisa

Nosso trabalho é fundamentado na pesquisa bibliográfica e simulação em computadores. Na fase inicial dos trabalhos fizemos um estudo bibliográfico com a finalidade de aprofundar os conhecimentos teóricos a respeito das tecnologias abordadas e fundamentar o trabalho a ser, então, realizado. Nesta etapa estudamos as tecnologias TCP/IP, ATM e MPLS, assim como diversas tecnologias de roteamento, tais como PNNI, IGP, OSPF, IS-IS. Também foram estudados os algoritmos de decisão de caminhos, como Dijikstra, Ford-Fulkerson e Bellman-Ford (CORMEN *et al*, 2002, p.465-520).

Na segunda fase foram realizadas simulações com a finalidade de avaliar a viabilidade de se fazer engenharia de tráfego fim a fim em uma rede com protocolos distintos. Para isso foi utilizado o simulador de redes NS-2, bastante usado no meio acadêmico e que permite simular redes TCP/IP e MPLS além de seus esquemas de roteamento estático e dinâmico. Esse software também permite criar túneis MPLS, essenciais para o desenvolvimento de nossa proposta.

1.4 Organização da Dissertação

A organização desta dissertação está descrita a seguir. No capítulo 2 mostramos o processo de roteamento segundo a perspectiva de diferentes tipos de redes. O capítulo 3 discute a engenharia de tráfego e alguns de seus parâmetros. No capítulo 4 mostramos as simulações, a topologia utilizada e o tráfego gerado nas redes simuladas. No capítulo 5 analisamos os resultados das simulações. No capítulo 6 apresentamos as conclusões e sugestões para trabalhos futuros. No apêndice A mostramos um teste preliminar feito em uma rede com roteadores e no apêndice B mostramos o código-fonte do programa TCL usado para simulação no NS-2, além do programa batch que automatiza a chamada das simulações com diferentes parâmetros.

2 O PROCESSO DE ROTEAMENTO EM DIFERENTES TIPOS DE REDES

Como mencionado na Seção 1.2, um ponto essencial deste trabalho é o conhecimento sobre o roteamento entre redes com protocolos distintos. A fim de abordarmos esse assunto, precisamos, primeiramente, entender como o roteamento é feito em redes de um único protocolo. Assim, nas seções 2.1 a 2.3, discutiremos como a comutação e o roteamento são feitos nas redes ATM, IP e MPLS. Por fim, nas seções 2.4 e 2.5 abordaremos as redes com protocolos distintos e o roteamento entre elas.

2.1 Redes ATM

Em ATM, podemos ter conexões criadas estaticamente usando-se PVC, que pode ser *Virtual Path Connection* (VPC) ou *Virtual Channel Connection* (VCC) (McDYSAN, 2000, p.333). As conexões estáticas podem ser criadas *hop-by-hop* (nó a nó), manualmente. Nesse caso, o operador da rede segue uma tabela pré-estabelecida pela engenharia de redes e cria em cada *switch* as entradas e saídas para cada conexão.

Outra possibilidade é criar as conexões ATM dinamicamente através do protocolo PNNI que estabelece os circuitos chamados *Soft Permanent Virtual Circuit* (SPVC) entre pontos da rede (AF-CS-0127.000, 1999).

O PNNI usa uma técnica de roteamento com rota na origem, disseminando as rotas através da rede como definido nas especificações do ATM Fórum (AF-PNNI-0055.001, 2002), mas nenhum algoritmo de seleção de caminho é especificado pelo PNNI para uso em seus nós (CONTE, 2003, p.38-40) , ficando sob responsabilidade de cada fabricante criar suas estratégias de

roteamento. Para a admissão de conexões em cada nó, o PNNI usa o algoritmo *Generic Call Admission Control* (GCAC). Quando estiver computando uma rota, o nó de origem usa o GCAC para estabelecer a conexão e trocar informações sobre as decisões dos GCAC dos outros nós.

Em uma rede ATM usando PNNI, o operador da rede seleciona a origem e o destino de uma conexão SVPC ou SVCC e a conexão é criada entre os nós intermediários seguindo o caminho calculado pelo algoritmo de roteamento. O protocolo PNNI é usado também para distribuir a topologia usando *PNNI Topology State Elements* (PTSE) para todos os nós pertencentes ao mesmo *peer group* na rede. Desse modo, um mecanismo de inundação (*flooding*) (McDYSAN, 2000, p.337) é usado para enviar parâmetros de mudanças de estado da topologia para o *peer group* inteiro.

2.2 Redes IP

As redes IP encaminham os pacotes de dados baseando-se em tabelas de roteamento presentes na memória de seus roteadores. Para enviar um pacote IP, um roteador deve verificar seu endereço de destino e verificar em sua tabela de roteamento se encontra uma rota correspondente. Ao encontrar uma rota correspondente, o roteador encaminhará o pacote para a interface determinada pela tabela.

Os protocolos de roteamento são divididos basicamente em *Interior Gateway Protocol* (IGP) e *Exterior Gateway Protocol* (EGP). Os protocolos do tipo IGP são usados no interior das redes privadas enquanto que os protocolos do tipo EGP permitem a comunicação entre redes diferentes, ou seja, quando um provedor tem uma saída para outra rede válida, usa um protocolo do tipo EGP (STEWART III, 1998, p.17-19). São exemplos de IGP o *Open Shortest Path First* (OSPF), o *Routing Information Protocol* (RIP) e o *Intermadiate System to Intermediate System* (IS-IS) (BLACK, 2000, p.43-121). Um exemplo comum de

protocolo EGP é o *Border Gateway Protocol* (BGP) usado para transferir rotas entre Sistemas Autônomos (Autonomous System – AS) na *Internet*.

O OSPF trabalha em áreas de redes privadas, com um ou mais ASs, (THOMAS II, 1998, p.173-211). Uma empresa pode ter diversos ASs comunicando-se entre si e trocando rotas a fim de que todos os roteadores de uma rede conheçam a topologia como um todo. A área principal é chamada de área 0 e as demais áreas devem receber numeração distinta. Quando uma rede privada tem saída para a Internet e deve enviar e receber rotas, pode usar o protocolo BGP para trocar suas rotas válidas (com endereços públicos) entre os ASs de provedores (RECKTER, 2006, p.53-89). Isso pode ser feito, por exemplo, exportando-se as rotas válidas do OSPF (IGP) para o BGP (EGP) e vice-versa.

2.3 Redes MPLS

A tecnologia MPLS trabalha com o conceito de rótulos (*label*). Cada pacote a ser transmitido recebe um cabeçalho de, no mínimo, 32 bits (ROSEN *et al*, 2001, p.2). Sendo 20 bits reservados para o rótulo, 3 bits experimentais, 1 bit para indicação de fim de empilhamento e 8 bits para indicar o tempo de vida de um pacote (TTL – *Time To Live*). O rótulo é o identificador de um fluxo de dados. É por meio do uso de rótulos que cada roteador identificará a entrada e a saída de um caminho na rede.

Na arquitetura MPLS, os roteadores que fazem exclusivamente encaminhamento de pacotes baseados em rótulos são chamados de *Label Switch Routers* (LSR). Os roteadores de borda que servem de entrada e saída para a rede MPLS são chamados de LSR-ingress e LSR-egress (BLACK, 2002, p.45-47), ou LSR de Ingresso e LSR de egresso respectivamente. Os rótulos são distribuídos entre os LSR usando-se o protocolo *Label Distribution Protocol* (LDP) (PEPELNJAK, 2003, p.118-121). O protocolo *Resource reSerVation Protocol* (RSVP) é usado para reservar recursos nos roteadores da rede e

também pode ser usado para distribuir rótulos. O protocolo LDP também pode ser usado para reserva de recursos através de sua extensão LDP-*Traffic Engineering* (LDP-TE). RSVP e LDP trabalham no Plano de Controle do MPLS, enquanto que o encaminhamento de pacotes MPLS ocorre no Plano de Dados conforme pode-se ver na Figura 1. Nessa mesma figura vemos a *Forward Information Base* (FIB) desse LSR, que alguns autores chamam de *Label Information Base* (LIB). A FIB mantém uma tabela de encaminhamento de rótulos onde se encontram as referências de rótulos e interfaces de entrada e saída associadas a esses rótulos. A FIB é consultada pelo plano de dados a fim de fazer o encaminhamento dos pacotes e é mantida pelo plano de controle, com base nas informações recebidas pelos protocolos de roteamento IGP, EGP, LDP ou RSVP.

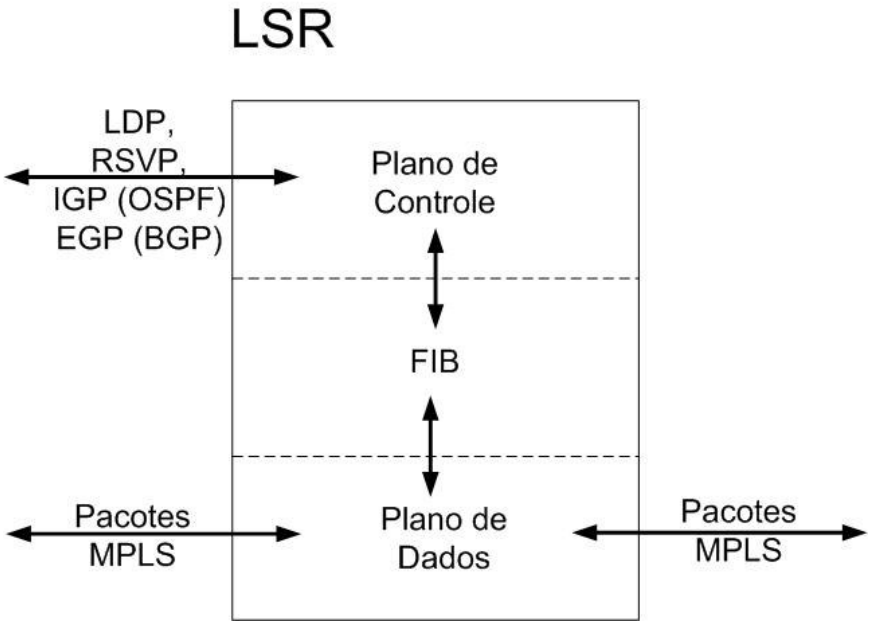

Figura 1 – Planos de Controle e de Dados de um LSR

Os nós de ingresso são capazes de criar *Label Switched Path* (LSP) na rede MPLS, que são caminhos para chavear os rótulos na rede. Dentro de um domínio MPLS um LSP pode cruzar por vários LSR. Cada LSP é atribuído a uma *Forward Equivalence Class* (FEC) a qual especifica os requisitos para o

encaminhamento de pacotes, podendo associar fluxos de dados de acordo com seu endereço de destino, o que é mais comum, ou então associá-los com sua classe de tráfego, ou sua porta (protocolo). Uma vez associado um pacote a uma FEC ele é encaminhado ao LSP, tendo sido associado um rótulo. Nos LSRs intermediários da rede, os pacotes são encaminhados de acordo com os rótulos associados ao túnel, e nenhuma outra informação dos pacotes tem que ser verificada.

Nas redes MPLS tem-se também a opção de criar as conexões estaticamente ou dinamicamente.

Para se criar as conexões estáticas, usamos o que é chamado de rotas explícitas, que são definidasdas pelo operador da rede, sendo que, o caminho inteiro é especificado manualmente.

O MPLS tem procedimentos bem definidos para estabelecer túneis dinamicamente. Usando RSVP associado a um protocolo de roteamento IGP como o OSPF ou o IS-IS, distribuem-se os estados dos enlaces através dos *switches* da rede, na topologia que foi conhecida através do OSPF ou do IS-IS. Há também a opção de se realizar engenharia de tráfego com o MPLS-*Traffic Engineering* (MPLS-TE) que usa para isso o RSVP-TE como explicado na RFC-3209 (AWDUCHE, 2001) em uma área MPLS ou entre várias áreas MPLS como mostrado na RFC-4105 (LEROUX *et al*, 2005). A Figura 2 demonstra várias áreas OSPF interconectadas, nessa figura consideramos que todas as áreas estão usando RSVP-TE. De acordo com essa mesma RFC-4105 o uso de *traffic engineering* é possível entre diversas redes MPLS que estejam usando RSVP-TE, desde que usem OSPF-TE ou IS-IS-TE que são adaptações dos protocolos IGP para engenharia de tráfego. Ainda nessa figura, vemos uma conexão de um túnel passando de uma área para outra e usando RSVP-TE, que só é possível devido ao uso de OSPF com TE. É importante observar que essa RFC só faz

menção ao uso de áreas diferentes em MPLS-TE, ou seja, redes unicamente MPLS.

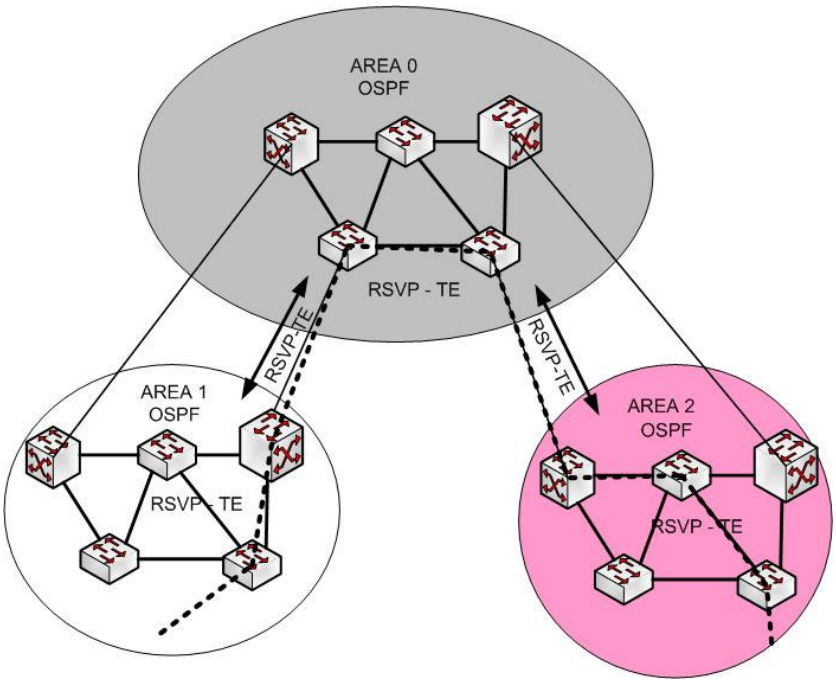

Figura 2 – RSVP-TE usado entre áreas OSPF

2.4 Redes mistas e pontos de tradução

No cenário atual das redes de telecomunicações é comum que as redes legadas sejam mantidas para atender aos clientes existentes. A interoperabilidade de diferentes tecnologias faz com que sejam necessários desenvolvimentos de traduções de uma tecnologia para outra, o que é conhecido na atual literatura como *Interworking Point* (MFA-FORUM-18.0.0, 2007). Existem muitos padrões de Interworking e podemos citar alguns deles: Frame-Relay para ATM como definido pelo Frame-Relay forum (FRF-5, 1994) ou (FRF-8.2, 2004), IP-para-ATM (AF-MPOA-014-000, 1999) e ATM e *Frame-Relay* para MPLS (AF-AIC-0178-001, 2003) (MFA-FORUM-10.0.0, 2006).

Todos esses *interworkings* têm um ponto em comum, que é o ponto de tradução, o qual é fixo e não pode ser movido dinamicamente. A Figura 3 mostra um exemplo de uma rede mista com ATM e MPLS, na qual uma conexão entre os pontos A e B é mostrada. A e B são as terminações da conexão. A mesma passa pelo ponto C, o qual está na área de *interworking*. O ponto C é fixo para essa conexão e tem que ser criado manualmente. Dentro da rede ATM pode-se criar o caminho dinamicamente usando-se para isso o protocolo PNNI e dentro da rede MPLS pode-se criar a conexão dinamicamente com RSVP e LDP. Porém, o ponto de tradução de um protocolo para outro é fixo e não pode ser movido. Recentemente o MFA-Forum (MFA-FORUM-18.0.0, 2007) lançou uma norma que especifica como deverá ser feita a inetroperabilidade entre ATM e MPLS no que se refere à criação de *Soft Permanent Virtual Circuits* (SPVC), na qual apresenta as políticas de tradução entre os protocolos e chega a mencionar um mecanismo de *crankback* para evitar nós ou enlaces defeituosos, mas não especifica de forma detalhada como o *crankback* deverá ser feito e também não menciona nem sugere se algum mecanismo mais complexo de engenharia de tráfego é usado. No melhor do nosso conhecimento, não foi desenvolvida nenhuma estratégia de engenharia de tráfego fim a fim para criar dinamicamente túneis iniciando na rede ATM e terminando na rede MPLS, usando engenharia de tráfego fim a fim. Quando muito, foram criadas especificações para tratar alguns tipos de serviços.

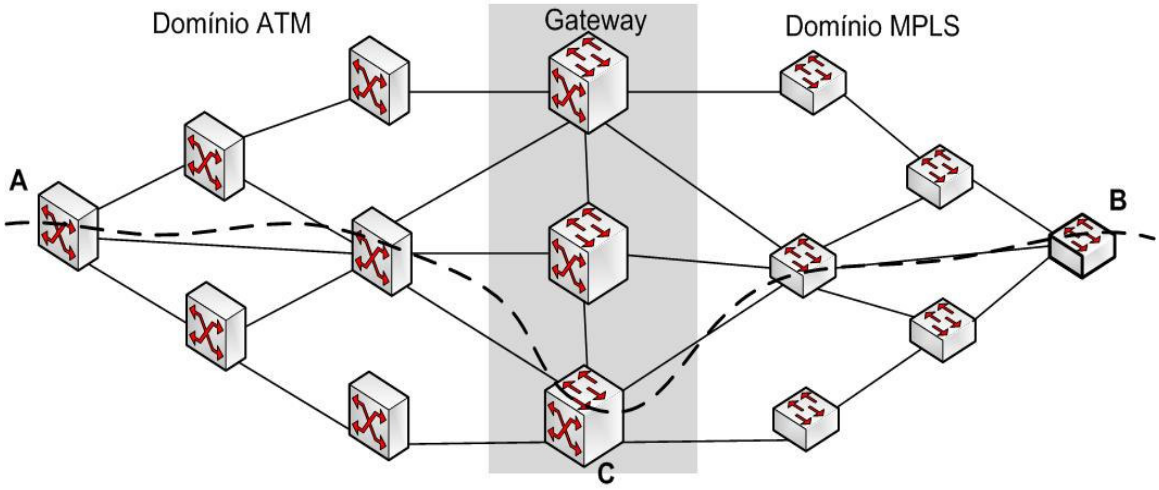

Figura 3 – Rede Mista ATM com MPLS

2.5 Redes mistas IP com MPLS e Roteamento em redes de múltiplos domínios

Está crescendo o uso do MPLS no *backbone* das redes metropolitanas transportando TCP/IP. As áreas IP e MPLS se comunicam, podendo estabelecer vizinhanças entre seus protocolos IGP (BLACK, 2002, p.194-210). Dessa forma, tanto os roteadores da área MPLS quanto os roteadores das áreas IP podem conhecer detalhes das topologias uns dos outros. Dentro do domínio MPLS pode-se estabelecer túneis a fim de transportar os pacotes entre um nó de ingresso e outro de egresso, sendo depois os pacotes roteados dentro da rede IP.

Os protocolos do tipo *Interior Gateway Protocol* (IGP) são usados dentro de redes em áreas estabelecidas. Os protocolos IGP mais usados no mercado são *Intermediate System to Intermediate System* (IS-IS) e *Open Shortest Path First*(OSPF). Tanto IS-IS quanto OSPF são protocolos do tipo *link-state* (THOMAS II, 1998, p.104), ou seja, montam suas tabelas de roteamento, levando-se em consideração o custo e o estado dos enlaces entre os nós da

rede. Esses protocolos permitem o uso dos algoritmos *link-state*, também chamados de algoritmos *Shortest Path First* (SPF) ou algoritmo de Dijkstra.

As tabelas dos protocolos IGP podem ser difundidas dentro do domínio IP e também dentro do domínio MPLS, de forma que os roteadores das duas redes conheçam a topologia como um todo (PEPELNJAK, 2003, p.305-336).

Caso dentro do domínio MPLS estejam se usando túneis, a tabela de rotas presente no equipamento de entrada (*ingress*) terá uma referência ao destino com apenas um salto, mesmo que o túnel esteja passando por mais de um roteador.

O estabelecimento de túneis pode ser feito de forma manual ou automatizada, sendo que quando criado manualmente, um túnel seguirá o caminho estabelecido pelo operador da rede. Já, quando criado automaticamente, o túnel seguirá um caminho estabelecido por um protocolo como o RSVP (OSBORNE, 2003, p.147-148). No modo automático, o operador da rede determina os pontos de entrada e saída da rede MPLS e o restante do caminho é determinado automaticamente pelo algoritmo de roteamento em uso na rede.

É importante observar que, no caso do uso de túneis, os pontos de tradução da rede são fixos, criados pelo operador da rede, e não podem trocar de roteador, mesmo que o caminho usado pelo túnel e os nós que passem pelo caminho estejam saturados.

Também cabe mencionar que, no caso do uso do protocolo IGP para estabelecimento de tráfego entre uma rede usando MPLS e outra usando IP, que os fluxos de dados seguirão os caminhos decididos pelos critérios do SPF.

Na Figura 4 vemos uma rede IP conectada a uma rede MPLS sendo que os roteadores A, B C, D e E pertencem à rede MPLS e os roteadores 1, 2, 3, e 4 à rede IP. O roteador E funciona como *gateway*, e tem enlaces para as duas redes. Nessa mesma figura, um túnel MPLS iniciando no roteador A (ingresso), passando por C e D e terminando em E (egresso). Esse túnel está representado na cor vermelha e o fluxo de dados segue pela rede IP, obedecendo às rotas aprendidas pelo IGP na rede IP, seguindo o caminho 1 - 2 - 3. Quando um fluxo de dados entra no roteador A com destino para o roteador 3, o roteador A olha em sua tabela de roteamento e encontra uma entrada para 3 através de E, ou seja, usando o túnel a ligação A – C – D – E aparece como se fosse um único salto, já que nos roteadores intermediários os pacotes serão encaminhados pelos seus rótulos. O caminho todo pode ser representado por A – E – 1 – 2 – 3, totalizando 5 saltos.

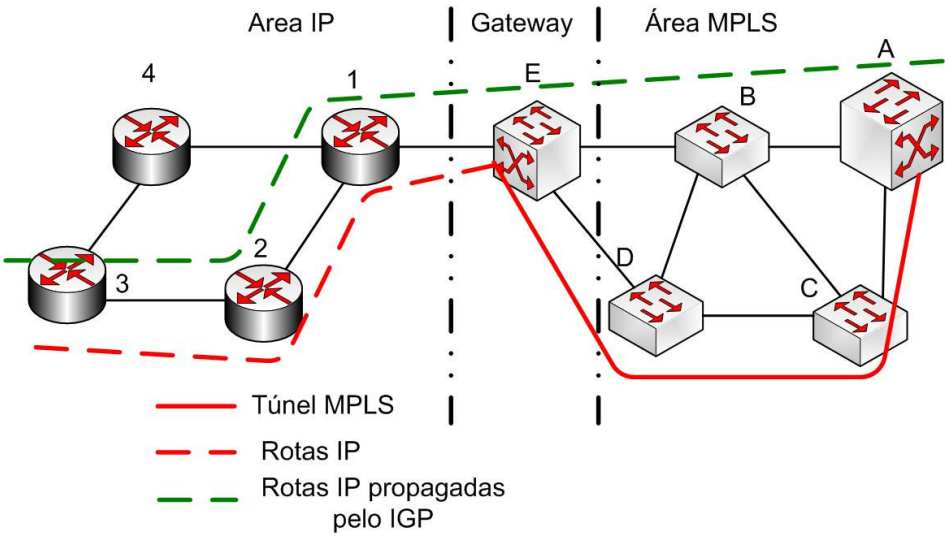

Figura 4 – Áreas MPLS e IP

Também há a possibilidade de rotear os pacotes dentro da rede MPLS, seguindo-se basicamente a tabela de rotas aprendida pelo IGP. Dessa forma a rede é vista como uma única rede IP sendo que pode ser dividida em áreas IGP. Essa possibilidade é vista na mesma Figura 4 no tráfego representado pela linha

verde tracejada. Supondo-se que o operador da rede não optou em usar túneis e que as duas áreas se conheçam através do IGP, que pode ser um protocolo como OSPF ou IS-IS, caso o roteador A receba um fluxo de dados com destino ao roteador 3, este seguirá sua tabela de roteamento e encontrará como próximo salto, o roteador B. B também tem uma entrada em sua tabela de roteamento, indicando como pode chegar a 3 através de E. O caminho todo seria A – B – E – 1 – 2 – 3, totalizando 6 saltos entre A e 3.

Da mesma forma, seguindo esse raciocínio, qualquer fluxo de pacotes entre os dois domínios, seguiria a tabela de roteamento IP, dentro da rede IP. Dentro da rede MPLS a tabela de roteamento só é seguida durante o estabelecimento do LSP. Após estabelecido o LSP, os fluxos de pacotes não olham mais a tabela de roteamento, e sim os rótulos. Caso o operador da rede opte por fazer engenharia de tráfego dentro da rede MPLS, ou MPLS-TE (MPLS *Traffic Engineering*), esta seria feita apenas na rede MPLS e não na rede como um todo.

2.6 Considerações finais

Em todas as opções apresentadas, observamos que sempre há algum problema quando a questão é engenharia de tráfego entre domínios de redes com protocolos distintos. Em geral, poucos ou nenhum mecanismo de engenharia de tráfego são considerados pelas normas dos diversos protocolos.

Em ATM usando PNNI como protocolo de roteamento, e atualmente entre ATM e MPLS também usando PNNI é especificado um mecanismo de *crankback* segundo o qual, quando um nó não pode estabelecer uma conexão, devolve a solicitação para o nó anterior para tentar um caminho alternativo.

Em redes MPLS comunicando-se com redes IP e usando o IGP – OSPF para divulgar suas rotas e estabelecer os fluxos de tráfego, apenas o mecanismo de roteamento é usado, seguindo-se a rota mais específica para determinar os caminhos que o tráfego seguirá. Novamente, nenhum mecanismo de engenharia de tráfego mais complexo é usado.

3 ENGENHARIA DE TRÁFEGO

Engenharia de tráfego em redes é o conjunto de técnicas usado para medir, predizer e planejar os recursos de uma rede. Na seção 1 deste capítulo vamos abordar as diferenças entre engenharia de redes e engenharia de tráfego. Na seção 2 faremos uma breve explicação sobre o algoritmo de Dijkstra. Na seção 3 mostramos as tabelas de custo e sua atualização. Na seção 4 mostramos os parâmetros utilizados em nosso trabalho.

3.1 Engenharia de redes e engenharia de tráfego

A engenharia de redes é o trabalho de organizar os recursos físicos a fim de se adequar às necessidades de tráfego, disponibilidade de circuitos e elementos de rede.

A engenharia de tráfego trabalha o tráfego a fim de ajustá-lo aos recursos de uma rede. O foco da engenharia de tráfego é a medida do tráfego e seu controle. Seus objetivos são a melhoria das redes e de seu desempenho. Em redes do tipo Internet, seus principais objetivos, no que se refere à engenharia de tráfego para melhoria de desempenho são (BLACK, 2002,p.162-163), minimizar a perda de pacotes, minimizar o atraso, minimizar a variação do atraso (*jitter*), diminuir a probabilidade de bloqueio, maximizar a capacidade de transferência e atender aos SLA (*Service Level Agreement*) propostos para as redes.

Em muitas situações faz sentido distribuir o tráfego através de diversos caminhos na rede física para que as conexões passem por diversos nós e enlaces a fim de evitar os caminhos mais congestionados da rede. Esse é um problema complexo e que exige estudo da topologia da rede, análise de parâmetros como capacidade dos enlaces, utilização desses enlaces, recursos

diversos dos nós da rede, como *backplane* e sua utilização, memória, assim como o perfil do tráfego que converge para a rede.

3.2 Algoritmos para encontrar o melhor caminho

A fim de executar a tarefa de engenharia de tráfego, precisamos estabelecer uma forma de que, ao inserir uma nova conexão, esta seja inserida em um caminho que evite os trechos mais congestionados da rede. Para isso é necessário que se use um algoritmo capaz de decidir qual o melhor caminho baseado em métricas. Essas métricas podem ser a distância entre dois pontos ou custos arbitrados aos enlaces de uma rede.

Existem diversos algoritmos para encontrar o melhor caminho. O primeiro e mais utilizado em problemas de roteamento em redes é o algoritmo de Dijkstra (DIJKSTRA, 1959). Esse algoritmo foi desenvolvido por Edsger Dijkstra e pode ser usado para encontrar o menor caminho entre dois nós referentemente a um dado conjunto de custos de enlaces conectando os nós. Pode ser usado para encontrar o melhor caminho entre dois pontos de um mapa, representando uma cidade, um país ou mesmo uma rede de computadores.

Outros algoritmos foram desenvolvidos para descobrir o menor caminho entre dois pontos, é o caso do algoritmo de Ford-Fulkerson, também chamado de *backward search* ou busca às avessas. Recebe esse nome, pois faz a busca a partir do destino até encontrar a origem (CONTE, 2003, p.26-28). Outro algoritmo, de Bellman-Ford, resolve o caso em que a métrica é negativa, no caso, os custos dos enlaces entre nós são negativos.

Decidimos usar em nosso trabalho o algoritmo de Dijkstra, por calcular o melhor caminho diretamente entre a origem e o destino, por não trabalharmos com custos negativos e também pela facilidade de programação. Usamos esse algoritmo com a finalidade de decidir o melhor caminho para estabelecer

conexões em uma rede onde os custos dos enlaces são representados por parâmetros dinâmicos e comparamos a técnica proposta com o roteamento clássico do OSPF.

3.3 O algoritmo de Dijkstra

O algoritmo de Dijkstra calcula o menor custo entre dois enlaces ou todos os custos mínimos entre nós de um grafo (CORMEN,2001, p.470-475). Os cálculos começam no nó de origem *o*, usando o algoritmo calculamos o menor caminho para o nó de destino pela rede. A cada passo, o menor caminho para o próximo nó é encontrado. Ao final do procedimento, o menor custo para o nó de destino é encontrado.

Para explicar o algoritmo consideramos que *c*(*i*,*j*) é o custo da linha entre os nós *i* e *j*, e que S é o conjunto que receberá o menor caminho entre os nós *o* e *n*. Os nós *i* e *j* são nós genéricos na rede enquanto *o* é o nó de origem e *n* é o nó de destino. G é um conjunto que representa todos os vértices do grafo. O algoritmo computa os passos representados no Quadro 1 (CORMEN, 2002, p.470) no qual *d*[*v*] é a distância de um nó para a origem. Para iniciar o cálculo, nas linhas 1 a 3 todos os *d*[*j*] são iniciados com o valor infinito, exceto o nó de origem d[o] que é iniciado com o valor zero. Na linha 6, o conjunto S é esvaziado e na linha 7 a Matriz Q recebe os vértices de G. Na linha 8 começa o algoritmo Dijkstra propriamente dito, repetindo a função enquanto tiver algum nó a ser testado, ou seja, enquanto Q não estiver vazio. Na linha 9 o vértice *i* recebe o valor mínimo de Q para a origem e na linha 10 S recebe *i*. A função RELAX(*i*,*j*,*c*), das linhas 1 a 14, muda o valor *d*[*j*] quando um novo caminho é encontrado do nó sob teste para o nó de origem. Repetindo o laço até que Q esteja vazio, o menor caminho será encontrado.

```
DIJKSTRA(G,w,s)
1:Para cada vértice v ∈ V[G]    -- (INICIALIZA FONTES)
2:  Faz   d[j] ← ∞
3:        π[j] ← NIL
4:d[o] ← 0
5:
6:S ← Ø
7:Q ← V[G]
8:Enquanto Q ≠ Ø                        -- CALCULA DIJKSTRA
9:  Faz i ← min (Q)
10:       S ← S U {i}
11:       Para cada vértice  v ∈ Adj[i]
12:             Se d[j] > d[i] + C(i,j)        --  RELAX(i,j,C)
13:                   Então d[j] ← d[i] + C(i,j)
14:                         π[j] ← i
```

Quadro 1 - Algoritmo DIJKSTRA para o cálculo do melhor caminho

3.4 Proposta de algoritmo para engenharia de tráfego

Nossa proposta de engenharia de tráfego é encontrar o melhor caminho para o estabelecimento de conexões em uma rede com domínios de protocolos distintos fim a fim. Para isso usamos uma tabela na memória da estação de gerência, a qual representa a topologia da rede e os custos dos enlaces os quais são atualizados dinamicamente de acordo com parâmetros de utilização dessa rede e que será melhor explicada na Seção 3.5. Dentre vários parâmetros para atualizar dinamicamente os custos dos enlaces optamos por usar a ocupação do enlace, capacidade e ocupação do *backplane* por serem parâmetros que podem ser conseguidos nos elementos da rede ou então calculados a partir de parâmetros disponíveis. Deixamos como sugestão outros parâmetros como o uso de CPU, ficando a cargo do operador da rede decidir quais parâmetros usar desde que tenha como medir ou abstrair esses valores dos elementos da rede.

Para fazer a atualização dos custos dinamicamente foi usada uma função representada como:

$$C_{(t)}(i,j) = w(x_1, x_2, ..., x_n)V(i,j), \quad (1).$$

Nessa função *C* (*i,j*) é o custo de um enlace conectando dois nós e *w(x1, x2, ...,xn)* é uma função que representa o peso aplicado ao custo de um enlace, onde x_1, x_2, ..., x_n são fatores utilizados para penalizar o custo dos enlaces. V(i,j) é a tabela de referência, com os recursos dos enlaces sem uso. A função *w* computa o fator que se deseja usar e aplicar ao custo prévio do enlace, através da relação:

$$w(x_1, x_2, ..., x_n) = 1 + a_1 f_1(x_1) + a_2 f_2(x_2) + ... + a_n f_n(x_n), \quad (2)$$

na qual f_i (i=1 até *n*) é uma função utilizada para penalizar o parâmetro x_i (que pode ser, por exemplo, a ocupação, o tamanho ou o atraso do enlace, a ocupação do *backplane* ou outra métrica qualquer) e a_i é um fator que deve ser determinado pelo operador de rede para otimizar a distribuição de tráfego. É importante notar que, em geral, a determinação dos a_i depende do parâmetro de engenharia de tráfego que está sendo considerado. Por exemplo, um conjunto de a_i que minimize a perda de pacotes de uma determinada rede pode não minimizar a latência média da mesma.

Neste trabalho, consideramos dois parâmetros para penalizar o custo dos enlaces. O primeiro deles é a ocupação do enlace definida por:

$$f(x_1) = x_1 = \frac{V_{lo}}{V_l}, \quad (3)$$

na qual V_l e V_{l0} são, respectivamente a capacidade e a taxa de ocupação do enlace.

O segundo parâmetro é a ocupação do *backplane* definida por:

$$f(x_1, x_2) = [1 + x_2]x_1 = \left[1 + \left(\frac{V_p - V_{pl}}{V_{bp}}\right)\right]x_1, \quad (4)$$

na qual V_{bp} é a capacidade do maior *backplane* da rede, V_p é a capacidade do backplane do roteador ou switch em questão e V_{pl} é a capacidade ociosa desse roteador.

Dessa forma, distinguir-se-á diferentes capacidades de *backplane*. Essa é uma forma de dizer que se existem diferentes caminhos a serem seguidos, um deles passando através de um roteador com muito mais capacidade de transferência que outros, que é preferível passar pelo roteador cujo *backplane* tem maior capacidade ociosa nesse momento do que por outros, mais ocupados.

Na Figura 5 vemos uma rede representada com roteadores de diferentes capacidades. Ao criar-se uma conexão de A para B, considerando-se todos os roteadores desocupados, será preferível que a conexão passe pelo roteador D do que pelo roteador C, já que D tem maior capacidade de backplane do que C.

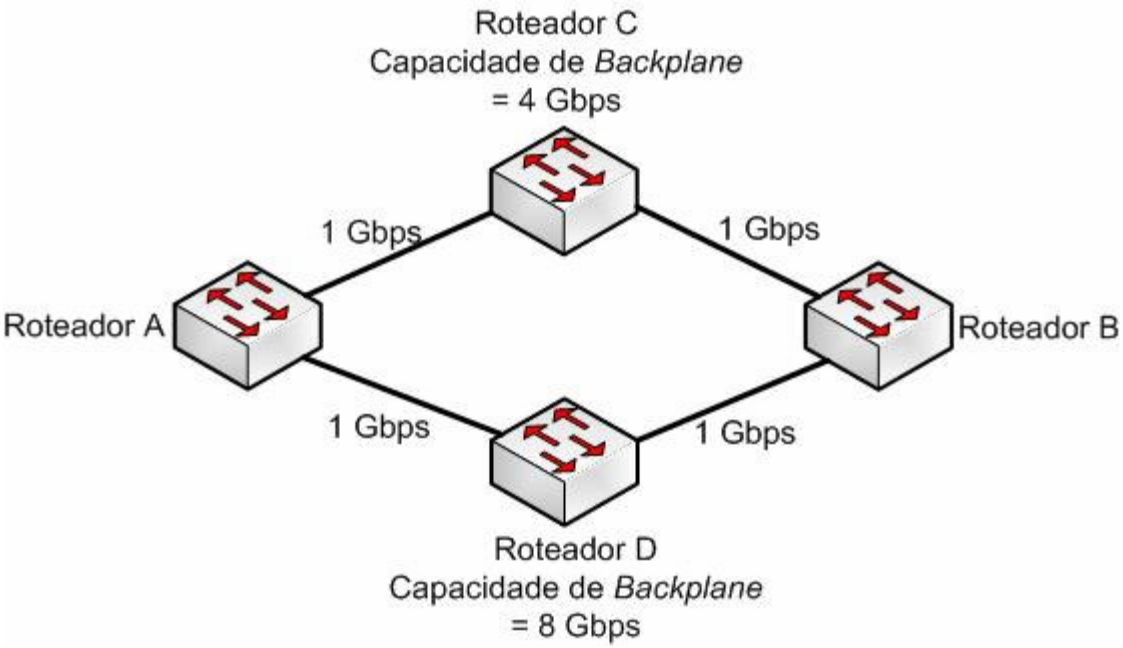

Figura 5 – *Backplanes* com diferentes capacidades

Se todos os roteadores da rede tiverem a mesma capacidade de *backplane* podemos reescrever a expressão 4, ficando assim:

$$f(x_1, x_2) = [1 + x_2]x_1 = \left(1 + \left(\frac{V_{po}}{V_p}\right)\right)x_1 \ , \ (5),$$

na qual V_{po} é a ocupação do *backplane* do roteador em questão e V_p é a capacidade do *backplane* desse roteador.

Alguns *switches* ou roteadores do mercado fornecem esse dado de ocupação do *backplane* em uma variável da *Management Information Base* (MIB) que é a base de dados de informação de gerência e que pode ser coletada através de comandos do *Simple Network Management Protocol* (SNMP). Mas mesmo que um roteador não forneça uma variável de ocupação do *backplane*, esse valor pode ser estimado através da medida da taxa de ocupação dos enlaces ligados ao roteador. Somando-se a taxa de ocupação de todos os enlaces ligados ao roteador, estimamos a taxa de ocupação de seu *backplane*.

Assim, a função de custo que será utilizada neste trabalho é descrita por:

$$w(x_1, x_2) = 1 + \alpha f_1(x_1) + \beta f_2(x_2), \quad (6)$$

na qual as constantes α e β representam, respectivamente, os fatores a_1 e a_2 definidos em (2).

3.5 Atualizando a tabela de custos e recalculando melhores caminhos

Para este trabalho foi adotado o uso de uma tabela de custos na memória da estação de gerência. Essa tabela é uma matriz que representa o custo dos enlaces que interligam os nós da rede. A atualização dos custos nessa tabela é feita dinamicamente em intervalos regulares de tempo definidos pelo operador da rede. A estação de gerência (*Network Management Station* - NMS) requisita determinados parâmetros aos elementos da rede e faz os cálculos mostrados nas equações (1) a (6) para atualizar a tabela de custos de enlaces. Esses valores podem ser a ocupação dos enlaces, ocupação do *backplane* ou outros conforme mostrado na Seção 3.4. Na Figura 6a vemos como essa atualização é feita, ao ser chamada. A rotina lê os valores de ocupação dos enlaces e dos backplanes nos elementos da rede, usando para isso SNMP, em seguida calcula os custos e os atualiza em uma tabela, que é representada pela matriz *C* (*i*,*j*).

Na Figura 6b vemos como uma nova conexão é criada, sobre a tabela atualizada. O NMS calcula o melhor caminho para a nova conexão executando Dijkstra e então, cria essa nova conexão pelo caminho escolhido.

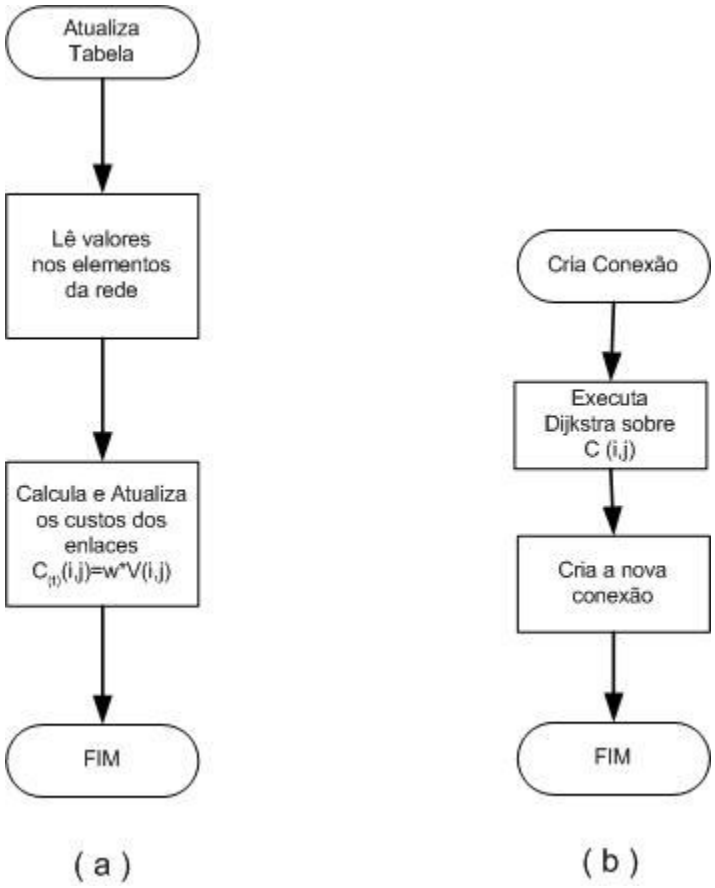

Figura 6 – Atualização da tabela (a) e Criação de uma nova conexão (b)

Com pequenas alterações, o algoritmo pode ser usado para monitorar conexões e recalcular o melhor caminho para as conexões que estejam sendo monitoradas, usando Dijkstra para isso. Se um caminho melhor for encontrado, a conexão será redirecionada. Na Figura 7 vemos o fluxograma que representa a realocação dinâmica de conexões. Quando essa rotina é chamada, lê valores nos elementos da rede, esses valores podem ser a ocupação dos enlaces, ocupação do *backplane* ou outros. Nesse ponto a rotina calcula e atualiza os custos dos enlaces e em seguida executa a rotina de Dijkstra considerando a origem e o destino da conexão monitorada, se houver mudança no caminho da conexão aplica-a.

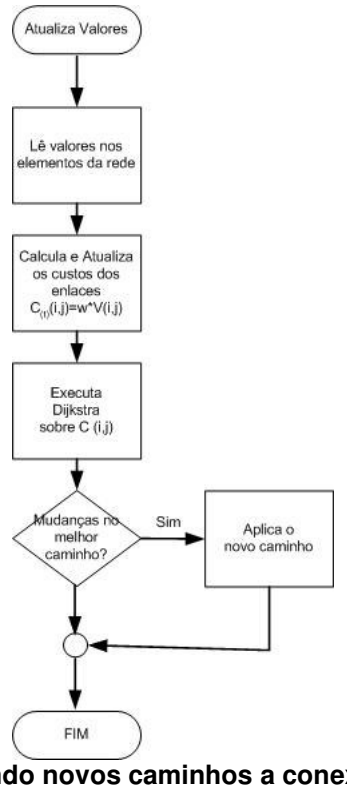

Figura 7 – Aplicando novos caminhos a conexões monitoradas

Em nosso trabalho decidimos usar a opção de inserir cada nova conexão por um caminho de menor custo calculado durante sua criação, não movendo as conexões anteriores. Uma tabela de custos é atualizada dinamicamente, coletando-se na rede dados sobre a utilização dos enlaces. Sobre essa tabela, será calculado o algoritmo de Dijkstra para encontrar o melhor caminho por onde passará um novo tráfego.

4 SIMULAÇÕES

As simulações de nossa proposta foram realizadas no simulador de redes NS-2. Entre outras características, este simulador permite criar redes IP e MPLS e usar diversos esquemas de roteamento, tais como rotas estáticas, roteamento dinâmico com OSPF e também permite a criação de túneis MPLS estaticamente ou dinamicamente. Em nosso trabalho consideramos roteamento explícito na rede MPLS combinado com roteamento estático na rede IP e comparamos os resultados com o roteamento OSPF dinâmico, como IGP. Usamos roteamento estático na rede IP, criando uma rota em cada nó da rede IP, fazemos com que um fluxo de dados com destino a um cliente da rede, siga o caminho desejado, como se fosse um PVC ATM. Dessa forma, simulamos uma rede ATM transportando pacotes IP sobre PVCs, como sugerido pelo grupo de desenvolvimento do NS-2. Na seção 4.1 abordamos a topologia *Manhatan Street* utilizada nas simulações. Na seção 4.2 descrevemos as redes com tamanhos 3 x 3 e 4 x 4. Na seção 4.3 mostramos como é a rede com tamanho 5 x 5. Na seção 4.4 falamos sobre as características do tráfego gerado e suas aplicações. Na seção 4.5 descrevemos as capacidades dos enlaces da rede. Na seção 4.6 falamos sobre o tráfego gerado e finalmente na seção 4.7 descrevemos como é feita a simulação da rede 5 x 5 com a distribuição dos servidores modificada.

4.1 Topologia utilizada

A topologia usada para o desenvolvimento de nosso trabalho é a *Manhatan Street*. Essa topologia possui N x M nós que compõe uma malha bidimensional, sendo que cada nó dá origem a quatro enlaces. Neste trabalho, consideramos que os nós na borda da rede pudessem ter um quinto enlace, para conectá-los a roteadores de concentração. Esses roteadores, por sua vez,

são responsáveis por agregar o tráfego dos clientes e servidores em direção à rede. As Figuras 8, 9 e 10 mostram as redes *Manhattan Street* com três, quatro e cinco nós de largura respectivamente . Observamos que todos os enlaces têm enlaces paralelos, por isso o nome *Manhattan Street*, como as ruas de *Manhattan*.

4.2 Simulações com redes de tamanhos 3 x 3 e 4 x 4

Iniciamos nossos trabalhos com uma rede *Manhattan Street* de tamanho pequeno, com 9 LSRs no núcleo e trabalhando com protocolo MPLS. Nessa etapa, o objetivo era verificar o funcionamento do algoritmo proposto e comparar o desempenho deste algoritmo com o do OSPF. Não era nosso objetivo comparar com MPLS-TE ou RSVP-TE, já que em nossa rede final de tamanho 5 x 5 vamos trabalhar com rede mista MPLS com IP simulando ATM.

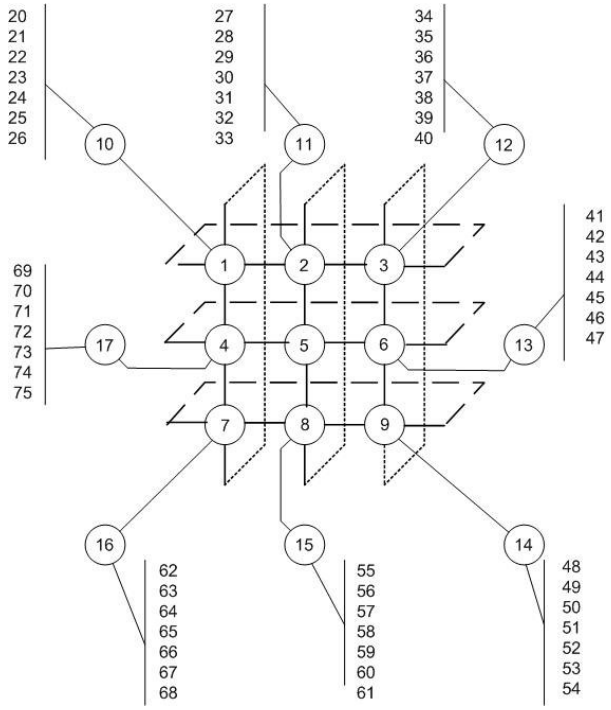

Figura 8 – Rede *Manhattan* com dimensões 3 x 3

A rede considerada nesta etapa, ilustrada na Figura 8, apresenta 27 servidores, numerados de 20 a 47 e 27 clientes numerados de 48 a 75. Os servidores geram tráfegos em direção aos clientes na seguinte ordem: servidor 20 para o cliente 48, servidor 21 para o cliente 49 e assim por diante até o servidor 47 que envia tráfego para o cliente 75. Dessa forma todos os tráfegos cruzam a rede em direção a nós de concentração diametralmente opostos. Nesta etapa do desenvolvimento de nosso trabalho, usamos o parâmetro α a fim de incrementar o custo dos enlaces em nossa tabela de referência.

Um esquema similar foi adotado para a rede de dimensões 4 x 4, ilustrada na Figura 9. Neste caso, as máquinas numeradas de 40 a 81 são servidores, enquanto as máquinas numeradas de 82 a 123 são clientes. O servidor 40 gera tráfego para o cliente 82, o servidor 41 gera tráfego para o cliente 83 e assim por diante até o servidor 81 que envia tráfego para o cliente 123. Nesta etapa passamos a usar os parâmetros α e β, testando suas combinações de valores inteiros de 0 a 10.

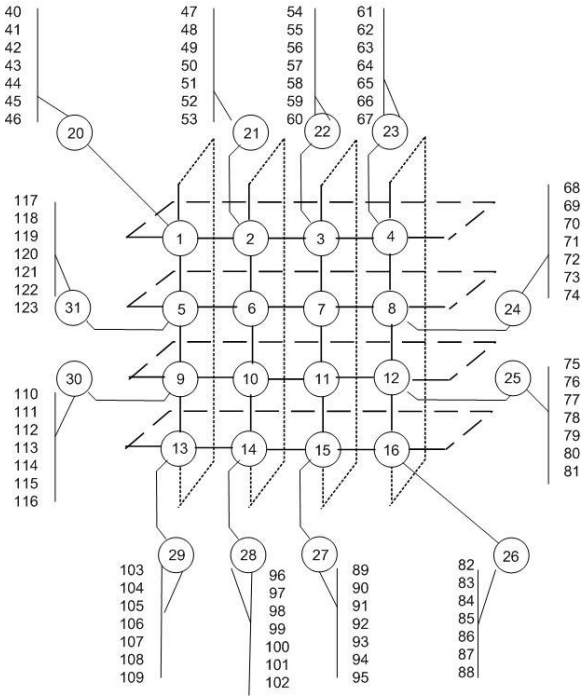

Figura 9 – Rede *Manhattan* com dimensões 4 x 4

4.3 Simulações com redes de tamanho 5 x 5 e divisão de domínios

As simulações em redes de dimensão 5 x 5, cuja topologia está indicada na Figura 10, consideraram dois domínios distintos: um domínio IP simulando ATM e outro MPLS, nos quais realizamos a engenharia de tráfego fim a fim proposta na Seção 3.4 . Da mesma forma que nas redes de tamanho 4 x 4 usamos α e β como parâmetros ajustáveis pelo operador da rede e executamos nossa simulação com combinações possíveis de α e β, inteiros de 0 a 10, perfazendo um total de 121 execuções. Também realizamos a simulação com OSPF a fim de comparar os resultados com os valores obtidos de descartes para α e β.

Nosso trabalho consiste em fazer a engenharia de trafego fim a fim, desde a entrada, pela rede MPLS até a saída na rede IP, distribuindo o tráfego nas duas redes como se fosse uma só rede e dessa forma minimizando a perda de pacotes total da rede.

Os nós numerados de 1 a 10 são nós da rede MPLS, enquanto os nós numerados de 16 a 25 são nós da rede IP e os nós numerados de 11 a 15 são os gateways entre as duas redes. Dessa forma os nós de 1 a 25 compõem o núcleo de nossa rede. A esse núcleo conectamos os nós de concentração que agregam o tráfego entre os servidores e os clientes da rede.

4.3.1 A rede 5 x 5 completa

Os nós de número 1, 2, 3, 4, 5, 6 e 10 receberam um quinto enlace que os conecta aos chamados nós de concentração, numerados de 30 a 36. A esses nós de concentração ligamos os servidores, que estão numerados de 50 a 98. Os nós de número 16, 20, 21, 22, 23, 24 e 25 também são ligados a nós de

concentração através de um quinto enlace e esses nós de concentração estão numerados de 37 a 44.

Cada nó de concentração tem sete máquinas conectadas a si, sendo que as máquinas numeradas de 50 a 98 são servidores enquanto que as máquinas numeradas de 99 a 154 são clientes. A Figura 10 mostra a topologia completa da rede utilizada.

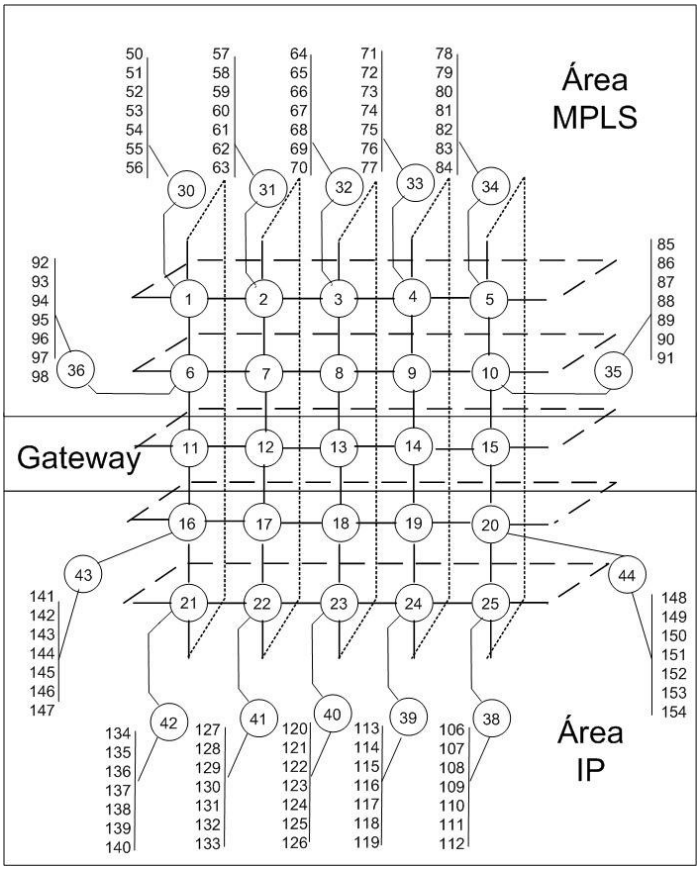

Figura 10 – Topologia completa da rede e divisão dos domínios

4.4 Características do tráfego e aplicações

Para a execução de nossos trabalhos adotamos tráfego UDP do tipo *unicast* e com taxas constantes CBR (*Constant Bit Rate*). Dessa forma simulamos um tipo de aplicação semelhante ao tráfego de Vídeo Sob Demanda

como o usado em locadoras virtuais, nas quais os clientes selecionam e assistem a um vídeo armazenado em servidores. Esse tipo de tráfego se distingue do tráfego de IPTV via Internet, pois não caracteriza vídeo ao vivo, que trafega normalmente sobre tráfego *Multicast*.

Outro tipo de aplicação que pode caracterizar nosso trabalho é o transporte de ATM CBR, AAL1 ou AAL2 sobre MPLS. Esse tipo de aplicação também gera tráfego a taxas constantes.

4.5 Capacidades dos enlaces

Nossa intenção inicial era avaliar a rede com enlaces da ordem de Gbps. Entretanto simulações com tráfego nessa ordem eram demasiadamente demoradas e comprometeriam o desenvolvimento do trabalho. Por essa razão trabalhamos com enlaces na ordem de Mbps, assim como taxas de transmissão máximas nessa ordem, o simulador NS-2 trata de criar os enlaces e seus buffers adequados às taxas de transmissão programadas e isso é bem conhecido. Dessa forma, consideramos que os enlaces entre os nós do núcleo da rede têm capacidade de 1 Mbps, enquanto que os enlaces entre os nós de concentração e o núcleo têm capacidade de 10 Mbps, usada nesses enlaces para evitar perda de pacotes fora do núcleo da rede, já que o objetivo é testar o núcleo e não as bordas da rede.

4.6 Taxa de comunicação

Admitiu-se um tráfego de 300 kbps entre cada servidor e seu cliente correspondente. Dessa forma, como são 7 servidores conectados a cada nó de concentração, cada um gera uma taxa máxima de 2,1 Mbps para passar ao núcleo da rede. Esse tráfego supera a capacidade de um enlace do núcleo e, portanto, deve ocasionar a perda de pacotes. O algoritmo proposto para o

estabelecimento de conexões deve reduzir essa perda em comparação à utilização do OSPF.

Na rede de topologia 3 x 3, admitimos que a cada segundo uma nova conexão é estabelecida, e seu tráfego de 300 kbps iniciado, entre servidores e clientes localizados em pontos diametralmente opostos na rede e interrompemos os tráfegos no período de 25 segundos. Na rede 4 x 4 também admitimos que a cada segundo uma nova conexão é estabelecida, e seu tráfego de 300 kbps iniciado, entre servidores e clientes localizados em pontos diametralmente opostos na rede até o período de 45 segundos e finalizamos interrompendo a simulação com 50 segundos. Já na topologia 5 x 5, admitimos que a cada segundo uma nova conexão é estabelecida, e o tráfego de 300 kbps iniciado, entre servidores e clientes diametralmente opostos na rede e, no período de 60 segundos, começamos a interromper os tráfegos até que não haja mais tráfego na rede e paramos a simulação com 120 segundos. Dessa forma testamos para a rede de topologia 5 x 5 uma situação onde os tráfegos são iniciados e depois de algum tempo interrompidos.

4.7 Simulação com a distribuição dos servidores modificada

Para verificarmos a robustez do algoritmo proposto, repetimos as simulações da topologia 5x5, alterando a disposição dos servidores para uma disposição na qual servidores em um mesmo nó de concentração sirvam a clientes localizados em diferentes nós de concentração de clientes. A nova configuração pode ser vista na Figura 11. O cliente de número 148 requisita tráfego ao servidor número 50, que está no nó de concentração 30. O cliente de número 149 requisita tráfego ao servidor número 51, que está no nó de concentração 31. Ambos os clientes estão no nó de concentração 44. E assim por diante, clientes de um mesmo nó de concentração estão solicitando dados a servidores em diferentes nós de concentração.

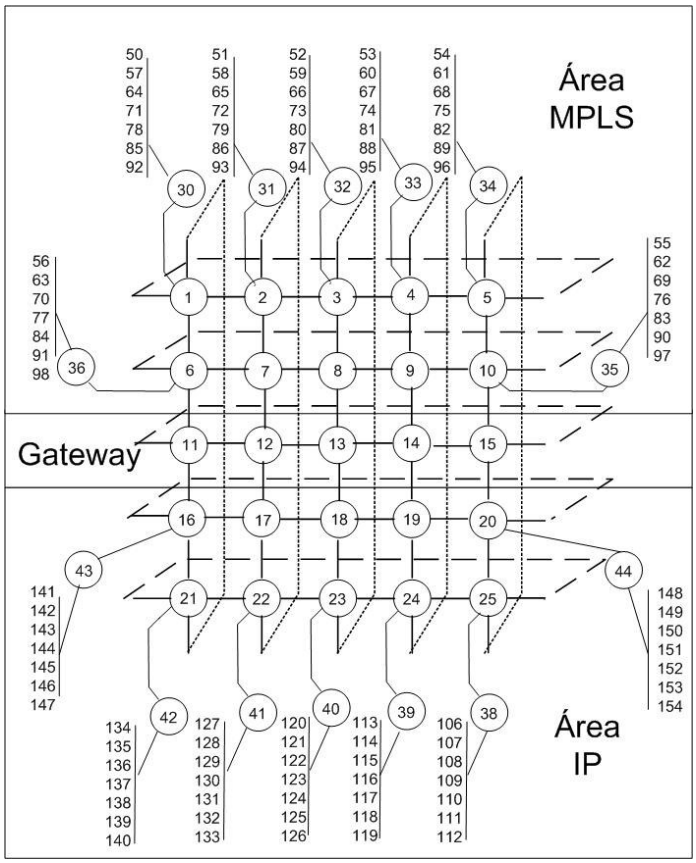

Figura 11 – Disposição dos servidores modificada

5 RESULTADOS

Para a análise do desempenho das redes realizamos as simulações descritas no capítulo anterior e avaliamos dados referentes ao tráfego gerado e recebido, tais como, número de pacotes recebidos, número de pacotes descartados e total de pacotes gerados. Na primeira seção deste capítulo apresentamos os resultados obtidos para uma rede *Manhattan Street* de tamanho 3 x 3. Na segunda seção apresentamos os resultados para uma rede 4 x 4. Na terceira seção mostramos os resultados para uma rede de tamanho 5 x 5 e na quarta seção, os resultados para uma rede 5 x 5 com a distribuição dos servidores modificada.

5.1 Resultados das Simulações Rede 3 x 3

Nesta etapa, conforme a descrição do Capítulo 4, consideramos uma rede MPLS de tamanho 3 x 3 no núcleo com tráfego de 300 kbps entre cada par servidor– cliente. O NS-2 fornece o número total de pacotes recebidos e perdidos para cada destino e, através destes resultados, podemos avaliar o percentual de pacotes perdidos em toda a rede. Repetimos as medições para α (ocupação do enlace) variando em valores inteiros entre 0 e 6 e também para OSPF. Os valores registrados estão apresentados no gráfico da Figura 12. Ao final do tempo de simulação, o OSPF apresenta uma perda de pacotes de aproximadamente 52%. Por outro lado, a perda de pacotes do algoritmo proposto é de 34% para α= 1 e, inicialmente, exibe um comportamento decrescente à medida que α é aumentado. De fato, a perda de pacotes é mínima e igual a 19% para $3 \leq \alpha \leq 5$, o que sugere que os valores de α neste intervalo distribuem o tráfego da maneira mais adequada possível. Se $\alpha > 5$, a perda de pacotes volta a aumentar devido à forma como atua o α. Com valores pequenos de α, como 1 e 2, o algoritmo tende a inicialmente ocupar várias

vezes o mesmo enlace em um nó antes de mudar para outro enlace. Nessa dimensão da rede, encontramos o valor ideal de $\alpha = 4$. Com valores maiores de α a distribuição é maior e as conexões iniciais tendem a ocupar áreas mais distribuídas da rede, o que acaba causando mais perdas de pacotes para as conexões posteriores, por já encontrarem segmentos da rede ocupados. Estes resultados sugerem que a engenharia de tráfego proposta pode oferecer uma melhora de até 2,7 (=52/19) vezes em relação ao desempenho do OSPF..

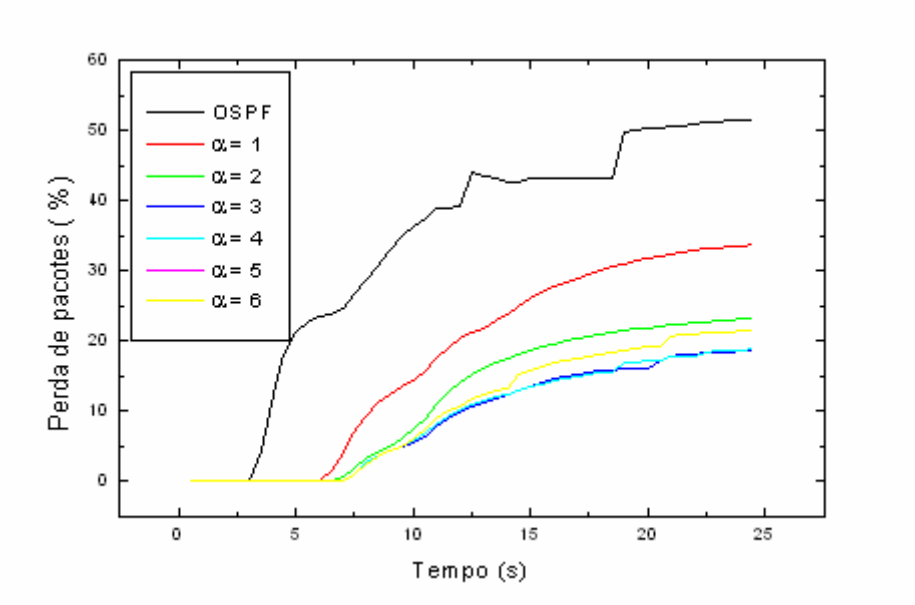

Figura 12 – Percentual de pacotes perdidos para a rede 3 x 3

5.2 Resultados das Simulações Rede 4 x 4

Em uma segunda etapa, aumentamos o nível de complexidade de nossas simulações, para avaliar a operação do algoritmo proposto em uma rede *Manhattan Street* de tamanho 4 x 4 e acrescentamos a variação do β (relativo à ocupação do *backplane*).

Na Figura 13 pode-se ver os resultados de percentual de perda de pacotes para $\alpha = 0$ e variando-se β. Enquanto que para OSPF verificamos perda

de pacotes de aproximadamente 50%, para β = 8 tivemos o menor valor de perda de pacotes, chegando a aproximadamente 25%.

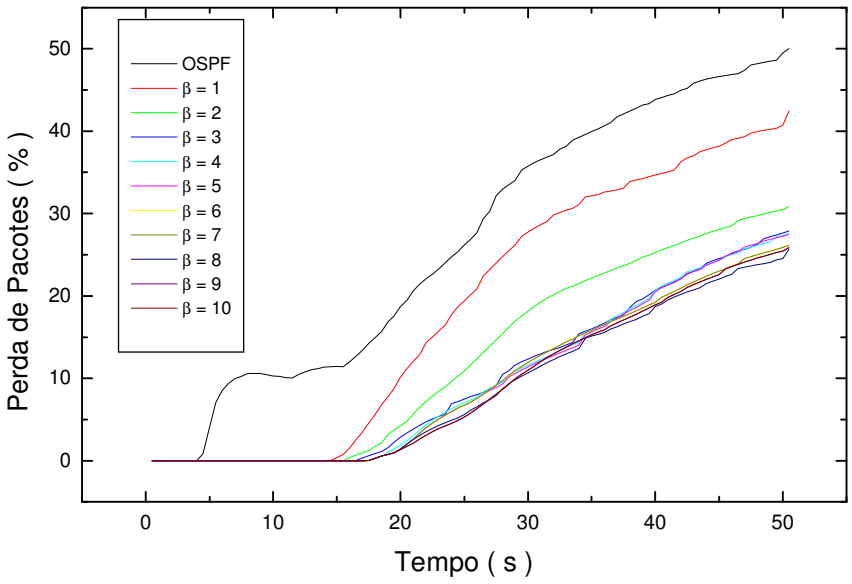

Figura 13 - Perda de pacotes para rede 4 x 4 para α = 0

No gráfico da Figura 14 podemos ver a perda de pacotes percentual e cumulativa para β = 0 e variando-se α. O valor mínimo foi conseguido com α = 10, de aproximadamente 28%.

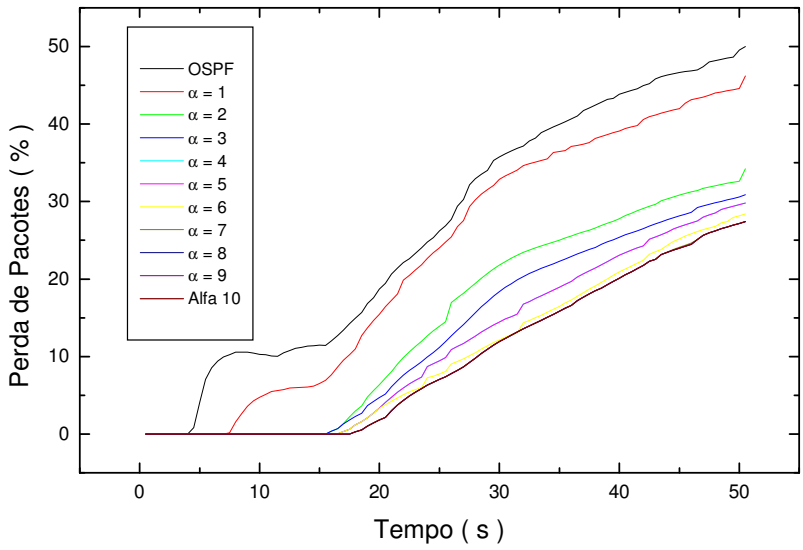

Figura 14 - Perda de pacotes para rede 4 x 4 para β = 0

A Figura 15 mostra um gráfico do percentual de perda de pacotes em função de α e β para o momento de 49,5 segundos. Escolhemos esse momento de 49,5s por ser o momento de maior perda percentual do OSPF. Nesse gráfico podemos observar a perda de pacotes para OSPF de aproximadamente 50% (ponto 0 x 0), e uma queda substancial para aproximadamente 25% para o algoritmo proposto, sendo que o percentual de descartes praticamente se estabiliza em 25% para combinações de maiores valores de α e β. Demarcamos no gráfico duas áreas. Na primeira, de cor azul, a perda de pacotes tem valores compreendidos dentro de uma variação de ±1% ao redor do valor de 26%. A segunda área, na cor violeta, é um quadrado delimitado pelos vértices α = β = 5 e α = β = 10. Destacamos essa área por estar dentro da mesma variação de ±1% e por não estar próxima da borda da área amarela, onde pequenas variações nos parâmetros resultam em grandes variações no percentual de perda de pacotes.

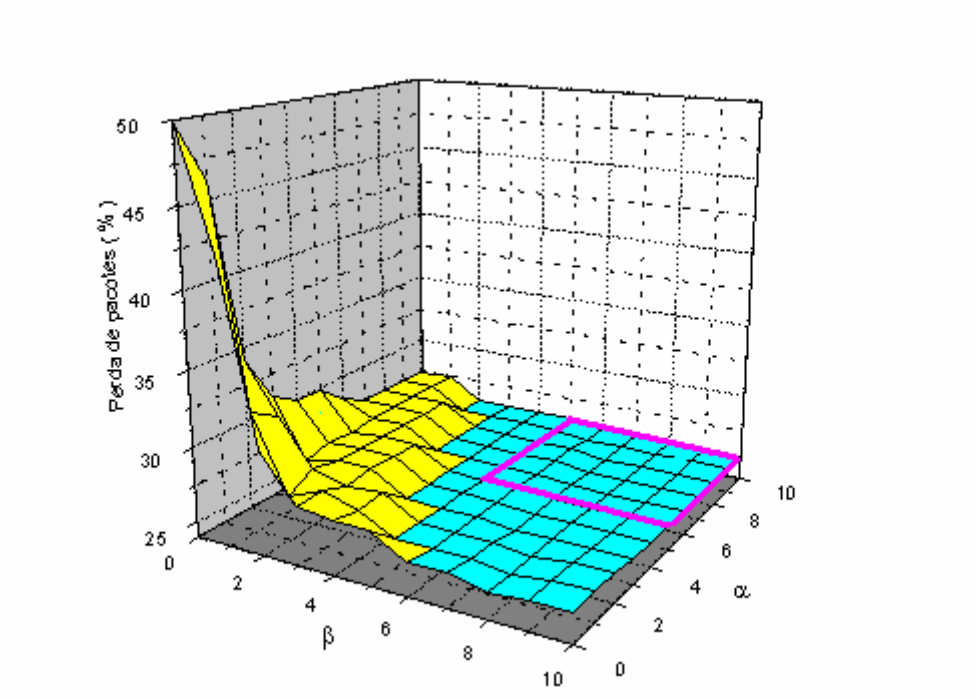

Figura 15 – Perda de pacotes (%) em função de α e β – Topologia 4 x 4

5.3 Resultados das Simulações Rede 5 x 5

Em uma terceira etapa, realizamos testes em uma rede de tamanho 5 x 5 e com domínios de protocolos distintos. Continuamos usando α e β como parâmetros relativos à ocupação dos enlaces e *backplane*, respectivamente.

5.3.1 Número de pacotes descartados

Em cada cliente contamos o número de pacotes descartados e contabilizamos o total em todos os nós a cada ciclo do programa de 0,5s, desde o início até o final do programa após 120 segundos. A Figura 16 mostra os resultados obtidos com α= 0 e para diferentes valores de β comparados ao OSPF. Nessa figura podemos observar que com o OSPF a perda de pacotes

chegou a 165.000 pacotes, enquanto que para β = 6 foi de aproximadamente 81.000 pacotes, demonstrando uma melhora de 51%.

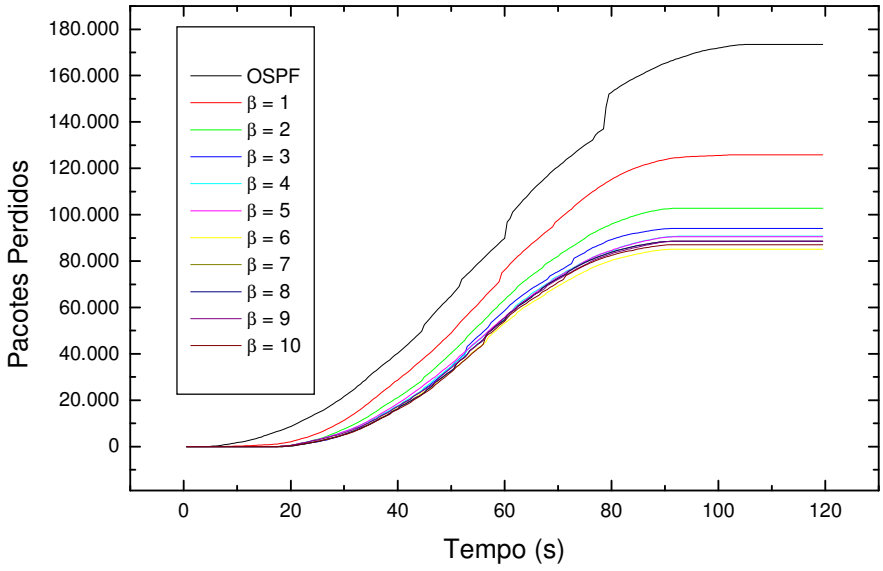

Figura 16 – Número absoluto de pacotes perdidos x tempo para α = 0.

De maneira similar, a Figura 17 mostra os resultados obtidos com β= 0 e para diferentes valores de α comparados ao OSPF. Observa-se que houve um valor mínimo de descartes para α = 6. Enquanto que para OSPF tivemos um total de 165.000 pacotes descartados, para α = 6 tivemos 85.000 pacotes descartados, mostrando uma melhora de 48,5%.

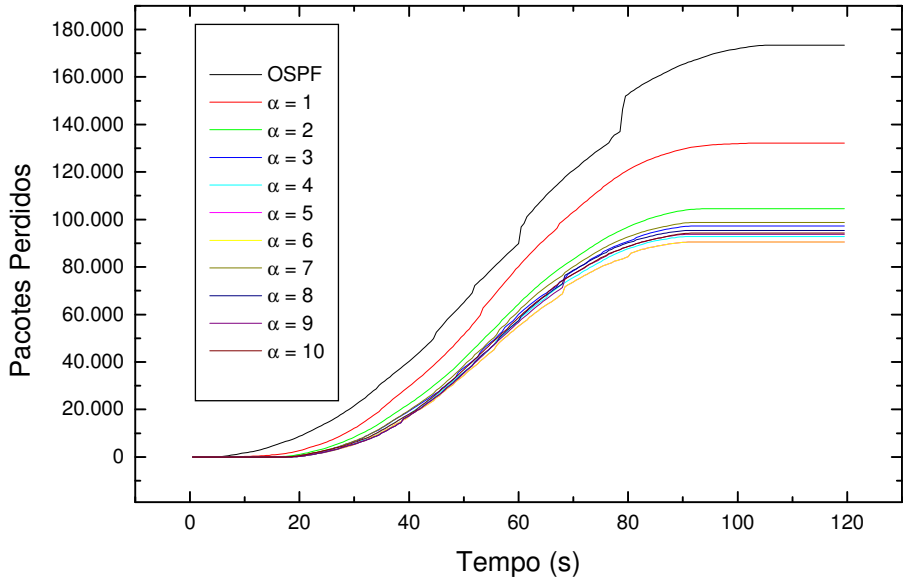

Figura 17 – Número absoluto de pacotes perdidos x tempo para β = 0.

Neste ponto, podemos observar que com a variação do parâmetro β (Figura 16), obtivemos valores melhores do que variando α (Figura 17). Pois enquanto β totalizou 81.000 pacotes descartados, α totalizou 85.000 pacotes descartados, sendo β aproximadamente 5% melhor do que α.Isso ocorre porque β faz com que haja uma distribuição melhor do tráfego, já que β penaliza os custos de todos os enlaces ligados a um nó, evitando o uso das áreas mais ocupadas da rede.

5.3.2 Número de pacotes recebidos

Consideramos que o número de pacotes recebidos é o número de pacotes que efetivamente atingem seu destino. Contabilizamos o número total de pacotes recebidos em todos os nós da rede em períodos de 0,5s, iniciando a contabilização no princípio do programa e terminando no período de 120s quando o programa se encerra.

A Figura 18 mostra o número absoluto de pacotes em função do Tempo para OSPF e diferentes valores de β e α = 0. Podemos observar que o máximo de pacotes recebidos se deu com β = 10. Enquanto com OSPF recebemos um total de 118.000 pacotes, com β = 10 recebemos 202.000 pacotes, demonstrando uma melhora de 71%.

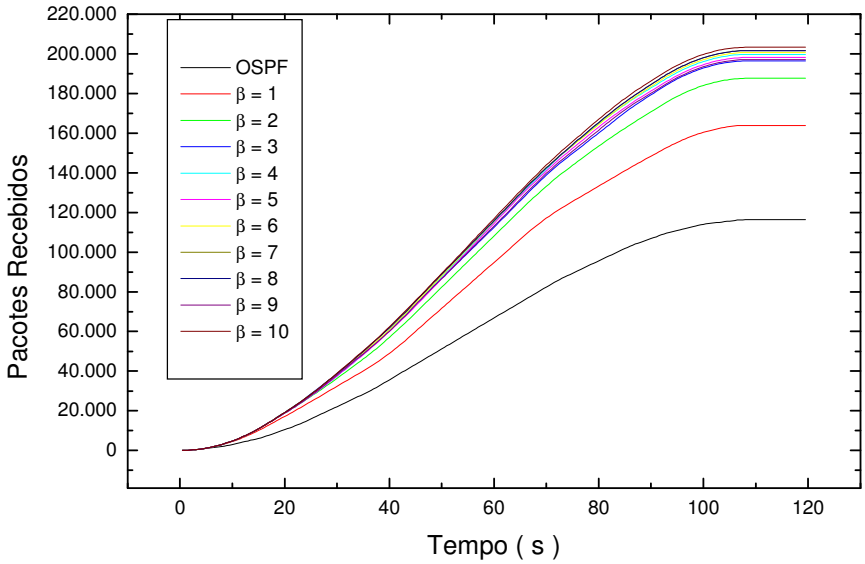

Figura 18 – Pacotes recebidos x tempo para α = 0.

A Figura 19 mostra o número absoluto de pacotes recebidos em função do tempo para OSPF e diferentes valores de α. Podemos observar que o maior número de pacotes recebidos foi conseguido com α = 6. Enquanto com OSPF recebemos 118.000 pacotes, com α = 6 recebemos 200.000 pacotes, resultando em uma melhora de 69,5%.

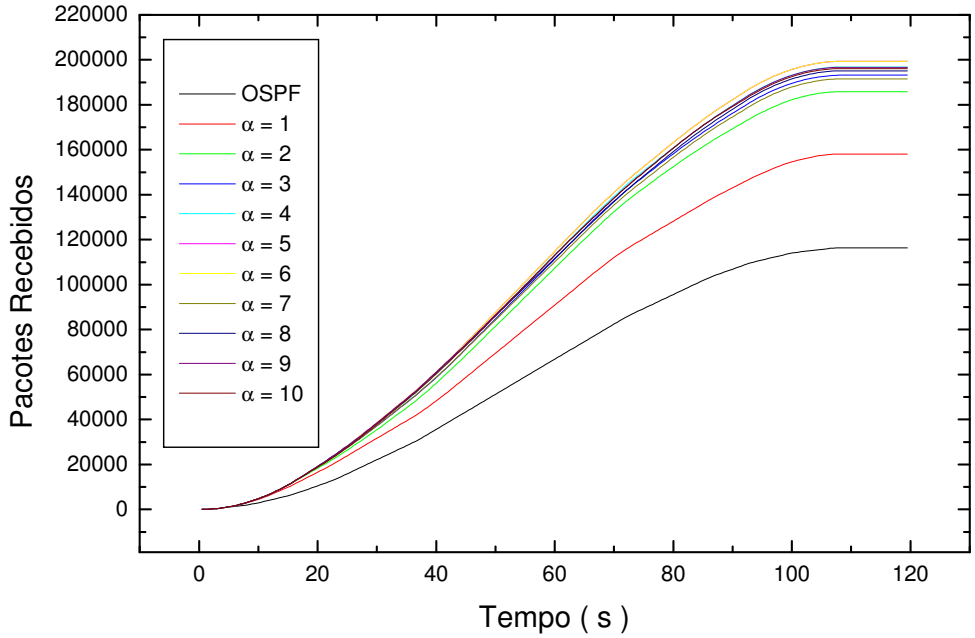

Figura 19 – Pacotes recebidos x tempo para β = 0

5.3.3 Percentual de pacotes descartados

Nesta sub-seção, consideramos o número de pacotes descartados e o número de pacotes recebidos em todos os nós da rede, sendo que, contabilizamos o percentual de pacotes descartados a cada ciclo de 0,5s de simulação, desde o início até o período de 120s, quando o programa é encerrado. Traçamos com esses valores gráficos em 3D, mostrados nas figuras 20 e 21, que demonstram o percentual de pacotes perdidos com um valor fixo de α e variando o valor de β de 1 a 10 sendo que β = 0 corresponde aos valores OBTIDOS para o caso com OSPF.

A Figura 20 mostra o gráfico de perda de pacotes para α = 0 e diferentes valores de β. Nessa figura podemos observar que a perda de pacotes para o OSPF estabilizou em aproximadamente 60% nos períodos entre 80 e 120

segundos e que teve um mínimo de aproximadamente 29% para β = 6 nesse mesmo período.

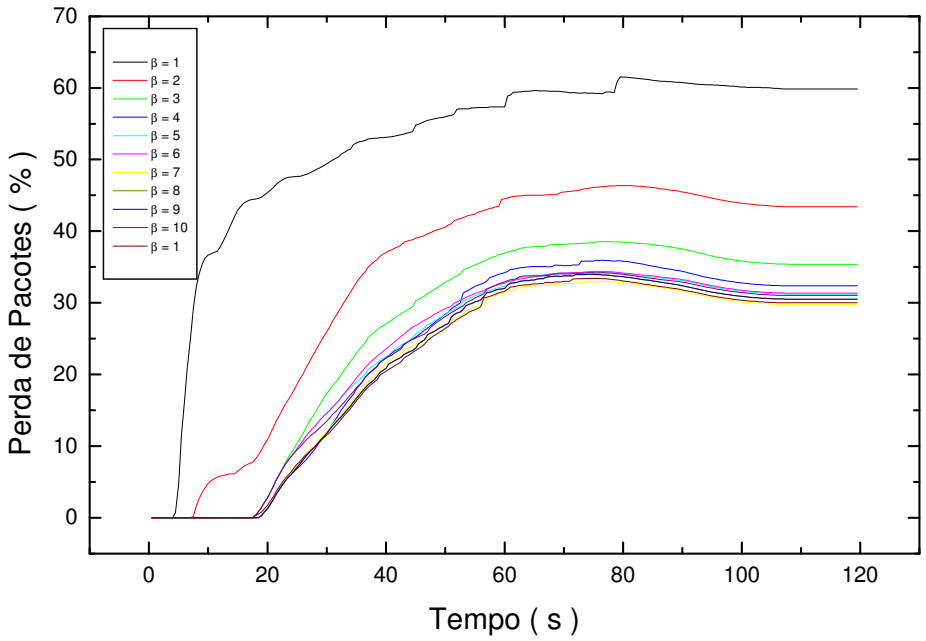

Figura 20 - Percentual de pacotes perdidos para α = 0 e diferentes valores de β

Na figura 21 temos representada a perda de pacotes para α = 10 e diferentes valores de β. Nesta figura, vemos que desde β = 1 até β = 10 os resultados foram praticamente estáveis sendo que, para β =1 mostrou descartes de aproximadamente 34% para o período de 120s enquanto que para β = 10 ocorreu aproximadamente 29% de descartes, com uma variação de apenas 5%.

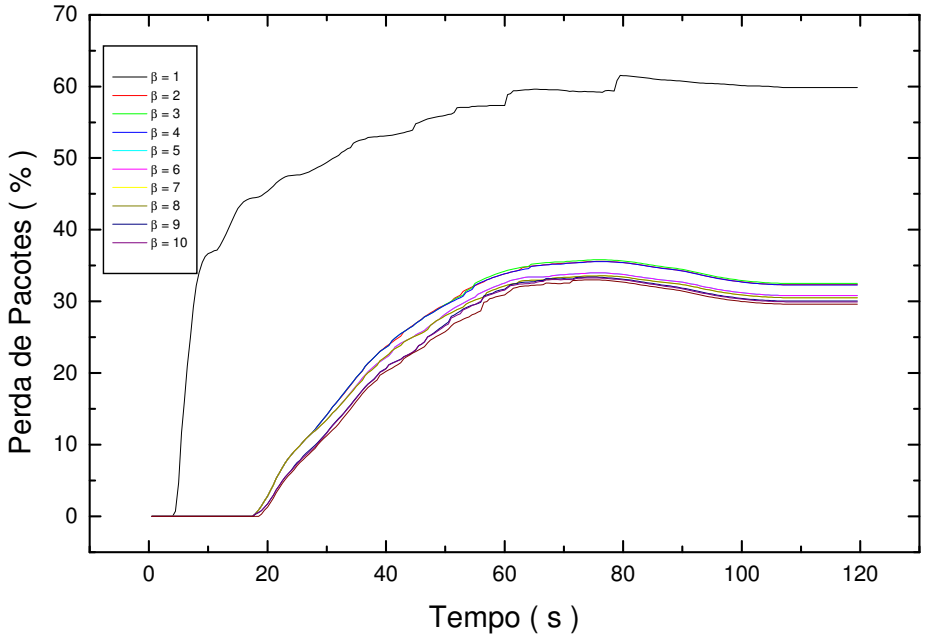

Figura 21 – Percentual de pacotes perdidos para α = 10 e diferentes valores de β

5.3.4 Perda de pacotes em função de α e β

A fim de confrontar a atuação dos parâmetros testados α e β, executamos o programa variando α e β com valores inteiros de 0 a 11.

A Figura 22 mostra um gráfico de percentual de perda de pacotes em função de α e β para o momento de 79,5 segundos. Escolhemos esse momento de 79,5s por ser o momento de maior perda percentual do OSPF. Nesse gráfico podemos observar a perda de pacotes para OSPF de aproximadamente 62% (ponto 0 x 0), e uma queda substancial usando o algoritmo proposto, sendo que o percentual de descartes praticamente se estabiliza para combinações de maiores valores de α e β. Demarcamos no gráfico duas áreas, a primeira de cor azul tem os valores todos compreendidos dentro de uma variação de ± 1% em torno do valor médio de 34%. A segunda área, na cor violeta, é um quadrado delimitado entre α = β = 5 e α = β = 10. Escolhemos essa área por estar dentro

da variação de 1% em relação ao valor médio de 34% e por não estar próxima da borda da área azul, onde, com pequenas variações nos parâmetros entramos na área amarela, onde há variações muito grandes, maiores que 5%, no percentual de perda de pacotes.

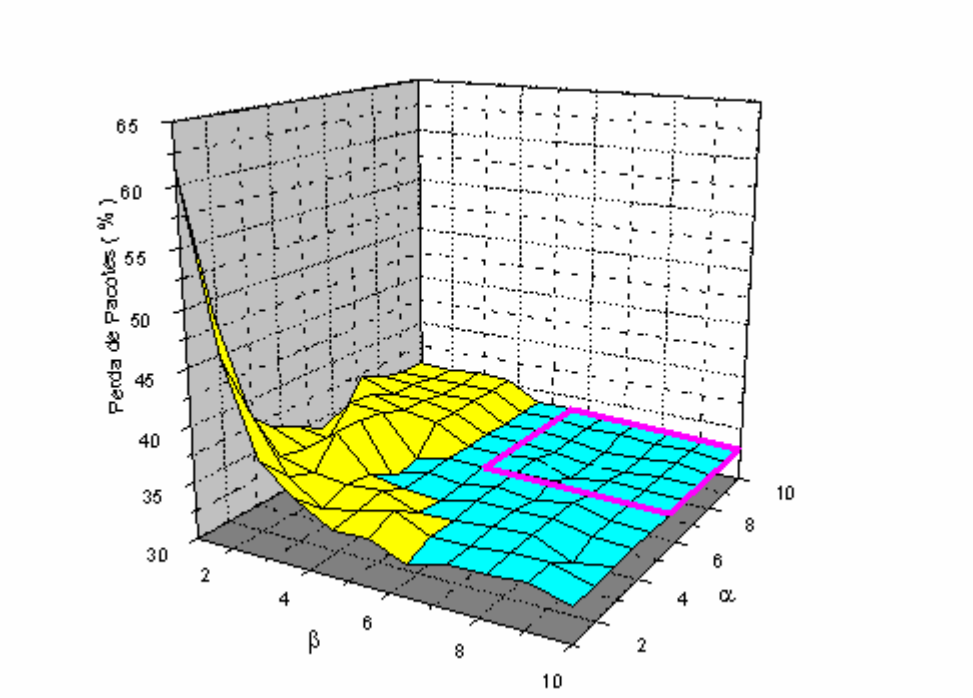

Figura 22 – Pacotes Perdidos em função de α e β – Topologia 5 x 5

Nessa mesma figura 22 pode-se observar um quadrado por nós delineado desde α igual a cinco com β igual a cinco até α e β iguais a 10 onde a variação do percentual de pacotes perdidos está dentro da variação de 1% em relação à média, ou seja, é a área mais estável da rede, e onde sugerimos que o operador da rede deva situar seus parâmetros de α e β.

5.3.5 Redistribuição do tráfego

Um ponto importante de nosso trabalho é mostrar que com a engenharia de tráfego proposta conseguimos redistribuir os fluxos de dados nos enlaces e nós da rede. Para comprovar que isso esteja acontecendo analisamos os dados

do número de pacotes enviados e recebidos nos enlaces de alguns nós da rede. Elegemos os nós 1 e 24 por estarem respectivamente na entrada da rede MPLS, e na saída da rede IP.

Nas Figuras 23 e 24 vemos o tráfego do nó 1 para OSPF e para $\alpha = \beta = 8$. Na figura 23 vemos que o tráfego desse nó foi todo transmitido pelo enlace 1-5, resultando em um total de 32500 pacotes transmitidos. Na Figura 24 vemos que todos os enlaces passaram a ter tráfego. O enlace 1-5 transmitiu aproximadamente 31000 pacotes enquanto, 1-2 transmitiu cerca de 28000, 1-21 transmitiu 27000 pacotes e finalmente 1-6 transmitiu aproximadamente 26000 pacotes.

Nas Figuras 25 e 26 vemos o tráfego do nó 24 para OSPF e para $\alpha = \beta = 8$. Da mesma forma que no caso do nó 1, o tráfego era transmitido apenas pelo enlace 24-8 e passou a ser transmitido por todos enlaces, mostrando claramente que houve redistribuição pelos enlaces ligados ao nó.

Podemos notar por esses gráficos que houve uma significativa melhora no uso dos enlaces já que seguindo-se as rotas estabelecidas pelo OSPF, tínhamos tráfego apenas em um enlace, 1 – 5 no nó 1 e 24 – 4 no nó 24 e depois, usando a engenharia de tráfego proposta passamos a ter tráfego em todos os enlaces ligados a esses nós.

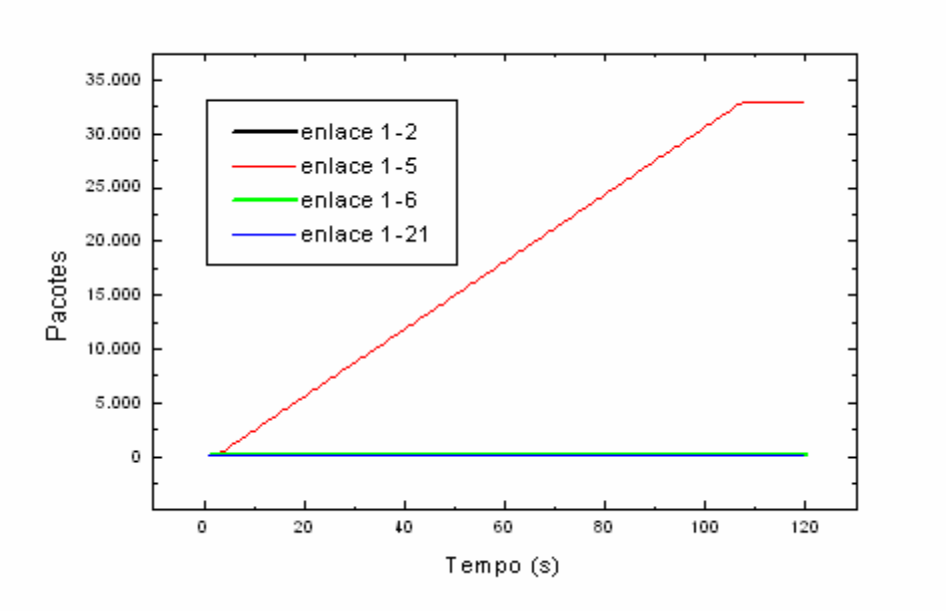

Figura 23 – Tráfego nos enlaces ligados ao nó 1 para OSPF.

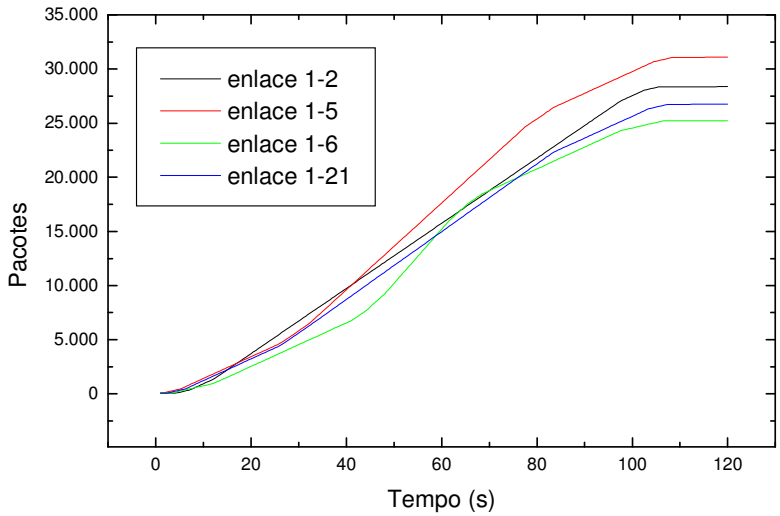

Figura 24 – Tráfego nos enlaces ligados ao nó 1 para $\alpha = \beta = 8$.

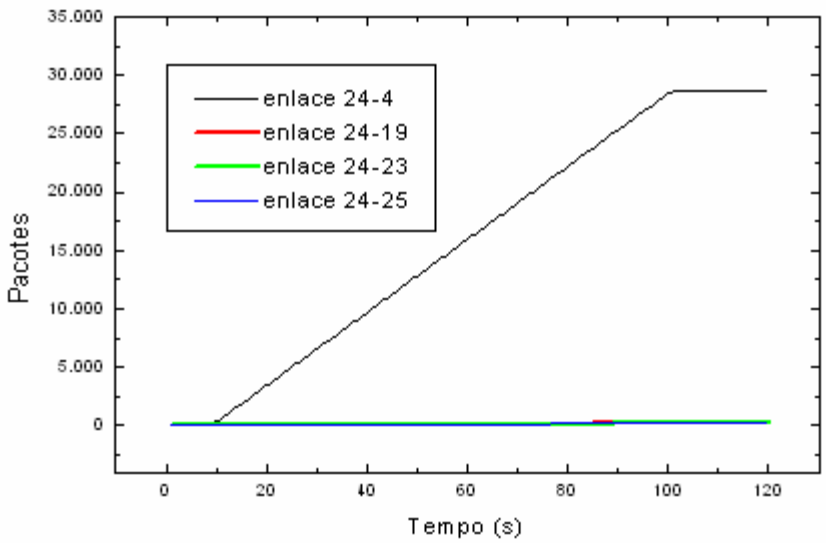

Figura 25 Tráfego nos enlaces ligados ao nó 24 para OSPF.

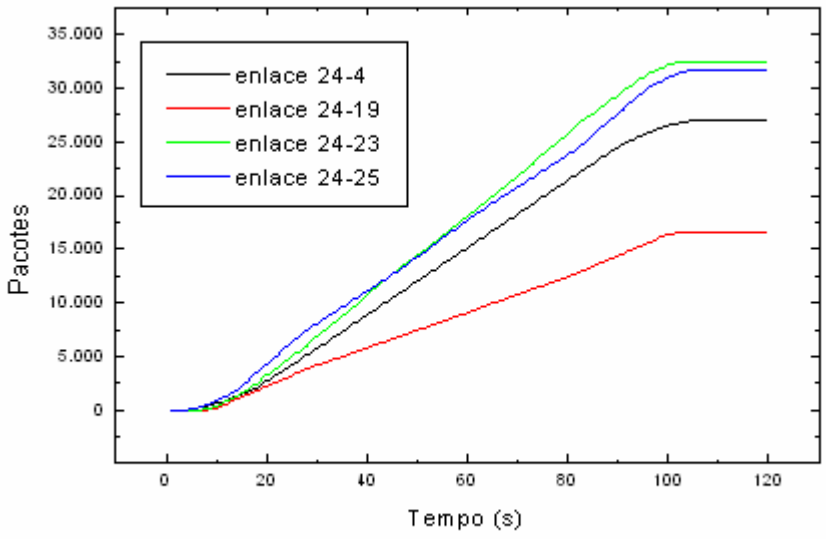

Figura 26 – Tráfego nos enlaces ligados ao nó 24 para $\alpha = \beta = 8$.

5.4 Resultados das Simulações Rede 5 x 5 com distribuição dos servidores modificada

Conforme explicamos anteriormente, a distribuição dos servidores foi modificada a fim de testar a robustez do algoritmo desenvolvido. Vemos aqui alguns dos principais resultados obtidos com a distribuição dos servidores modificada.

5.4.1 Percentual de pacotes descartados

Comparamos novamente os valores de percentual de perda de pacotes. Nos gráficos das Figuras 27 e 28 temos respectivamente o percentual de pacotes perdidos para $\alpha = 0$ e para $\alpha = 10$ e o período variando de zero a 120 s.

Na Figura 27 vemos que o mínimo para $\alpha = 0$, foi obtido com $\beta = 4$ e $\beta = 8$, ficando em torno de 27%, ao passo que em OSPF, representado pelo valor zero do eixo de β. Se considerarmos o período de 120s, e tomarmos a média do percentual de descartes, obteremos o valor de 29,47%. Em relação a essa média, temos uma variação de +6,27% o máximo de 35,74% com $\beta = 1$ e variação de -2,4% para o mínimo de 27,11% com $\beta = 4$.

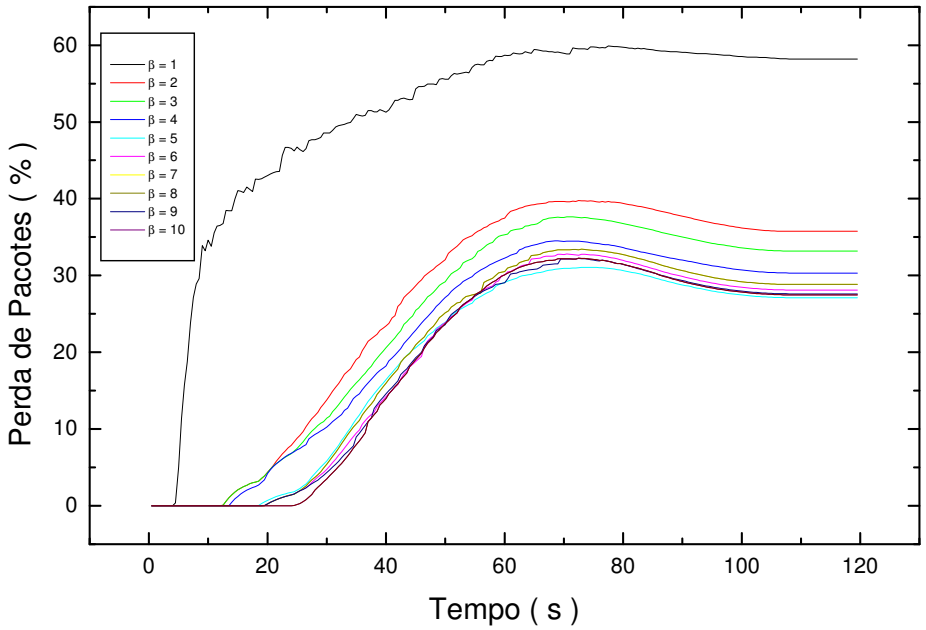

Figura 27 - Percentual de pacotes perdidos para α = 0 e diferentes valores de β

Na Figura 28 observamos que o mínimo para α = 10 foi obtido com β = 3 e β = 10, com valor de aproximadamente 25%. Para o período de 120s, podemos observar também que todos os valores estão bastante próximos, dentro de uma variação de aproximadamente 1% em relação à média de 28,06%. Isso demonstra que o algoritmo está muito mais estável nessa faixa, mas ainda não temos a visão de todos os parâmetros confrontados.

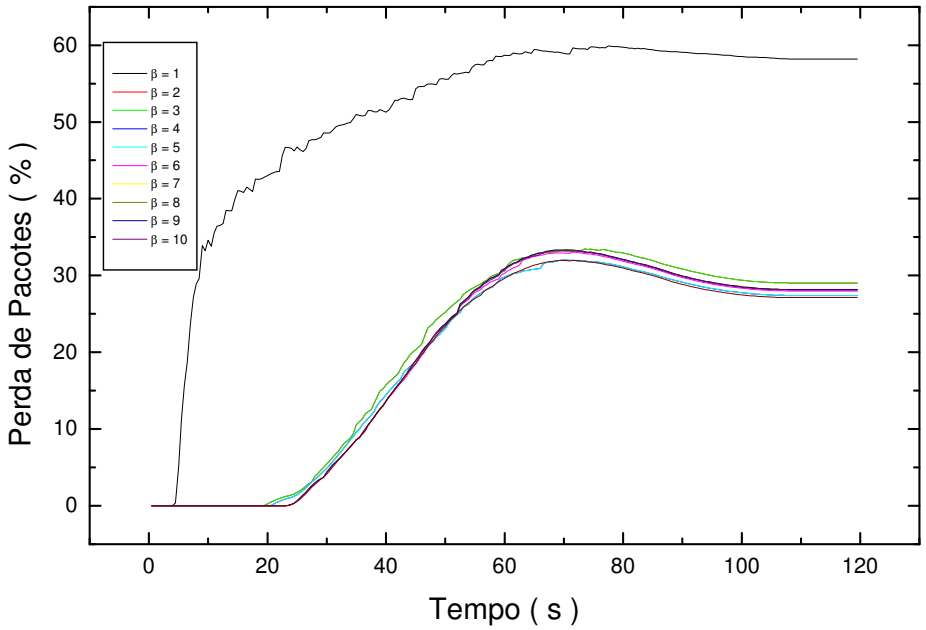

Figura 28 - Percentual de pacotes perdidos para α = 10 e diferentes valores de β

5.4.2 Perda de pacotes em função de α e β

Novamente, confrontamos a atuação dos parâmetros testados α e β, executamos o programa combinando esses parâmetros em valores inteiros de 0 a 11 e salvamos os resultados. Para a rede 5 x 5 com a distribuição dos servidores modificada o momento de maior número de descartes para o OSPF foi 43,5 s, fizemos então o gráfico da Figura 29 combinando valores de α e β e apresentando o percentual de perda de pacotes para cada combinação nesse instante de 43,5s. Demarcamos em azul a área onde os valores estão dentro de uma variação de ± 1% de uma média de 17,8%. Nessa área escolhemos o quadrado delimitado pelas coordenadas α = β = 5 até α = β = 10 como a área sugerida para o operador da rede situar α e β devido a que essa área além de

estar na mesma variação de ± 1%, não tem vizinhança com áreas de maiores variações, evitando assim instabilidades.

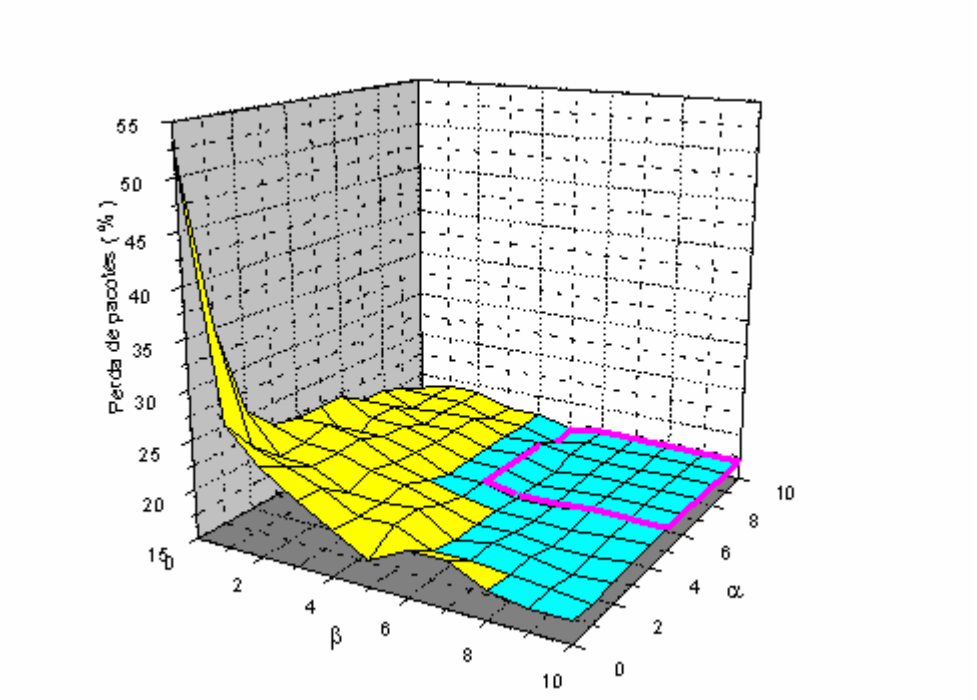

Figura 29 – Pacotes Perdidos (%) em função de α e β para t = 43,5 s

5.4.3 Redistribuição do tráfego

Testamos a redistribuição do tráfego novamente para os nós 1 e 24. Os gráficos das Figuras 30 e 31 mostram o tráfego nos enlaces ligados ao nó 1 enquanto que os gráficos das Figuras 32 e 33 mostram o tráfego nos enlaces ligados ao nó 24. Nos enlaces do nó 1 notamos que a redistribuição não foi significativa, já nos enlaces ligados ao nó 24, na Figura 32, notamos que com o OSPF havia tráfego majoritariamente no enlace 24 – 4 e pouquíssimo tráfego nos enlaces 24 – 23 e 24 – 25. Já com a solução proposta, com α = β = 8 notamos que houve mais incidência de tráfego nos enlaces 24 – 4, 24 – 19 e 24 – 23, sendo que o enlace 24 – 25 praticamente manteve o mesmo tráfego que houve com o OSPF. O tráfego nos enlaces ligados ao nó 1já se mostravam mais

distribuídos com o OSPF, se comparado com o tráfego das simulações anteriores da rede 5 x 5 sem modificar a posição dos servidores, porque, ao modificar a disposição dos mesmos, já redistribuímos o tráfego. Mesmo nessa condição, o algoritmo proposto se mostra mais eficiente do que o OSPF.

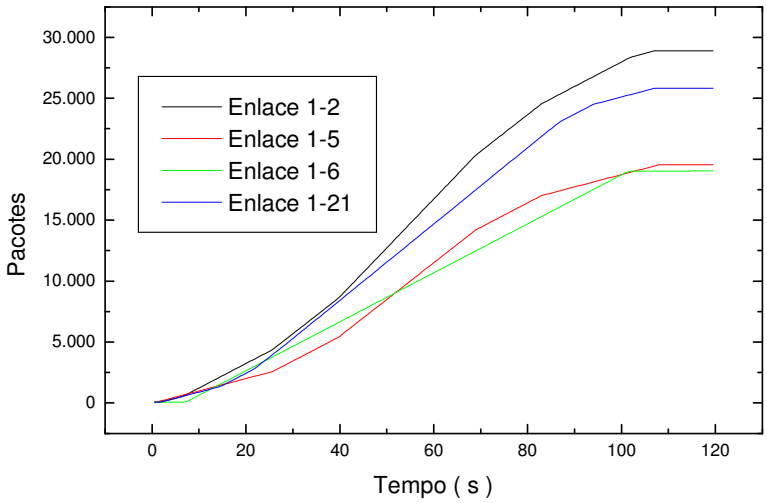

Figura 30 – Tráfego nos enlaces do nó 1 com OSPF

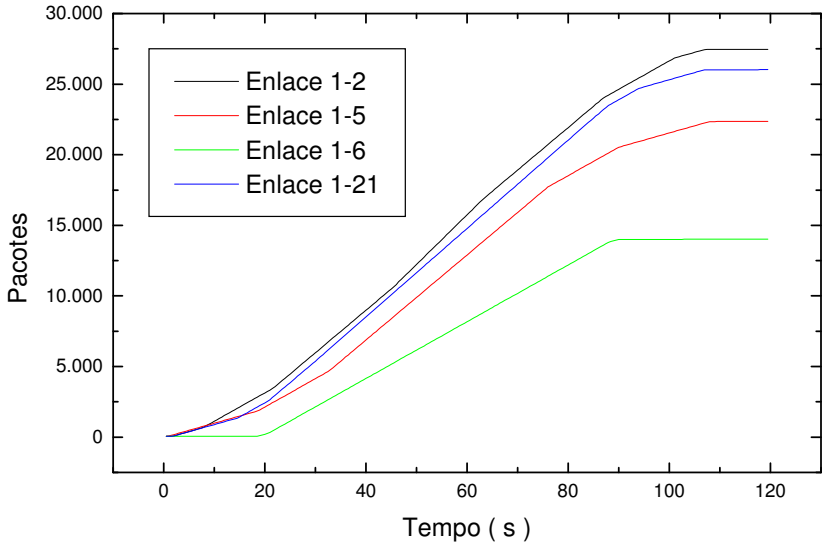

Figura 31 – Tráfego nos enlaces do nó 1 com $\alpha = \beta = 8$

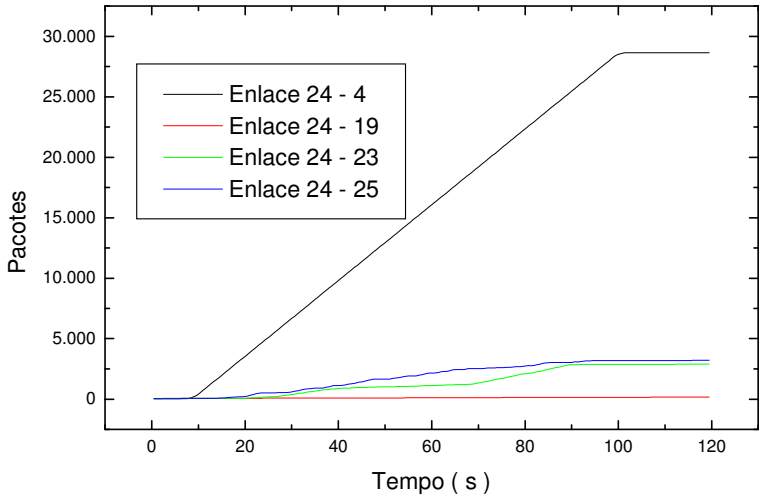

Figura 32 - Tráfego nos enlaces do nó 24 para OSPF

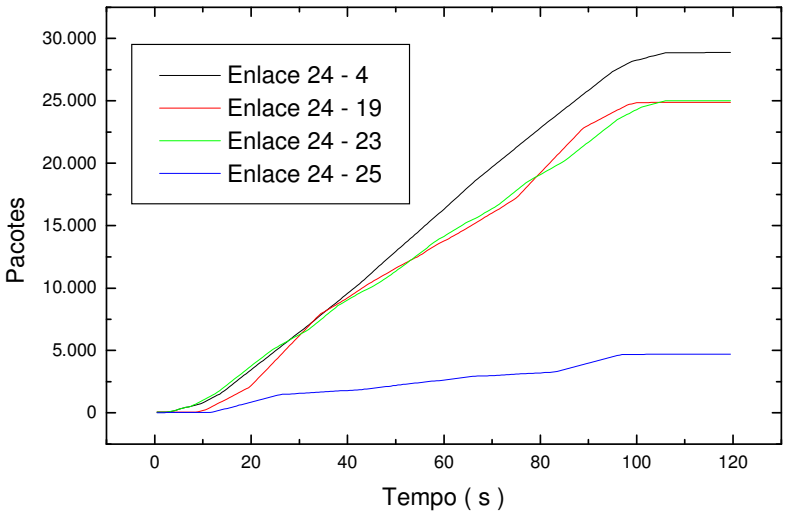

Figura 33 - Tráfego nos enlaces do nó 24 para $\alpha = \beta = 8$

6 CONCLUSÕES

Concluímos através de nosso trabalho que a tarefa de engenharia de tráfego pode ser realizada fim a fim em um ambiente de redes de distintos protocolos. A distribuição do tráfego em uma rede *Manhattan Street* foi conseguida e verificada através de simulações para os enlaces ligados aos roteadores das redes.

Nosso principal resultado foi conseguir reduzir o percentual de pacotes perdidos e aumento do número de pacotes recebidos em redes com protocolos distintos em aproximadamente 50% comparado com o OSPF e com diferentes distribuições das máquinas conectadas às redes, além de termos conseguido usar dois parâmetros diferentes para a engenharia de redes proposta.

Além dos trabalhos realizados, seria interessante que trabalhos futuros considerassem:

- A questão de manutenção de enlaces, queda e recuperação desses enlaces.
- A possibilidade de usar enlaces paralelos entre os nós da rede.
- Modificar o antigo código ATM do NS versão 1 para que funcione no NS-2 e testar o algoritmo em redes com os protocolos distintos ATM e MPLS.

A implementação prática desse trabalho, utilizaria uma estação de gerência com *scripts* que atualizariam os valores nas tabelas de custos dos enlaces. Essas tabelas seriam atualizadas a partir de valores de ocupação dos enlaces e *backplane* nos elementos da rede obtidos através do SNMP. Fizemos alguns trabalhos práticos inicialmente em uma rede de laboratório, os quais são mostrados no Apêndice A.

7 REFERÊNCIAS

AF-AIC-0178-001, ATM Forum, *ATM-MPLS Network Interworking, Version 2.0, af-aic-0178-001*, 2003.

AF-MPOA-014-000, *ATM Forum- Multi-protocol Over ATM Specification, Version 1.1, af-mpoa-014-000*, 1999.

AF-PNNI-0055.001, *ATM FORUM- Private Network Network Specification version 1.1, MFA Forum*, 2002.

AWDUCHE *et al*, *RSVP-TE: Extensions to RSVP for LSP Tunnels* , McLean: IETF, 2001.

BLACK, Uyless, MPLS and Label Switching Networks, Second Edition, Upper Saddle River: Prentice Hall, 2002. 314p.

BLACK, Uyless, *IP Routing Protocols – RIP, OSPF, BGP, PNNI & Cisco Routing Protocols*, Upper Saddle River: Prentice Hall, 2000. 287p.

BURIOL, Luciana Salete. *Roteamento do tráfego na internet: algoritmos para projeto e operação de redes com protocolo OSPF*, Unicamp: Campinas, SP, 2003.

CONTE, Marco. *Dynamic Routing in Broadband Networks,* Boston: Kluwer Academic Publishers, 2003. 214p.

CORMEN, T. *et al*. *Algoritmos: teoria e prática*, 4ª edição, Rio de Janeiro : Elsevier, 2002. 916p.

DAVIE B., J. Lawrence, K. McCloghrie, E. Rosen, G. Swallow, Y. Rekhter, P. Doolan, *RFC-3035 - MPLS using LDP and ATM VC Switch*, 2001.

DIJKSTRA, E. W. *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik, 1959. pp 269-271

FRF-5, *Frame Relay Forum, Frame Relay/ATM PVC Network Interworking Implementation FRF.5*, 1994.

FRF-8,2 *Frame Relay Forum, Frame Relay/ATM PVC Service Interworking Implementation Agreement FRF.8.2*, 2004.

KUMAR, Balaji, *Broadband Communications*, New York: McGrall-Hill, 2000. 624p.

KYAS, O., Crawford, G. *ATM Networks.* New Jersey: Prentice Hall, 2002. 526p.

LOTITO, Alberto. *A new approach on using Dijkstra on multiprotocols networks,* Santa Rita do Sapucaí, IWT-2007.

McDYSAN, D.; Spohn, D. *ATM Theory and Applications,* New York: McGrall Hill, 2000. 1011p.

McDYSAN, David. *QoS & Traffic Management in IP & ATM Networks*, NY, McGrall Hill, 2000. 456p.

MFA-FORUM-10.0.0, *ATM and Frame Relay to MPLS Control Plane Interworking: Client-Server*, 2006.

MFA-FORUM-18.0.0, *Soft Permanent Virtual Circuit Interworking between MPLS Pseudowires and ATM*, 2007.

NADEAU, T.; Hedge S., *RFC-4368 Multiprotocol Label Switch (MPLS) Label-Controlled Asynchronous Transfer Mode (ATM) and Frame-Relay Management Interface Definition, Boxboro*, MA, 2006.

OSBORNE, Eric, *Engenharia de Tráfego com OSPF: projeto, configuração e gerenciamento do MPLS para otimização de desempenho*, Rio de Janeiro: Campus, 2003. 614p.

PEPELNJAK, Ivan. Guichard, Jim*. MPLS and VPN Architectures, CCIP Edition*, 2a edição, Indianapolis: Cisco Press, 2003. 481p.

RECKTER *et al*, *RFC-4271 A Border gateway Protocol BGP-4*, Ann Arbor: IETF, 2006.

RESENDE, R. A. et al. *A New adaptive Traffic Shortest Path Routing for IP Architecture*, Innsbruck: Applied Informatics, 2002.

ROSEN *et al, RFC-3032 MPLS Label Stack Encoding*, Chelmsford: IETF, 2001.

STALLINGS, William, *Data and Computer Communications*, Seventh Edition, New Jersey: Prentice Hall, 2004.

STEWART III, *John W. BGP4 Inter-Domain Routing in the Internet*, Upper Saddle River: Addison-Wesley, 1998. 137p.

THOMAS II, T. M. *OSPF Network Design Solutions*, Indianapolis: CiscoPress, 1998. 776p.

VERDI, Fábio L. *Using Virtualization to Provide Interdomain QoS-enabled Routing*, Journal of Networks (JNW), V.2, 2007. p.23-32.

8 BIBLIOGRAFIA

MATOSO, Maria Cristina. *Orientações para apresentação de trabalhos acadêmicos*. 18ª edição. Campinas : PUC-Campinas, 2007. 43p.

EMEDIATO, Wander. *A Fórmula do Texto*. 1a edição. São Paulo : Geração Editorial, 2004. 295p.

FRANCA, Alexander. *Tcl/Tk Programação Linux*. 1ª edição. Rio de Janeiro: Brasport, 2005. 373p.

SEVERINO, Antônio Joaquim. *Metodologia do trabalho Científico*. 22ª edição. São Paulo : Cortez, 2002. 335p.

APÊNDICE A – TESTES PRELIMINARES EM LABORATÓRIO

Nossos trabalhos começaram com a realização de alguns testes em laboratório, realizados no laboratório da Lucent Technologies em Campinas. Publicamos baseados nesses trabalhos iniciais o artigo *A new approach on using Dijkstra algorithm in multiple protocols networks* (LOTITO, 2007) no congresso IWT2007.

Usamos para esses testes um gerador de tráfego e analisador de protocolos capaz de ser controlado remotamente com uso de *scripts*. Nos *script*s fazíamos as requisições de informação de uso da rede, usando para isso protocolo SNMP, de modo que atualizávamos as tabelas de ocupação dos enlaces. Uma vez atualizada a tabela, calculamos o melhor caminho sobre esses valores e criamos as rotas estáticas nos roteadores da rede, de maneira que os fluxos de dados *unicast* seguirão as rotas criadas.

Na Figura 34 vemos o diagrama da rede usada em nosso teste. Estão representados os roteadores da rede e os links usados. Os roteadores R7, R8 e R9 com capacidades de 1, 1 e 2 Gbps respectivamente, são o gargalo da rede.

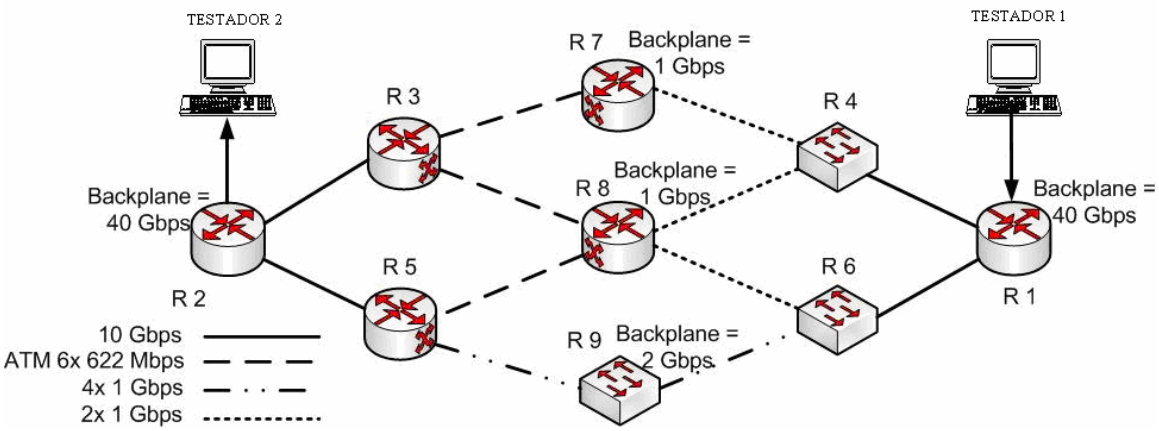

Figura 34 – Diagrama da rede de teste

Ainda na Figura 34, temos o gerador de tráfego representado pelo Testador 1 e o receptor do tráfego representado pelo Testador 2, que são, na verdade, duas placas do mesmo equipamento de teste, esse equipamento permite numerar os pacotes na origem e verificar sua perda no destino. As conexões ATM dessa figura terminam em IP usando o *interwork* descrito na RFC-1483 e usou-se OSPF em toda a rede para os Roteadores conhecerem a rede como um todo.

Geramos dez fluxos de dados de 50 Mbps a cada 50 segundos. Começamos com 1,5 Gbps até um limite de 8 Gbps. De 14 minutos em diante, não aplicamos mais crescimento de tráfego. Os pacotes recebidos pelo testador 2 são contados e sua seqüência verificada. Pacotes perdidos foram registrados para contabilizar os resultados.

Na figura 35, podemos ver os resultados do OSPF comparados ao método proposto onde usamos $\alpha = \beta = 10$ como parâmetros para a engenharia de tráfego da rede,conforme artigo publicado no IWT-2007 (LOTITO2007). Os resultados são acumulativos e mostram o percentual de pacotes perdidos com o passar do tempo. Podemos observar que, em geral, a perda de pacotes com o método proposto foi a metade do que com o OSPF. Isso se deve ao fato de que, com o algoritmo proposto, as novas conexões tendem a usar caminhos mais livres, diminuindo a perda de pacotes.

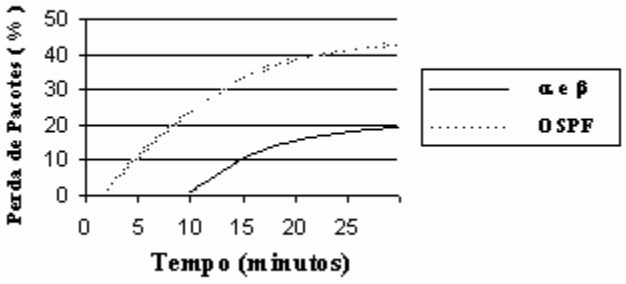

Figura 35 – Perda de Pacotes para OSPF e para o algoritmo proposto

Concluímos com este teste em laboratório que a redistribuição de tráfego funcionou e que houve melhoria de mais de 50% no número de pacotes perdidos se compararmos com o OSPF.

APÊNDICE B – CÓDIGOS FONTE

Apresentamos aquí o código fonte utilizado em nossas simulações. Para executálo deve-se chamar “ns 25mistaB.tcl <v1> <v2>” onde v1 é o α e v2 corresponde ao β.

```
#=====================================================#
#  Simulacao MPLS sobre rede em arquitetura Manhattan #
#  Alberto Lotito                                     #
#  Mestrado PUC - Campinas                            #
#=====================================================#

#set simulator instance
set ::ns [new Simulator]

# Se não houver 2 argumentos, sai do programa emitindo mensagem de erro
if {[llength $argv] != 2} {
        puts "Missing parameters... link_bw tcp_window"
        exit 1
}

# Atribui os argumentos a ALFA, BETA, 'y' e 'e'
# ALFA e BETA são parametros e y/e nomearão arquivos
set ::y [lindex $argv 0]
set ::e [lindex $argv 1]
set ::ALFA [lindex $argv 0]
set ::BETA [lindex $argv 1]

# Nomeia os arquivos texto onde serão depositados os dados coletados
# out são dados totalizados de descartes, pacotes recebidos e
percentual de descartes
# des
set ::nx [open y$y/y$y%$e%out.dat w]
set ::nt [open y$y/d$y%$e%des.dat w]
set ::nz [open y$y/a$y%$e%atr.dat w]

# define o arquivo nf a ser usado pelo Network Animator (NAM)
#set nf [open mpls-out-mista.nam w]

#Define cores diferentes a serem usadas por fluxos de dados
$ns color 0 purple
$ns color 1 blue
$ns color 2 red
$ns color 3 green

# Atribui o arquivo nf ao trace do Network Animator (NAM)
# Caso não seja usado, deve ser comentado para tornar
# a simulação mais rápida
#$ns namtrace-all $nf

# Define uma rotina de finalização
# Fecha os arquivos usados e chama o NAM. Caso esteja usando NAM,
```

```
# deve retirar o comentário da linha 'exec nam'
proc finish {} {
      global  ns nt nf nx nz
      $ns flush-trace
      #close $nf
      close $nx
      close $nt

      #exec nam mpls-out-mista.nam &
      #exec xgraph out.tr &
      exit 0
}


# Faz com que o algoritmo de Bellman-Ford seja usado (LS= Link State)
# Como estratégia de roteamento dinâmico
# Deve ser retirado o comentário para usar essa opção de roteamento
#$ns rtproto LS
#$ns rtproto DV

#=================================================================
# Definição dos nós, o MPLS é definido entre esses dois comandos
# "$ns node-config -MPLS ON" ou "$ns node-config - MPLS OFF"
#

# 9 nós MPLS a partir daqui, MPLS ON
$ns node-config -MPLS ON

# Cria o Label Switch Router 0
set ::LSR(0)   [$ns mpls-node]
# faz com que a côr do nó MPLS seja rôxa, para o NAM
$LSR(0) color "purple"

set ::LSR(1)   [$ns mpls-node]
$LSR(1) color "purple"
set ::LSR(2)   [$ns mpls-node]
$LSR(2) color "purple"
set ::LSR(3)   [$ns mpls-node]
$LSR(3) color "purple"
set ::LSR(4)   [$ns mpls-node]
$LSR(4) color "purple"
set ::LSR(5)   [$ns mpls-node]
$LSR(5) color "purple"
set ::LSR(6)   [$ns mpls-node]
$LSR(6) color "purple"
set ::LSR(7)   [$ns mpls-node]
$LSR(7) color "purple"
set ::LSR(8)   [$ns mpls-node]
$LSR(8) color "purple"
set ::LSR(9)   [$ns mpls-node]
$LSR(9) color "purple"
set ::LSR(10)   [$ns mpls-node]
$LSR(10) color "purple"
set ::LSR(11)   [$ns mpls-node]
$LSR(11) color "purple"
set ::LSR(12)   [$ns mpls-node]
$LSR(12) color "purple"
```

```
set ::LSR(13)   [$ns mpls-node]
$LSR(13) color "purple"
set ::LSR(14)   [$ns mpls-node]
$LSR(14) color "purple"
set ::LSR(15)   [$ns mpls-node]
$LSR(15) color "purple"

#set LSR(1)6   [$ns mpls-node]
#$LSR(1)6 color "purple"
#set LSR(1)7   [$ns mpls-node]
#$LSR(1)7 color "purple"
#set LSR(1)8   [$ns mpls-node]
#$LSR(1)8 color "purple"
#set LSR(1)9   [$ns mpls-node]
#$LSR(1)9 color "purple"
#set LSR(2)0   [$ns mpls-node]
#$LSR(2)0 color "purple"
#set LSR(2)1   [$ns mpls-node]
#$LSR(2)1 color "purple"
#set LSR(2)2   [$ns mpls-node]
#$LSR(2)2 color "purple"
#set LSR(2)3   [$ns mpls-node]
#$LSR(2)3 color "purple"
#set LSR(2)4   [$ns mpls-node]
#$LSR(2)4 color "purple"
#set LSR(2)5   [$ns mpls-node]
#$LSR(2)5 color "purple"

#termina de configurar o MPLS
$ns node-config -MPLS OFF

#Cria os nós IP usados no CORE da rede mista
for {set i 16} {$i < 26} {incr i} {
      set ::n_($i) [$ns node]
      $n_($i) color "green"
}

#Cria os nós IP usados apenas para manter a sequencia numerica
for {set i 26} {$i < 50} {incr i} {
      set ::n_($i) [$ns node]
      $n_($i) color "red"
}

#============================================================
# Cria os hosts 50 a 161
for {set f 50} {$f < 162} {incr f} {
      set ::n_($f) [$ns node]
      $n_($f) color "blue"
}
#============================================================


#============================================================
# cria os links entre os LSR          B/W  Delay  Queue
$ns duplex-link $LSR(1)   $LSR(2)     1Mb  1ms    DropTail
$ns duplex-link $LSR(1)   $LSR(5)     1Mb  1ms    DropTail
$ns duplex-link $LSR(1)   $LSR(6)     1Mb  1ms    DropTail
```

```
$ns duplex-link $LSR(1)   $n_(21)     1Mb  1ms    DropTail

$ns duplex-link $LSR(2)   $LSR(3)     1Mb  1ms    DropTail
$ns duplex-link $LSR(2)   $LSR(7)     1Mb  1ms    DropTail
$ns duplex-link $LSR(2)   $n_(22)     1Mb  1ms    DropTail

$ns duplex-link $LSR(3)   $LSR(4)     1Mb  1ms    DropTail
$ns duplex-link $LSR(3)   $LSR(8)     1Mb  1ms    DropTail
$ns duplex-link $LSR(3)   $n_(23)     1Mb  1ms    DropTail

$ns duplex-link $LSR(4)   $LSR(5)     1Mb  1ms    DropTail
$ns duplex-link $LSR(4)   $LSR(9)     1Mb  1ms    DropTail
$ns duplex-link $LSR(4)   $n_(24)     1Mb  1ms    DropTail

$ns duplex-link $LSR(5)   $LSR(10)     1Mb  1ms    DropTail
$ns duplex-link $LSR(5)   $n_(25)     1Mb  1ms    DropTail

$ns duplex-link $LSR(6)   $LSR(7)     1Mb  1ms    DropTail
$ns duplex-link $LSR(6)   $LSR(10)     1Mb  1ms    DropTail
$ns duplex-link $LSR(6)   $LSR(11)     1Mb  1ms    DropTail

$ns duplex-link $LSR(7)   $LSR(8)     1Mb  1ms    DropTail
$ns duplex-link $LSR(7)   $LSR(12)     1Mb  1ms    DropTail

$ns duplex-link $LSR(8)   $LSR(9)     1Mb  1ms    DropTail
$ns duplex-link $LSR(8)   $LSR(13)     1Mb  1ms    DropTail

$ns duplex-link $LSR(9)   $LSR(10)     1Mb  1ms    DropTail
$ns duplex-link $LSR(9)   $LSR(14)     1Mb  1ms    DropTail

$ns duplex-link $LSR(10)   $LSR(15)     1Mb  1ms    DropTail

$ns duplex-link $LSR(11)   $LSR(12)     1Mb  1ms    DropTail
$ns duplex-link $LSR(11)   $LSR(15)     1Mb  1ms    DropTail
$ns duplex-link $LSR(11)   $n_(16)     1Mb  1ms    DropTail

$ns duplex-link $LSR(12)   $LSR(13)     1Mb  1ms    DropTail
$ns duplex-link $LSR(12)   $n_(17)     1Mb  1ms    DropTail

$ns duplex-link $LSR(13)   $LSR(14)     1Mb  1ms    DropTail
$ns duplex-link $LSR(13)   $n_(18)     1Mb  1ms    DropTail

$ns duplex-link $LSR(14)   $LSR(15)     1Mb  1ms    DropTail
$ns duplex-link $LSR(14)   $n_(19)     1Mb  1ms    DropTail

$ns duplex-link $LSR(15)   $n_(20)     1Mb  1ms    DropTail

$ns duplex-link $n_(16)   $n_(17)     1Mb  1ms    DropTail
$ns duplex-link $n_(16)   $n_(20)     1Mb  1ms    DropTail
$ns duplex-link $n_(16)   $n_(21)     1Mb  1ms    DropTail

$ns duplex-link $n_(17)   $n_(18)     1Mb  1ms    DropTail
$ns duplex-link $n_(17)   $n_(22)     1Mb  1ms    DropTail

$ns duplex-link $n_(18)   $n_(19)     1Mb  1ms    DropTail
$ns duplex-link $n_(18)   $n_(23)     1Mb  1ms    DropTail
```

```
$ns duplex-link $n_(19)   $n_(20)     1Mb  1ms    DropTail
$ns duplex-link $n_(19)   $n_(24)     1Mb  1ms    DropTail

$ns duplex-link $n_(20)   $n_(25)     1Mb  1ms    DropTail

$ns duplex-link $n_(21)   $n_(22)     1Mb  1ms    DropTail
$ns duplex-link $n_(21)   $n_(25)     1Mb  1ms    DropTail

$ns duplex-link $n_(22)   $n_(23)     1Mb  1ms    DropTail

$ns duplex-link $n_(23)   $n_(24)     1Mb  1ms    DropTail

$ns duplex-link $n_(24)   $n_(25)     1Mb  1ms    DropTail

#=============================================================

#=============================================================
#cria os enlaces entre o NUCLEO e os nós de concentracao
$ns duplex-link $n_(30)   $LSR(1)     10Mb  1ms    DropTail
$ns duplex-link $n_(31)   $LSR(2)     10Mb  1ms    DropTail
$ns duplex-link $n_(32)   $LSR(3)     10Mb  1ms    DropTail
$ns duplex-link $n_(33)   $LSR(4)     10Mb  1ms    DropTail
$ns duplex-link $n_(34)   $LSR(5)     10Mb  1ms    DropTail
$ns duplex-link $n_(35)   $LSR(10)    10Mb  1ms    DropTail
$ns duplex-link $n_(36)   $LSR(6)     10Mb  1ms    DropTail

#$ns duplex-link $n_(37)   $LSR(6)    10Mb  1ms    DropTail
$ns duplex-link $n_(38)   $n_(25)    10Mb  1ms    DropTail
$ns duplex-link $n_(39)   $n_(24)    10Mb  1ms    DropTail
$ns duplex-link $n_(40)   $n_(23)    10Mb  1ms    DropTail
$ns duplex-link $n_(41)   $n_(22)    10Mb  1ms    DropTail
$ns duplex-link $n_(42)   $n_(21)    10Mb  1ms    DropTail
$ns duplex-link $n_(43)   $n_(16)    10Mb  1ms    DropTail

$ns duplex-link $n_(44)   $n_(20)    10Mb  1ms    DropTail
#=============================================================


#=============================================================
#cria os enlaces entre estacões e nós de concentracao
$ns duplex-link $n_(50)   $n_(30)    1Mb  1ms    DropTail
$ns duplex-link $n_(51)   $n_(30)    1Mb  1ms    DropTail
$ns duplex-link $n_(52)   $n_(30)    1Mb  1ms    DropTail
$ns duplex-link $n_(53)   $n_(30)    1Mb  1ms    DropTail
$ns duplex-link $n_(54)   $n_(30)    1Mb  1ms    DropTail
$ns duplex-link $n_(55)   $n_(30)    1Mb  1ms    DropTail
$ns duplex-link $n_(56)   $n_(30)    1Mb  1ms    DropTail

$ns duplex-link $n_(57)   $n_(31)    1Mb  1ms    DropTail
$ns duplex-link $n_(58)   $n_(31)    1Mb  1ms    DropTail
$ns duplex-link $n_(59)   $n_(31)    1Mb  1ms    DropTail
$ns duplex-link $n_(60)   $n_(31)    1Mb  1ms    DropTail
$ns duplex-link $n_(61)   $n_(31)    1Mb  1ms    DropTail
$ns duplex-link $n_(62)   $n_(31)    1Mb  1ms    DropTail
$ns duplex-link $n_(63)   $n_(31)    1Mb  1ms    DropTail

$ns duplex-link $n_(64)   $n_(32)    1Mb  1ms    DropTail
```

```
$ns duplex-link $n_(65)   $n_(32)    1Mb  1ms    DropTail
$ns duplex-link $n_(66)   $n_(32)    1Mb  1ms    DropTail
$ns duplex-link $n_(67)   $n_(32)    1Mb  1ms    DropTail
$ns duplex-link $n_(68)   $n_(32)    1Mb  1ms    DropTail
$ns duplex-link $n_(69)   $n_(32)    1Mb  1ms    DropTail
$ns duplex-link $n_(70)   $n_(32)    1Mb  1ms    DropTail

$ns duplex-link $n_(71)   $n_(33)    1Mb  1ms    DropTail
$ns duplex-link $n_(72)   $n_(33)    1Mb  1ms    DropTail
$ns duplex-link $n_(73)   $n_(33)    1Mb  1ms    DropTail
$ns duplex-link $n_(74)   $n_(33)    1Mb  1ms    DropTail
$ns duplex-link $n_(75)   $n_(33)    1Mb  1ms    DropTail
$ns duplex-link $n_(76)   $n_(33)    1Mb  1ms    DropTail
$ns duplex-link $n_(77)   $n_(33)    1Mb  1ms    DropTail

$ns duplex-link $n_(78)   $n_(34)    1Mb  1ms    DropTail
$ns duplex-link $n_(79)   $n_(34)    1Mb  1ms    DropTail
$ns duplex-link $n_(80)   $n_(34)    1Mb  1ms    DropTail
$ns duplex-link $n_(81)   $n_(34)    1Mb  1ms    DropTail
$ns duplex-link $n_(82)   $n_(34)    1Mb  1ms    DropTail
$ns duplex-link $n_(83)   $n_(34)    1Mb  1ms    DropTail
$ns duplex-link $n_(84)   $n_(34)    1Mb  1ms    DropTail

$ns duplex-link $n_(85)   $n_(35)    1Mb  1ms    DropTail
$ns duplex-link $n_(86)   $n_(35)    1Mb  1ms    DropTail
$ns duplex-link $n_(87)   $n_(35)    1Mb  1ms    DropTail
$ns duplex-link $n_(88)   $n_(35)    1Mb  1ms    DropTail
$ns duplex-link $n_(89)   $n_(35)    1Mb  1ms    DropTail
$ns duplex-link $n_(90)   $n_(35)    1Mb  1ms    DropTail
$ns duplex-link $n_(91)   $n_(35)    1Mb  1ms    DropTail

$ns duplex-link $n_(92)   $n_(36)    1Mb  1ms    DropTail
$ns duplex-link $n_(93)   $n_(36)    1Mb  1ms    DropTail
$ns duplex-link $n_(94)   $n_(36)    1Mb  1ms    DropTail
$ns duplex-link $n_(95)   $n_(36)    1Mb  1ms    DropTail
$ns duplex-link $n_(96)   $n_(36)    1Mb  1ms    DropTail
$ns duplex-link $n_(97)   $n_(36)    1Mb  1ms    DropTail
$ns duplex-link $n_(98)   $n_(36)    1Mb  1ms    DropTail

$ns duplex-link $n_(99)   $n_(37)    1Mb  1ms    DropTail
$ns duplex-link $n_(100)   $n_(37)    1Mb  1ms    DropTail
$ns duplex-link $n_(101)   $n_(37)    1Mb  1ms    DropTail
$ns duplex-link $n_(102)   $n_(37)    1Mb  1ms    DropTail
$ns duplex-link $n_(103)   $n_(37)    1Mb  1ms    DropTail
$ns duplex-link $n_(104)   $n_(37)    1Mb  1ms    DropTail
$ns duplex-link $n_(105)   $n_(37)    1Mb  1ms    DropTail

$ns duplex-link $n_(106)   $n_(38)    1Mb  1ms    DropTail
$ns duplex-link $n_(107)   $n_(38)    1Mb  1ms    DropTail
$ns duplex-link $n_(108)   $n_(38)    1Mb  1ms    DropTail
$ns duplex-link $n_(109)   $n_(38)    1Mb  1ms    DropTail
$ns duplex-link $n_(110)   $n_(38)    1Mb  1ms    DropTail
$ns duplex-link $n_(111)   $n_(38)    1Mb  1ms    DropTail
$ns duplex-link $n_(112)   $n_(38)    1Mb  1ms    DropTail

$ns duplex-link $n_(113)   $n_(39)    1Mb  1ms    DropTail
$ns duplex-link $n_(114)   $n_(39)    1Mb  1ms    DropTail
```

```
$ns duplex-link $n_(115)   $n_(39)    1Mb  1ms    DropTail
$ns duplex-link $n_(116)   $n_(39)    1Mb  1ms    DropTail
$ns duplex-link $n_(117)   $n_(39)    1Mb  1ms    DropTail
$ns duplex-link $n_(118)   $n_(39)    1Mb  1ms    DropTail
$ns duplex-link $n_(119)   $n_(39)    1Mb  1ms    DropTail

$ns duplex-link $n_(120)   $n_(40)    1Mb  1ms    DropTail
$ns duplex-link $n_(121)   $n_(40)    1Mb  1ms    DropTail
$ns duplex-link $n_(122)   $n_(40)    1Mb  1ms    DropTail
$ns duplex-link $n_(123)   $n_(40)    1Mb  1ms    DropTail
$ns duplex-link $n_(124)   $n_(40)    1Mb  1ms    DropTail
$ns duplex-link $n_(125)   $n_(40)    1Mb  1ms    DropTail
$ns duplex-link $n_(126)   $n_(40)    1Mb  1ms    DropTail

$ns duplex-link $n_(127)   $n_(41)    1Mb  1ms    DropTail
$ns duplex-link $n_(128)   $n_(41)    1Mb  1ms    DropTail
$ns duplex-link $n_(129)   $n_(41)    1Mb  1ms    DropTail
$ns duplex-link $n_(130)   $n_(41)    1Mb  1ms    DropTail
$ns duplex-link $n_(131)   $n_(41)    1Mb  1ms    DropTail
$ns duplex-link $n_(132)   $n_(41)    1Mb  1ms    DropTail
$ns duplex-link $n_(133)   $n_(41)    1Mb  1ms    DropTail

$ns duplex-link $n_(134)   $n_(42)    1Mb  1ms    DropTail
$ns duplex-link $n_(135)   $n_(42)    1Mb  1ms    DropTail
$ns duplex-link $n_(136)   $n_(42)    1Mb  1ms    DropTail
$ns duplex-link $n_(137)   $n_(42)    1Mb  1ms    DropTail
$ns duplex-link $n_(138)   $n_(42)    1Mb  1ms    DropTail
$ns duplex-link $n_(139)   $n_(42)    1Mb  1ms    DropTail
$ns duplex-link $n_(140)   $n_(42)    1Mb  1ms    DropTail

$ns duplex-link $n_(141)   $n_(43)    1Mb  1ms    DropTail
$ns duplex-link $n_(142)   $n_(43)    1Mb  1ms    DropTail
$ns duplex-link $n_(143)   $n_(43)    1Mb  1ms    DropTail
$ns duplex-link $n_(144)   $n_(43)    1Mb  1ms    DropTail
$ns duplex-link $n_(145)   $n_(43)    1Mb  1ms    DropTail
$ns duplex-link $n_(146)   $n_(43)    1Mb  1ms    DropTail
$ns duplex-link $n_(147)   $n_(43)    1Mb  1ms    DropTail

$ns duplex-link $n_(148)   $n_(44)    1Mb  1ms    DropTail
$ns duplex-link $n_(149)   $n_(44)    1Mb  1ms    DropTail
$ns duplex-link $n_(150)   $n_(44)    1Mb  1ms    DropTail
$ns duplex-link $n_(151)   $n_(44)    1Mb  1ms    DropTail
$ns duplex-link $n_(152)   $n_(44)    1Mb  1ms    DropTail
$ns duplex-link $n_(153)   $n_(44)    1Mb  1ms    DropTail
$ns duplex-link $n_(154)   $n_(44)    1Mb  1ms    DropTail

$ns duplex-link $n_(155)   $n_(45)    1Mb  1ms    DropTail
$ns duplex-link $n_(156)   $n_(45)    1Mb  1ms    DropTail
$ns duplex-link $n_(157)   $n_(45)    1Mb  1ms    DropTail
$ns duplex-link $n_(158)   $n_(45)    1Mb  1ms    DropTail
$ns duplex-link $n_(159)   $n_(45)    1Mb  1ms    DropTail
$ns duplex-link $n_(160)   $n_(45)    1Mb  1ms    DropTail
$ns duplex-link $n_(161)   $n_(45)    1Mb  1ms    DropTail
#=========================================================


#=========================================================
```

```
# cria os monitores de links
# os quais monitorarão o tráfego nesses enlaces
set qmon12 [$ns monitor-queue $LSR(1) $LSR(2) stdout]
set qmon21 [$ns monitor-queue $LSR(2) $LSR(1) stdout]
set qmon15 [$ns monitor-queue $LSR(1) $LSR(5) stdout]
set qmon51 [$ns monitor-queue $LSR(5) $LSR(1) stdout]
set qmon16 [$ns monitor-queue $LSR(1) $LSR(6) stdout]
set qmon61 [$ns monitor-queue $LSR(6) $LSR(1) stdout]
set qmon121 [$ns monitor-queue $LSR(1) $n_(21) stdout]
set qmon211 [$ns monitor-queue $n_(21) $LSR(1) stdout]

set qmon23 [$ns monitor-queue $LSR(2) $LSR(3) stdout]
set qmon32 [$ns monitor-queue $LSR(3) $LSR(2) stdout]
set qmon27 [$ns monitor-queue $LSR(2) $LSR(7) stdout]
set qmon72 [$ns monitor-queue $LSR(7) $LSR(2) stdout]
set qmon222 [$ns monitor-queue $LSR(2) $n_(22) stdout]
set qmon222v [$ns monitor-queue $n_(22) $LSR(2) stdout]

set qmon34 [$ns monitor-queue $LSR(3) $LSR(4) stdout]
set qmon43 [$ns monitor-queue $LSR(4) $LSR(3) stdout]
set qmon38 [$ns monitor-queue $LSR(3) $LSR(8) stdout]
set qmon83 [$ns monitor-queue $LSR(8) $LSR(3) stdout]
set qmon323 [$ns monitor-queue $LSR(3) $n_(23) stdout]
set qmon233 [$ns monitor-queue $n_(23) $LSR(3) stdout]

set qmon45 [$ns monitor-queue $LSR(4) $LSR(5) stdout]
set qmon54 [$ns monitor-queue $LSR(5) $LSR(4) stdout]
set qmon49 [$ns monitor-queue $LSR(4) $LSR(9) stdout]
set qmon94 [$ns monitor-queue $LSR(9) $LSR(4) stdout]
set qmon424 [$ns monitor-queue $LSR(4) $n_(24) stdout]
set qmon244 [$ns monitor-queue $n_(24) $LSR(4) stdout]

set qmon510 [$ns monitor-queue $LSR(5) $LSR(10) stdout]
set qmon105 [$ns monitor-queue $LSR(10) $LSR(5) stdout]
set qmon525 [$ns monitor-queue $LSR(5) $n_(25) stdout]
set qmon255 [$ns monitor-queue $n_(25) $LSR(5) stdout]

set qmon67 [$ns monitor-queue $LSR(6) $LSR(7) stdout]
set qmon76 [$ns monitor-queue $LSR(7) $LSR(6) stdout]
set qmon610 [$ns monitor-queue $LSR(6) $LSR(10) stdout]
set qmon106 [$ns monitor-queue $LSR(10) $LSR(6) stdout]
set qmon611 [$ns monitor-queue $LSR(6) $LSR(11) stdout]
set qmon116 [$ns monitor-queue $LSR(11) $LSR(6) stdout]

set qmon78 [$ns monitor-queue $LSR(7) $LSR(8) stdout]
set qmon87 [$ns monitor-queue $LSR(8) $LSR(7) stdout]
set qmon712 [$ns monitor-queue $LSR(7) $LSR(12) stdout]
set qmon127 [$ns monitor-queue $LSR(12) $LSR(7) stdout]

set qmon89 [$ns monitor-queue $LSR(8) $LSR(9) stdout]
set qmon98 [$ns monitor-queue $LSR(9) $LSR(8) stdout]
set qmon813 [$ns monitor-queue $LSR(8) $LSR(13) stdout]
set qmon138 [$ns monitor-queue $LSR(13) $LSR(8) stdout]

set qmon910 [$ns monitor-queue $LSR(9) $LSR(10) stdout]
set qmon109 [$ns monitor-queue $LSR(10) $LSR(9) stdout]
set qmon914 [$ns monitor-queue $LSR(9) $LSR(14) stdout]
```

```
set qmon149 [$ns monitor-queue $LSR(14) $LSR(9) stdout]

set qmon1015 [$ns monitor-queue $LSR(10) $LSR(15) stdout]
set qmon1510 [$ns monitor-queue $LSR(15) $LSR(10) stdout]

set qmon1112 [$ns monitor-queue $LSR(11) $LSR(12) stdout]
set qmon1211 [$ns monitor-queue $LSR(12) $LSR(11) stdout]
set qmon1115 [$ns monitor-queue $LSR(11) $LSR(15) stdout]
set qmon1511 [$ns monitor-queue $LSR(15) $LSR(11) stdout]
set qmon1116 [$ns monitor-queue $LSR(11) $n_(16) stdout]
set qmon1611 [$ns monitor-queue $n_(16) $LSR(11) stdout]

set qmon1213 [$ns monitor-queue $LSR(12) $LSR(13) stdout]
set qmon1312 [$ns monitor-queue $LSR(13) $LSR(12) stdout]
set qmon1217 [$ns monitor-queue $LSR(12) $n_(17) stdout]
set qmon1712 [$ns monitor-queue $n_(17) $LSR(12) stdout]

set qmon1314 [$ns monitor-queue $LSR(13) $LSR(14) stdout]
set qmon1413 [$ns monitor-queue $LSR(14) $LSR(13) stdout]
set qmon1318 [$ns monitor-queue $LSR(13) $n_(18) stdout]
set qmon1813 [$ns monitor-queue $n_(18) $LSR(13) stdout]

set qmon1415 [$ns monitor-queue $LSR(14) $LSR(15) stdout]
set qmon1514 [$ns monitor-queue $LSR(15) $LSR(14) stdout]
set qmon1419 [$ns monitor-queue $LSR(14) $n_(19) stdout]
set qmon1914 [$ns monitor-queue $n_(19) $LSR(14) stdout]

set qmon1520 [$ns monitor-queue $LSR(15) $n_(20) stdout]
set qmon2015 [$ns monitor-queue $n_(20) $LSR(15) stdout]

set qmon1617 [$ns monitor-queue $n_(16) $n_(17) stdout]
set qmon1716 [$ns monitor-queue $n_(17) $n_(16) stdout]
set qmon1620 [$ns monitor-queue $n_(16) $n_(20) stdout]
set qmon2016 [$ns monitor-queue $n_(20) $n_(16) stdout]
set qmon1621 [$ns monitor-queue $n_(16) $n_(21) stdout]
set qmon2116 [$ns monitor-queue $n_(21) $n_(16) stdout]

set qmon1718 [$ns monitor-queue $n_(17) $n_(18) stdout]
set qmon1817 [$ns monitor-queue $n_(18) $n_(17) stdout]
set qmon1722 [$ns monitor-queue $n_(17) $n_(22) stdout]
set qmon2217 [$ns monitor-queue $n_(22) $n_(17) stdout]

set qmon1819 [$ns monitor-queue $n_(18) $n_(19) stdout]
set qmon1918 [$ns monitor-queue $n_(19) $n_(18) stdout]
set qmon1823 [$ns monitor-queue $n_(18) $n_(23) stdout]
set qmon2318 [$ns monitor-queue $n_(23) $n_(18) stdout]

set qmon1920 [$ns monitor-queue $n_(19) $n_(20) stdout]
set qmon2019 [$ns monitor-queue $n_(20) $n_(19) stdout]
set qmon1924 [$ns monitor-queue $n_(19) $n_(24) stdout]
set qmon2419 [$ns monitor-queue $n_(24) $n_(19) stdout]

set qmon2025 [$ns monitor-queue $n_(20) $n_(25) stdout]
set qmon2520 [$ns monitor-queue $n_(25) $n_(20) stdout]

set qmon2122 [$ns monitor-queue $n_(21) $n_(22) stdout]
set qmon2221 [$ns monitor-queue $n_(22) $n_(21) stdout]
```

```
set qmon2125 [$ns monitor-queue $n_(21) $n_(25) stdout]
set qmon2521 [$ns monitor-queue $n_(25) $n_(21) stdout]

set qmon2223 [$ns monitor-queue $n_(22) $n_(23) stdout]
set qmon2322 [$ns monitor-queue $n_(23) $n_(22) stdout]

set qmon2324 [$ns monitor-queue $n_(23) $n_(24) stdout]
set qmon2423 [$ns monitor-queue $n_(24) $n_(23) stdout]

set qmon2425 [$ns monitor-queue $n_(24) $n_(25) stdout]
set qmon2524 [$ns monitor-queue $n_(25) $n_(24) stdout]
#===========================================================

#===========================================================
# Cria as funções e procedures


#Define a procedure que atualiza os custos
proc atualiza_tab { custoi custoa Sink_ } {
      #cria ponteiros para as tabelas de custos
      upvar $custoi inic
      upvar $custoa atual
      upvar $Sink_ recv

      #variaveis globais
      global time nx nt nz banda1 ALFA BETA
      global qmon12 qmon21 qmon15 qmon51 qmon16 qmon61 qmon121 qmon211
      global qmon23 qmon32 qmon27 qmon72 qmon222 qmon222v
      global qmon34 qmon43 qmon38 qmon83 qmon323 qmon233
      global qmon45 qmon54 qmon49 qmon94 qmon424 qmon244
      global qmon510 qmon105 qmon525 qmon255
      global qmon67 qmon76 qmon610 qmon106 qmon611 qmon116
      global qmon78 qmon87 qmon712 qmon127
      global qmon89 qmon98 qmon813 qmon138
      global qmon910 qmon109 qmon914 qmon149 qmon1015 qmon1510
      global qmon1112 qmon1211 qmon1115 qmon1511 qmon1116 qmon1611
      global qmon1213 qmon1312 qmon1217 qmon1712
      global qmon1314 qmon1413 qmon1318 qmon1813
      global qmon1415 qmon1514 qmon1419 qmon1914
      global qmon2015 qmon1520
      global qmon1617 qmon1716 qmon1620 qmon2016 qmon1621 qmon2116
      global qmon1718 qmon1817 qmon1722 qmon2217
      global qmon1819 qmon1918 qmon1823 qmon2318
      global qmon1920 qmon2019 qmon1924 qmon2419
      global qmon2025 qmon2520
      global qmon2122 qmon2221 qmon2125 qmon2521
      global qmon2223 qmon2322 qmon2324 qmon2423
      global qmon2425 qmon2524



      set ns [Simulator instance]

      #cria as variáveis bw que carregam os "bytes recebidos" de qmon
      set bw_(1,2) [$qmon12 set bdepartures_]
      set bw_(2,1) [$qmon21 set bdepartures_]
      set bw_(1,5) [$qmon15 set bdepartures_]
```

```
set bw_(5,1) [$qmon51 set bdepartures_]
set bw_(1,6) [$qmon16 set bdepartures_]
set bw_(6,1) [$qmon61 set bdepartures_]
set bw_(1,21) [$qmon121 set bdepartures_]
set bw_(21,1) [$qmon211 set bdepartures_]

set bw_(2,3) [$qmon23 set bdepartures_]
set bw_(3,2) [$qmon32 set bdepartures_]
set bw_(2,7) [$qmon27 set bdepartures_]
set bw_(7,2) [$qmon72 set bdepartures_]
set bw_(2,22) [$qmon222 set bdepartures_]
set bw_(22,2) [$qmon222v set bdepartures_]

set bw_(3,4) [$qmon34 set bdepartures_]
set bw_(4,3) [$qmon43 set bdepartures_]
set bw_(3,8) [$qmon38 set bdepartures_]
set bw_(8,3) [$qmon83 set bdepartures_]
set bw_(3,23) [$qmon323 set bdepartures_]
set bw_(23,3) [$qmon233 set bdepartures_]

set bw_(4,5) [$qmon45 set bdepartures_]
set bw_(5,4) [$qmon54 set bdepartures_]
set bw_(4,9) [$qmon49 set bdepartures_]
set bw_(9,4) [$qmon94 set bdepartures_]
set bw_(4,24) [$qmon424 set bdepartures_]
set bw_(24,4) [$qmon244 set bdepartures_]

set bw_(5,10) [$qmon510 set bdepartures_]
set bw_(10,5) [$qmon105 set bdepartures_]
set bw_(5,25) [$qmon525 set bdepartures_]
set bw_(25,5) [$qmon255 set bdepartures_]

set bw_(6,7) [$qmon67 set bdepartures_]
set bw_(7,6) [$qmon76 set bdepartures_]
set bw_(6,10) [$qmon610 set bdepartures_]
set bw_(10,6) [$qmon106 set bdepartures_]
set bw_(6,11) [$qmon611 set bdepartures_]
set bw_(11,6) [$qmon116 set bdepartures_]

set bw_(7,8) [$qmon78 set bdepartures_]
set bw_(8,7) [$qmon87 set bdepartures_]
set bw_(7,12) [$qmon712 set bdepartures_]
set bw_(12,7) [$qmon127 set bdepartures_]

set bw_(8,9) [$qmon89 set bdepartures_]
set bw_(9,8) [$qmon98 set bdepartures_]
set bw_(8,13) [$qmon813 set bdepartures_]
set bw_(13,8) [$qmon138 set bdepartures_]

set bw_(9,10) [$qmon910 set bdepartures_]
set bw_(10,9) [$qmon109 set bdepartures_]
set bw_(9,14) [$qmon914 set bdepartures_]
set bw_(14,9) [$qmon149 set bdepartures_]

set bw_(10,15) [$qmon1015 set bdepartures_]
set bw_(15,10) [$qmon1510 set bdepartures_]
```

```
set bw_(11,12) [$qmon1112 set bdepartures_]
set bw_(12,11) [$qmon1211 set bdepartures_]
set bw_(11,15) [$qmon1115 set bdepartures_]
set bw_(15,11) [$qmon1511 set bdepartures_]
set bw_(11,16) [$qmon1116 set bdepartures_]
set bw_(16,11) [$qmon1611 set bdepartures_]

set bw_(12,13) [$qmon1213 set bdepartures_]
set bw_(13,12) [$qmon1312 set bdepartures_]
set bw_(12,17) [$qmon1217 set bdepartures_]
set bw_(17,12) [$qmon1712 set bdepartures_]

set bw_(13,14) [$qmon1314 set bdepartures_]
set bw_(14,13) [$qmon1413 set bdepartures_]
set bw_(13,18) [$qmon1318 set bdepartures_]
set bw_(18,13) [$qmon1813 set bdepartures_]

set bw_(14,15) [$qmon1415 set bdepartures_]
set bw_(15,14) [$qmon1514 set bdepartures_]
set bw_(14,19) [$qmon1419 set bdepartures_]
set bw_(19,14) [$qmon1914 set bdepartures_]

set bw_(15,20) [$qmon1520 set bdepartures_]
set bw_(20,15) [$qmon2015 set bdepartures_]

set bw_(16,17) [$qmon1617 set bdepartures_]
set bw_(17,16) [$qmon1716 set bdepartures_]
set bw_(16,20) [$qmon1620 set bdepartures_]
set bw_(20,16) [$qmon2016 set bdepartures_]
set bw_(16,21) [$qmon1621 set bdepartures_]
set bw_(21,16) [$qmon2116 set bdepartures_]

set bw_(17,18) [$qmon1718 set bdepartures_]
set bw_(18,17) [$qmon1817 set bdepartures_]
set bw_(17,22) [$qmon1722 set bdepartures_]
set bw_(22,17) [$qmon2217 set bdepartures_]

set bw_(18,19) [$qmon1819 set bdepartures_]
set bw_(19,18) [$qmon1918 set bdepartures_]
set bw_(18,23) [$qmon1823 set bdepartures_]
set bw_(23,18) [$qmon2318 set bdepartures_]

set bw_(19,20) [$qmon1920 set bdepartures_]
set bw_(20,19) [$qmon2019 set bdepartures_]
set bw_(19,24) [$qmon1924 set bdepartures_]
set bw_(24,19) [$qmon2419 set bdepartures_]

set bw_(20,25) [$qmon2025 set bdepartures_]
set bw_(25,20) [$qmon2520 set bdepartures_]

set bw_(21,22) [$qmon2122 set bdepartures_]
set bw_(22,21) [$qmon2221 set bdepartures_]
set bw_(21,25) [$qmon2125 set bdepartures_]
set bw_(25,21) [$qmon2521 set bdepartures_]

set bw_(22,23) [$qmon2223 set bdepartures_]
set bw_(23,22) [$qmon2322 set bdepartures_]
```

```
set bw_(23,24) [$qmon2324 set bdepartures_]
set bw_(24,23) [$qmon2423 set bdepartures_]

set bw_(24,25) [$qmon2425 set bdepartures_]
set bw_(25,24) [$qmon2524 set bdepartures_]


#Monitora Pacotes perdidos e enviados para os enlaces do nó 1
set pdrop_(1,2) [$qmon12 set pdrops_]
set pdrop_(2,1) [$qmon21 set pdrops_]
set pdrop_(1,5) [$qmon15 set pdrops_]
set pdrop_(5,1) [$qmon51 set pdrops_]
set pdrop_(1,6) [$qmon16 set pdrops_]
set pdrop_(6,1) [$qmon61 set pdrops_]
set pdrop_(1,21) [$qmon121 set pdrops_]
set pdrop_(21,1) [$qmon211 set pdrops_]

set pdep_(1,2) [$qmon12 set pdepartures_]
set pdep_(2,1) [$qmon21 set pdepartures_]
set pdep_(1,5) [$qmon15 set pdepartures_]
set pdep_(5,1) [$qmon51 set pdepartures_]
set pdep_(1,6) [$qmon16 set pdepartures_]
set pdep_(6,1) [$qmon61 set pdepartures_]
set pdep_(1,21) [$qmon121 set pdepartures_]
set pdep_(21,1) [$qmon211 set pdepartures_]


#Monitora Pacotes perdidos e enviados para os enlaces do nó 24
set pdrop_(24,4) [$qmon244 set pdrops_]
set pdrop_(4,24) [$qmon424 set pdrops_]
set pdrop_(24,19) [$qmon2419 set pdrops_]
set pdrop_(19,24) [$qmon1924 set pdrops_]
set pdrop_(24,23) [$qmon2423 set pdrops_]
set pdrop_(23,24) [$qmon2324 set pdrops_]
set pdrop_(24,25) [$qmon2425 set pdrops_]
set pdrop_(25,24) [$qmon2524 set pdrops_]

set pdep_(24,4) [$qmon244 set pdepartures_]
set pdep_(4,24) [$qmon424 set pdepartures_]
set pdep_(24,19) [$qmon2419 set pdepartures_]
set pdep_(19,24) [$qmon1924 set pdepartures_]
set pdep_(24,23) [$qmon2423 set pdepartures_]
set pdep_(23,24) [$qmon2324 set pdepartures_]
set pdep_(24,25) [$qmon2425 set pdepartures_]
set pdep_(25,24) [$qmon2524 set pdepartures_]

#=========================================================
# Cria os indices de ocupacao dos links
for {set t 1} {$t < 26} {incr t} {
      for {set s 1} {$s < 26} {incr s} {
            set ocupa_($s,$t) 1
      }
}

#calcula o indice de ocupação dos links
set ocupa_(1,2) [expr ($bw_(1,2) * 8/($time * $banda1))]
```

```
set ocupa_(2,1) [expr ($bw_(2,1) * 8/($time * $banda1))]
set ocupa_(1,5) [expr ($bw_(1,5) * 8/($time * $banda1))]
set ocupa_(5,1) [expr ($bw_(5,1) * 8/($time * $banda1))]
set ocupa_(1,6) [expr ($bw_(1,6) * 8/($time * $banda1))]
set ocupa_(6,1) [expr ($bw_(6,1) * 8/($time * $banda1))]
set ocupa_(1,21) [expr ($bw_(1,21) * 8/($time * $banda1))]
set ocupa_(21,1) [expr ($bw_(21,1) * 8/($time * $banda1))]

set ocupa_(2,3) [expr ($bw_(2,3) * 8/($time * $banda1))]
set ocupa_(3,2) [expr ($bw_(3,2) * 8/($time * $banda1))]
set ocupa_(2,7) [expr ($bw_(2,7) * 8/($time * $banda1))]
set ocupa_(7,2) [expr ($bw_(7,2) * 8/($time * $banda1))]
set ocupa_(2,22) [expr ($bw_(2,22) * 8/($time * $banda1))]
set ocupa_(22,2) [expr ($bw_(22,2) * 8/($time * $banda1))]

set ocupa_(3,4) [expr ($bw_(3,4) * 8/($time * $banda1))]
set ocupa_(4,3) [expr ($bw_(4,3) * 8/($time * $banda1))]
set ocupa_(3,8) [expr ($bw_(3,8) * 8/($time * $banda1))]
set ocupa_(8,3) [expr ($bw_(8,3) * 8/($time * $banda1))]
set ocupa_(3,23) [expr ($bw_(3,23) * 8/($time * $banda1))]
set ocupa_(23,3) [expr ($bw_(23,3) * 8/($time * $banda1))]

set ocupa_(4,5) [expr ($bw_(4,5) * 8/($time * $banda1))]
set ocupa_(5,4) [expr ($bw_(5,4) * 8/($time * $banda1))]
set ocupa_(4,9) [expr ($bw_(4,9) * 8/($time * $banda1))]
set ocupa_(9,4) [expr ($bw_(9,4) * 8/($time * $banda1))]
set ocupa_(4,24) [expr ($bw_(4,24) * 8/($time * $banda1))]
set ocupa_(24,4) [expr ($bw_(24,4) * 8/($time * $banda1))]

set ocupa_(5,10) [expr ($bw_(5,10) * 8/($time * $banda1))]
set ocupa_(10,5) [expr ($bw_(10,5) * 8/($time * $banda1))]
set ocupa_(5,25) [expr ($bw_(5,25) * 8/($time * $banda1))]
set ocupa_(25,5) [expr ($bw_(25,5) * 8/($time * $banda1))]

set ocupa_(6,7) [expr ($bw_(6,7) * 8/($time * $banda1))]
set ocupa_(7,6) [expr ($bw_(7,6) * 8/($time * $banda1))]
set ocupa_(6,10) [expr ($bw_(6,10) * 8/($time * $banda1))]
set ocupa_(10,6) [expr ($bw_(10,6) * 8/($time * $banda1))]
set ocupa_(6,11) [expr ($bw_(6,11) * 8/($time * $banda1))]
set ocupa_(11,6) [expr ($bw_(11,6) * 8/($time * $banda1))]

set ocupa_(7,8) [expr ($bw_(7,8) * 8/($time * $banda1))]
set ocupa_(8,7) [expr ($bw_(8,7) * 8/($time * $banda1))]
set ocupa_(7,12) [expr ($bw_(7,12) * 8/($time * $banda1))]
set ocupa_(12,7) [expr ($bw_(12,7) * 8/($time * $banda1))]

set ocupa_(8,9) [expr ($bw_(8,9) * 8/($time * $banda1))]
set ocupa_(9,8) [expr ($bw_(9,8) * 8/($time * $banda1))]
set ocupa_(8,13) [expr ($bw_(8,13) * 8/($time * $banda1))]
set ocupa_(13,8) [expr ($bw_(13,8) * 8/($time * $banda1))]

set ocupa_(9,10) [expr ($bw_(9,10) * 8/($time * $banda1))]
set ocupa_(10,9) [expr ($bw_(10,9) * 8/($time * $banda1))]
set ocupa_(9,14) [expr ($bw_(9,14) * 8/($time * $banda1))]
set ocupa_(14,9) [expr ($bw_(14,9) * 8/($time * $banda1))]

set ocupa_(10,15) [expr ($bw_(10,15) * 8/($time * $banda1))]
```

```
set ocupa_(15,10) [expr ($bw_(15,10) * 8/($time * $banda1))]

set ocupa_(11,12) [expr ($bw_(11,12) * 8/($time * $banda1))]
set ocupa_(12,11) [expr ($bw_(12,11) * 8/($time * $banda1))]
set ocupa_(11,15) [expr ($bw_(11,15) * 8/($time * $banda1))]
set ocupa_(15,11) [expr ($bw_(15,11) * 8/($time * $banda1))]
set ocupa_(11,16) [expr ($bw_(11,16) * 8/($time * $banda1))]
set ocupa_(16,11) [expr ($bw_(16,11) * 8/($time * $banda1))]

set ocupa_(12,13) [expr ($bw_(12,13) * 8/($time * $banda1))]
set ocupa_(13,12) [expr ($bw_(13,12) * 8/($time * $banda1))]
set ocupa_(12,17) [expr ($bw_(12,17) * 8/($time * $banda1))]
set ocupa_(17,12) [expr ($bw_(17,12) * 8/($time * $banda1))]

set ocupa_(13,14) [expr ($bw_(13,14) * 8/($time * $banda1))]
set ocupa_(14,13) [expr ($bw_(14,13) * 8/($time * $banda1))]
set ocupa_(13,18) [expr ($bw_(13,18) * 8/($time * $banda1))]
set ocupa_(18,13) [expr ($bw_(18,13) * 8/($time * $banda1))]

set ocupa_(14,15) [expr ($bw_(14,15) * 8/($time * $banda1))]
set ocupa_(15,14) [expr ($bw_(15,14) * 8/($time * $banda1))]
set ocupa_(14,19) [expr ($bw_(14,19) * 8/($time * $banda1))]
set ocupa_(19,14) [expr ($bw_(19,14) * 8/($time * $banda1))]

set ocupa_(15,20) [expr ($bw_(15,20) * 8/($time * $banda1))]
set ocupa_(20,15) [expr ($bw_(20,15) * 8/($time * $banda1))]

set ocupa_(16,17) [expr ($bw_(16,17) * 8/($time * $banda1))]
set ocupa_(17,16) [expr ($bw_(17,16) * 8/($time * $banda1))]
set ocupa_(16,20) [expr ($bw_(16,20) * 8/($time * $banda1))]
set ocupa_(20,16) [expr ($bw_(20,16) * 8/($time * $banda1))]
set ocupa_(16,21) [expr ($bw_(16,21) * 8/($time * $banda1))]
set ocupa_(21,16) [expr ($bw_(21,16) * 8/($time * $banda1))]

set ocupa_(17,18) [expr ($bw_(17,18) * 8/($time * $banda1))]
set ocupa_(18,17) [expr ($bw_(18,17) * 8/($time * $banda1))]
set ocupa_(17,22) [expr ($bw_(17,22) * 8/($time * $banda1))]
set ocupa_(22,17) [expr ($bw_(22,17) * 8/($time * $banda1))]

set ocupa_(18,19) [expr ($bw_(18,19) * 8/($time * $banda1))]
set ocupa_(19,18) [expr ($bw_(19,18) * 8/($time * $banda1))]
set ocupa_(18,23) [expr ($bw_(18,23) * 8/($time * $banda1))]
set ocupa_(23,18) [expr ($bw_(23,18) * 8/($time * $banda1))]

set ocupa_(19,20) [expr ($bw_(19,20) * 8/($time * $banda1))]
set ocupa_(20,19) [expr ($bw_(20,19) * 8/($time * $banda1))]
set ocupa_(19,24) [expr ($bw_(19,24) * 8/($time * $banda1))]
set ocupa_(24,19) [expr ($bw_(24,19) * 8/($time * $banda1))]

set ocupa_(20,25) [expr ($bw_(20,25) * 8/($time * $banda1))]
set ocupa_(25,20) [expr ($bw_(25,20) * 8/($time * $banda1))]

set ocupa_(21,22) [expr ($bw_(21,22) * 8/($time * $banda1))]
set ocupa_(22,21) [expr ($bw_(22,21) * 8/($time * $banda1))]
set ocupa_(21,25) [expr ($bw_(21,25) * 8/($time * $banda1))]
set ocupa_(25,21) [expr ($bw_(25,21) * 8/($time * $banda1))]
```

```
      set ocupa_(22,23) [expr ($bw_(22,23) * 8/($time * $banda1))]
      set ocupa_(23,22) [expr ($bw_(23,22) * 8/($time * $banda1))]

      set ocupa_(23,24) [expr ($bw_(23,24) * 8/($time * $banda1))]
      set ocupa_(24,23) [expr ($bw_(24,23) * 8/($time * $banda1))]

      set ocupa_(24,25) [expr ($bw_(24,25) * 8/($time * $banda1))]
      set ocupa_(25,24) [expr ($bw_(25,24) * 8/($time * $banda1))]

      ###############################################################
      # Beta
      # A varável back determina a ocupação do backplane
      # Foram criadas 4 variáveis back para cada nó, todas recebem o
mesmo valor
      # o qual será usado como multiplicador para o custo dos enlaces
      for {set t 1} {$t < 26} {incr t} {
            for {set s 1} {$s < 26} {incr s} {
                  set back($s,$t) 1
            }
      }

      # Nó 1
      set back(1,2) [expr  ( 1  - (( 4000000 - (($bw_(1,2) + $bw_(1,5)
+ $bw_(1,6) + $bw_(1,21)) * 8 / $time)) /4000000 ))]
      set back(1,5) [expr  ( 1  - (( 4000000 - (($bw_(1,2) + $bw_(1,5)
+ $bw_(1,6) + $bw_(1,21)) * 8 / $time)) /4000000 ))]
      set back(1,6) [expr  ( 1  - (( 4000000 - (($bw_(1,2) + $bw_(1,5)
+ $bw_(1,6) + $bw_(1,21)) * 8 / $time)) /4000000 ))]
      set back(1,21) [expr  ( 1  - (( 4000000 - (($bw_(1,2) + $bw_(1,5)
+ $bw_(1,6) + $bw_(1,21)) * 8 / $time)) /4000000 ))]

      # Nó 2
      set back(2,1) [expr  ( 1  - (( 4000000 - (($bw_(2,1) + $bw_(2,3)
+ $bw_(2,7) + $bw_(2,22)) * 8 / $time)) /4000000 ))]
      set back(2,3) [expr  ( 1  - (( 4000000 - (($bw_(2,1) + $bw_(2,3)
+ $bw_(2,7) + $bw_(2,22)) * 8 / $time)) /4000000 ))]
      set back(2,7) [expr  ( 1  - (( 4000000 - (($bw_(2,1) + $bw_(2,3)
+ $bw_(2,7) + $bw_(2,22)) * 8 / $time)) /4000000 ))]
      set back(2,22) [expr  ( 1  - (( 4000000 - (($bw_(2,1) + $bw_(2,3)
+ $bw_(2,7) + $bw_(2,22)) * 8 / $time)) /4000000 ))]

      # Nó 3
      set back(3,2) [expr  ( 1  - (( 4000000 - (($bw_(3,2) + $bw_(3,4)
+ $bw_(3,8) + $bw_(3,23)) * 8 / $time)) /4000000 ))]
      set back(3,4) [expr  ( 1  - (( 4000000 - (($bw_(3,2) + $bw_(3,4)
+ $bw_(3,8) + $bw_(3,23)) * 8 / $time)) /4000000 ))]
      set back(3,8) [expr  ( 1  - (( 4000000 - (($bw_(3,2) + $bw_(3,4)
+ $bw_(3,8) + $bw_(3,23)) * 8 / $time)) /4000000 ))]
      set back(3,23) [expr  ( 1  - (( 4000000 - (($bw_(3,2) + $bw_(3,4)
+ $bw_(3,8) + $bw_(3,23)) * 8 / $time)) /4000000 ))]

      # Nó 4
      set back(4,3) [expr  ( 1  - (( 4000000 - (($bw_(4,3) + $bw_(4,5)
+ $bw_(4,9) + $bw_(4,24)) * 8 / $time)) /4000000 ))]
      set back(4,5) [expr  ( 1  - (( 4000000 - (($bw_(4,3) + $bw_(4,5)
+ $bw_(4,9) + $bw_(4,24)) * 8 / $time)) /4000000 ))]
```

```
      set back(4,9) [expr  ( 1  - (( 4000000 - (($bw_(4,3) + $bw_(4,5)
+ $bw_(4,9) + $bw_(4,24)) * 8 / $time)) /4000000 ))]
      set back(4,24) [expr  ( 1  - (( 4000000 - (($bw_(4,3) + $bw_(4,5)
+ $bw_(4,9) + $bw_(4,24)) * 8 / $time)) /4000000 ))]

      # Nó 5
      set back(5,1) [expr  ( 1  - (( 4000000 - (($bw_(5,1) + $bw_(5,4)
+ $bw_(5,10) + $bw_(5,25)) * 8 / $time)) /4000000 ))]
      set back(5,4) [expr  ( 1  - (( 4000000 - (($bw_(5,1) + $bw_(5,4)
+ $bw_(5,10) + $bw_(5,25)) * 8 / $time)) /4000000 ))]
      set back(5,10) [expr  ( 1  - (( 4000000 - (($bw_(5,1) + $bw_(5,4)
+ $bw_(5,10) + $bw_(5,25)) * 8 / $time)) /4000000 ))]
      set back(5,25) [expr  ( 1  - (( 4000000 - (($bw_(5,1) + $bw_(5,4)
+ $bw_(5,10) + $bw_(5,25)) * 8 / $time)) /4000000 ))]

      # Nó 6
      set back(6,1) [expr  ( 1  - (( 4000000 - (($bw_(6,1) + $bw_(6,7)
+ $bw_(6,10) + $bw_(6,11)) * 8 / $time)) /4000000 ))]
      set back(6,7) [expr  ( 1  - (( 4000000 - (($bw_(6,1) + $bw_(6,7)
+ $bw_(6,10) + $bw_(6,11)) * 8 / $time)) /4000000 ))]
      set back(6,10) [expr  ( 1  - (( 4000000 - (($bw_(6,1) + $bw_(6,7)
+ $bw_(6,10) + $bw_(6,11)) * 8 / $time)) /4000000 ))]
      set back(6,11) [expr  ( 1  - (( 4000000 - (($bw_(6,1) + $bw_(6,7)
+ $bw_(6,10) + $bw_(6,11)) * 8 / $time)) /4000000 ))]

      # Nó 7
      set back(7,2) [expr  ( 1  - (( 4000000 - (($bw_(7,2) + $bw_(7,6)
+ $bw_(7,8) + $bw_(7,12)) * 8 / $time)) /4000000 ))]
      set back(7,6) [expr  ( 1  - (( 4000000 - (($bw_(7,2) + $bw_(7,6)
+ $bw_(7,8) + $bw_(7,12)) * 8 / $time)) /4000000 ))]
      set back(7,8) [expr  ( 1  - (( 4000000 - (($bw_(7,2) + $bw_(7,6)
+ $bw_(7,8) + $bw_(7,12)) * 8 / $time)) /4000000 ))]
      set back(7,12) [expr  ( 1  - (( 4000000 - (($bw_(7,2) + $bw_(7,6)
+ $bw_(7,8) + $bw_(7,12)) * 8 / $time)) /4000000 ))]

      # Nó 8
      set back(8,3) [expr  ( 1  - (( 4000000 - (($bw_(8,3) + $bw_(8,7)
+ $bw_(8,9) + $bw_(8,13)) * 8 / $time)) /4000000 ))]
      set back(8,7) [expr  ( 1  - (( 4000000 - (($bw_(8,3) + $bw_(8,7)
+ $bw_(8,9) + $bw_(8,13)) * 8 / $time)) /4000000 ))]
      set back(8,9) [expr  ( 1  - (( 4000000 - (($bw_(8,3) + $bw_(8,7)
+ $bw_(8,9) + $bw_(8,13)) * 8 / $time)) /4000000 ))]
      set back(8,13) [expr  ( 1  - (( 4000000 - (($bw_(8,3) + $bw_(8,7)
+ $bw_(8,9) + $bw_(8,13)) * 8 / $time)) /4000000 ))]

      # Nó 9
      set back(9,4) [expr  ( 1  - (( 4000000 - (($bw_(9,4) + $bw_(9,8)
+ $bw_(9,10) + $bw_(9,14)) * 8 / $time)) /4000000 ))]
      set back(9,8) [expr  ( 1  - (( 4000000 - (($bw_(9,4) + $bw_(9,8)
+ $bw_(9,10) + $bw_(9,14)) * 8 / $time)) /4000000 ))]
      set back(9,10) [expr  ( 1  - (( 4000000 - (($bw_(9,4) + $bw_(9,8)
+ $bw_(9,10) + $bw_(9,14)) * 8 / $time)) /4000000 ))]
      set back(9,14) [expr  ( 1  - (( 4000000 - (($bw_(9,4) + $bw_(9,8)
+ $bw_(9,10) + $bw_(9,14)) * 8 / $time)) /4000000 ))]

      # Nó 10
```

```
      set back(10,5) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]
      set back(10,6) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]
      set back(10,9) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]
      set back(10,15) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]

      # Nó 11
      set back(11,6) [expr  ( 1  - (( 4000000 - (($bw_(11,6) +
$bw_(11,12) + $bw_(11,15) + $bw_(11,16)) * 8 / $time)) /4000000 ))]
      set back(11,12) [expr  ( 1  - (( 4000000 - (($bw_(11,6) +
$bw_(11,12) + $bw_(11,15) + $bw_(11,16)) * 8 / $time)) /4000000 ))]
      set back(11,15) [expr  ( 1  - (( 4000000 - (($bw_(11,6) +
$bw_(11,12) + $bw_(11,15) + $bw_(11,16)) * 8 / $time)) /4000000 ))]
      set back(11,16) [expr  ( 1  - (( 4000000 - (($bw_(11,6) +
$bw_(11,12) + $bw_(11,15) + $bw_(11,16)) * 8 / $time)) /4000000 ))]

      # Nó 12
      set back(12,7) [expr  ( 1  - (( 4000000 - (($bw_(12,7) +
$bw_(12,11) + $bw_(12,13) + $bw_(12,17)) * 8 / $time)) /4000000 ))]
      set back(12,11) [expr  ( 1  - (( 4000000 - (($bw_(12,7) +
$bw_(12,11) + $bw_(12,13) + $bw_(12,17)) * 8 / $time)) /4000000 ))]
      set back(12,13) [expr  ( 1  - (( 4000000 - (($bw_(12,7) +
$bw_(12,11) + $bw_(12,13) + $bw_(12,17)) * 8 / $time)) /4000000 ))]
      set back(12,17) [expr  ( 1  - (( 4000000 - (($bw_(12,7) +
$bw_(12,11) + $bw_(12,13) + $bw_(12,17)) * 8 / $time)) /4000000 ))]

      # Nó 13
      set back(13,8) [expr  ( 1  - (( 4000000 - (($bw_(13,8) +
$bw_(13,12) + $bw_(13,14) + $bw_(13,18)) * 8 / $time)) /4000000 ))]
      set back(13,12) [expr  ( 1  - (( 4000000 - (($bw_(13,8) +
$bw_(13,12) + $bw_(13,14) + $bw_(13,18)) * 8 / $time)) /4000000 ))]
      set back(13,14) [expr  ( 1  - (( 4000000 - (($bw_(13,8) +
$bw_(13,12) + $bw_(13,14) + $bw_(13,18)) * 8 / $time)) /4000000 ))]
      set back(13,18) [expr  ( 1  - (( 4000000 - (($bw_(13,8) +
$bw_(13,12) + $bw_(13,14) + $bw_(13,18)) * 8 / $time)) /4000000 ))]

      # Nó 14
      set back(14,9) [expr  ( 1  - (( 4000000 - (($bw_(14,9) +
$bw_(14,13) + $bw_(14,15) + $bw_(14,19)) * 8 / $time)) /4000000 ))]
      set back(14,13) [expr  ( 1  - (( 4000000 - (($bw_(14,9) +
$bw_(14,13) + $bw_(14,15) + $bw_(14,19)) * 8 / $time)) /4000000 ))]
      set back(14,15) [expr  ( 1  - (( 4000000 - (($bw_(14,9) +
$bw_(14,13) + $bw_(14,15) + $bw_(14,19)) * 8 / $time)) /4000000 ))]
      set back(14,19) [expr  ( 1  - (( 4000000 - (($bw_(14,9) +
$bw_(14,13) + $bw_(14,15) + $bw_(14,19)) * 8 / $time)) /4000000 ))]

      # Nó 15
      set back(15,10) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]
      set back(15,11) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]
      set back(15,14) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]
```

```
      set back(15,20) [expr  ( 1  - (( 4000000 - (($bw_(10,5) +
$bw_(10,6) + $bw_(10,9) + $bw_(10,15)) * 8 / $time)) /4000000 ))]

      # Nó 16
      set back(16,11) [expr  ( 1  - (( 4000000 - (($bw_(16,11) +
$bw_(16,17) + $bw_(16,20) + $bw_(16,21)) * 8 / $time)) /4000000 ))]
      set back(16,17) [expr  ( 1  - (( 4000000 - (($bw_(16,11) +
$bw_(16,17) + $bw_(16,20) + $bw_(16,21)) * 8 / $time)) /4000000 ))]
      set back(16,20) [expr  ( 1  - (( 4000000 - (($bw_(16,11) +
$bw_(16,17) + $bw_(16,20) + $bw_(16,21)) * 8 / $time)) /4000000 ))]
      set back(16,21) [expr  ( 1  - (( 4000000 - (($bw_(16,11) +
$bw_(16,17) + $bw_(16,20) + $bw_(16,21)) * 8 / $time)) /4000000 ))]

      # Nó 17
      set back(17,12) [expr  ( 1  - (( 4000000 - (($bw_(17,12) +
$bw_(17,16) + $bw_(17,18) + $bw_(17,22)) * 8 / $time)) /4000000 ))]
      set back(17,16) [expr  ( 1  - (( 4000000 - (($bw_(17,12) +
$bw_(17,16) + $bw_(17,18) + $bw_(17,22)) * 8 / $time)) /4000000 ))]
      set back(17,18) [expr  ( 1  - (( 4000000 - (($bw_(17,12) +
$bw_(17,16) + $bw_(17,18) + $bw_(17,22)) * 8 / $time)) /4000000 ))]
      set back(17,22) [expr  ( 1  - (( 4000000 - (($bw_(17,12) +
$bw_(17,16) + $bw_(17,18) + $bw_(17,22)) * 8 / $time)) /4000000 ))]

      # Nó 18
      set back(18,13) [expr  ( 1  - (( 4000000 - (($bw_(18,13) +
$bw_(18,17) + $bw_(18,19) + $bw_(18,23)) * 8 / $time)) /4000000 ))]
      set back(18,17) [expr  ( 1  - (( 4000000 - (($bw_(18,13) +
$bw_(18,17) + $bw_(18,19) + $bw_(18,23)) * 8 / $time)) /4000000 ))]
      set back(18,19) [expr  ( 1  - (( 4000000 - (($bw_(18,13) +
$bw_(18,17) + $bw_(18,19) + $bw_(18,23)) * 8 / $time)) /4000000 ))]
      set back(18,23) [expr  ( 1  - (( 4000000 - (($bw_(18,13) +
$bw_(18,17) + $bw_(18,19) + $bw_(18,23)) * 8 / $time)) /4000000 ))]

      # Nó 19
      set back(19,14) [expr  ( 1  - (( 4000000 - (($bw_(19,14) +
$bw_(19,18) + $bw_(19,20) + $bw_(19,24)) * 8 / $time)) /4000000 ))]
      set back(19,18) [expr  ( 1  - (( 4000000 - (($bw_(19,14) +
$bw_(19,18) + $bw_(19,20) + $bw_(19,24)) * 8 / $time)) /4000000 ))]
      set back(19,20) [expr  ( 1  - (( 4000000 - (($bw_(19,14) +
$bw_(19,18) + $bw_(19,20) + $bw_(19,24)) * 8 / $time)) /4000000 ))]
      set back(19,24) [expr  ( 1  - (( 4000000 - (($bw_(19,14) +
$bw_(19,18) + $bw_(19,20) + $bw_(19,24)) * 8 / $time)) /4000000 ))]

      # Nó 20
      set back(20,15) [expr  ( 1  - (( 4000000 - (($bw_(20,15) +
$bw_(20,16) + $bw_(20,19) + $bw_(20,25)) * 8 / $time)) /4000000 ))]
      set back(20,16) [expr  ( 1  - (( 4000000 - (($bw_(20,15) +
$bw_(20,16) + $bw_(20,19) + $bw_(20,25)) * 8 / $time)) /4000000 ))]
      set back(20,19) [expr  ( 1  - (( 4000000 - (($bw_(20,15) +
$bw_(20,16) + $bw_(20,19) + $bw_(20,25)) * 8 / $time)) /4000000 ))]
      set back(20,25) [expr  ( 1  - (( 4000000 - (($bw_(20,15) +
$bw_(20,16) + $bw_(20,19) + $bw_(20,25)) * 8 / $time)) /4000000 ))]

      # Nó 21
      set back(21,1) [expr  ( 1  - (( 4000000 - (($bw_(21,1) +
$bw_(21,16) + $bw_(21,22) + $bw_(21,25)) * 8 / $time)) /4000000 ))]
```

```
      set back(21,16) [expr  ( 1  - (( 4000000 - (($bw_(21,1) +
$bw_(21,16) + $bw_(21,22) + $bw_(21,25)) * 8 / $time)) /4000000 ))]
      set back(21,22) [expr  ( 1  - (( 4000000 - (($bw_(21,1) +
$bw_(21,16) + $bw_(21,22) + $bw_(21,25)) * 8 / $time)) /4000000 ))]
      set back(21,25) [expr  ( 1  - (( 4000000 - (($bw_(21,1) +
$bw_(21,16) + $bw_(21,22) + $bw_(21,25)) * 8 / $time)) /4000000 ))]

      # Nó 22
      set back(22,2) [expr  ( 1  - (( 4000000 - (($bw_(22,2) +
$bw_(22,17) + $bw_(22,21) + $bw_(22,23)) * 8 / $time)) /4000000 ))]
      set back(22,17) [expr  ( 1  - (( 4000000 - (($bw_(22,2) +
$bw_(22,17) + $bw_(22,21) + $bw_(22,23)) * 8 / $time)) /4000000 ))]
      set back(22,21) [expr  ( 1  - (( 4000000 - (($bw_(22,2) +
$bw_(22,17) + $bw_(22,21) + $bw_(22,23)) * 8 / $time)) /4000000 ))]
      set back(22,23) [expr  ( 1  - (( 4000000 - (($bw_(22,2) +
$bw_(22,17) + $bw_(22,21) + $bw_(22,23)) * 8 / $time)) /4000000 ))]

      # Nó 23
      set back(23,3) [expr  ( 1  - (( 4000000 - (($bw_(23,3) +
$bw_(23,18) + $bw_(23,22) + $bw_(23,24)) * 8 / $time)) /4000000 ))]
      set back(23,18) [expr  ( 1  - (( 4000000 - (($bw_(23,3) +
$bw_(23,18) + $bw_(23,22) + $bw_(23,24)) * 8 / $time)) /4000000 ))]
      set back(23,22) [expr  ( 1  - (( 4000000 - (($bw_(23,3) +
$bw_(23,18) + $bw_(23,22) + $bw_(23,24)) * 8 / $time)) /4000000 ))]
      set back(23,24) [expr  ( 1  - (( 4000000 - (($bw_(23,3) +
$bw_(23,18) + $bw_(23,22) + $bw_(23,24)) * 8 / $time)) /4000000 ))]

      # Nó 24
      set back(24,4) [expr  ( 1  - (( 4000000 - (($bw_(24,4) +
$bw_(24,19) + $bw_(24,23) + $bw_(24,25)) * 8 / $time)) /4000000 ))]
      set back(24,19) [expr  ( 1  - (( 4000000 - (($bw_(24,4) +
$bw_(24,19) + $bw_(24,23) + $bw_(24,25)) * 8 / $time)) /4000000 ))]
      set back(24,23) [expr  ( 1  - (( 4000000 - (($bw_(24,4) +
$bw_(24,19) + $bw_(24,23) + $bw_(24,25)) * 8 / $time)) /4000000 ))]
      set back(24,25) [expr  ( 1  - (( 4000000 - (($bw_(24,4) +
$bw_(24,19) + $bw_(24,23) + $bw_(24,25)) * 8 / $time)) /4000000 ))]

      # Nó 25
      set back(25,5) [expr  ( 1  - (( 4000000 - (($bw_(25,5) +
$bw_(25,20) + $bw_(25,21) + $bw_(25,24)) * 8 / $time)) /4000000 ))]
      set back(25,20) [expr  ( 1  - (( 4000000 - (($bw_(25,5) +
$bw_(25,20) + $bw_(25,21) + $bw_(25,24)) * 8 / $time)) /4000000 ))]
      set back(25,21) [expr  ( 1  - (( 4000000 - (($bw_(25,5) +
$bw_(25,20) + $bw_(25,21) + $bw_(25,24)) * 8 / $time)) /4000000 ))]
      set back(25,24) [expr  ( 1  - (( 4000000 - (($bw_(25,5) +
$bw_(25,20) + $bw_(25,21) + $bw_(25,24)) * 8 / $time)) /4000000 ))]

      #============================================================
      # Calcula o valor ATUAL dos custos e poe na variavel atual
      # ALFA é o 1o fator de ajuste
      #set ALFA 1
      #set BETA 1

      for {set t 1} {$t < 26} {incr t} {
            for {set s 1} {$s < 26} {incr s} {
                  # só Alfa
```

```
                    #set atual($t,$s) [expr ($inic($s,$t) * (1 + ( $ALPHA
* $ocupa_($s,$t))))]
                    # Alfa e Beta
                    set atual($t,$s) [expr ($inic($s,$t) * (1 + (( $ALFA
* $ocupa_($s,$t)) + ( $BETA * $ocupa_($s,$t) * ( 1 + $back($s,$t))))))]
                }
        }

        puts "$atual(1,21)"
        puts "$atual(21,1)"
        puts "$atual(24,25)"
        puts "$atual(25,24)"

        # cria a variavel now com o momento atual da simulação
        set now [$ns now]
        puts "$now \n"


#=================================================
# Zera a variavel qmon barrivals_  dos links
# Facilita o calculo da ocupação entre o momento atual e a próxima
medida

        $qmon12 set bdepartures_ 0
        $qmon21 set bdepartures_ 0
        $qmon15 set bdepartures_ 0
        $qmon51 set bdepartures_ 0
        $qmon16 set bdepartures_ 0
        $qmon61 set bdepartures_ 0
        $qmon121 set bdepartures_ 0
        $qmon211 set bdepartures_ 0

        $qmon23 set bdepartures_ 0
        $qmon32 set bdepartures_ 0
        $qmon27 set bdepartures_ 0
        $qmon72 set bdepartures_ 0
        $qmon222 set bdepartures_ 0
        $qmon222v set bdepartures_ 0

        $qmon34 set bdepartures_ 0
        $qmon43 set bdepartures_ 0
        $qmon38 set bdepartures_ 0
        $qmon83 set bdepartures_ 0
        $qmon323 set bdepartures_ 0
        $qmon233 set bdepartures_ 0

        $qmon45 set bdepartures_ 0
        $qmon54 set bdepartures_ 0
        $qmon49 set bdepartures_ 0
        $qmon94 set bdepartures_ 0
        $qmon424 set bdepartures_ 0
        $qmon244 set bdepartures_ 0

        $qmon510 set bdepartures_ 0
        $qmon105 set bdepartures_ 0
        $qmon525 set bdepartures_ 0
        $qmon255 set bdepartures_ 0
```

```
$qmon67 set bdepartures_ 0
$qmon76 set bdepartures_ 0
$qmon610 set bdepartures_ 0
$qmon106 set bdepartures_ 0
$qmon611 set bdepartures_ 0
$qmon116 set bdepartures_ 0

$qmon78 set bdepartures_ 0
$qmon87 set bdepartures_ 0
$qmon712 set bdepartures_ 0
$qmon127 set bdepartures_ 0

$qmon89 set bdepartures_ 0
$qmon98 set bdepartures_ 0
$qmon813 set bdepartures_ 0
$qmon138 set bdepartures_ 0

$qmon910 set bdepartures_ 0
$qmon109 set bdepartures_ 0
$qmon914 set bdepartures_ 0
$qmon149 set bdepartures_ 0

$qmon1015 set bdepartures_ 0
$qmon1510 set bdepartures_ 0

$qmon1112 set bdepartures_ 0
$qmon1211 set bdepartures_ 0
$qmon1115 set bdepartures_ 0
$qmon1511 set bdepartures_ 0
$qmon1116 set bdepartures_ 0
$qmon1611 set bdepartures_ 0

$qmon1213 set bdepartures_ 0
$qmon1312 set bdepartures_ 0
$qmon1217 set bdepartures_ 0
$qmon1712 set bdepartures_ 0

$qmon1314 set bdepartures_ 0
$qmon1413 set bdepartures_ 0
$qmon1318 set bdepartures_ 0
$qmon1813 set bdepartures_ 0

$qmon1415 set bdepartures_ 0
$qmon1514 set bdepartures_ 0
$qmon1419 set bdepartures_ 0
$qmon1914 set bdepartures_ 0

$qmon1520 set bdepartures_ 0
$qmon2015 set bdepartures_ 0

$qmon1617 set bdepartures_ 0
$qmon1716 set bdepartures_ 0
$qmon1620 set bdepartures_ 0
$qmon2016 set bdepartures_ 0
$qmon1621 set bdepartures_ 0
$qmon2116 set bdepartures_ 0
```

```
        $qmon1718 set bdepartures_ 0
        $qmon1817 set bdepartures_ 0
        $qmon1722 set bdepartures_ 0
        $qmon2217 set bdepartures_ 0

        $qmon1819 set bdepartures_ 0
        $qmon1918 set bdepartures_ 0
        $qmon1823 set bdepartures_ 0
        $qmon2318 set bdepartures_ 0

        $qmon1920 set bdepartures_ 0
        $qmon2019 set bdepartures_ 0
        $qmon1924 set bdepartures_ 0
        $qmon2419 set bdepartures_ 0

        $qmon2025 set bdepartures_ 0
        $qmon2520 set bdepartures_ 0

        $qmon2122 set bdepartures_ 0
        $qmon2221 set bdepartures_ 0
        $qmon2125 set bdepartures_ 0
        $qmon2521 set bdepartures_ 0

        $qmon2223 set bdepartures_ 0
        $qmon2322 set bdepartures_ 0

        $qmon2324 set bdepartures_ 0
        $qmon2423 set bdepartures_ 0

        $qmon2425 set bdepartures_ 0
        $qmon2524 set bdepartures_ 0

#       Computa os dados recebidos e perdidos para cada Sink_
Momentaneamente
#       perd é o numero de packets lost
#       recb é o numero de packets recebidos
#       Total é o total enviado da fonte (src) correspondente
        for {set i 106} {$i < 162} {incr i} {
              set perd($i) [$recv($i) set nlost_]
              set recb($i) [$recv($i) set npkts_]
              set total($i) [expr $perd($i) + $recb($i)]
        }
        set perda 0
        set recebido 0
        set totaliza 1

#       Computa os dados recebidos e perdidos TOTAIS para cada Sink_
        for {set i 106} {$i < 162} {incr i} {
              set perda [expr $perda + $perd($i)]
              set recebido [expr $recebido + $recb($i)]
              set totaliza [expr $totaliza + $total($i)]
              set percent [expr (($perda * 100.00)/$totaliza)]
        }
        # Imprime os valores para o arquivo nx
        puts $nx "$now,$perda,$recebido,$totaliza,$percent"
```

```
      # Imprime dados diversos para os nós 1 e 24
      puts $nt
"$pdrop_(1,2),$pdrop_(2,1),$pdrop_(1,5),$pdrop_(5,1),$pdrop_(1,6),$pdro
p_(6,1),$pdrop_(1,21),$pdrop_(21,1),$pdrop_(24,4),$pdrop_(4,24),$pdrop_
(24,19),$pdrop_(19,24),$pdrop_(24,23),$pdrop_(23,24),$pdrop_(24,25),$pd
rop_(25,24),$pdep_(1,2),$pdep_(2,1),$pdep_(1,5),$pdep_(5,1),$pdep_(1,6)
,$pdep_(6,1),$pdep_(1,21),$pdep_(21,1),$pdep_(24,4),$pdep_(4,24),$pdep_
(24,19),$pdep_(19,24),$pdep_(24,23),$pdep_(23,24),$pdep_(24,25),$pdep_(
25,24)"

      # Reprograma a chamada da função de atualização para o período
especificado
      $ns at [expr $now+0.1] "atualiza_tab custoi custoa Sink_"

#fim da funcao atualiza
}

#=================================================================
# procedure que executa a Rotina de DIJKSTRA
# sr = source = nó de origem
# ds = destination = nó de destino
# no_nodes (numero de nós)
# custoa é a matriz de custos dos enlaces
# caminhoIP é a string que receberá o caminho na rede IP
# nucleo é a matriz para identificar se um nó é IP ou MPLS

proc calculaDJ { sr ds no_nodes custoa caminhoIP nucleo} {
      # passa a matriz 'custoa' para a matriz 'a'
      # essa passagem de parametros é necessária devido aos mecanismos
internos do NS
      # mudanças nas matrizes internas às funções são repassadas às
matrizes externas
      upvar $custoa a

      # passa a matriz 'caminhoIP' para a matriz 'caIP'
      upvar $caminhoIP caIP

      # passa a matriz 'nucleo' para a matriz 'nucleoRT'
      upvar $nucleo nucleoRT

      global caminho gateway contaMPLS nz

      set Infinito 100000
      for {set x 0} {$x < 26} {incr x} {
            set state($x,pre) -1
            set state($x,length) $Infinito
            set state($x,rotulo) "tent"
      }
      set i 0
      set ii 0
      set k 0
      set p 0
      set state($ds,length) 0
      set state($ds,rotulo) "perm"
      set k $ds

      while {$k != $sr} {
```

```
            for {set i 0} {$i < [expr $no_nodes + 1]} {incr i} {
                  if {$a($k,$i) != 0} {
                        if {$state($i,rotulo) eq "tent"} {
                              if {[expr $state($k,length) + $a($k,$i)]
< $state($i,length)} {
                                    set state($i,pre) $k
                                    set state($i,length) [expr
$state($k,length) + $a($k,$i)]
                              }
                        }
                  }
            }

            set k 0
            set min $Infinito
            for {set ii 0} {$ii < [expr $no_nodes + 1]} {incr ii} {
                  if {$state($ii,rotulo) eq "tent"} {
                        if {$state($ii,length)< $min} {
                              set min $state($ii,length)
                              set k $ii
                        }
                  }
            }
            set state($k,rotulo) "perm"

      }

      set i 0
      set k $sr
      set saltos 0
      # A variavel saltos conta o numero de nós por onde o caminho
passa

      while {$k >= 0} {
            set path($i) $k
            set k $state($k,pre)
            puts "$path($i) \n"
            set i [expr $i + 1]
            set saltos [expr $saltos +1]
      }
      set path($i) $k
      puts $nz "$saltos"

      # Aqui começa a organizar a variavel bst
      # que guarda o melhor caminho MPLS (antes em path)
      set i 0
      set bst "$path($i)"
      set gateway "$path(0)"
      # Gateway é o router entre as redes IP e MPLS

      # Variavel que conta o numero de routers MPLS
      set contaMPLS 1

      # j é o ponteiro para caminho IP
      set j 0

      # i é o ponteiro para a variável path
```

```
	set i 1

	# necessario para identificar se a rota é toda MPLS
	set caIP(0) -1

	while {$x != -1} {
		set x $path($i)

		if {$x > -1} {
		#Se o ponteiro é > -1 faz parte do caminho
			if { $nucleoRT($x) == "MPLS" } {
			#Se for MPLS põe no caminho MPLS
				set bst "$bst\_$x"
				set gateway $x
				set contaMPLS [expr $contaMPLS + 1]
			} else {
				set caIP($j) $x
				set j [expr $j + 1]
			}
		} elseif { $x == -1 } {
			set caIP($j) $x
			set j [expr $j + 1]
		}

		set i [expr $i + 1]
	}

	#################################################################
	# Imprime o caminho MPLS
	puts "$bst"
	set caminho $bst

	#################################################################
	# Imprime o caminho IP
	set j 0
	set t 0
	while {$t != -1} {
		set t $caIP($j)
		puts "$t"
		set j [expr $j + 1]
	}
#Fim da rotina de Dijkstra
}


# Rotina que cria os caminhos na rede MPLS
proc cria_caminho { lsrorig noegress lspid nodest tempo caminhoIP n_
LSR} {
	global caminho ns nc gateway contaMPLS

	# passagem de matrizes externas para matrizes internas à função
	# mudanças nas matrizes internas são automaticamente repassadas
às externas
	upvar $caminhoIP camIP
	upvar $n_ no
	upvar $LSR LSRG
	#puts $nc "$caminho , $noegress , $lspid , $nodest , $tempo"
```

```
      puts "ContaMPLS == $contaMPLS"
      puts "Gateway  $gateway   =====   puts Caminho  $caminho"

      #Cria o caminho MPLS
      if { $contaMPLS > 1 } {
            # o gateway é usado como nó egress
            $ns at $tempo "[$lsrorig get-module MPLS] make-explicit-
route $gateway $caminho $lspid -1"
            $ns at [expr $tempo + 0.10] "[$lsrorig get-module MPLS]
flow-erlsp-install $nodest -1 $lspid"
      }

      #Cria o caminho IP
      set t 0
      if { $camIP(0) > -1  }  {
            $LSRG($gateway) add-route [$no($nodest) id] [[$ns link
$LSRG($gateway) $no($camIP(0))] head]
            set z 0
            while { $z > -1 } {
                  set z $camIP($t)
                  set u [expr $t + 1]
                  if { $camIP($t) > -1 }  {
                        if { $camIP($u) > -1 }  {
                              $no($camIP($t)) add-route [$no($nodest)
id] [[$ns link $no($camIP($t)) $no($camIP($u))] head]
                        }
                  }
                  set t [expr $t + 1]
            }
      }

}

#=========================================================
# Cria agentes LDP nos nós MPLS,
for {set i 1} {$i < 16} {incr i} {
        set a LSR($i)
         for {set j [expr $i+1]} {$j < 16} {incr j} {
                set b LSR($j)
                eval $ns LDP-peer $$a $$b
         }
         set m [eval $$a get-module "MPLS"]
         $m enable-reroute "drop"

}

#=========================================================
# Variaveis GLOBAIS
set ::caminho 1_2
set ::contaMPLS 0
set ::gateway 0

#=========================================================
# Cria a tabela de custos dos links
# Todos em Infinito = 100000
set Infinito 100000
```

```
for {set i 0} {$i < 26} {incr i} {
    for {set j 0} {$j < 26} {incr j} {
       set ::custoi($i,$j) $Infinito
       set ::custoa($i,$j) $Infinito

    }
}

#============================================================
#A variável nucleo distingue switches IP de MPLS (nós do núcleo da rede)

for {set g 1} {$g < 16} {incr g} {
      set ::nucleo($g) "MPLS"
}

for {set g 16} {$g < 26} {incr g} {
      set ::nucleo($g) "IP"
}
#============================================================


#============================================================
# Cria os receptores Sink_
for {set i 0} {$i < 162} {incr i} {
      set ::Sink_($i) 0
}

#============================================================
# Cria a matriz caminhoIP que receberá o caminho a ser
# roteado para um determinado tráfego na rede IP.
for {set b 0} {$b < 25} {incr b} {
      set ::caminhoIP($b) -1
}


#============================================================
# Cria a tabela de custos dos links
# tabela de referencia para os enlaces sem uso

set custoi(1,2) 10
set custoi(2,1) 10
set custoi(1,5) 10
set custoi(5,1) 10
set custoi(1,6) 10
set custoi(6,1) 10
set custoi(1,21) 10
set custoi(21,1) 10

set custoi(2,3) 10
set custoi(3,2) 10
set custoi(2,7) 10
set custoi(7,2) 10
set custoi(2,22) 10
set custoi(22,2) 10

set custoi(3,4) 10
set custoi(4,3) 10
```

```
set custoi(3,8) 10
set custoi(8,3) 10
set custoi(3,23) 10
set custoi(23,3) 10

set custoi(4,5) 10
set custoi(5,4) 10
set custoi(4,9) 10
set custoi(9,4) 10
set custoi(4,24) 10
set custoi(24,4) 10

set custoi(5,10) 10
set custoi(10,5) 10
set custoi(5,25) 10
set custoi(25,5) 10

set custoi(6,7) 10
set custoi(7,6) 10
set custoi(6,10) 10
set custoi(10,6) 10
set custoi(6,11) 10
set custoi(11,6) 10

set custoi(7,8) 10
set custoi(8,7) 10
set custoi(7,12) 10
set custoi(12,7) 10

set custoi(8,9) 10
set custoi(9,8) 10
set custoi(8,13) 10
set custoi(13,8) 10

set custoi(9,10) 10
set custoi(10,9) 10
set custoi(9,14) 10
set custoi(14,9) 10

set custoi(10,15) 10
set custoi(15,10) 10

set custoi(11,12) 10
set custoi(12,11) 10
set custoi(11,15) 10
set custoi(15,11) 10
set custoi(11,16) 10
set custoi(16,11) 10

set custoi(12,13) 10
set custoi(13,12) 10
set custoi(12,17) 10
set custoi(17,12) 10

set custoi(13,14) 10
set custoi(14,13) 10
set custoi(13,18) 10
```

```
set custoi(18,13) 10

set custoi(14,15) 10
set custoi(15,14) 10
set custoi(14,19) 10
set custoi(19,14) 10

set custoi(15,20) 10
set custoi(20,15) 10

set custoi(16,17) 10
set custoi(17,16) 10
set custoi(16,20) 10
set custoi(20,16) 10
set custoi(16,21) 10
set custoi(21,16) 10

set custoi(17,18) 10
set custoi(18,17) 10
set custoi(17,22) 10
set custoi(22,17) 10

set custoi(18,19) 10
set custoi(19,18) 10
set custoi(18,23) 10
set custoi(23,18) 10

set custoi(19,20) 10
set custoi(20,19) 10
set custoi(19,24) 10
set custoi(24,19) 10

set custoi(20,25) 10
set custoi(25,20) 10

set custoi(21,22) 10
set custoi(22,21) 10
set custoi(21,25) 10
set custoi(25,21) 10

set custoi(22,23) 10
set custoi(23,22) 10

set custoi(23,24) 10
set custoi(24,23) 10

set custoi(24,25) 10
set custoi(25,24) 10

# Tabela de trabalho
# recebe o custo atualizado == custoa
# para os enlaces
set custoa(1,2) 10
set custoa(2,1) 10
set custoa(1,5) 10
set custoa(5,1) 10
set custoa(1,6) 10
```

```
set custoa(6,1) 10
set custoa(1,21) 10
set custoa(21,1) 10

set custoa(2,3) 10
set custoa(3,2) 10
set custoa(2,7) 10
set custoa(7,2) 10
set custoa(2,22) 10
set custoa(22,2) 10

set custoa(3,4) 10
set custoa(4,3) 10
set custoa(3,8) 10
set custoa(8,3) 10
set custoa(3,23) 10
set custoa(23,3) 10

set custoa(4,5) 10
set custoa(5,4) 10
set custoa(4,9) 10
set custoa(9,4) 10
set custoa(4,24) 10
set custoa(24,4) 10

set custoa(5,10) 10
set custoa(10,5) 10
set custoa(5,25) 10
set custoa(25,5) 10

set custoa(6,7) 10
set custoa(7,6) 10
set custoa(6,10) 10
set custoa(10,6) 10
set custoa(6,11) 10
set custoa(11,6) 10

set custoa(7,8) 10
set custoa(8,7) 10
set custoa(7,12) 10
set custoa(12,7) 10

set custoa(8,9) 10
set custoa(9,8) 10
set custoa(8,13) 10
set custoa(13,8) 10

set custoa(9,10) 10
set custoa(10,9) 10
set custoa(9,14) 10
set custoa(14,9) 10

set custoa(10,15) 10
set custoa(15,10) 10

set custoa(11,12) 10
set custoa(12,11) 10
```

```
set custoa(11,15) 10
set custoa(15,11) 10
set custoa(11,16) 10
set custoa(16,11) 10

set custoa(12,13) 10
set custoa(13,12) 10
set custoa(12,17) 10
set custoa(17,12) 10

set custoa(13,14) 10
set custoa(14,13) 10
set custoa(13,18) 10
set custoa(18,13) 10

set custoa(14,15) 10
set custoa(15,14) 10
set custoa(14,19) 10
set custoa(19,14) 10

set custoa(15,20) 10
set custoa(20,15) 10

set custoa(16,17) 10
set custoa(17,16) 10
set custoa(16,20) 10
set custoa(20,16) 10
set custoa(16,21) 10
set custoa(21,16) 10

set custoa(17,18) 10
set custoa(18,17) 10
set custoa(17,22) 10
set custoa(22,17) 10

set custoa(18,19) 10
set custoa(19,18) 10
set custoa(18,23) 10
set custoa(23,18) 10

set custoa(19,20) 10
set custoa(20,19) 10
set custoa(19,24) 10
set custoa(24,19) 10

set custoa(20,25) 10
set custoa(25,20) 10

set custoa(21,22) 10
set custoa(22,21) 10
set custoa(21,25) 10
set custoa(25,21) 10

set custoa(22,23) 10
set custoa(23,22) 10

set custoa(23,24) 10
```

```
set custoa(24,23) 10

set custoa(24,25) 10
set custoa(25,24) 10
#============================================================



#============================================================
# define a cor das mensagens ldp-label
$ns ldp-request-color       magenta
$ns ldp-mapping-color       red
$ns ldp-withdraw-color      green
$ns ldp-release-color       black
$ns ldp-notification-color  yellow
#============================================================

#============================================================
# Define a estrategia de trigger, o modo de distribuição de rótulos
# e o esquema de distribuição e alocação de rótulos
Classifier/Addr/MPLS enable-control_driven
#Classifier/Addr/MPLS enable-on-demand
#Classifier/Addr/MPLS enable-ordered-control

# liga todos os traces para stdout
Agent/LDP set trace_ldp_ 1
Classifier/Addr/MPLS set trace_mpls_ 1
# usa 'List' como método de scheduling (agendamento) de eventos
$ns use-scheduler List

#=================================================================
# cria as fontes (Source=src) e receptores(Sink) e associa aos nós
#0
for {set s 0} {$s < 56} {incr s} {
      set n [expr $s + 50]
      set d [expr $s + 106]

      set Src_($s) [new Agent/CBR]
      $ns attach-agent $n_($n) $Src_($s)
      $Src_($s) set packetSize_ 400k
      $Src_($s) set interval_ 0.01

      #create sinks and attach to nodes
      set Sink_($d) [new Agent/LossMonitor]
      #set Sink48 [new Agent/LossMonitor]
      $ns attach-agent $n_($d) $Sink_($d)

      #connect traffic sources to sinks
      $ns connect $Src_($s) $Sink_($d)
}
#=================================================================


#=================================================================
# Cria o intervalo de tempo de 0.5
set time 0.5
set banda1 1000000
```

```
#=============================================================
# Chama pela primeira vez rotina que atualiza a tabela de custos
# As demais vezes são reprogramadas automaticamente pela própria
# rotina

$ns at 0.5 "atualiza_tab custoi custoa Sink_"


#=============================================================
# Início das rotinas MPLS
#=============================================================

## Calcula a rotina de Dijkstra sobre a tabela de custos atualizada
(custoa)
## Cria o melhor caminho entre os LSRs e também na rede IP
## Inicia o trafego

##proc calculaDJ { sr ds no_nodes custoa caminhoIP } {
$ns at 0.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
##proc cria_caminho { lsrorig noegress lspid nodest tempo } {
$ns at 0.52 "cria_caminho $LSR(1) 25 3600 106 0.52 caminhoIP n_ LSR"
$ns at 0.72 "$Src_(0) start"

$ns at 1.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
$ns at 1.52 "cria_caminho $LSR(1) 25 3601 107 1.52 caminhoIP n_ LSR"
$ns at 1.72 "$Src_(1) start"

$ns at 2.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
$ns at 2.52 "cria_caminho $LSR(1) 25 3602 108 2.52 caminhoIP n_ LSR"
$ns at 2.72 "$Src_(2) start"

$ns at 3.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
$ns at 3.52 "cria_caminho $LSR(1) 25 3603 109 3.52 caminhoIP n_ LSR"
$ns at 3.72 "$Src_(3) start"

$ns at 4.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
$ns at 4.52 "cria_caminho $LSR(1) 25 3604 110 4.52 caminhoIP n_ LSR"
$ns at 4.72 "$Src_(4) start"

$ns at 5.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
$ns at 5.52 "cria_caminho $LSR(1) 25 3605 111 5.52 caminhoIP n_ LSR"
$ns at 5.72 "$Src_(5) start"

$ns at 6.51 "calculaDJ 1 25 25 custoa caminhoIP nucleo"
$ns at 6.52 "cria_caminho $LSR(1) 25 3606 112 6.52 caminhoIP n_ LSR"
$ns at 6.72 "$Src_(6) start"
##

$ns at 7.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 7.52 "cria_caminho $LSR(2) 24 3607 113 7.52 caminhoIP n_ LSR"
$ns at 7.72 "$Src_(7) start"

$ns at 8.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 8.52 "cria_caminho $LSR(2) 24 3608 114 8.52 caminhoIP n_ LSR"
$ns at 8.72 "$Src_(8) start"
```

```
$ns at 9.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 9.52 "cria_caminho $LSR(2) 24 3609 115 9.52 caminhoIP n_ LSR"
$ns at 9.72 "$Src_(9) start"

$ns at 10.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 10.52 "cria_caminho $LSR(2) 24 3610 116 10.52 caminhoIP n_ LSR"
$ns at 10.72 "$Src_(10) start"

$ns at 11.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 11.52 "cria_caminho $LSR(2) 24 3611 117 11.52 caminhoIP n_ LSR"
$ns at 11.72 "$Src_(11) start"

$ns at 12.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 12.52 "cria_caminho $LSR(2) 24 3612 118 12.52 caminhoIP n_ LSR"
$ns at 12.72 "$Src_(12) start"

$ns at 13.51 "calculaDJ 2 24 25 custoa caminhoIP nucleo"
$ns at 13.52 "cria_caminho $LSR(2) 24 3613 119 13.52 caminhoIP n_ LSR"
$ns at 13.72 "$Src_(13) start"
##
$ns at 14.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 14.52 "cria_caminho $LSR(3) 23 3614 120 14.52 caminhoIP n_ LSR"
$ns at 14.72 "$Src_(14) start"

$ns at 15.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 15.52 "cria_caminho $LSR(3) 23 3615 121 15.52 caminhoIP n_ LSR"
$ns at 15.72 "$Src_(15) start"

$ns at 16.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 16.52 "cria_caminho $LSR(3) 23 3616 122 16.52 caminhoIP n_ LSR"
$ns at 16.72 "$Src_(16) start"

$ns at 17.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 17.52 "cria_caminho $LSR(3) 23 3617 123 17.52 caminhoIP n_ LSR"
$ns at 17.72 "$Src_(17) start"

$ns at 18.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 18.52 "cria_caminho $LSR(3) 23 3618 124 18.52 caminhoIP n_ LSR"
$ns at 18.72 "$Src_(18) start"

$ns at 19.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 19.52 "cria_caminho $LSR(3) 23 3619 125 19.52 caminhoIP n_ LSR"
$ns at 19.72 "$Src_(19) start"

$ns at 20.51 "calculaDJ 3 23 25 custoa caminhoIP nucleo"
$ns at 20.52 "cria_caminho $LSR(3) 23 3620 126 20.52 caminhoIP n_ LSR"
$ns at 20.72 "$Src_(20) start"
##
$ns at 21.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
$ns at 21.52 "cria_caminho $LSR(4) 22 3621 127 21.52 caminhoIP n_ LSR"
$ns at 21.72 "$Src_(21) start"

$ns at 22.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
$ns at 22.52 "cria_caminho $LSR(4) 22 3622 128 22.52 caminhoIP n_ LSR"
$ns at 22.72 "$Src_(22) start"

$ns at 23.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
```

```
$ns at 23.52 "cria_caminho $LSR(4) 22 3623 129 23.52 caminhoIP n_ LSR"
$ns at 23.72 "$Src_(23) start"

$ns at 24.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
$ns at 24.52 "cria_caminho $LSR(4) 22 3624 130 24.52 caminhoIP n_ LSR"
$ns at 24.72 "$Src_(24) start"

$ns at 25.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
$ns at 25.52 "cria_caminho $LSR(4) 22 3625 131 25.52 caminhoIP n_ LSR"
$ns at 25.72 "$Src_(25) start"

$ns at 26.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
$ns at 26.52 "cria_caminho $LSR(4) 22 3626 132 26.52 caminhoIP n_ LSR"
$ns at 26.72 "$Src_(26) start"

$ns at 27.51 "calculaDJ 4 22 25 custoa caminhoIP nucleo"
$ns at 27.52 "cria_caminho $LSR(4) 22 3627 133 27.52 caminhoIP n_ LSR"
$ns at 27.72 "$Src_(27) start"

##
$ns at 28.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 28.52 "cria_caminho $LSR(5) 21 3621 134 28.52 caminhoIP n_ LSR"
$ns at 28.72 "$Src_(28) start"

$ns at 29.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 29.52 "cria_caminho $LSR(5) 21 3622 135 29.52 caminhoIP n_ LSR"
$ns at 29.72 "$Src_(29) start"

$ns at 30.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 30.52 "cria_caminho $LSR(5) 21 3623 136 30.52 caminhoIP n_ LSR"
$ns at 30.72 "$Src_(30) start"

$ns at 31.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 31.52 "cria_caminho $LSR(5) 21 3624 137 31.52 caminhoIP n_ LSR"
$ns at 31.72 "$Src_(31) start"

$ns at 32.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 32.52 "cria_caminho $LSR(5) 21 3625 138 32.52 caminhoIP n_ LSR"
$ns at 32.72 "$Src_(32) start"

$ns at 33.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 33.52 "cria_caminho $LSR(5) 21 3626 139 33.52 caminhoIP n_ LSR"
$ns at 33.72 "$Src_(33) start"

$ns at 34.51 "calculaDJ 5 21 25 custoa caminhoIP nucleo"
$ns at 34.52 "cria_caminho $LSR(5) 21 3627 140 34.52 caminhoIP n_ LSR"
$ns at 34.72 "$Src_(34) start"

##
$ns at 35.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 35.52 "cria_caminho $LSR(10) 16 3621 141 35.52 caminhoIP n_ LSR"
$ns at 35.72 "$Src_(35) start"

$ns at 36.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 36.52 "cria_caminho $LSR(10) 16 3622 142 36.52 caminhoIP n_ LSR"
$ns at 36.72 "$Src_(36) start"
```

```
$ns at 37.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 37.52 "cria_caminho $LSR(10) 16 3623 143 37.52 caminhoIP n_ LSR"
$ns at 37.72 "$Src_(37) start"

$ns at 38.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 38.52 "cria_caminho $LSR(10) 16 3624 144 38.52 caminhoIP n_ LSR"
$ns at 38.72 "$Src_(38) start"

$ns at 39.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 39.52 "cria_caminho $LSR(10) 16 3625 145 39.52 caminhoIP n_ LSR"
$ns at 39.72 "$Src_(39) start"

$ns at 40.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 40.52 "cria_caminho $LSR(10) 16 3626 146 40.52 caminhoIP n_ LSR"
$ns at 40.72 "$Src_(40) start"

$ns at 41.51 "calculaDJ 10 16 25 custoa caminhoIP nucleo"
$ns at 41.52 "cria_caminho $LSR(10) 16 3627 147 41.52 caminhoIP n_ LSR"
$ns at 41.72 "$Src_(41) start"

##
$ns at 42.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 42.52 "cria_caminho $LSR(6) 20 3621 148 42.52 caminhoIP n_ LSR"
$ns at 42.72 "$Src_(42) start"

$ns at 43.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 43.52 "cria_caminho $LSR(6) 20 3622 149 43.52 caminhoIP n_ LSR"
$ns at 43.72 "$Src_(43) start"

$ns at 44.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 44.52 "cria_caminho $LSR(6) 20 3623 150 44.52 caminhoIP n_ LSR"
$ns at 44.72 "$Src_(44) start"

$ns at 45.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 45.52 "cria_caminho $LSR(6) 20 3624 151 45.52 caminhoIP n_ LSR"
$ns at 45.72 "$Src_(45) start"

$ns at 46.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 46.52 "cria_caminho $LSR(6) 20 3625 152 46.52 caminhoIP n_ LSR"
$ns at 46.72 "$Src_(46) start"

$ns at 47.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 47.52 "cria_caminho $LSR(6) 20 3626 153 47.52 caminhoIP n_ LSR"
$ns at 47.72 "$Src_(47) start"

$ns at 48.51 "calculaDJ 6 20 25 custoa caminhoIP nucleo"
$ns at 48.52 "cria_caminho $LSR(6) 20 3627 154 48.52 caminhoIP n_ LSR"
$ns at 48.72 "$Src_(48) start"

#=====================================================================
=
# Interrompe os tráfegos, um a cada segundo
# a partir do momento de 60 segundos
set tempo_parada 60
set n 48
while {$n > -1}   {
      $ns at $tempo_parada "$Src_($n) stop"
```

```
      set tempo_parada [expr ($tempo_parada + 1)]
      set n [expr ($n - 1)]
}

#for {set n 0} {$n < 56} {incr n} {
#     $ns at 300.0 "$Src_($n) stop"
#}



# Finaliza a simulação
$ns at 120.0 "finish"

# executa a simulação
$ns run
```