Pedro Dias de Oliveira Carvalho

Um Sistema de Engenharia de Tráfego Autogerenciável para Redes IP

Belo Horizonte

Dezembro de 2013

Pedro Dias de Oliveira Carvalho

Um Sistema de Engenharia de Tráfego Autogerenciável para Redes IP

Dissertação apresentada à Comissão Examinadora, designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Professor Luciano de Errico

Belo Horizonte

Dezembro de 2013

Dedico este trabalho a meus pais
que, desde a minha infância, incentivam minha curiosidade e interesse pela ciência.

AGRADECIMENTOS

Agradeço ao meu orientador, Prof. Luciano de Errico, pela confiança e oportunidade de trabalhar em seu grupo de pesquisa e pela orientação e acompanhamento que se iniciou na graduação e se estende até hoje.

A todos os colegas que passaram pelo grupo LabCOM após a minha entrada e que certamente deixaram suas contribuições nas reuniões do grupo. Agradeço especialmente ao Orlewilson, Álvaro e John por suas valiosas contribuições na implementação do projeto.

Agradeço também aos que passaram pelo grupo antes de mim, Nilton, Alessandro e Arlindo, por estarem disponíveis a esclarecer dúvidas sobre seus trabalhos, que serviram de base e inspiração para o desenvolvimento deste.

Aos coordenadores, funcionários e professores do PPGEE pelo apoio e acompanhamento da minha vida acadêmica.

Aos orgãos de fomento pelo apoio financeiro: da FAPEMIG no desenvolvimento do projeto, e da CAPES na apresentação e divulgação do trabalho em evento científico.

Aos colegas de trabalho da RemOpt por entenderem a necessidade dos meus horários flexíveis e me ajudarem a conciliar trabalho e mestrado.

E aos amigos e familiares por me desejarem sucesso, por sentirem minha ausência e por compreenderem a razão da minha menor disponibilidade ao longo deste último ano de mestrado.

"Computers do the calculating
to allow people to transform the world."
(Conrad Wolfram)

Resumo

Embora consolidada entre as operadoras de telecomunicações, a oferta de um pacote de serviços que agrega telefonia, televisão e Internet ainda é entregue, na maioria das vezes, através de diferentes redes de telecomunicações. A convergência desses serviços em uma única rede permitiria a redução nos custos de implantação, expansão e manutenção. A crescente diversidade de equipamentos multimídia como netbooks, smartphones, tablets, smartTVs, óculos de realidade aumentada, entre outros, compatíveis com a rede IP, indicam que esta é a rede ideal para a convergência. Entretanto, os diferentes serviços somados à diversidade de dispositivos multimídia, geram um perfil de tráfego dinâmico, complexo e imprevisível, para o qual o atual esquema de gerência da rede se torna lento e ineficiente.

A proposta deste trabalho é um sistema de engenharia de tráfego autogerenciado, onde a operação e a manutenção da rede sejam realizadas com eficiência e sem a intervenção humana. Para avaliação de desempenho do sistema proposto, foi utilizado o simulador ns-2. Nesse ambiente foram configurados diferentes cenários, com fluxos de taxa variável (*Variable Bit Rate* - VBR), provenientes de vídeos reais, e fluxos de taxa constante (*Constant Bit Rate* - CBR). Os resultados mostram que quando os recursos disponíveis na rede são maiores do que a demanda dos serviços, o sistema autogerenciável é capaz de otimizar a alocação destes recursos, de forma a buscar atender toda a demanda com a qualidade necessária, distribuindo quanto possível a carga na rede. Já quando a demanda é maior do que os recursos de que a rede dispõe, o sistema busca alocar o máximo de demandas, sem prejudicar a qualidade de suas respectivas transmissões. Também foi avaliado o comportamento do sistema na ocorrência de uma falha de enlace, mostrando que o mesmo é capaz de reagir, sem a interferência humana, adaptando-se ao novo estado da rede para continuar atendendo os serviços afetados.

Abstract

The offer of a service bundle, which includes phone, television and Internet, is still delivered most often through separate telecommunications networks even though it is consolidated among telecom operators. The convergence of these services into a single network would save on costs of implementation, expansion and maintenance. The growing diversity of multimedia devices such as netbooks, smartphones, tablets, smartTVs and augmented reality glasses, among others compatible with the IP networks is an indication that this is the ideal network for convergence. However, different services in addition to the diversity of multimedia devices generate a dynamic, complex and unpredictable traffic profile, for which the current network management scheme becomes slow and inefficient.

The proposal of this work is a self-managed traffic engineering system in which the network operation and maintenance are performed efficiently and without human intervention. The ns-2 simulator was used for the performance evaluation of the proposed system. Different scenarios were configured in this environment with Variable Bit Rate (VBR) flows from real videos and Constant Bit Rate (CBR) flows. The results show that when the available resources in the network are greater than the services demand, the self-managing system is able to optimize the resources allocation in order to meet the demand with the required quality, distributing the network load when possible. However, when demand is greater than the resources that the network can provide, the system seeks to maximize demand allocation without sacrificing the broadcast quality. The behavior of the system in the occurrence of a link failure was also assessed and showed that the system is capable of reacting without human interference, adapting to the new network state to continue delivering the affected services.

Lista de ilustrações

Lista de tabelas

LISTA DE ABREVIATURAS E SIGLAS

CBQ	Class-Based Queuing
CBR	Constant Bit Rate
FIFO	First In First Out
FMM	Fuzzy Mixed Metric
FTP	File Transfer Protocol
GA	Genetic Algorithm
HD	High Definition
IP	Internet Protocol
ISDB-T	Integrated Services Digital Broadcasting Terrestrial
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switched Path
LSR	Label Switching Router
MHA	Minimum Hop Algorithm
MIRA	Minimum Interference Routing Algorithm
MOPS	Multi-Objective Path Selection
MPLS	Multiprotocol Label Switching

ns-2	Network Simulator 2
OSPF	Open Shortest Path First
ppm	parte por milhão
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RED	Random Early Detection
SD	Standard Definition
SFQ	Stochastic Fair Queuing
SLA	Service-Level Agreement
SMM	Single Mixed Metric
SRS	Stochastic Remainder Sampling
TCL	Tool Command Language
TCP	Transmission Control Protocol
TE	Traffic Engineering
VBR	Variable Bit Rate
VoD	Video on Demand
VPN	Virtual Private Network

Sumário

CAPÍTULO 1

INTRODUÇÃO

Observou-se nos últimos anos uma corrida das empresas de telecomunicações de todo o mundo pela oferta de pacotes conhecidos como *triple play* e que incluem os serviços de televisão, telefonia e Internet. Companhias que se encontravam antes em nichos separados, como telefonia, provedores de TV a cabo, ou Internet banda larga, passaram então a fazer parte do mesmo negócio. Operadoras de telefonia passaram a oferecer TV a cabo, e operadoras de TV a cabo passaram a ofertar serviços de telefonia e banda larga. Embora a maioria das empresas de telecomunicações já ofertem pacotes *triple play*, a entrega dos serviços ainda é feita por infraestruturas diferentes. O serviço de televisão é geralmente entregue via satélite ou cabo por meio de tecnologias de *broadcast* como a Transmissão Digital de Serviços Integrados Terrestres (*Integrated Services Digital Broadcasting Terrestrial* - ISDB-T). O serviço de telefonia é entregue pela Rede Pública de Telefonia Comutada (*Public Switched Telephone Network* - PSTN) e em alguns poucos casos já se observa a entrega via sistemas de voz sobre IP (*Voice over IP* - VoIP).

A convergência de todos os serviços em uma mesma infraestrutura tem sido alvo da indústria de telecomunicações, visto as vantagens das redes convergentes na redução dos custos totais de rede, pois permite o compartilhamento da operação, da administração, e manutenção de equipamentos. Além de poder transmitir voz, dados, imagens, som e vídeo, e com isso aumentar as receitas através da oferta de novos serviços (NASSIF, 2004).

Entretanto para usufruir dos benefícios da convergência não basta transferir todos os serviços para uma única infraestrurura qualquer. As tecnologias de redes tradicionais foram projetadas para atender as necessidades de um serviço específico, como a PSTN que foi projetada para transmitir tráfego de voz, ou o ISDB-T projetado para a transmissão de vídeo por rádio difusão. Esses são dois exemplos de tecnologias que não poderiam

ser utilizadas para a convergência, pois assim como a PSTN não é adequada para a transmissão de vídeos, o ISDB-T não é adequado para o fornecimento do serviço de telefonia. Até mesmo a tecnologia de rede da Internet, a rede IP (*Internet Protocol*), não foi projetada inicialmente para atender as diferentes necessidades de cada serviço. Porém ao contrário das outras tecnologias, a rede IP é uma tecnologia aberta à adaptação e pode ser adequada para se alcançar a convergência.

Embora a rede IP tradicional possa transmitir voz, dados, imagens, som e vídeo, esta adota o modelo de serviço Melhor Esforço (*Best Effort*), que não garante a Qualidade de Serviço (*Quality of Service* - QoS) necessária para cumprir com os níveis de serviço (*Service-Level Agreement* - SLA), como tempo máximo de indisponibilidade, tempo médio entre falhas, etc., geralmente presentes nos contratos de prestação de serviços de telecomunicação. Portanto os principais desafios da convergência dos serviços de telecomunicação na rede IP são prover um serviço confiável e aderente ao SLA, monitorar a QoS enquanto o serviço é provido e reagir em caso de desempenho insatisfatório e falhas (DéSOBLIN; PAPINI, 2001). Esses desafios são o foco deste trabalho, que busca resolvê-los através da engenharia de tráfego (*Traffic Engineering* - TE) (AWDUCHE et al., 1999) e de conceitos da computação autonômica (HORN; IBM, 2001). A engenharia de tráfego é a área que trabalha com a análise e otimização do desempenho de operação das redes IP. A computação autonômica, motivada pela crescente complexidade dos sistemas computacionais e inspirada no sistema nervoso autônomo, busca conferir a capacidade de autogerência e com isso minimizar a necessidade da interferência humana para o funcionamento do sistema. Ambos conceitos serão apresentados com mais detalhes no capítulo 2.

1.1 Objetivos

1.1.1 Geral

Este trabalho visa implementar e avaliar o desempenho de um sistema autogerenciável como solução para a Engenharia de Tráfego IP em um cenário de serviços multimídia. O sistema deve otimizar a gerência dos recursos de rede disponíveis de forma a garantir a QoS necessária para cada fluxo/serviço admitido e também maximar o número de admissões buscando uma maior eficiência de utilização dos recursos de rede.

1.1.2 Específicos

- Configurar um ambiente de simulação de redes autonômicas para que seja possível gerar não somente os resultados do sistema proposto neste trabalho, mas que também sirva de ferramenta para experimentação de futuros trabalhos nessa mesma linha de pesquisa.

- Implementar um mecanismo para controle de admissão que permita ao sistema aceitar ou recusar a alocação de recursos solicitada por cada fluxo/serviço.
- Implementar um módulo de medição, responsável por coletar informações de disponibilidade de recursos para cada enlace da rede, e também monitorar a QoS obtida para cada fluxo admitido.
- Implementar um módulo de otimização, responsável por sugerir os melhores conjuntos de rotas para um determinado conjunto de requisições de recursos.
- Implementar um mecanismo para garantia de QoS, responsável por detectar quando a QoS prometida não está sendo atingida e então propor soluções alternativas.
- Implementar um mecanismo para reagir a falhas de enlaces, responsável por ativar o módulo de otimização dado o novo estado da rede.
- Utilizar quando possível tecnologias já existentes no mercado, evitando uma renovação de toda a arquitetura da rede.
- Concentrar a complexidade e inteligência do sistema nos roteadores de borda, permitindo o uso de equipamentos mais simples no núcleo da rede.

1.2 Estrutura do Trabalho

Este trabalho está estruturado em cinco capítulos. No capítulo 2 são apresentados os conceitos básicos de QoS, engenharia de tráfego, computação autonômica e algoritmos genéticos. Ainda no capítulo 2 são apresentados os trabalhos relacionados.

O sistema de TE autogerenciável proposto é apresentado no capítulo 3, no qual são detalhadas as funções e estruturas de cada módulo que compõe a arquitetura do sistema. Nesse capítulo também é apresentado o ambiente de simulação desenvolvido para implementar e testar o sistema proposto.

Experimentos e análise dos resultados são apresentados no capítulo 4. E, finalmente, no capítulo 5 são apresentadas as conclusões, considerações finais e trabalhos futuros.

CAPÍTULO 2

ENGENHARIA DE TRÁFEGO E AUTOGERENCIAMENTO

A abordagem proposta neste trabalho envolve conceitos não só de Qualidade de Serviço e Engenharia de Tráfego na rede IP, como também de Computação Autonômica e de Inteligência Computacional, mais especificamente nesta última sobre a classe de algoritmos genéticos. Este capítulo detalha estes conceitos e tecnologias.

2.1 Qualidade de Serviço

Qualidade de Serviço compreende todas as características de um serviço de telecomunicações que afetam a capacidade de satisfazer necessidades do usuário do serviço desde relações sociais, como a presteza do serviço de atendimento ao cliente, a parâmetros técnicos como a capacidade da infraestrutura fornecida. A QoS de um serviço pode ainda ser avaliada sob quatro diferentes pontos de vista: a QoS desejada pelo usuário, a oferecida pelo provedor de serviços, a alcançada pelo provedor e a percebida pelo usuário. A primeira e a última são geralmente formadas por parâmetros subjetivos, afetados por fatores psicológicos, e seu entendimento é fundamental para a escolha e otimização dos parâmetros objetivos do provedor (ITU-T, 2008).

No âmbito do desempenho de rede, ou seja da capacidade de operação e manutenção da infraestrutura, diversos trabalhos têm utilizado as seguintes métricas como principais parâmetros de QoS (CHEN; FARLEY; YE, 2004):

- tempo de resposta: tempo gasto entre o envio da requisição do usuário e o recebi-

mento da resposta;

- atraso: tempo gasto entre o envio do primeiro bit de um pacote de dados pelo nó emissor e o recebimento desse mesmo bit pelo nó receptor;
- *jitter*: variação do atraso de pacotes de um mesmo fluxo de dados;
- vazão: a taxa de bits por segundo de uma transmissão incluindo os dados de conteúdo e cabeçalho;
- taxa de perda: parcela da informação perdida durante a transmissão;

Alguns desses parâmetros são utilizados inclusive em recomendações de QoS para que serviços específicos sob a rede IP propiciem uma boa experiência ao usuário, como mostra a Tabela 1.

Tabela 1 – Recomendações de QoS.

Serviço	Banda (Mbps)	Atraso (ms)	Jitter (ms)	Perda
SDTV (MPEG-2)	3.0	<200	<50	<5.85 ppm
SDTV (MPEG-4)	1.75	<200	<50	<6.68 ppm
HDTV (MPEG-4)	8.0	<200	<50	<1.28 ppm
Vídeo conferência	0.460	<150	<30	<1%
VoIP (G.711 @ 50 pps)	0.093	<150	<30	<1%

Fonte: (PAUL, 2011) e (LEWIS; PICKAVANCE, 2006)

2.1.1 Modelos de Serviço

A rede IP tradicional adota o modelo de serviço Melhor Esforço (*Best Effort*) que não oferece garantias de QoS, apenas faz um esforço para transmitir a informação, sem compromisso com o tempo e a integridade dos dados. Nesse modelo os dados são enviados em qualquer quantidade, sem solicitar permissão ou informar a rede. Portanto, uma eventual falha de enlace ou mesmo o congestionamento da fila de um roteador poderia ocasionar a perda de parte da informação transmitida, sendo que o transmissor sequer é avisado da não entrega da informação. Esse tipo de problema é evitado por aplicações que se utilizam do protocolo de transporte TCP e seus mecanismos, que aumentam a confiabilidade e robustez da comunicação, garantindo pelo menos a entrega íntegra dos dados. Entretanto esse modelo TCP/IP atende a apenas um perfil de aplicação, o perfil de transferência de arquivos, no qual a integridade dos dados é muito mais importante do que o tempo gasto na sua transferência. De fato por vários anos esse foi o perfil de uso da Internet, que começou a mudar aos poucos na década de 90 e mais radicalmente na primeira década do terceiro milênio, quando o uso de aplicações em tempo real como

streaming de vídeo, conferência de voz e jogos *online*, cresceram rapidamente. Logo tornou-se evidente que tais aplicações necessitam de tratamentos diferenciados para se alcançar uma boa experiência de uso e com isso outros dois modelos de serviço ganharam destaque: o modelo integrado e o diferenciado.

No modelo integrado, cada aplicação solicita à rede uma QoS específica e inicia sua transmissão apenas após receber uma confirmação de que a rede dispõe dos recursos necessários e irá oferecer uma transmissão com a qualidade solicitada. Em contrapartida é esperado da aplicação que envie somente dados segundo o perfil de QoS aprovado. Assim que a transmissão é estabelecida, a rede se encarrega de monitorar e manter a QoS acordada (CISCO, 2011).

O modelo diferenciado também é capaz de atender diferentes perfis de QoS. Entretanto, diferentemente do modelo integrado, a aplicação não solicita explicitamente pela alocação de recursos específicos antes de iniciar a transmissão. Ao invés disso, os pacotes de uma determinada aplicação devem ser classificados em uma das classes de QoS pré-definidas no sistema. É com base nessa classificação que a rede então buscará fazer o melhor para transmitir aquele pacote (CISCO, 2011). Portanto o sucesso em atender a QoS necessária de uma aplicação depende de quão bem uma das classes pré-definidas a representa.

A implementação desses modelos de serviço é possível com a atuação de um sistema de engenharia de tráfego, explicado a seguir.

2.2 Engenharia de Tráfego IP

A Engenharia de Tráfego trabalha com a análise e a otimização do desempenho da operação de redes IP, utilizando princípios tecnológicos e científicos para medir, caracterizar, modelar e controlar o tráfego na rede (AWDUCHE et al., 1999). Seu principal objetivo é a melhoria do desempenho operacional, que se desdobra em diversos objetivos específicos como:

- garantir o nível de qualidade de serviço demandado pelas aplicações;
- melhorar a eficiência de utilização dos recursos de rede;
- aumentar a confiabilidade da rede, reduzindo erros e vulnerabilidades a falhas.

Tais objetivos podem ser alcançados através do gerenciamento da capacidade da rede e do gerenciamento do tráfego. O gerenciamento de capacidade envolve ações como o planejamento de capacidade da rede, o controle de roteamento na mesma e o gerenciamento de recursos como largura de banda, espaço em *buffers* e recursos computacionais.

Por outro lado, o gerenciamento do tráfego pode compreender atuações em condicionamento de tráfego (na entrada da rede) e em gerenciamento de filas e controle de escalonamento (nos roteadores) (AWDUCHE et al., 2002).

Os métodos de controle da TE podem ser reativos, quando se utilizam de informações passadas para corrigir e adaptar o atual estado da rede, ou podem ser pró-ativos, tomando ações preventivas que evitem estados indesejados, seja através da predição ou de ações que induzam a um estado desejado. Os métodos são ainda classificados quanto ao tempo de resposta, podendo ser lentos, médios ou rápidos. Por exemplo, o planejamento de capacidade e instalação da infraestrutura se enquadra no tempo de resposta lento, podendo levar dias ou até mesmo anos. Já mecanismos de roteamento possuem tempo de resposta médio, atuando na casa dos milisegundos. E finalmente existem os métodos rápidos, que atuam em pico ou nanosegundos, como o gerenciamento de fila.

Para a entrega de um serviço é necessário estabelecer pelo menos um fluxo, sendo que fluxo é uma sequencia de pacotes que possuem a mesma origem e destino. Os enlaces que um pacote percorre, durante seu trajeto até o destino, afetam diretamente a QoS percebida pelo usuário, pois cada enlace pode possuir diferentes recursos disponíveis. Portanto se os pacotes de um mesmo fluxo passam por caminhos diferentes até atingir seu destino, a QoS percebida pelo usuário será o resultado de uma complexa combinação da QoS oferecida por cada um dos caminhos. Por outro lado se todos os pacotes de um mesmo fluxo percorrem o mesmo caminho, a relação entre a QoS oferecida pelo caminho e a QoS percebida pelo usuário se torna muito mais simples. Assim a busca por uma solução que atenda a solicitação de QoS de uma aplicação passa a ser uma busca por um caminho que disponha dos recursos necessários.

Porém, na rede IP cada roteador intermediário de um trajeto entre a origem e o destino encaminha o pacote com base no endereço IP de destino e em sua própria tabela de roteamento (PETERSON; DAVIE, 2003), que é construída por um protocolo de roteamento, sendo mais comum o protocolo *Open Shortest Path First* (OSPF). Os roteadores OSPF possuem uma base da dados que descrevem o estado de toda a rede, pois trocam entre si informações sobre os estados de seus enlaces e por isso é classificado como um protocolo de roteamento *Link State*. Todos os roteadores aplicam paralelamente um mesmo algoritmo sobre suas respectivas bases de dados e com isso calculam os caminhos mais curtos para qualquer outro roteador da rede (MOY, 1991).

Esse modelo de roteamento salto-a-salto não estabelece uma rota explícita na rede, o que pode ser obtido com auxílio da tecnologia *MultiProtocol Label Switching* (MPLS).

2.2.1 MPLS

A tecnologia MPLS permite o encaminhamento de pacotes através da utilização de rótulos cujos valores, de tamanho fixo, determinam o próximo salto do pacote. Para que isso seja possível dois roteadores vizinhos devem ser capazes de entrar em acordo quanto ao significado de um determinado rótulo. Há portanto uma negociação e distribuição de rótulos através de uma série de procedimentos executados por um protocolo de distribuição de rótulos (*Label Distribution Protocol* - LDP) (THOMAS; GRAY, 2001). Roteadores IPs convencionais não são capazes de executar esses procedimentos, são necessários para isso o uso de roteadores compatíveis com a tecnologia MPLS.

Em outras palavras os roteadores intermediários, conhecidos como *Label Switching Router* (LSR), não consultam suas próprias tabelas de roteamento e nem o endereço IP de destino, mas encaminham o pacote de acordo com o rótulo MPLS inserido pelo roteador anterior. Porém antes de encaminhar o pacote MPLS, um LSR remove o rótulo recebido e adiciona um novo que indicará ao roteador seguinte por qual caminho o pacote deverá ser encaminhado. Esses rótulos não existem em uma rede IP tradicional, portanto para que uma rede MPLS funcione, de forma interconectada a outras redes IP, são necessários os roteadores de borda conhecidos como *Label Edge Router* (LER). Esses são responsáveis por rotular o pacote IP que ingressa no domínio MPLS, e também por retirar o rótulo do pacote IP que deixa o domínio. É o valor desse rótulo, que o pacote recebe ao ingressar no domínio, que irá determinar por quais LSRs o pacote irá passar até sair do domínio. A essa sequência de LSRs que um pacote percorre no domínio MPLS é dada o nome de *Label Switched Path* (LSP) (GHEIN, 2007).

No exemplo ilustrado na Figura 1, um pacote IP ingressa em um domínio MPLS através de um LER que atribui o rótulo de valor 45 e encaminha o pacote para o próximo roteador do LSP. O LSR que recebe o pacote, identifica o rótulo 45 e então o troca pelo rótulo 32 antes de encaminhar ao próximo LSR, este por sua vez troca o rótulo 32 pelo valor 10 e encaminha ao último roteador do LSP que finalmente remove o rótulo do pacote e o encaminha para fora do domínio MPLS através da análise de seu endereço IP de destino. Essa troca de rótulos a cada salto é uma estratégia do LDP para simplificar e viabilizar a configuração dos caminhos, pois dessa forma a negociação quanto ao significado do valor de um rótulo é limitada a dois LSRs vizinhos. Utilizar um único valor de rótulo para todo o caminho tornaria o procedimento de configuração de caminhos muito ineficiente, pois seria necessário encontrar um valor que estivesse disponível em todos os LSRs do caminho. Tal busca envolveria muitos LSRs na negociação e prolongaria o tempo de configuração do caminho.

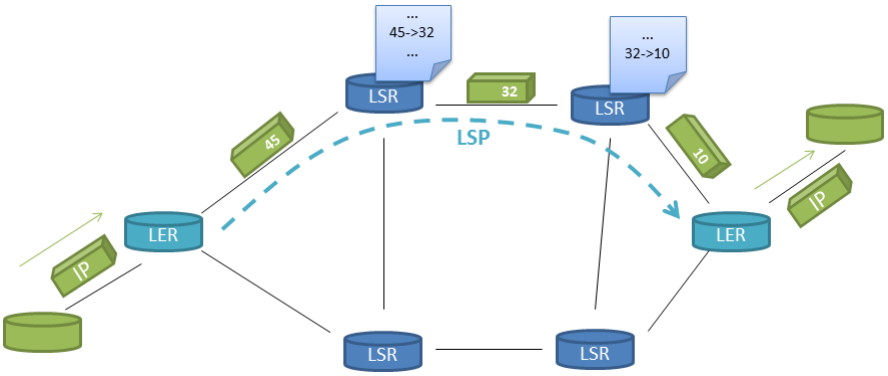


Figura 1 – Exemplo de um domínio MPLS e seus componentes.

O MPLS pode ser utilizado para diversos fins como a implementação de redes virtuais privadas (*Virtual Private Network* - VPN) ou para conectar subredes de diferentes arquiteturas. Neste trabalho é utilizado pela sua capacidade de criar rotas explícitas na rede e com isso o sistema de TE possa alocar aplicações em LSPs específicas que atendam aos requisitos de QoS. No entanto o tráfego IP não gerenciado pelo sistema de TE poderia congestionar os enlaces de uma LSP, degradando a QoS de uma aplicação prioritária. Isso pode ser evitado com a adoção de uma política de fila que favoreça a aplicação prioritária.

2.2.2 Políticas de Fila

Um pacote, quando recebido por um roteador, é processado para se determinar o destino de seu próximo salto e então colocado na fila de transmissão do enlace de destino. Esse tempo que o pacote espera na fila para ser transmitido contribui para o atraso total fim-a-fim, que é um parâmetro de QoS importante para muitos serviços. Portanto a forma como os pacotes de um fluxo são tratados na fila afeta diretamente a QoS percebida pelo usuário.

Uma fila é caracterizada por um tamanho máximo, um algoritmo de escalonamento e um método de descarte. O tamanho máximo determina o número máximo de pacotes que podem coexistir na fila e, portanto, se faz necessário definir uma ação para tratar os pacotes que encontram uma fila já totalmente cheia. Tal ação é determinada pelo método de descarte que pode optar por sempre eliminar o pacote que tenta entrar em uma fila cheia, como é de fato feito no modelo mais comum, o *DropTail*. Outro método de descarte é o algoritmo *Random Early Detection* (RED) que descarta pacotes com uma probabilidade que aumenta à medida que a fila enche. Essa estratégia, proposta em Floyd e Jacobson (1993), possui maior eficiência em evitar congestionamentos quando combinada com o mecanismo de controle de congestionamento do protocolo de transporte TCP, que controla sua taxa de transmissão com base na taxa de perdas de pacotes. Por outro lado possui a desvantagem de gerar perdas mesmo quando o enlace ainda não se

encontra congestionado.

Os algoritmos de escalonamento definem como os pacotes são tratados pela fila, sendo o modelo mais comum o *First In First Out* (FIFO), no qual o primeiro pacote a entrar é também o primeiro pacote a sair da fila. A única vantagem desse modelo é sua simplicidade de implementação, mas não há um tratamento justo já que a aplicação que enviar pacotes com maior frequência tem mais chances de ocupar a maior parte da fila. Foi pensando em um tratamento mais justo que o algoritmo *Stochastic Fair Queuing* (SFQ) foi proposto em McKenney (1990). Nesse algoritmo é implementado um número limitado de filas FIFO pelas quais os fluxos são distribuídos através de uma função *hash*. A vantagem dessa abordagem é que uma aplicação com taxa mais elevada irá interferir apenas nas transmissões das aplicações que dividem a mesma chave *hash*, enquanto os fluxos que se enquadram nas demais chaves continuarão suas transmissões sem sofrer impacto algum. Uma outra alternativa de escalonamento é o *Class-Based Queuing* (CBQ), proposto em Floyd e Jacobson (1995) e adotado nesse trabalho, que divide os fluxos em classes. A cada uma dessas classe é dedicada uma fila FIFO assim como no SFQ, porém no CBQ a distribuição dos fluxos não é aleatória, mas baseada no perfil de tráfego da aplicação. O tempo que cada fila FIFO tem para escoar seus pacotes também não é uniforme como no caso do SFQ e sim baseado em um nível de prioridade que cada classe possui, de forma que uma fila só poderá transmitir seus pacotes quando as filas com prioridades maiores não estiverem transmitindo. Há também a possibilidade de se aplicar restrições de banda, em cada classe, e com isso evitar a situação em que uma classe de baixa prioridade nunca consiga esvaziar sua fila porque outras classes ocupam o enlace o tempo todo. Em outras palavras, um fluxo classificado com a mais alta prioridade terá sempre a preferência de transmissão enquanto sua taxa for menor ou igual à restrição de banda de sua classe, de forma que a banda mínima requerida por um serviço seja garantida. O CBQ permite ainda que as classes sejam estruturadas de forma hierárquica, possibilitando que classes filhas compartilhem recursos de uma classe pai.

Essa seção apresentou alguns conceitos básicos da engenharia de tráfego com foco nas tecnologias MPLS e CBQ, por serem aquelas adotadas nesse trabalho. Foi também colocado que a tecnologia MPLS pode ser utilizada para definir caminhos explícitos na rede, facilitando a alocação e o controle dos recursos necessários para o oferecimento da QoS solicitada por uma aplicação. E que uma política de fila como o CBQ deve ser aplicada para evitar que os recursos alocados a uma aplicação prioritária sejam compartilhados com outras aplicações não gerenciadas pelo sistema de TE. Entretanto nada foi dito sobre como os caminhos ou LSPs são escolhidos. Essa escolha, como já foi dito, deve levar em consideração a demanda de QoS da aplicação e o nível de QoS que cada caminho pode oferecer. Muitas vezes essa análise de capacidade e configuração das LSPs é feita manualmente pelo gerente da rede, porém com o surgimento cada vez mais acelerado de novos perfis de tráfego somados à velocidade com que os estados de utilização dos enlaces de uma

rede sofrem alterações, tal gerência e configuração manual se torna demasiadamente lenta e ineficiente. Faz-se necessário portanto um sistema computacional que atue com rapidez e a todo momento na análise da demanda de QoS, assim como também na análise do estado da rede, e que seja capaz de configurar as LSPs sem a necessidade da intervenção humana. Os princípios básicos desse tipo de sistema são apresentados a seguir.

2.3 Sistemas Computacionais Autonômicos

Motivada pela crescente complexidade dos sistemas computacionais, a IBM lança em 2001 sua perspectiva sobre o que seria o maior desafio para a tecnologia da informação. A complexidade dos futuros sistemas computacionais chegaria a um nível tal que seres humanos não seriam capazes de operar e manter os sistemas com eficiência. É apresentado em Horn e IBM (2001) o conceito de sistema computacional autonômico, um sistema capaz de se autogerenciar com eficiência, inspirado no sistema nervoso autônomo. Ainda em Horn e IBM (2001) são definidas oito características necessárias para o funcionamento autonômico:

1. autoconhecimento: o sistema deve ter conhecimento de seus componentes, de suas capacidades e estado corrente (um sistema não pode gerenciar o que não conhece);
2. autoconfiguração: capacidade de se configurar, sem a intervenção humana, assim como adaptar dinamicamente suas configurações perante mudanças no ambiente;
3. auto-otimização: o sistema deve buscar continuamente por soluções melhores do que a atual no sentido de atender melhor seus objetivos pré-definidos;
4. autocura: capacidade de descobrir e reagir a falhas, contornando o problema através da redundância, reconfiguração ou redistribuição dos recursos;
5. autoproteção: um sistema autonômico deve estar preparado para responder a ataques digitais com sistemas de detecção e prevenção de intrusão;
6. sensibilidade ao contexto: o sistema deve procurar a melhor forma de interagir com sistemas vizinhos; semelhante à auto-otimização, mas voltado para o ambiente externo;
7. padronização: diante da heterogeneidade de dispositivos e sistemas que coexistem no mundo digital, um sistema autonômico não deve ser uma solução proprietária mas implementar um padrão aberto;
8. antecipação: capacidade de antecipar um estado de recursos otimizado para atender uma próxima decisão do usuário.

Desde a publicação da IBM inúmeros pesquisadores têm tentado definir diferentes níveis de autonomia de acordo com o subconjunto de características que um sistema apresenta. Como exemplo, Macedo (2012) propôs uma escala de quatro níveis na qual um sistema é classificado por sua capacidade adaptativa: sistemas não adaptativos, no nível um; sistemas adaptativos *offline*, ou sistemas incapazes de se adaptar em tempo de execução, no nível dois; no nível três estão os sistemas adaptativos *online* capazes de se adaptar em tempo real com base em objetivos pré-definidos; e finalmente no quarto e último nível os sistemas adaptativos autonômicos, capazes de adaptar suas próprias políticas e objetivos em tempo real.

Já em Truszkowski et al. (2009), essa capacidade de autogoverno (*self governance*) está associada aos sistemas autônomos (*autonomous systems*), que diferem dos sistemas autonômicos (*autonomic systems*) e sua capacidade de autogestão (*self management*). De fato, embora exista uma discordância quanto ao uso do termo autogoverno, na proposta inicial da IBM a definição das políticas e objetivos do sistema é de responsabilidade do projetista humano. O sistema autonômico não deve substituir a atividade humana de governa-lo, mas sim buscar atender os objetivos que lhe foram impostos com eficiência através da gerência de seus recursos.

Do conceito de computação autonômica, derivou-se o conceito de rede autonômica (*autonomic networking*), caracterizada pela capacidade de se recuperar de falhas e agilidade em se adaptar às mudanças do ambiente de rede através da auto-otimização. Espera-se também que a capacidade de autocontrole e gerência dos recursos ajude a superar a complexidade e heterogeneidade das redes de comunicação atuais (DENKO; YANG; ZHANG, 2009).

Como apresentado nessa seção, um sistema autonômico é composto, dentre outras coisas, de um algoritmo de otimização. Nesse trabalho é adotado, como otimizador, um Algoritmo Genético (*Genetic Algorithm* - GA) e portanto seus conceitos, nomenclatura e características básicas são apresentadas na seção seguinte. Algumas alternativas ao GA são apresentadas na seção de trabalhos relacionados. A escolha de uma heurística, como o GA, deve-se ao fato do problema, de encontrar caminhos sujeitos a qualquer combinação de duas ou mais restrições, ser NP-completo como mostrado em Wang e Crowcroft (1996). O que implica que a solução ótima é desconhecida e não pode ser encontrada por métodos exatos em tempo computacional razoável.

2.4 Algoritmos Genéticos

Essa seção descreve os conceitos básicos de algoritmos genéticos em geral. Detalhes de implementação do GA utilizado nesse trabalho são apresentados no capítulo 3.

Desenvolvido inicialmente por Holland (1989), o Algoritmo Genético é um método

de otimização e faz parte da família de métodos da computação natural que se baseiam na teoria da evolução das espécies de Charles Robert Darwin. Assim como indivíduos mais aptos tendem a prevalecer sobre os menos aptos, nos algoritmos genéticos as melhores soluções tendem a predominar a população de soluções candidatas a um problema de otimização.

Em um GA uma solução candidata é codificada de acordo com a modelagem do problema, sendo tal solução codificada denominada indivíduo ou cromossomo. Dentro do espaço de busca são avaliados diversos indivíduos simultaneamente a cada iteração e a esse conjunto de indivíduos é atribuido o nome de população. O papel da avaliação de cada indivíduo é atribuir uma nota de aptidão que represente quão bom é aquele indivíduo para a solução do problema.

A forma como uma população inicial evolui através das gerações depende dos operadores genéticos de seleção, cruzamento e mutação empregados. A natureza do problema deve ser levada em consideração na escolha desses métodos que influenciam diretamente no sucesso do algoritmo (VASCONCELOS et al., 1995).

Algoritmos de seleção determinam quais indivíduos de uma população devem passar seu material genético para a próxima geração. Esse processo geralmente possui um fator aleatório, pois uma seleção determinística dos mais aptos tende a causar a convergência prematura para um ótimo local. Existem diversos métodos de seleção, uns mais aleatórios que outros, tais como o método da roleta em que a chance de qualquer indivíduo ser selecionado é proporcional a sua aptidão (SRINIVAS; PATNAIK, 1994). O método do torneio no qual indivíduos escolhidos aleatoriamente participam de uma competição em que o indivíduo de maior aptidão tem maior probabilidade de vencer e por consequência ser selecionado (CHAKRABORTY; CHAKRABORTY, 1997). E o método *Stochastic Remainder Sampling* (SRS) em que a aptidão de cada indivíduo é dividia pela aptidão média da população e aqueles indivíduos cujos resultados forem maior do que um são selecionados. Possíveis vagas restantes são preenchidas pelo método da roleta, sendo que a probabilidade de um indivíduo ser escolhido é proporcional à parte fracionária do resultado da razão entre sua aptidão e a aptidão média da população (ANDRADE et al., 2008).

Os métodos de cruzamento são os responsáveis pela exploração direcionada do espaço, através da troca de material genético entre indivíduos. Dentre os vários métodos de cruzamentos, alguns exemplos são: cruzamento em um ponto, cruzamento em dois pontos, e cruzamento uniforme (ANDRADE et al., 2008). Os métodos diferem na forma como é feita a troca do material genético, sendo realizada por trechos de genes que são determinados por um ponto que separa o cromossomo em duas partes, no caso do primeiro método, ou dois pontos que separam o cromossomo em três partes, no segundo método. No método de cruzamento uniforme cada gene tem 50% de chance de se tornar um ponto de

corte. Já os métodos de mutação têm por objetivo aumentar a diversidade da população e garantir a exploração de regiões aleatórias do espaço de soluções, que não foram incluídas na população inicial (HOLLAND, 1989).

Os GAs diferem dos métodos clássicos de otimização em quatro aspectos principais (GOLDBERG, 1989):

1. GAs não trabalham diretamente com os parâmetros, mas com uma codificação destes;
2. GAs trabalham simultaneamente com uma população de soluções candidatas, ao invés de apenas uma solução a cada iteração;
3. GAs não utilizam derivadas ou gradientes de funções, mas o conceito de custo e aptidão para avaliar um ponto do espaço de busca;
4. GAs são algoritmos estocásticos, baseiam-se em regras probabilísticas e aleatórias.

O custo computacional do GA varia de acordo com os métodos empregados, mas é afetado principalmente pelo número de gerações e número de indivíduos na população. A escolha do número de indivíduos deve levar em consideração a complexidade do problema assim como o tempo computacional disponível. Quanto maior o número de indivíduos mais rápida é a exploração do espaço de busca e também maior é o tempo computacional gasto (ANDRADE, 2008). Definir um número máximo de gerações permite que o GA funcione dentro de um tempo computacional disponível, interrompendo o algoritmo mesmo em situações em que os critérios de convergência não sejam alcançados.

Embora o GA não garanta a solução ótima global, tem como principal vantagem, sobre outros métodos, o melhor desempenho computacional e a simplicidade de implantação (GARCIA; GARCIA; FRIEDMANN, 2000).

2.5 Trabalhos Relacionados

O problema de alocação de LSPs para provimento de QoS e maior eficiência de utilização da rede foi tratado por outros autores.

Em sua pesquisa Majd e Yaghmaee (2006) propõem a alocação de uma LSP por fluxo através da busca do caminho de menor custo com um algoritmo de Dijkstra adaptado, nomeado *Fuzzy Mixed Metric* (FMM). O FMM se utiliza de uma lógica nebulosa para combinar os custos relativos a banda disponível, atraso e taxa de perdas em uma só métrica. Embora três parâmetros de QoS sejam considerados na busca, a aplicação pode solicitar apenas um requisito, a banda mínima. Não há garantias portanto quanto a um atraso máximo ou taxa de perdas, mas sim uma tentativa de minimizar esses dois últimos

parâmetros. Os resultados desse trabalho mostram que o FMM apresenta maior eficiência quanto a utilização da rede e redução de perdas, embora possua complexidade computacional similar às variações de Dijkstra com restrição de banda e minimização de atraso proposta por Wang e Crowcroft (1996) e o *Single Mixed Metric* (SMM) com restrição de banda e minimização da soma dos parâmetros de atraso e perdas proposto por Costa e Duarte (1999).

Pant e Sanguankotchakorn (2010) buscaram combinar os conceitos do *Minimum Hop Algorithm* (MHA), que procura preservar os recursos da rede através da minimização do número de saltos, com os do *Minimum Interference Routing Algorithm* (MIRA) (KAR; KODIALAM; LAKSHMAN, 2000), que procura soluções que minimizem a interferência com requisições futuras. Entretanto a execução sob demanda do MIRA tornaria muito lento o tempo de resposta para alocar uma requisição, portanto os autores propõem um algoritmo dividido em duas etapas. Uma primeira etapa *offline* faz um pré-processamento de todos os caminhos da rede, no qual é calculada a criticalidade de cada caminho. O cálculo da criticalidade é uma média ponderada de três métricas:

- o balanço de carga que considera a banda disponível em cada enlace;
- o potencial de um caminho para receber futuras requisições, que considera a largura de banda do caminho e o número de caminhos alternativos com capacidades inferiores;
- a preservação de recursos, que considera o número de enlaces que formam o caminho;

Em uma segunda etapa *online*, i.e executada no momento em que é feita uma requisição de transmissão, procura-se dentre os K caminhos menos críticos aquele que minimiza a interferência em possíveis requisições futuras. A métrica de interferência de um caminho é calculada com base na queda do fluxo máximo que a solução provocaria, e também da probabilidade de novas requisições necessitarem desse mesmo caminho. O fluxo máximo representa a vazão total entre dois nós considerando a soma da banda disponível em cada caminho que interliga esses nós. Esse algoritmo nomeado *Multi-Objective Path Selection* (MOPS) foi testado e comparado, por seus autores, com o MHA e MIRA por meio de simulações. Nos resultados o MOPS obteve maior vazão total na rede e menor probabilidade de bloqueio das requisições.

Em um trabalho mais recente Kulkarni, Sharma e Mishra (2012) propõem a aplicação do algoritmo de Dijkstra com uma função de custo inversamente proporcional à largura de banda residual e proporcional ao nível de criticalidade do enlace. Esta criticalidade é calculada através da razão entre a capacidade máxima de largura de banda e o número total de caminhos que possam utilizar o enlace. Procura-se com isso evitar a utilização dos enlaces críticos da rede e minimizar o número de requisições negadas. Cami-

nhos que não atendam aos requisitos de banda mínima e atraso máximo especificados pela aplicação são descartados e no caso de não haver um caminho que atenda aos requisitos, a alocação é negada. Os resultados obtidos com esse algoritmo, através da simulação de diferentes topologias e demandas, mostraram melhor desempenho na redução de rejeições e no tempo computacional em redes mais complexas.

Um sistema de engenharia de tráfego completo com controle de admissão, alocação de LSPs, manutenção da QoS e reação a falhas na rede é proposto em Maia (2006). Nesse trabalho a alocação de LSPs é também por fluxo, porém não necessariamente um fluxo por vez. O sistema de TE proposto é capaz de considerar a demanda de uma ou mais aplicações simultaneamente e alocar de uma só vez um conjunto de LSPs. Para isso o sistema avalia e seleciona, dentre os k caminhos mais curtos, aqueles caminhos que possuem banda disponível maior do que a banda mínima desejada e atraso menor do que o atraso máximo requerido por cada aplicação. A cada caminho selecionado é atribuído um custo que representa quão bem o caminho candidato atende a demanda de sua respectiva aplicação. O valor do custo é resultado de uma lógica nebulosa que considera o tamanho da folga entre a banda disponível e a banda mínima desejada assim como a folga entre o atraso máximo permitido e o atraso característico do caminho. Quanto maiores são as folgas, menor é o custo nebuloso e portanto melhor é o caminho. As aplicações são então alocadas uma a uma em seus respectivos melhores caminhos conforme suas prioridades, sendo que a cada alocação é atualizada a banda disponível dos enlaces que compõem o caminho alocado. Portanto uma alocação pode alterar o estado do melhor caminho das aplicações seguintes, de menor prioridade. Caso o melhor caminho candidato de uma aplicação não possua mais a disponibilidade da banda requerida, é verificado o segundo melhor caminho e assim por diante. Se não houver caminho que atenda aos requisitos, a aplicação é bloqueada. Dessa forma esse sistema não apenas tenta maximizar a QoS, mas garante que se uma aplicação é alocada então sua QoS mínima é atendida. Após a alocação o sistema continua monitorando periodicamente a QoS alcançada pela aplicação, realocando-a se necessário. No ato da alocação são definidas também LSPs reservas que são acionadas em resposta a falha em enlaces da LSP principal. O sistema de TE conta ainda com uma otimização periódica a médio prazo, com algoritmo genético, que realoca todos os fluxos correntes em busca de um uso mais eficiente dos recursos da rede. Experimentos em diferentes topologias mostraram que o sistema de TE proposto, quando comparado ao OSPF tradicional, obteve ganhos significativos quanto a eficiência de utilização da rede e principalmente na manutenção da QoS solicitada, que foi atendida até mesmo na ocorrência de falhas de enlace da rede. Esse trabalho foi utilizado como base e inspiração para o desenvolvimento do sistema de TE que será apresentado no capítulo 3.

Em Andrade (2008) é proposto um algoritmo genético como sistema de alocação centralizado, capaz de alocar múltiplas LSPs entre quaisquer nós de origem e destino. A codificação binária é adotada para representar o indivíduo, sendo que cada gene repre-

senta uma demanda de LSP. Os experimentos realizados consideram restrições de banda e atraso, e exploram um espaço de soluções que incluem os dois caminhos mais curtos para cada demanda, que chegam a até 500 LSPs em uma única alocação. A busca pelos dois caminhos mais curtos de cada demanda é realizada com a aplicação do algoritmo dos k-caminhos mínimos.

Também é proposta a alocação dinâmica de multiplas LSPs em Santos e Mateus (2009), mas com a aplicação de um algoritmo genético multiobjetivo que busca minimizar três parâmetros: a vazão do enlace mais utilizado, o número de requisições negadas e o número de enlaces utilizados. O objetivo com o primeiro parâmetro é balancear a carga na rede, com o segundo é maximizar o número de admissões e com o terceiro é diminuir o atraso médio das aplicações, pois segundo o autor quanto menos enlaces forem utilizados mais rapidamente os dados são transferidos da origem ao destino. Os três objetivos são concorrentes entre si, motivo pelo qual se optou pelo algoritmo multiobjetivo NSGA-II. É proposto que o indivíduo codificado armazene a informação de um caminho por requisição e, para que não sejam geradas soluções inválidas, a população inicial de caminhos é gerada com o algoritmo Dijkstra aplicado à topologia do problema com custos aleatórios atribuídos aos enlaces. Assim são gerados caminhos aleatórios e válidos que irão gerar sempre novos indivíduos também válidos através de uma operação de cruzamento adequada. Porém a mutação é realizada removendo-se um enlace de um caminho, o que o torna inválido e portanto se faz necessária a busca por novos caminhos válidos. Experimentos com multiplas requisições em diversas topologias mostram que o algoritmo proposto possui maior eficiência no balanceamento de carga e maximização das admissões do que na redução do atraso médio.

Os autores Xueshun et al. (2009) também optaram pelo algoritmo genético multiobjetivo. Porém nesse trabalho a modelagem proposta foi limitada a uma alocação por vez e as restrições de QoS não foram definidas, com exceção da restrição de banda. A solução proposta é portanto um algoritmo genérico para a busca de caminhos com multiplas restrições de QoS, além da restrição de banda. Um algoritmo de busca aleatória de caminhos em um grafo garante que a população inicial contenha apenas soluções válidas. A partir daí a população se mantém válida por todas as gerações através da aplicação de operadores genéticos de cruzamento e mutação, que também foram projetados para manter as soluções dentro do espaço viável. Todos as restrições de QoS são modeladas como objetivos e portanto consideradas na função de aptidão, com exceção da restrição de banda que é considerada em um pré-processamento que descarta todos os enlaces da rede com banda insuficiente. Portanto a solução proposta oferece garantia de banda mínima e minimização de multiplos parâmetros de QoS. Resultados de simulações mostraram que o algoritmo converge para a fronteira pareto-ótima, embora sua complexidade seja inferior a do NSGA-II.

CAPÍTULO 3

SISTEMA DE ENGENHARIA DE TRÁFEGO AUTOGERENCIÁVEL

A proposta deste trabalho é o desenvolvimento de um Sistema de Engenharia de Tráfego (Sistema de TE) autogerenciável para redes IP com tráfego multimídia. Este sistema deve ser capaz de, sem intervenção humana, gerenciar a rede de modo a garantir a QoS de cada fluxo de comunicação que atravessa a mesma, assim como maximizar a quantidade destes fluxos admitidos na rede. A complexidade e a inteligência do sistema estarão concentradas nos roteadores de borda, permitindo o uso de equipamentos mais simples no núcleo da rede. Este capítulo detalha a arquitetura deste sistema e a implementação realizada da mesma.

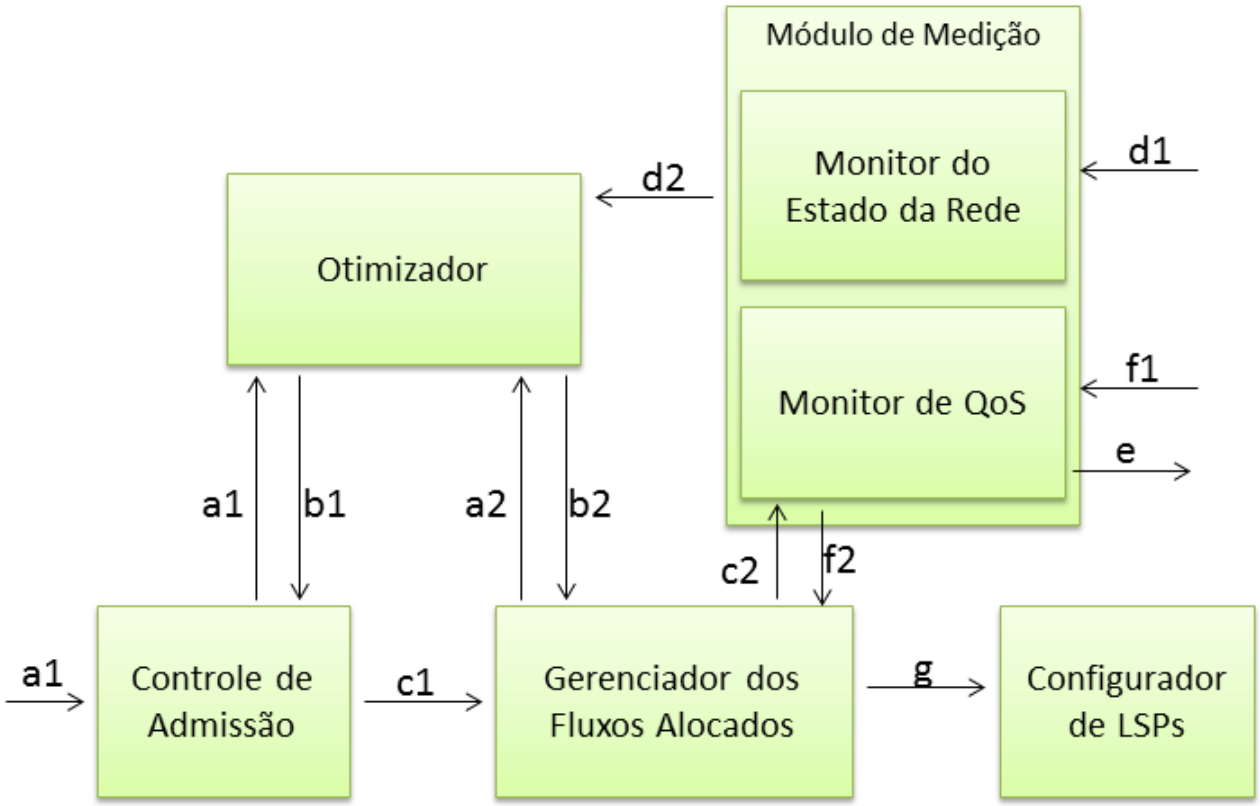
3.1 Arquitetura do Sistema

Com base nas premissas acima foi desenvolvida a arquitetura mostrada na Figura 2, composta por cinco módulos: módulo de controle de admissão, módulo otimizador, módulo de medição, módulo gerenciador de fluxos alocados e módulo configurador de LSPs. As aplicações devem colocar seus requisitos de QoS (vazão, atraso, *jitter* e perdas) ao módulo de controle de admissão. A demanda é repassada ao módulo otimizador, responsável por encontrar um conjunto de caminhos no domínio MPLS que atenda às necessidades das aplicações. O controle de admissão avalia a melhor solução encontrada, sendo que aquelas aplicações que não tiverem seus requisitos atendidos tem sua admissão negada. Caso contrário, a aplicação é admitida e seu fluxo é transmitido pelo caminho LSP estabelecido. O módulo gerenciador dos fluxos alocados monitora periodicamente,

através do módulo de medição, o nível de QoS alcançado. Quando é detectada uma QoS abaixo do acordado, o módulo otimizador é acionado para encontrar uma nova solução.

Aplicações que não solicitarem alocação de recursos são consideradas fluxos não prioritários e transmitidas por meio de *best effort*. A discriminação do tráfego prioritário e não prioritário é baseada no rótulo MPLS. Através da aplicação de filas CBQ, os pacotes rotulados pelo MPLS, e portanto prioritários, são enfileirados em uma fila de maior prioridade do que a fila dos pacotes IPs dos fluxos não prioritários. Para que exista sempre uma disponibilidade mínima para o tráfego não prioritário, o sistema de TE pode alocar apenas uma fração da capacidade máxima dos enlaces.

O sistema formado pelo conjunto de módulos apresentado na Figura 2 deve ser implementado em cada roteador de entrada do domínio MPLS. Assim o funcionamento da rede como um todo se torna distribuído, evitando-se portanto os pontos críticos de um sistema centralizado. Entretanto neste trabalho o sistema de TE é implementado somente em um roteador de borda. Como consequência desta limitação, é possível realizar apenas experimentos com cenários nos quais todas as requisições de QoS sejam realizadas em um mesmo roteador de entrada, como será mostrado no capítulo de experimentos.



Legenda:

a1) Demanda de QoS de novas requisições;
a2) Demanda de QoS de fluxos que precisam ser realocados;
b1) Conjunto solução de caminhos para novas requisições;
b2) Conjunto solução de caminhos para fluxos realocados;
c1) Demanda de QoS e caminhos dos novos fluxos admitidos;
c2) Demanda de QoS e caminhos dos fluxos alocados;

d1) Estatística de QoS dos enlaces de outros roteadores;
d2) Estado da rede (todos enlaces);
e) Solicitação de medição da QoS de um fluxo ao seu roteador de egresso;
f1) QoS de um fluxo mensurada por seu roteador de egresso;
f2) QoS alcançada pelos fluxos alocados;
g) Lista de novas LSPs;

a letra identifica o tipo de dado

a 1

o número identifica as diferentes instâncias

Figura 2 – Relação entre os módulos que compõem o sistema de TE.

3.1.1 Módulo de Medição

O módulo de medição é dividido em duas partes, o módulo de medição do estado de enlace e o módulo de medição de QoS fim-a-fim. O primeiro módulo está presente em todos os roteadores do domínio MPLS, capacitando cada roteador a coletar estatísticas de banda disponível, atraso, *jitter* e perdas, para todos os seus enlaces. A medição considera

apenas os pacotes de aplicações prioritárias e a atualização do estado da rede é propagada periodicamente para os roteadores de borda através de técnica de inundação presente em tecnologias *Link State*, como o OSPF.

Já o módulo de medição de QoS fim-a-fim é instalado apenas nos roteadores de borda, que coletam estatísticas de vazão, atraso, *jitter* e perdas, para cada aplicação prioritária que entre ou saia do domínio MPLS. As estatísticas de QoS obtidas pelo roteador de saída, para uma dada aplicação, são enviadas apenas para o roteador de entrada daquela aplicação. A atualização das estatísticas é periódica e configurável. Utilizou-se nesse trabalho intervalos de cinco segundos.

3.1.2 Módulo Controle de Admissão

O módulo de controle de admissão é o responsável por receber as requisições de QoS das aplicações, sendo que estas devem informar a vazão mínima necessária, atraso, *jitter* e perda máximos admitido. Pode-se ainda especificar pesos de importância para cada um desses parâmetro de QoS. O controle de admissão aceita uma ou mais requisições simultaneamente, podendo alocar vários fluxos de uma só vez. Toda essa demanda é organizada e passada para o módulo de otimização que devolve uma ou mais soluções ótimas. Cada solução é um conjunto de caminhos que melhor se aproxima do perfil de QoS de cada aplicação, porém há situações em que a rede não dispõe dos recursos necessários e mesmo a melhor solução não atenderia a aplicação. Nesse caso cabe ao controle de admissão avaliar as soluções e então recusar a admissão daquelas aplicações que não tiveram solução adequada encontrada. Por fim, o módulo de controle de admissão repassa a lista de aplicações, apenas as admitidas, com suas respectivas demandas e caminhos, para o gerenciador dos fluxos alocados.

3.1.3 Módulo Otimizador

O módulo otimizador tem a função de encontrar um conjunto de rotas que atenda as necessidades de QoS de um conjunto de aplicações. Podem existir diversas rotas que atendem as necessidades de uma aplicação, assim como uma rota pode atender a demanda de mais de uma aplicação. Além disso a alocação de uma aplicação altera os recursos disponíveis em vários enlaces na rede, afetando consequentemente vários caminhos. Portanto alocar os recursos disponíveis de forma a maximizar o número de aplicações atendidas é uma tarefa dinâmica e envolve um grande espaço de possibilidades que precisa ser explorado em tempo hábil. Esse problema de alocação de rotas LSPs foi abordado no trabalho (ANDRADE, 2008), no qual é feito um estudo comparativo do desempenho de diversas variações de algoritmos genéticos, e também com algoritmos imunológicos em Andrade, Errico e Assis (2009).

3.1.4 Módulo Gerenciador dos Fluxos Alocados

O gerenciador dos Fluxos alocados mantém organizada a informação de cada aplicação que se encontra alocada. É o módulo central do sistema, possui interface com todos os outros e portanto é o responsável por integrar as várias funcionalidades, conferindo as características autonômicas descritas a seguir.

3.1.4.1 Auto-Otimização

O gerenciador informa ao sistema de medição, quais são as aplicações que devem ter sua QoS fim-a-fim monitoradas periodicamente. Quando o gerenciador recebe um nível de QoS inadequado para uma aplicação, aciona então o módulo de otimização para que este encontre uma outra solução de caminho para a aplicação prejudicada. Portanto o sistema de TE não faz apenas uma alocação ótima offline, está sempre se adaptando e buscando por melhores soluções.

3.1.4.2 Auto-Configuração

É o gerenciador quem mantém a relação dos caminhos necessários na rede, portanto quem gerencia quando um caminho deve ser utilizado por uma aplicação, ou quando um novo caminho precisa ser criado. Sempre que necessário o gerenciador dispara então os mecanismos de configuração do domínio MPLS. Portanto o sistema não depende da interferência humana para modificar suas configurações.

3.1.4.3 Auto-Cura

Mesmo que o sistema de TE tenha alcançado um estado ótimo no qual todas as aplicações se encontram estabelecidas dentro de seus respectivos requisitos de QoS, uma falha de enlace imprevista, provocada por eventos externos, poderia transformar a solução ótima em uma solução insatisfatória. Portanto, assim que detectada uma falha desse tipo, a informação é atualizada em toda a rede através de mecanismos da tecnologia OSPF, e tão logo o módulo de TE tem conhecimento da falha, o sistema de medição informa o gerenciador que por sua vez ativa o módulo de otimização em procura de uma nova solução ótima. Por fim, com uma nova solução encontrada, o configurador de LSPs é acionado para reconfigurar os caminhos na rede.

3.1.5 Módulo Configurador de LSPs

O módulo configurador de LSPs é responsável por atuar no domínio MPLS, instalando os fluxos alocados em suas respectivas rotas que foram selecionadas previamente pelo módulo otimizador. Caso já exista uma LSP correspondente a rota necessária, o mó-

dulo apenas associa o fluxo a ela, caso contrário é realizado o processo de configuração da nova LSP.

3.2 Implementação do Sistema

Para realização de experimentos, criou-se um ambiente de simulação de redes autonômicas. Procurou-se implementar o máximo possível do ambiente através do simulador de redes Network Simulator 2 (ns-2), responsável por simular toda a arquitetura de rede, protocolos, eventos, tráfego, etc. Apenas recursos ausentes no ns-2, como a gerência autonômica, algoritmos de otimização e tomadas de decisão, foram implementados em C++ externamente ao simulador. Portanto o ambiente de simulação é composto por um sistema de controle de simulação, desenvolvido nesse trabalho, que se utiliza do software ns-2.

O funcionamento do ambiente de simulação pode ser dividido em duas etapas, uma primeira etapa ilustrada na Figura 3, executada uma única vez, na qual o ambiente é configurado para realizar o experimento desejado e descrito em um arquivo de entrada. E uma segunda etapa, ilustrada na Figura 4, na qual é estabelecido um *loop* em que ocorre a simulação. O *loop* é repetido quantas vezes forem necessárias até que todos os eventos descritos no arquivo de entrada do simulador sejam processados.

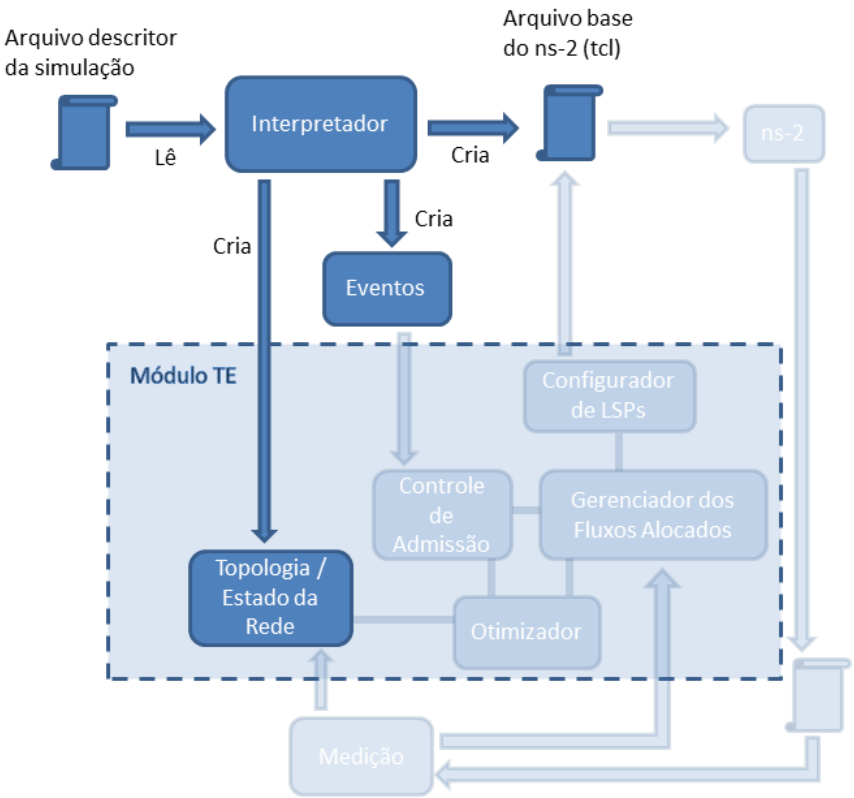


Figura 3 – Fluxograma destacando a primeira etapa do funcionamento do ambiente de simulação.

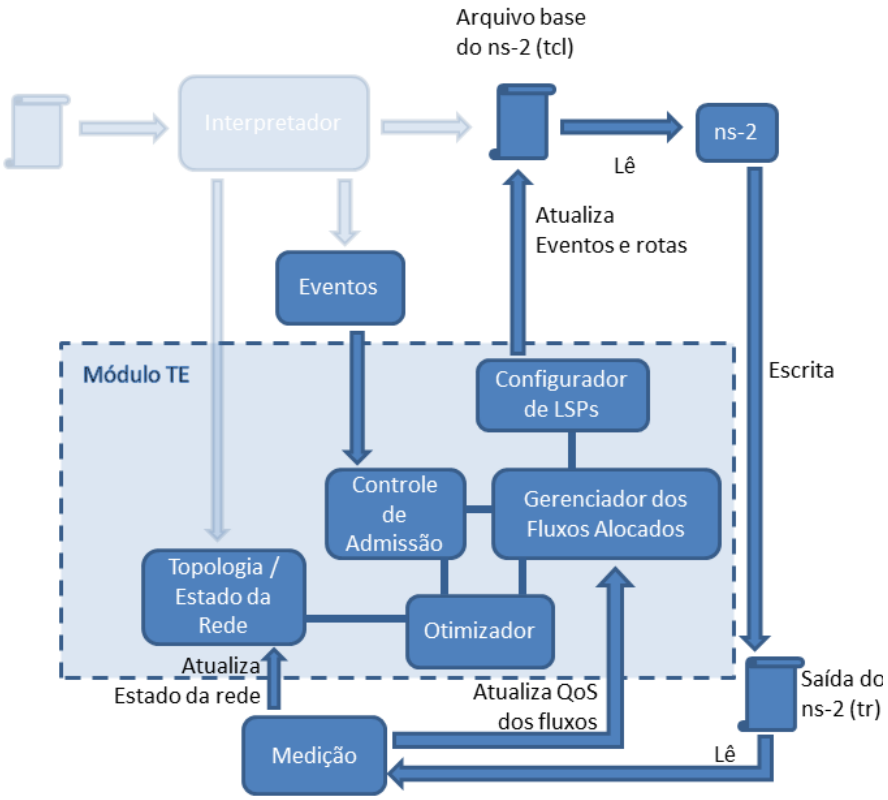


Figura 4 – Fluxograma destacando a segunda etapa do funcionamento do ambiente de simulação.

3.2.1 Inicialização

A seguir é detalhado o funcionamento do ambiente de simulação com o auxílio de trechos de pseudocódigo do programa principal, começando com o Algoritmo 1.

Em um primeiro passo algumas variáveis são inicializadas e o interpretador é acionado para ler o Arquivo Descritor da simulação.

Algoritmo 1 Primeira parte do pseudocódigo principal do ambiente de simulação.

```
1: [topologia, eventos] ← interpretador(ArquivoDescritor)
2: instanteAtual ← 0                    ▷ Instante atual da simulacao (em segundos)
3: ultimaAvaliacao ← 0   ▷ Ultimo instante em que a QoS foi avaliada (em segundos)
4: periodoAvaliacao ← 5                 ▷ Periodo das avaliacoes de QoS (em segundos)
5: fluxosAlocados ← []                  ▷ Inicialmente nao ha fluxos alocados na rede
```

O **Arquivo Descritor** é a interface pela qual o usuário interage com o sistema de simulação. Trata-se de um arquivo que contém todas as informações necessárias para que o sistema realize a simulação do cenário desejado. O arquivo deve seguir um formato específico, conforme exemplo na Figura 5, para que o sistema o interprete corretamente. Deve-se descrever primeiro as informações da topologia, características dos enlaces, e em seguida os eventos: início e fim das transmissões e falhas de enlace. O caracter "#" é um

identificador de comentário, ao encontrar esse caracter o interpretador ignora o restante da linha.

```
1  # Configuration file for network simulations
2  # Topology definition
3  # Set the number of nodes
4  NumberOfMPLSNodes = 5 # the ID of MPLS nodes start from 0 to "
       NumberOfMPLSNodes-1"
5  NumberOfClientNodes = 10 # the ID of client nodes start from "
       NumberOfMPLSNodes" to "NumberOfMPLSNodes + NumberOfClientNodes-1"
6  #Topology = {manual|mesh|fullyConnected}
7  Topology = manual
8  #For manual topology each link must be configured following the instructions
       below
9  #All links are full-duplex, so it is not necessary to define the two
       directions of the link.
10 Links = {
11 #SourceNode TargetNode BW(Mbps) Delay(ms) Cost
12 0 1 10 30 1
13 0 3 1 20 1
14 0 4 5 30 1
15 1 2 10 30 1
16 2 3 10 30 1
17 3 4 5 20 1
18 5 0 3 1 1
19 6 0 1 1 1
20 7 0 100 1 1
21 8 0 100 1 1
22 9 0 1 1 1
23 1 10 100 1 1
24 3 11 100 1 1
25 3 12 100 1 1
26 3 13 100 1 1
27 4 14 100 1 1
28 }
29 Events = {
30 #Time action
31 #Time: {0-9}*{.}?{0-9}+ (in seconds)
32 #actions:
33 #start CBR id srcNode dstNode pktSize(bytes) interval(s) reqThroughput(bps)
       reqDelay(ms) reqLoss([0-1]) reqJitter(ms) WeightThroughput WeightDelay
       WeightLoss WeightJitter
34 #start VIDEO id srcNode dstNode "traceFileName" reqThroughput reqDelay reqLoss
        reqJitter WeightThroughput WeightDelay WeightLoss WeightJitter
35 #start LinkFail srcNode dstNode
36 #stop CBR id
37 #stop VIDEO id
38 #stop LinkFail srcNode dstNode
39 #stop Sim
40 2.0 start VIDEO 1 5 10 "soccer_high.dat" 3000000 150 0.05 5 30 30 0 40
41 2.0 start VIDEO 2 6 11 "oktober_medium.dat" 1500000 150 0.05 5 30 30 0 40
42 2.0 start CBR 3 9 14 300 0.004 700000 150 0.05 5 10 50 0 40
43 9.0 start LinkFail 0 1
44 16.0 stop VIDEO 1
45 16.0 stop VIDEO 2
46 20.0 stop Sim
47 }
48 #end of the configuration file
```

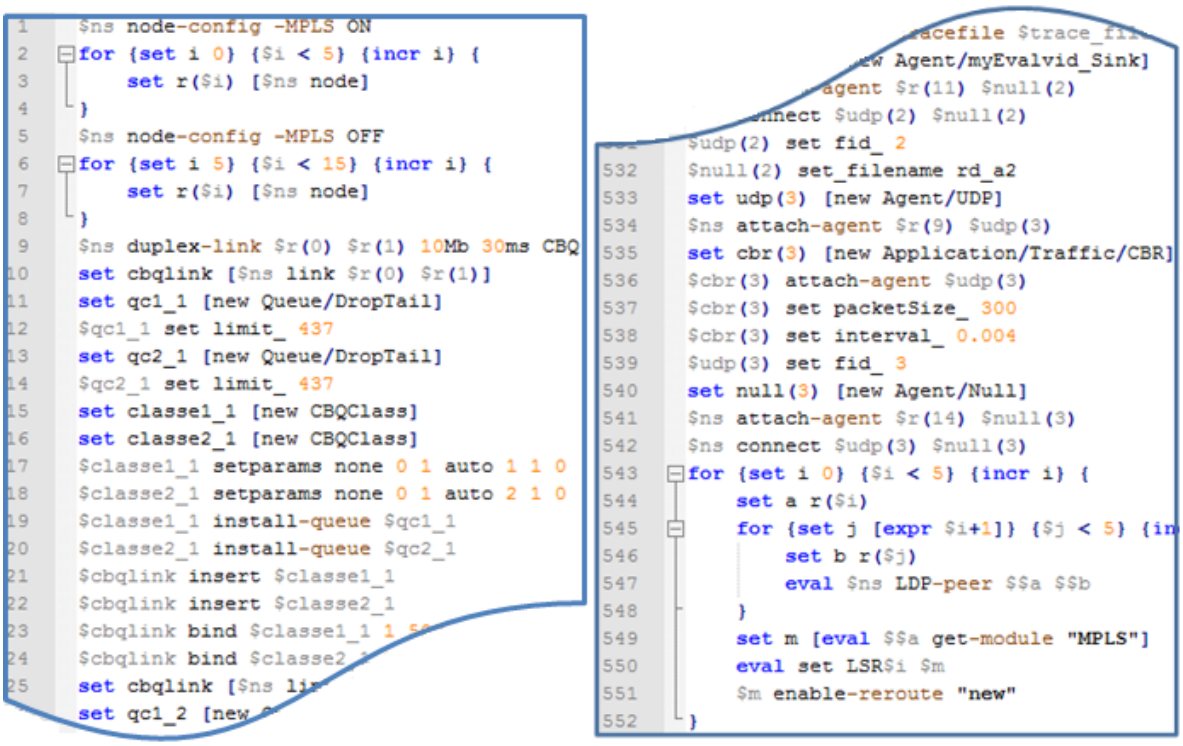
Figura 5 – Exemplo de arquivo descritor da simulação.

3.2.2 Interpretador

O Interpretador é uma função, implementada na linguagem de programação C++, que recebe como parâmetro de entrada um Arquivo Descritor de Simulação e tem o papel de ler o arquivo, apontar possíveis erros de sintaxe e, no caso de não haver erros, extrair a informação organizando-a nas estruturas de dados "Topologia" e "Eventos". Tem ainda a função de escrever um arquivo de código TCL descrevendo a topologia do cenário para o ns-2. Tomando o exemplo anterior da Figura 5, o conteúdo válido lido pelo interpretador seria aquele apresentado na Figura 6 e o arquivo TCL gerado seria o da Figura 7.

```
1  NumberOfMPLSNodes = 5
2  NumberOfClientNodes = 10
3  Topology = manual
4  Links = {
5  0 1 10 30 1
6  0 3 1 20 1
7  0 4 5 30 1
8  1 2 10 30 1
9  2 3 10 30 1
10 3 4 5 20 1
11 5 0 3 1 1
12 6 0 1 1 1
13 7 0 100 1 1
14 8 0 100 1 1
15 9 0 1 1 1
16 1 10 100 1 1
17 3 11 100 1 1
18 3 12 100 1 1
19 3 13 100 1 1
20 4 14 100 1 1
21 }
22 Events = {
23 2.0 start VIDEO 1 5 10 "soccer_high.dat" 3000000 150 0.05 5 30 30 0 40
24 2.0 start VIDEO 2 6 11 "oktober_medium.dat" 1500000 150 0.05 5 30 30 0 40
25 2.0 start CBR 3 9 14 300 0.004 700000 150 0.05 5 10 50 0 40
26 9.0 start LinkFail 0 1
27 16.0 stop VIDEO 1
28 16.0 stop VIDEO 2
29 20.0 stop Sim
30 }
```

Figura 6 – Conteúdo lido pelo sistema para o arquivo exemplo da Figura 5.



```
1   $ns node-config -MPLS ON
2   for {set i 0} {$i < 5} {incr i} {
3       set r($i) [$ns node]
4   }
5   $ns node-config -MPLS OFF
6   for {set i 5} {$i < 15} {incr i} {
7       set r($i) [$ns node]
8   }
9   $ns duplex-link $r(0) $r(1) 10Mb 30ms CBQ
10  set cbqlink [$ns link $r(0) $r(1)]
11  set qc1_1 [new Queue/DropTail]
12  $qc1_1 set limit_ 437
13  set qc2_1 [new Queue/DropTail]
14  $qc2_1 set limit_ 437
15  set classe1_1 [new CBQClass]
16  set classe2_1 [new CBQClass]
17  $classe1_1 setparams none 0 1 auto 1 1 0
18  $classe2_1 setparams none 0 1 auto 2 1 0
19  $classe1_1 install-queue $qc1_1
20  $classe2_1 install-queue $qc2_1
21  $cbqlink insert $classe1_1
22  $cbqlink insert $classe2_1
23  $cbqlink bind $classe1_1 1
24  $cbqlink bind $classe2
25  set cbqlink [$ns li
    set qc1_2 [new
```

```
    ...racefile $trace_fil
    ...w Agent/myEvalvid_Sink]
    ...agent $r(11) $null(2)
    ...nnect $udp(2) $null(2)
532 $udp(2) set fid_ 2
533 $null(2) set_filename rd_a2
534 set udp(3) [new Agent/UDP]
535 $ns attach-agent $r(9) $udp(3)
536 set cbr(3) [new Application/Traffic/CBR]
537 $cbr(3) attach-agent $udp(3)
538 $cbr(3) set packetSize_ 300
539 $cbr(3) set interval_ 0.004
540 $udp(3) set fid_ 3
541 set null(3) [new Agent/Null]
542 $ns attach-agent $r(14) $null(3)
543 $ns connect $udp(3) $null(3)
544 for {set i 0} {$i < 5} {incr i} {
545     set a r($i)
546     for {set j [expr $i+1]} {$j < 5} {in
547         set b r($j)
548         eval $ns LDP-peer $$a $$b
549     }
550     set m [eval $$a get-module "MPLS"]
551     eval set LSR$i $m
552     $m enable-reroute "new"
    }
```

Figura 7 – Exemplo de arquivo TCL gerado pelo módulo Interpretador.

Topologia é a estrutura de dados que representa o conhecimento que o sistema de TE tem sobre o domínio MPLS, e portanto é parte do módulo de medição da arquitetura proposta no capítulo anterior. Nessa estrutura são armazenadas as informações de todos os enlaces.

Eventos é uma estrutura de dados que controla a cinemática da simulação. Não implementa qualquer parte do sistema de TE, mas é necessária para o ambiente de simulação, pois substitui as variáveis externas que geram eventos em uma rede real, como aplicações clientes e falhas de enlace. Nessa estrutura são armazenadas as informações de quando e quais eventos irão ocorrer na rede durante a simulação. Mantém também a informação do estado de cada evento, e.g se o evento foi aceito, negado ou se ainda não foi processado.

O sistema de TE não tem acesso a toda informação da estrutura Eventos desde o início da simulação, mas recebe a informação de cada evento no momento certo. A partir da linha 6 do pseudocódigo, Algoritmo 2, começa o *loop* do ambiente de simulação que irá, a cada iteração, fornecer em ordem cronológica os eventos a serem tratados. Continuando o algoritmo, na linha 7 é obtido o instante do próximo evento dado que a simulação se encontra no *instanteAtual*. Em seguida é verificado se o período entre o próximo evento e o instante atual é maior do que o período de avaliação de QoS, se for maior então a simulação do *loop* atual será executada até o próximo instante de avaliação. Caso contrário será executada até o instante do próximo evento. Uma vez determinado o tamanho do

passo na simulação, ou seja o novo *instanteAtual*, todos os eventos até este instante são obtidos com o comando da linha 15. Se houverem novas requisições de aplicações que desejam alocar recursos da rede, estas serão armazenadas em *novasRequisicoes*. Caso exista algum evento de falha, a variável *falha* assumirá o valor *verdadeiro*. Esse comando também atualiza o arquivo de simulação TCL incluindo todos os eventos já processados até o *instanteAtual*, arquivo esse que é executado pelo ns-2 no comando seguinte (linha 16).

Algoritmo 2 Segunda parte do pseudocódigo principal do ambiente de simulação.

```
6:  repeat                                                          ▷ Inicio do loop principal
7:      proximoEvento ← eventos.proximo(instanteAtual)              ▷ Instante do proximo evento
8:      if ultimaAvaliacao + periodoAvaliacao ≥ proximoEvento then  ▷ Verifica se e necessario avaliar a QoS antes do proximo evento
9:          instanteAtual ← proximoEvento
10:         avaliacao ← falso
11:     else
12:         instanteAtual ← ultimaAvaliacao + periodoAvaliacao
13:         avaliacao ← verdadeiro
14:     end if
15:     [novasRequisicoes, falha] ← eventos.executar(instanteAtual)   ▷ obtem novos eventos e atualiza o codigo do ArquivoTCL
16:     ns2(ArquivoTCL)                                             ▷ executa a simulacao do ArquivoTCL no ns-2 gerando o ArquivoTR
```

3.2.3 Network Simulator 2

Network Simulator 2, ou ns-2, é um simulador de eventos discretos desenvolvido para a pesquisa na área de redes (NS2, 2011). Embora dentre os principais simuladores de rede, o ns-2 seja considerado um dos que mais demandam tempo de aprendizagem (ORFANUS et al., 2008), foi escolhido por ser um simulador de código aberto amplamente aceito pela comunidade científica e também por oferecer uma grande variedade de tecnologias e protocolos de rede. Na presente data, encontra-se disponível o ns-3, sucessor do ns-2, porém esse é muito novo na comunidade científica e não possui ainda a mesma quantidade de trabalhos de validação que atestam o uso do simulador para pesquisas.

O ns-2 foi desenvolvido parte na linguagem C++ e parte em OTcl. É nessa segunda linguagem que os cenários de simulação devem ser descritos para que o ns-2 execute a simulação e gere como saída um arquivo texto, como o da Figura 8, que contém a informação de todos os eventos ocorridos. Cada linha desse arquivo descreve uma ação aplicada a um pacote em determinado momento e local, além de informar o tipo do pacote

e algumas outras informações que este carrega como o tamanho, endereço de origem e destino.

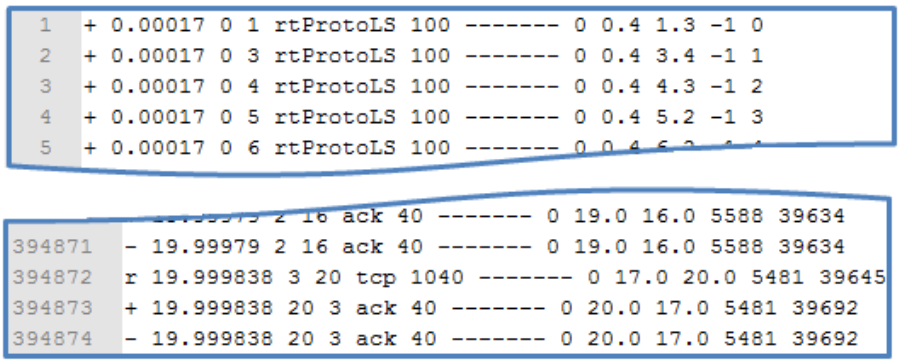


Figura 8 – Exemplo de arquivo de saída gerado pelo ns-2.

O arquivo de saida do ns-2 é então lido e processado pelo módulo de medição, através da chamada de método da linha 18 do Algoritmo 3, que extrai as informações necessárias para o cálculo dos parâmetros de QoS vazão, atraso, *jitter* e perdas, para cada enlace conforme as fórmulas apresentadas a seguir.

A média da vazão em *bits* por segundo, recebida por um roteador é:

$$\overline{vazão} = \sum_{i=2}^{n} \frac{tamanho_i}{t_n - t_1}[bps], \tag{3.1}$$

onde $tamanho_i$ é o tamanho em *bits* do i-ésimo pacote recebido, t_1 e t_n são respectivamente os instantes em segundos que o primeiro e o n-ésimo pacote são recebidos, e n é o número total de pacotes considerados no cálculo.

A média aritmética dos atrasos individuais de vários pacotes em sequência é:

$$\overline{atraso} = \sum_{i=1}^{n} \frac{t_i^{recebido} - t_i^{enviado}}{n}[s], \tag{3.2}$$

onde $t_i^{enviado}$ e $t_i^{recebido}$ são respectivamente os instantes em segundos que o pacote i foi enviado e recebido, e n é o número total de pacotes considerados no cálculo.

A média aritmética das diferenças de atrasos individuais de vários pacotes em sequência é:

$$\overline{jitter} = \sum_{i=2}^{n} \frac{atraso_i - atraso_{i-1}}{n-1}[s], \tag{3.3}$$

onde $atraso_i$ é o atraso em segundos do i-ésimo pacote, e n é o número total de pacotes considerados no cálculo.

A fração perdida de uma sequência de pacotes enviados.

$$perda = 1 - \frac{nPacotes^{recebidos}}{nPacotes^{enviados}}, \tag{3.4}$$

onde $nPacotes^{enviados}$ e $nPacotes^{recebidos}$ são respectivamente o número de pacotes enviados e recebidos.

Se alguma falha é detectada então todos os fluxos já estabelecidos são marcados para realocação, como se pode ver no pseudocódigo entre as linhas 33 e 39. Isso não significa que necessariamente o fluxo terá seu caminho alterado, mas apenas que o otimizador será acionado para procurar soluções melhores dado o novo estado da rede, portanto a nova solução pode manter um fluxo em seu caminho atual ou sugerir a alteração para um novo caminho. Já no caso em que nenhuma falha de enlace é detectada, os fluxos terão suas QoS avaliadas e serão marcados para realocação somente aqueles fluxos que obterem nível de QoS abaixo de seus respectivos requisitos, como apresentado no Algoritmo 3 entre as linhas 20 e 30.

Algoritmo 3 Terceira parte do pseudocódigo principal do ambiente de simulação.

```
17:     if avaliacao = verdadeiro then
18:         estadoDaRede ← medicao.enlaces(ArquivoTR)        ▷ Le a saida do ns-2 e calcula o estado dos enlaces
19:         topologia.atualizaEstado(estadoDaRede)           ▷ Atualiza os estado da rede
20:         if falha = falso then
21:             QoS ← medicao.qos(ArquivoTR, fluxosAlocados)   ▷ calcula a QoS dos fluxos alocados
22:             fluxosRealocar ← []      ▷ A principio nenhum fluxo precisa ser realocado
23:             for i ← 1 to tamanho(fluxosAlocados) do
24:                 fluxosAlocados[i].atualizaQoS(QoS[i])
25:                 if fluxosAlocados[i].QoS < fluxosAlocados[i].requisitos then
26:                     fluxosRealocar.add(fluxosAlocados[i])
27:                     topologia.liberaReservaBW(fluxosAlocados[i])
28:                 end if
29:             end for
30:         end if
31:         ultimaAvaliacao ← instanteAtual
32:     end if
33:     if falha = verdadeiro then
34:         fluxosRealocar ← []
35:         for i ← 1 to tamanho(fluxosAlocados) do
36:             fluxosRealocar.add(fluxosAlocados[i])
37:             topologia.liberaReservaBW(fluxosAlocados[i])
38:         end for
39:     end if
```

3.2.4 Otimizador

Uma vez definidas as novas requisições e as realocações de fluxos, o otimizador é acionado, na linha 42 do algorítimo 4, para buscar uma solução de conjunto de caminhos que atendam aos requisitos de QoS dos fluxos. O otimizador retorna então a melhor

solução encontrada para o controle de admissão que verifica para cada fluxo se a solução encontrada é suficiente para sua transmissão. Os fluxos já estabelecidos tem sua realocação garantida, enquanto as novas requisições são alocadas somente se a solução for aprovada pelo controle de admissão.

Na linha 54 o *loop* é fechado, começando tudo de novo na linha 6 do Algoritmo 2 até que todos os eventos da simulação sejam tratados.

Algoritmo 4 Quarta parte do pseudocódigo principal do ambiente de simulação.

```
40:     fluxosAlocar ← fluxosRealocar + novasRequisicoes
41:     if tamanho(fluxosAlocar) > 0 then
42:         caminhos ← otimizador.Executa(fluxosAlocar, topologia)    ▷ executa o GA
43:         for i ← 1 to tamanho(caminhos) do
44:             if i ≤ tamanho(fluxosRealocar) then
45:                 topologia.alocaReservaBW(caminhos[i])   ▷ atualiza reserva de banda
46:                 configuradorLSPs(caminhos[i])           ▷ atualiza o ArquivoTCL
47:             else if caminhos[i].QoS ≥ fluxosAlocar[i].requisitos then  ▷ controle de
    admissao
48:                 topologia.alocaReservaBW(caminhos[i])   ▷ atualiza reserva de banda
49:                 configuradorLSPs(caminhos[i])           ▷ atualiza o ArquivoTCL
50:                 fluxosAlocados.add(fluxosAlocar[i])
51:             end if
52:         end for
53:     end if
54: until instanteAtual ≥ eventos.fimSimulacao
```

3.2.4.1 Modelagem do problema

Para aplicar o GA é necessário antes modelar o problema, sendo assim a solução candidata, também conhecida como indivíduo, deve representar um conjunto de N caminhos para atender a demanda de N aplicações. Adotou-se a codificação real, na qual o indivíduo é formado por uma sequência de números inteiros, sendo cada um desses números um índice que representa um caminho de um fluxo na rede, conforme mostrado na Figura 9. A indexação de todos os caminhos possíveis deve ser executada uma única vez, sendo necessário atualizar apenas diante de uma modificação na topologia da rede, como o acréscimo de um roteador ou de um novo enlace. Tal indexação pode ser realizada com uma exploração de todas as arestas do grafo que representa os roteadores e seus respectivos enlaces. Uma solução alternativa para redes maiores é limitar a indexação a K caminhos com a aplicação do algoritmo dos K-caminhos mínimos de Yen (1970), que encontra os K menores caminhos para todos os destinos a partir de uma única fonte. O algoritmo encontra primeiro o menor caminho através da aplicação do algoritmo de Dijkstra e então procura o segundo menor caminho aplicando-se novamente Dijkstra em subgrafos gerados com a remoção de um enlace por vez do melhor caminho. O processo se repete até encontrar os K-caminhos mínimos (YEN, 1971).

Figura 9 – Indivíduo é composto por N genes que representam os caminhos dos fluxos a serem alocados.

Uma vez com o indivíduo definido é necessário uma forma de avaliar a qualidade da solução que o indivíduo representa. Dado que cada aplicação demanda um conjunto de parâmetros de QoS (vazão, atraso, *jitter* e perdas), o problema original se configura como um problema multi-objetivo, sendo portanto necessário transforma-lo em um problema mono-objetivo para que possa ser solucionado por um GA simples. Isso é obtido com a função-objetivo da Eq. 3.5:

$$CustoGene = \left[\sum_{i=1}^{k}\left(\frac{BW_r}{BW_i}\right)^{P_{BW}}\right] + \left(\frac{L_d}{L_r}\right)^{P_L} + \left(\frac{J_d}{J_r}\right)^{P_J} + \left(\frac{P_d}{P_r}\right)^{P_P} \tag{3.5}$$

onde BW_i é a banda disponível no i-ésimo enlace de um caminho, BW_r, L_r, J_r e P_r são respectivamente a banda mínima, latência máxima, *jitter* máximo e perdas máximas especificadas pela aplicação. L_d, J_d e P_d são respectivamente a latência, *jitter* e perdas mensuradas no caminho em questão. Finalmente P_{BW}, P_L, P_J e P_P são pesos de relevância, indicando quais parâmetros possuem maior ou menor importância para a aplicação. A soma de todos os pesos é sempre constante e igual a dez, de forma que ao aumentar um peso todos os outros são reduzidos. A Equação 3.5 representa o cálculo do custo de um gene. Já o custo total do indivíduo é a soma dos custos individuais de cada gene que o compõe. Sendo uma função de custo, quanto mais próximo de zero for o custo, melhor é o indivíduo. Em algoritmos genéticos é comum a utilização do termo aptidão, que possui uma relação contrária, quanto menor a aptidão pior é o indivíduo. Portanto, quando for utilizado o termo aptidão nesse texto, entenda-o como o inverso da função de custo $\frac{1}{Custo}$. A escolha por uma somatória de exponenciais de frações se deve a duas grandes vantagens. A primeira é que a razão entre variáveis de mesma unidade torna a somatória adimensional e portanto coerente. A segunda vantagem é que quando uma restrição da aplicação é violada, a razão *requerido/disponível* é maior do que um e a penalização do indivíduo cresce exponencialmente com o aumento da margem de violação. É importante que os pesos sejam utilizados na forma exponencial ao invés de multiplicativa para que indivíduos com uma violação grande sejam mais penalizados do que por exemplo indivíduos com duas violações pequenas. A exceção é a violação de banda, que recebe tratamento diferenciado. Quando detectada tal violação, a função de custo é substituída por uma penalização muito alta (e.g 10^6), indicando que o indivíduo propõe uma solução na qual essa aplicação não é alocada. Indivíduos penalizados tendem a desaparecer ao longo das gerações do GA, a menos que a rede não disponha dos recursos necessários para atender a todas as aplicações. Nesse caso todos indivíduos são penalizados, sendo que aquele que conseguir alocar mais

aplicações continuará tendo menor custo e terá mais chances de ser selecionado como solução.

O fluxograma de como é executado o algoritmo genético está ilustrado na Figura 10. Nas seções seguintes serão detalhados os operadores de seleção, cruzamento e mutação, assim como outros detalhes de implementação.

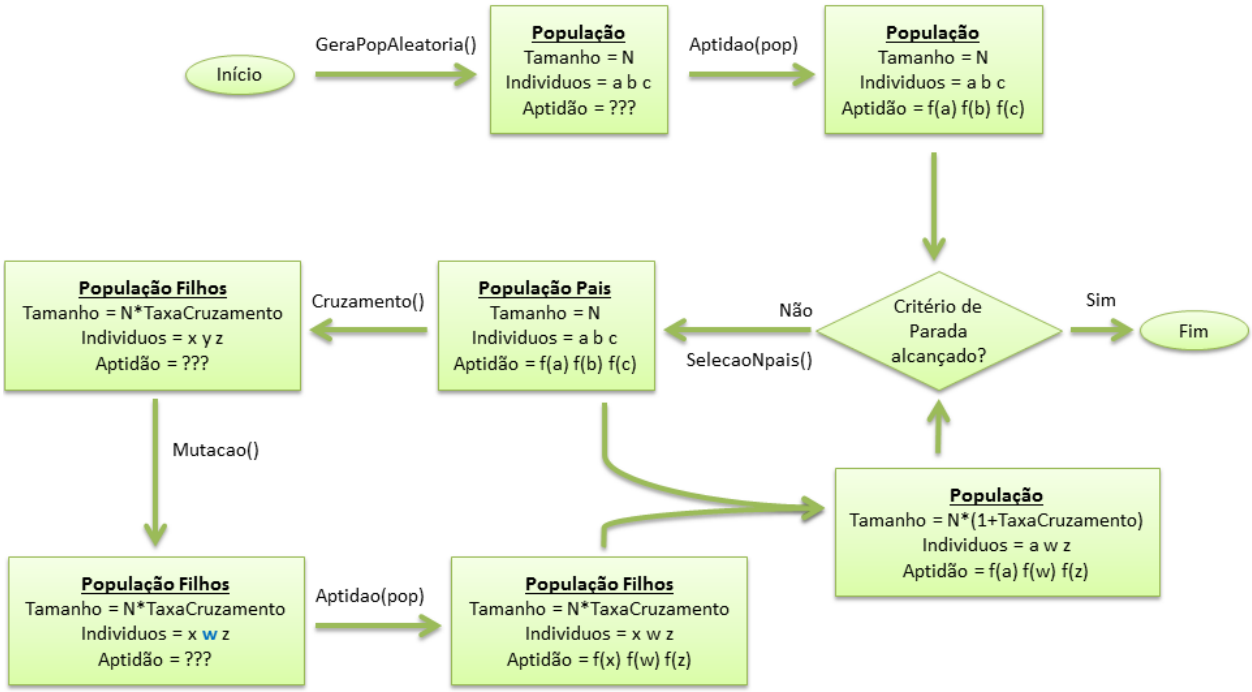


Figura 10 – Fluxograma do algoritmo genético responsável pela busca de uma solução ótima para alocação de recursos.

3.2.4.2 Seleção

O operador seleção tem a função de escolher N indivíduos para permanecer na próxima geração. Essa seleção considera a aptidão do indivíduo e é realizada em duas etapas. Uma primeira etapa elitista garante a permanência dos M melhores indivíduos. Uma segunda etapa preenche o restante das vagas (N-M) com o algoritmo Torneio. A escolha desse algoritmo foi baseada no estudo de Andrade (2008), no qual diversas variações de GAs foram aplicadas ao problema de alocação de LSPs com restrições de QoS.

No algoritmo Torneio é realizada uma competição de aptidão entre k indivíduos de um subconjunto aleatório da população. O indivíduo desse subconjunto que possui a maior aptidão é o ganhador do torneio, sendo assim selecionado para permanecer na próxima geração. O torneio é repetido, sempre tomando um novo subconjunto aleatório, quantas vezes forem necessárias até que os N indivíduos da próxima geração estejam selecionados.

Quanto maiores forem os parâmetros M e k, mais rápida será a convergência para

uma solução, mas também maior é a chance de se convergir para um ótimo local. No entanto, o objetivo do módulo otimizador é encontrar uma solução satisfatória, que atenda ao conjunto de demandas recebido. Portanto não importa se a solução encontrada é ou não a melhor. É mais importante que a solução atenda a demanda requerida e que seja encontrada o mais rápido possível, o que justifica a aplicação de uma estratégia elitista.

3.2.4.3 Cruzamento

Adotou-se, também com base nos estudos de Andrade (2008), o método de cruzamento em dois pontos variáveis, que consiste em definir aleatoriamente dois pontos que separam a codificação genética do indivíduo em duas partes, uma externa e outra interna aos pontos. Para cada par de pais em que é aplicado o cruzamento, são gerados dois filhos. O primeiro filho herda a informação genética da parte externa do Pai1 e a parte interna do Pai2, o segundo filho herda a parte interna do Pai1 e a externa do Pai2. Esse procedimento está ilustrado na Figura 11.

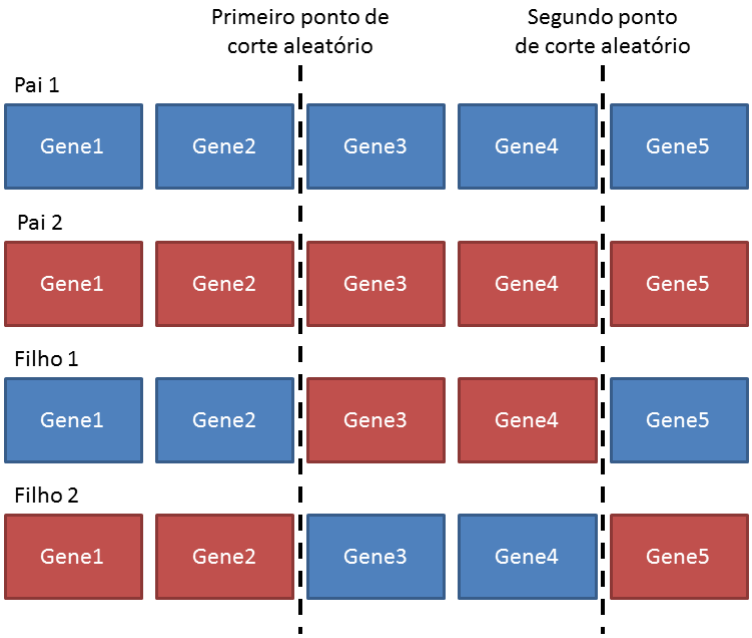


Figura 11 – Método de cruzamento com dois pontos variáveis.

O operador cruzamento é aplicado sobre pares de indivíduos do conjunto selecionado. O número de cruzamentos é definido por uma variável TaxaCruzamento, que deve ser maior ou igual a zero e menor ou igual a um. Quando igual a um, ocorre o número máximo de N/2 cruzamentos, gerando consequentemente N filhos. Quanto maior é a TaxaCruzamento, maior é o custo computacional para calcular a aptidão dos novos indivíduos, porém melhor é a exploração do espaço.

3.2.4.4 Mutação

O operador mutação confere diversidade de soluções ao modificar de forma completamente aleatória a informação genética de alguns indivíduos. A mutação pode ocorrer em qualquer indivíduo da população de filhos, afetando aleatóriamente qualquer um dos genes do indivíduo. A Figura 12 ilustra um exemplo de mutação.

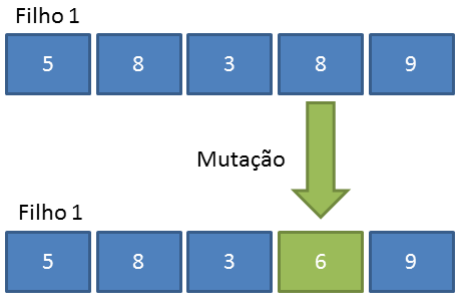


Figura 12 – Exemplo de mutação aleatória.

A frequência com que ocorrem as mutações é definida pela variável TaxaMutacao. Sem a mutação a exploração do espaço de soluções se torna muito dependente da população inicial. Quanto maior a taxa de mutação, maior a diversidade de informação genética na população, porém uma taxa muito elevada pode atrasar o processo de convergência.

3.2.4.5 Critério de Parada

É necessário definir critérios para que o algoritmo genético decida quando interromper a busca por soluções melhores. Optou-se por disponibilizar três critérios diferentes, que podem ser usados simultaneamente ou sozinhos, sendo que ao menos um deve ser adotado. Os critérios são:

Parada por tempo de execução: um tempo de execução limite, em milisegundos, é definido. Assim, ao alcançar o tempo limite, o GA irá parar de procurar por soluções melhores, independentemente do que ele já tenha encontrado.

Parada por número de gerações: é definido um número máximo de gerações, novamente independentemente das soluções já encontradas. O GA irá parar ao alcançar a geração limite.

Parada por convergência: a cada geração é realizada uma comparação da aptidão do melhor indivíduo com a aptidão média da população. Se essa diferença for menor do que uma tolerância definida, o GA interrompe sua execução.

O terceiro critério tem a vantagem de parar apenas quando detectada uma convergência para uma solução, o que geralmente é sinal de que o ótimo foi alcançado. Entretanto existem situações em que o GA pode não convergir, ou pelo menos demorar muito para alcançar o valor de tolerancia escolhido. Portanto é recomendado combinar esse critério com um dos dois primeiros critérios citados. Ambos tem função semelhante, porém a parada por tempo limite tem a vantagem de ser independente do poder de processamento do computador utilizado, retornando uma solução sempre em tempo inferior ao tempo limite definido. Por outro lado, o número de gerações possui a vantagem de ser um parâmetro possível de se relacionar a uma probabilidade de convergência, e com isso não corre o risco de interromper o algoritmo antes de uma quantidade mínima de gerações que estatisticamente garante uma boa exploração do espaço. Naturalmente, se mais de um critério é adotado, o GA irá parar quando qualquer um deles for satisfeito.

CAPÍTULO 4

EXPERIMENTOS E RESULTADOS

Com a finalidade de avaliar o funcionamento do Sistema de Engenharia de Tráfego, foram realizados três experimentos. No primeiro experimento foi avaliada a capacidade do sistema em otimizar a alocação de rotas, buscando a melhor utilização dos recursos disponíveis na rede. O segundo experimento testa o controle de admissão, que deve apenas aceitar novas aplicações comunicando na rede se for possível alocar recursos suficientes para as mesmas. O terceiro experimento força a queda de um enlace da rede, obrigando o sistema a realocar os fluxos de comunicação. Nestes experimentos foram utilizadas redes mesh, com topologias de pequeno número de nós (cinco a sete), o que facilita a geração de situações de congestionamento e a observação do que acontece na rede. Para as aplicações que se comunicam através da rede foram escolhidos vídeos com taxa de bit variável, gerados a partir do conteúdo de vídeos reais, e fluxos de taxa de bit constante, com conteúdo aleatório e representando o tráfego de fundo. Para o terceiro experimento foram acrescentados fluxos FTP, que não possuem taxa fixa e se comportam como aplicações oportunistas, que utilizam toda a banda disponível. Estes fluxos FTP não solicitam recursos ao sistema e portanto são consideradas como não prioritárias.

4.1 Experimento 1 - Otimização

4.1.1 Descrição do Cenário

O objetivo, com esse experimento, é avaliar a otimização na alocação de rotas e portanto a eficiência de distribuição dos recursos de rede. Criou-se para isso uma demanda de recursos formada pelo conjunto de aplicações apresentado na Tabela 2. As aplicações de taxa de bit variável são geradas a partir do conteúdo de vídeos reais, enquanto que as

de taxa de bit constante não possuem um conteúdo válido, são bits aleatórios transmitidos em taxa constante e representam qualquer aplicação de taxa constante. O vídeo entitulado *Old Town* é uma filmagem aérea que cruza uma cidade. Este vídeo foi publicado em Keimel et al. (2010). Os vídeos *Carving* e *Oktoberfest* são respectivamente uma filmagem em que a câmera se movimenta por uma loja de esculturas e uma filmagem, com câmera parada, da festa alemã *Oktoberfest* em Munique. Ambos vídeos foram retirados de Technische Universität München e Institute for Data Processing (2011).

Tabela 2 – Perfil das aplicações do experimento1.

Aplicação	Icone	Vazão Média / Pico	Descrição
CBR1		1.0 Mbps / 1.0 Mbps	Taxa de bit constante
Vídeo1		2.3 Mbps / 4.0 Mbps	Taxa de bit variável vídeo "Old Town"em HD
Vídeo2		2.5 Mbps / 4.0 Mbps	Taxa de bit variável vídeo "Carving"em HD
CBR2		2.0 Mbps / 2.0 Mbps	Taxa de bit constante
Vídeo3		1.0 Mbps / 1.8 Mbps	Taxa de bit variável vídeo "Oktoberfest"em SD

A topologia adotada é uma rede Mesh com sete nós roteadores ilustrada na Figura 13. A escolha por uma topologia pequena se deve à maior facilidade em gerar situações de congestionamento e também por facilitar a observação do que acontece na rede. O momento em que cada aplicação solicita entrada na rede, assim como seus respectivos requisitos de QoS estão descritos na Tabela 3.

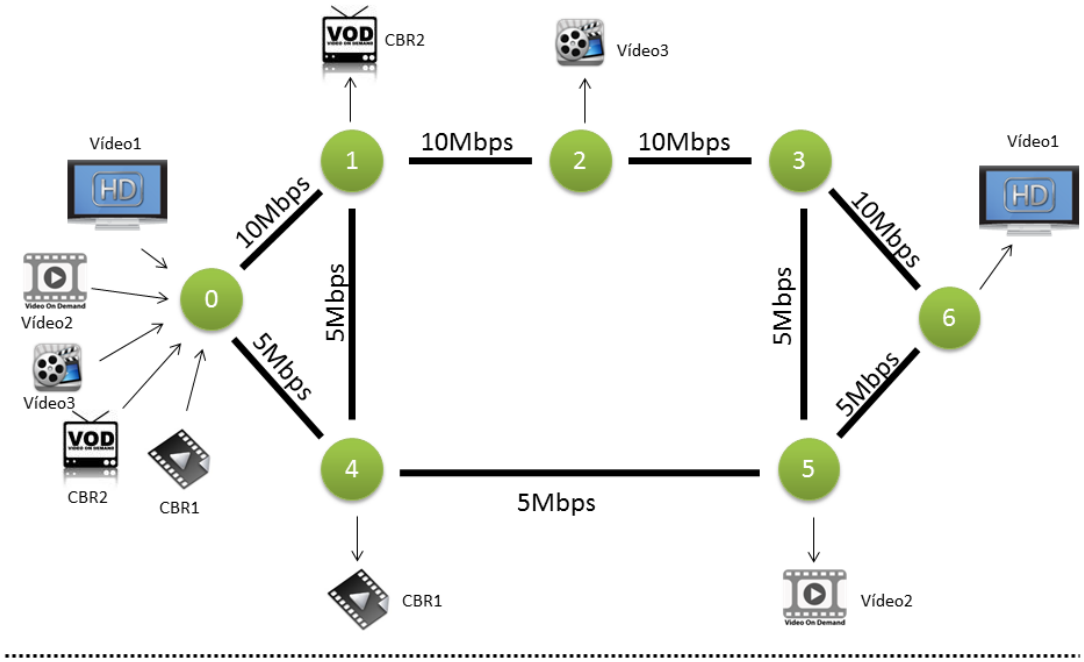


Figura 13 – Cenário do experimento 1.

Tabela 3 – Requisitos de QoS.

Aplicação	Icone	Banda Mínima	Atraso Máximo	Início	Término
CBR1		1.1Mbps	150 ms	2 s	16 s
Vídeo1		3.0 Mbps	150 ms	5 s	15 s
Vídeo2		3.0 Mbps	150 ms	5 s	15 s
CBR2		2.2 Mbps	150 ms	8 s	22 s
Vídeo3		1.5 Mbps	150 ms	8 s	22 s

4.1.2 Resultado

Para comparação, o experimento foi realizado com e sem o sistema de TE proposto. Na ausência do sistema de TE a rede funciona com o protocolo de roteamento OSPF. As rotas alocadas pelos dois sistemas coincidiram apenas para os fluxos CBR1 e CBR2, como pode se observar na Tabela 4.

Tabela 4 – Relação de rotas alocadas pelos sistemas com e sem TE.

Aplicação	Rota - semTE	Rota - comTE
CBR1	$0 \rightarrow 4$	$0 \rightarrow 4$
Vídeo1	$0 \rightarrow 4 \rightarrow 5 \rightarrow 6$	$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$
Vídeo2	$0 \rightarrow 4 \rightarrow 5$	$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5$
CBR2	$0 \rightarrow 1$	$0 \rightarrow 1$
Vídeo3	$0 \rightarrow 1 \rightarrow 2$	$0 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 2$

4.1.2.1 Vazão

A vazão instantânea, no receptor, das aplicações CBR1 e Vídeo1 com e sem o sistema de TE são apresentadas na Figura 14. A vazão instantânea das aplicações Vídeo2, CBR2 e Vídeo3, também com e sem o sistema de TE, são apresentadas na Figura 15. O resultado foi separado em dois gráficos para facilitar a visualização, pois o total de 10 séries de dados fica inviável em um único gráfico.

Os fluxos CBR possuem taxas de transmissão constantes e portanto a variação de sua vazão instantânea, como a que ocorre com cbr1-semTE, representa perturbações causadas por sobrecargas na rede. Já os fluxos VBR possuem taxas variáveis e portanto a variação de sua vazão é considerada normal, mas ainda sim é possível verificar que os vídeos 1 e 2 sem TE não alcançam o mesmo nível de vazão alcançado com TE, o que é também um indício de congestionamento. Os fluxos CBR2 e Vídeo3 apresentaram pouca diferença com ou sem TE.

4.1.2.2 Atraso

O tempo de atraso total dos pacotes entre sua transmissão e recepção também são apresentados em dois gráficos, Figura 16 para os fluxos CBR1 e Vídeo1, e Figura 17 para Vídeo2, CBR2 e Vídeo3.

Observa-se claramente a perturbação que os fluxos CBR1, Vídeo1 e Vídeo2 sofrem sem o sistema de TE a medida que mais fluxos disputam por recursos da rede. Com o sistema de TE todos os fluxos apresentam atraso menor que 100ms, sendo que quatro dos cinco fluxos se mantem abaixo de 50ms. Percebe-se que a aplicação Vídeo3 apresentou

atraso maior no sistema com TE, entretanto esse aumento no atraso não infrige o requisito de QoS estabelecido e portanto satisfaz a demanda. Tal aumento de atraso ocorre no sistema com TE, pois este optou por alocar o Vídeo3 em um caminho mais longo com fins de liberar recursos para atender a demanda de todas aplicações.

4.1.2.3 Perda

As perdas de pacotes são apresentadas, na Figura 18, apenas para os fluxos CBR1, Vídeo1 e Vídeo2 no sistema semTE, pois nenhum fluxo do sistema comTE apresentou perdas. Os fluxos CBR2 e Vídeo3 não apresentaram perdas em ambos sistemas. Sem o sistema de TE, os fluxos CBR1, Vídeo1 e Vídeo2, sofreram perdas em média de 8% a 16% continuamente a partir do instante 9 s.

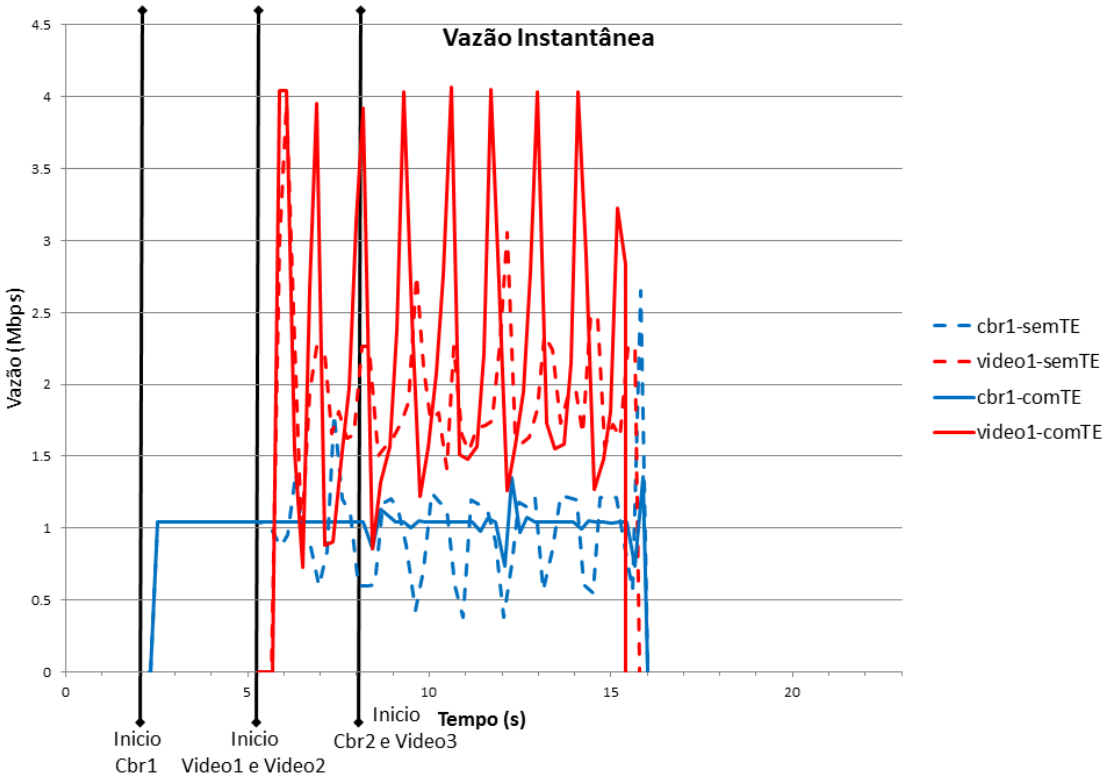


Figura 14 – Vazão instantânea das aplicações CBR1 e Vídeo1 do primeiro experimento.

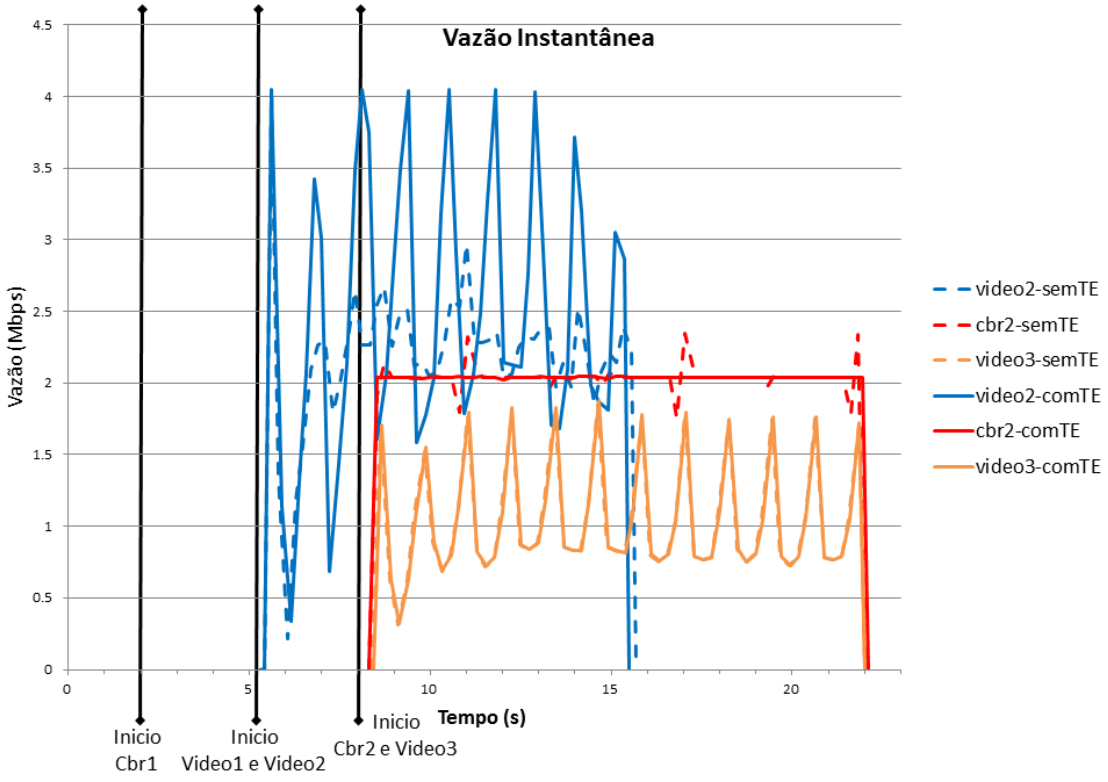


Figura 15 – Vazão instantânea das aplicações Vídeo2, CBR2 e Vídeo3 do primeiro experimento.

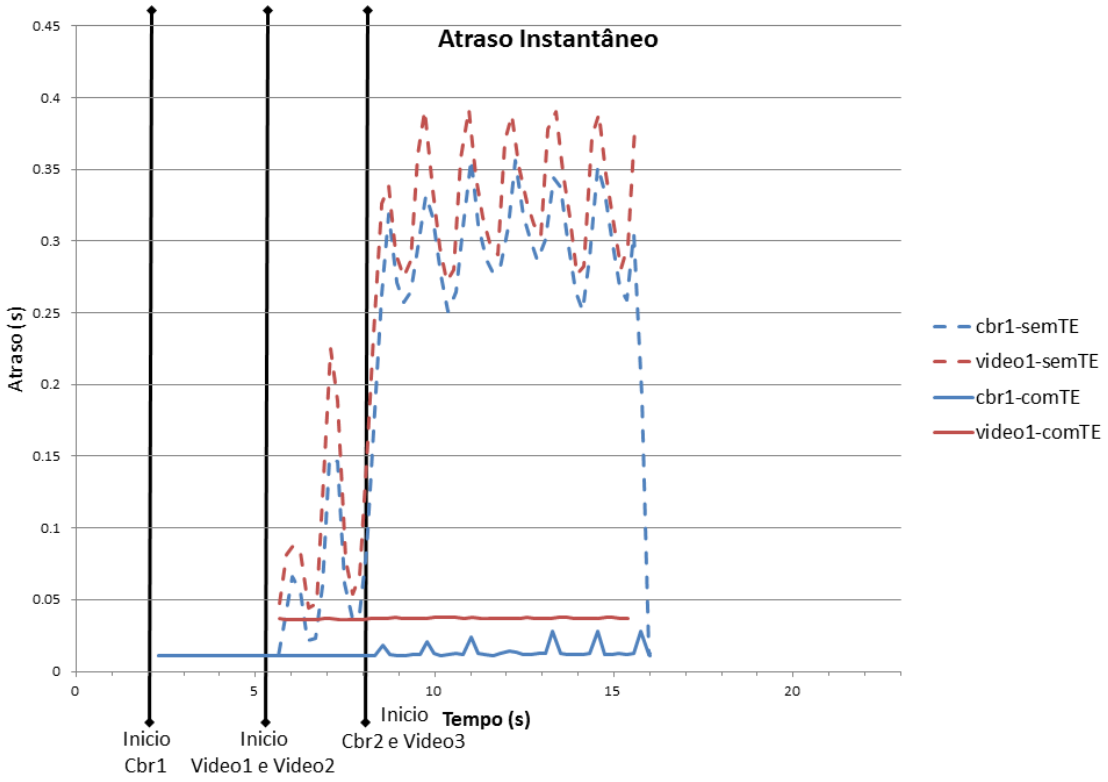


Figura 16 – Atraso instantâneo das aplicações CBR1 e Vídeo1 do primeiro experimento.

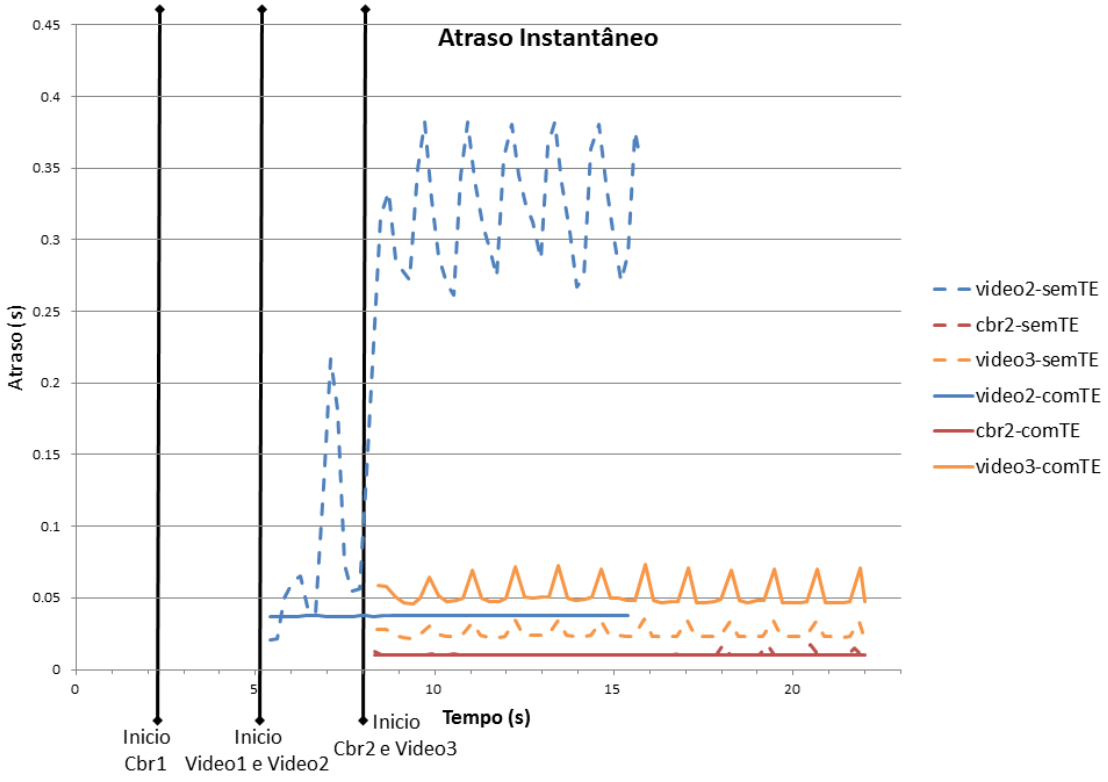


Figura 17 – Atraso instantâneo das aplicações Vídeo2, CBR2 e Vídeo3 do primeiro experimento.

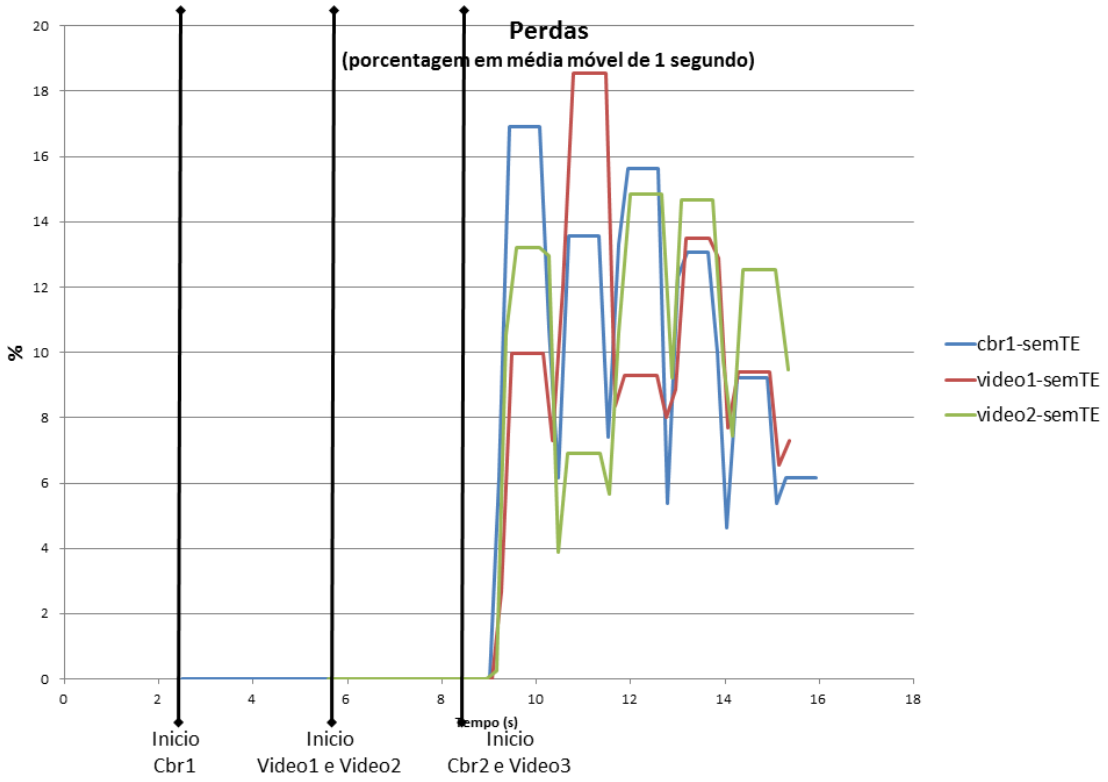


Figura 18 – Porcentagem de perdas das aplicações CBR1, Vídeo1 e Vídeo2 para o sistema semTE do primeiro experimento.

4.1.3 Análise dos Resultados

Os resultados mostram que o módulo otimizador foi capaz de alocar os recursos disponíveis de forma a garantir a QoS de todos os fluxos admitidos. E também que essa solução nem sempre coincide com o menor caminho em número de saltos.

Esse cenário demonstrou a importância de otimizar a utilização dos recursos da rede e que para alcançar uma maior eficiência é necessário considerar a capacidade máxima de recursos e a demanda do conjunto de aplicações. Esse objetivo é abordado por todos os trabalhos apresentados no capítulo 2, porém tratado de formas diferentes. Nos trabalhos de Majd e Yaghmaee (2006) e Xueshun et al. (2009) a alocação é feita considerando apenas a demanda individual de uma aplicação, uma abordagem pouco eficiente que pode resultar em maiores taxas de bloqueio de admissão. Já em Kulkarni, Sharma e Mishra (2012) e Pant e Sanguankotchakorn (2010) além da demanda da aplicação é considerado a criticalidade de cada enlace, em outras palavras procura-se evitar os enlaces que possuem maior potencial em atender futuras requisições. Essa estratégia ajuda a reduzir a taxa de bloqueios e a aumentar a eficiência de utilização dos recursos, embora ainda seja menos eficiente do que a estratégia adotada pelo sistema proposto, pois aqueles se baseiam em previsão enquanto o sistema proposto é capaz de considerar a demanda real de um

conjunto de aplicações simultaneamente. De todos os trabalhos apresentados, apenas as propostas de Santos e Mateus (2009) e Andrade (2008) também consideram a demanda de um conjunto de aplicações através de um algoritmo genético multiobjetivo e mono-objetivo respectivamente. Por ser um GA multiobjetivo com implementação do NSGA-II, pode-se concluir que a solução de Santos e Mateus (2009) possui maior custo computacional e portanto confere maior atraso no tempo de resposta entre a requisição e a alocação. A solução do sistema proposto é baseada em um GA simples e portanto, para a alocação de uma única aplicação, tende a alcançar um tempo de resposta mediano, entre o GA multiobjetivo e as soluções baseadas em Dijkstra.

4.2 Experimento 2 - Controle de Admissão

4.2.1 Descrição do Cenário

Esse experimento demonstra a atuação do mecanismo de controle de admissão e seu impacto no provimento da QoS. O cenário utilizado é igual ao anterior, com exceção dos enlaces de 10Mbps que foram reduzidos para 5Mbps como pode se observar na Figura 19. Portanto foi utilizado o mesmo conjunto de aplicações apresentado na Tabela 2, assim como seus respectivos requisitos de QoS na Tabela 3.

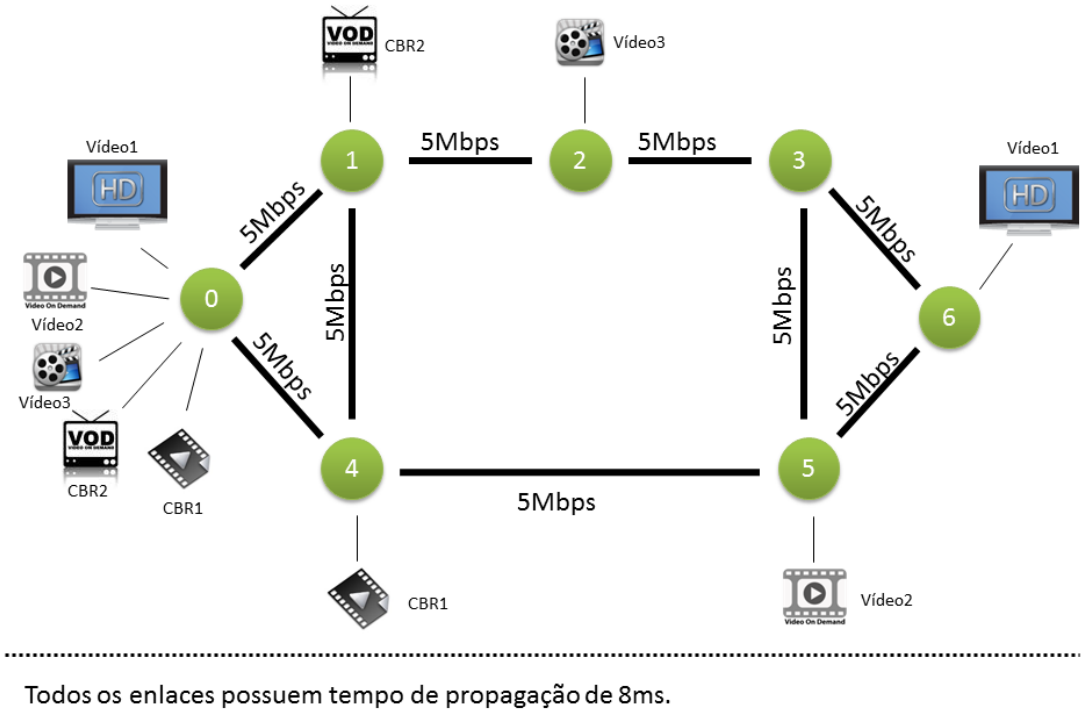


Figura 19 – Cenário do experimento 2.

4.2.2 Resultado

O experimento foi realizado com o sistema de TE proposto completo e com o sistema de TE sem o controle de admissão. As rotas alocadas pelos dois sistemas diferem apenas para o fluxo CBR2 que teve sua admissão negada pelo sistema de TE completo, como mostra a Tabela 5.

Tabela 5 – Relação de rotas alocadas pelos sistemas de TE com e sem controle de admissão.

Aplicação	Rota - semCtrlAdmissão	Rota - comCtrlAdmissão
CBR1	$0 \rightarrow 4$	$0 \rightarrow 4$
Vídeo1	$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$	$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$
Vídeo2	$0 \rightarrow 4 \rightarrow 5$	$0 \rightarrow 4 \rightarrow 5$
CBR2	$0 \rightarrow 1$	não alocado
Vídeo3	$0 \rightarrow 1 \rightarrow 2$	$0 \rightarrow 1 \rightarrow 2$

4.2.2.1 Vazão

A vazão instantânea, no receptor, das aplicações CBR1 e Vídeo1 com e sem o controle de admissão são apresentadas na Figura 20. A vazão instantânea das aplicações Vídeo2, CBR2 e Vídeo3 são apresentadas na Figura 21.

A vazão dos fluxos CBR1 e Vídeo2 não apresentaram diferenças com ou sem o controle de admissão, pois suas rotas não compartilham enlace com a rota alocada para o fluxo CBR2 no sistema sem controle. Já o Vídeo1 apresentou vazão reduzida após o instante 8 s, momento em que o fluxo CBR2 é alocado pelo sistema sem controle de admissão. O Vídeo3 apresentou pequenas diferenças, enquanto o CBR2 teve sua vazão, originalmente constante, bastante perturbada.

4.2.2.2 Atraso

O tempo de atraso total dos pacotes são apresentados na Figura 22 para os fluxos CBR1 e Vídeo1, e Figura 23 para Vídeo2, CBR2 e Vídeo3. Novamente os fluxos CBR1 e Vídeo2 não apresentaram diferenças. Já os fluxos Vídeo1, Vídeo3 e CBR2, que compartilham o enlace $0 \rightarrow 1$, apresentaram grande aumento do atraso no sistema sem controle de admissão.

4.2.2.3 Perda

Os fluxos Vídeo1, CBR2 e Vídeo3 sem controle de admissão apresentaram perda de até 6%, 2% e 14% respectivamente, como mostra a Figura 24. Todos os demais fluxos ocultados apresentaram perda nula durante toda a simulação.

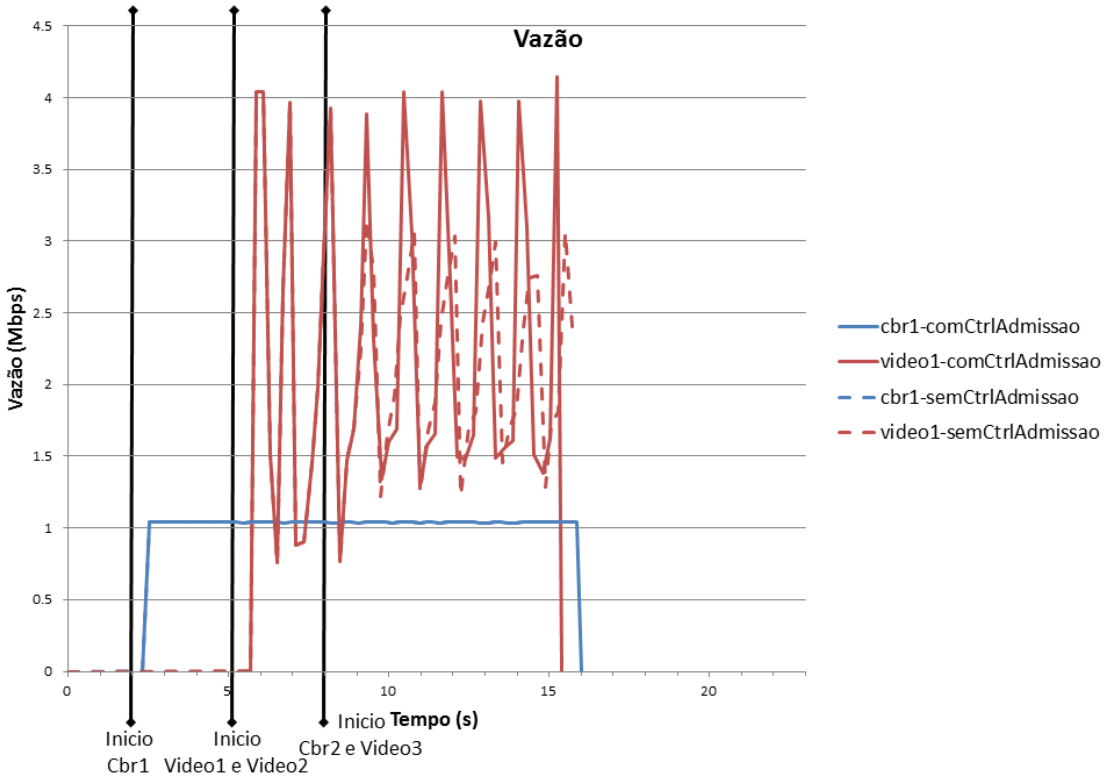


Figura 20 – Vazão instantânea das aplicações CBR1 e Vídeo1 do segundo experimento.

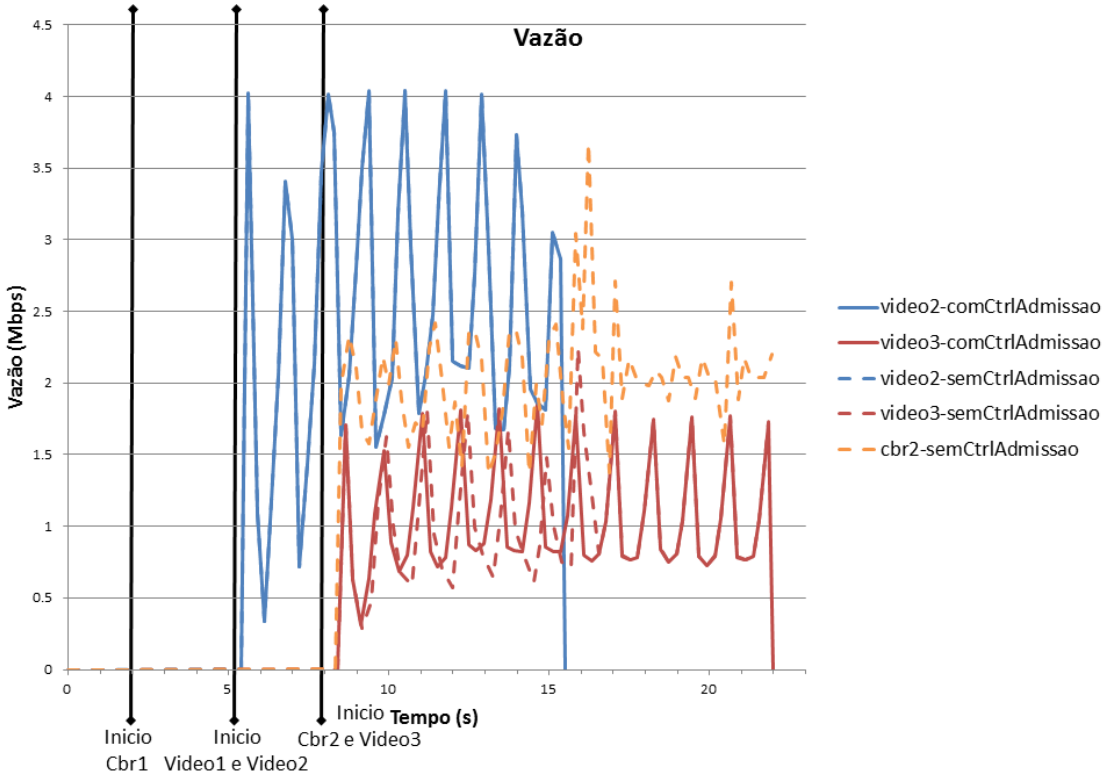


Figura 21 – Vazão instantânea das aplicações Vídeo2, CBR2 e Vídeo3 do segundo experimento.

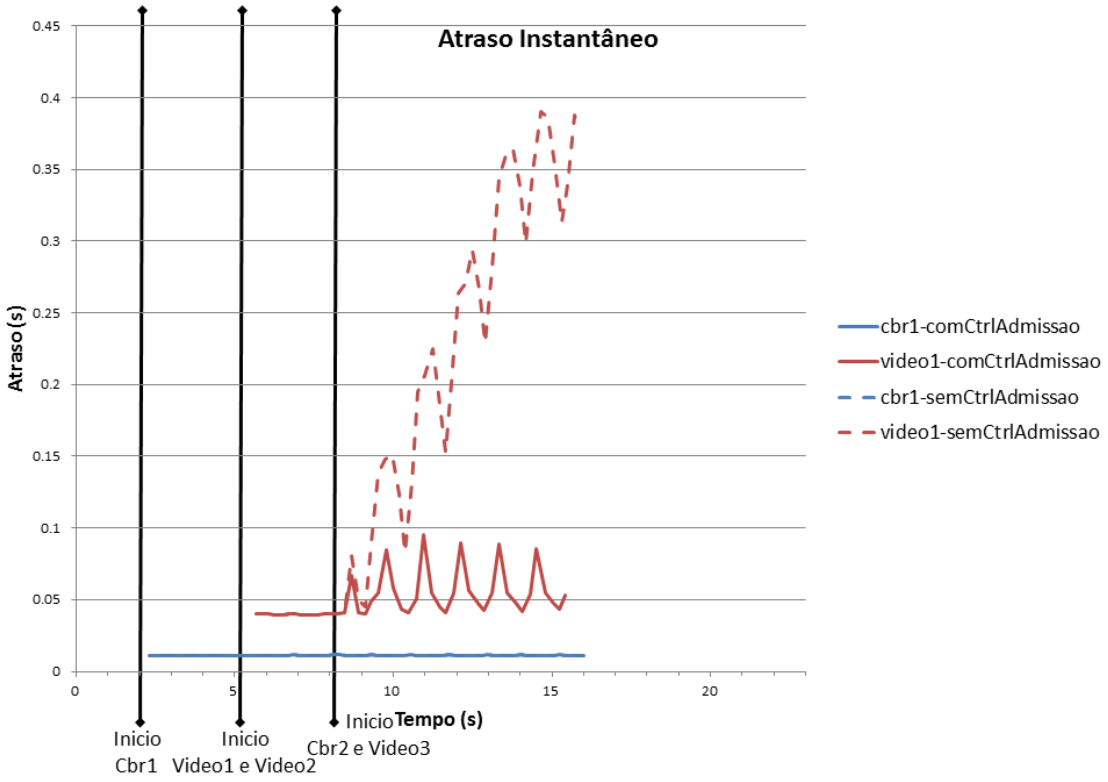


Figura 22 – Atraso instantâneo das aplicações CBR1 e Vídeo1 do segundo experimento.

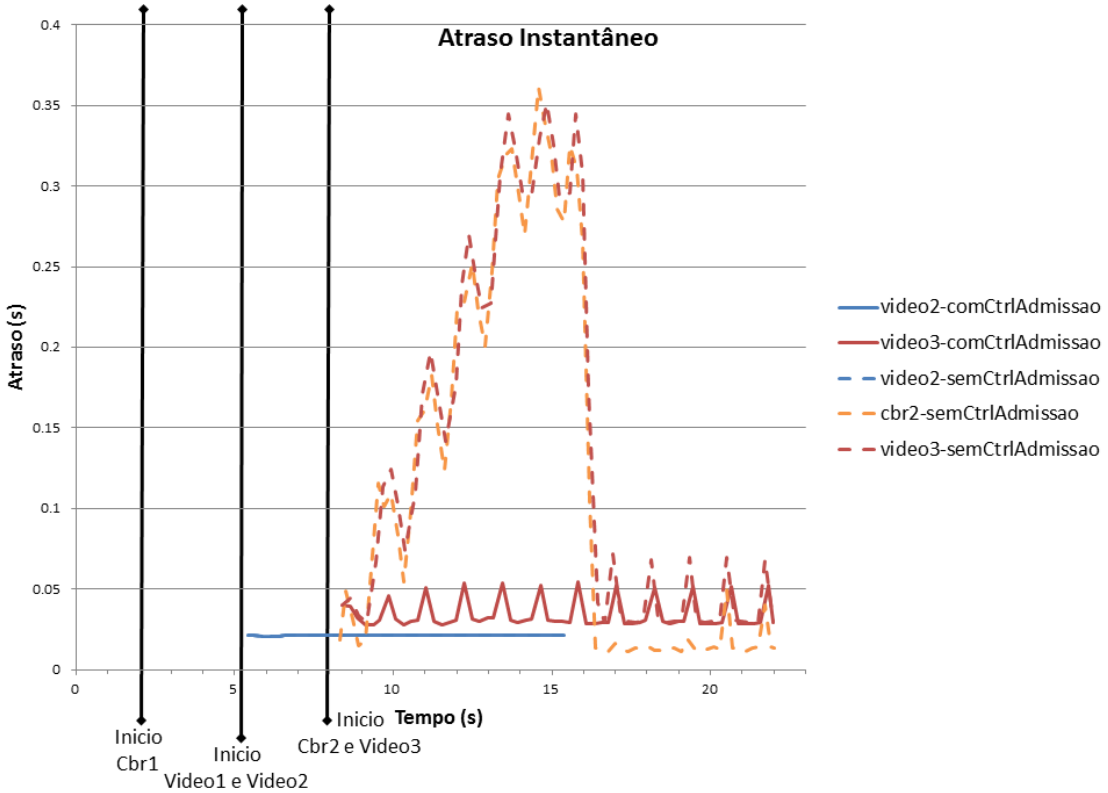


Figura 23 – Atraso instantâneo das aplicações Vídeo2, CBR2 e Vídeo3 do segundo experimento.

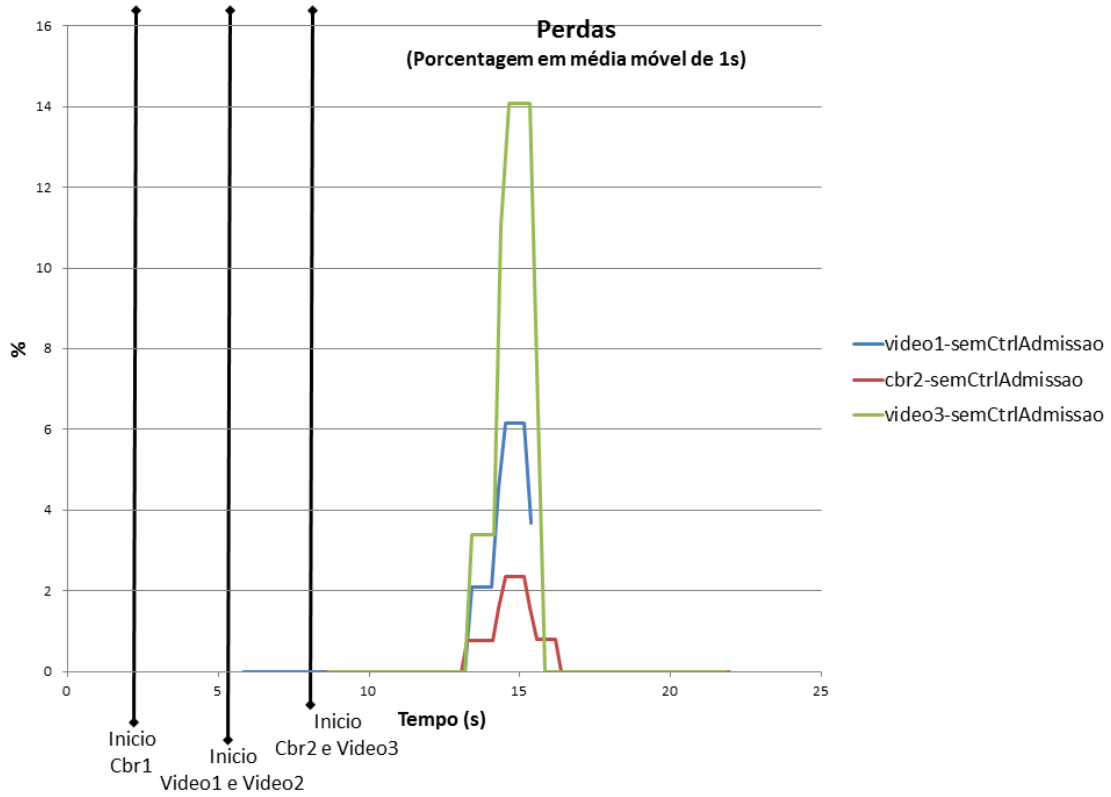


Figura 24 – Porcentagem de perdas das aplicações Vídeo1, CBR2 e Vídeo3 para o sistema semTE do segundo experimento.

4.2.3 Análise dos Resultados

O cenário configurado funcionou como o esperado, gerando uma situação em que a demanda é maior do que os recursos disponíveis. Situação essa que foi detectada e tratada com sucesso pelo mecanismo de controle de admissão. Os resultados de vazão, atraso e perdas mostram que a decisão de negar a admissão do fluxo CBR2 foi realmente necessária para manter a QoS dos fluxos estabelecidos.

A garantia de QoS, necessária para se cumprir contratos SLA, só é possível com a combinação de um mecanismo de controle de admissão e a constante manutenção da QoS, através da monitoração e atuação do sistema de engenharia de tráfego. O controle de admissão é indispensável para tratar situações em que a demanda é maior do que os recursos disponíveis, e portanto trabalhos como de Pant e Sanguankotchakorn (2010), que sempre alocam todas as requisições, apenas fazem um esforço para maximizar a QoS, mas não garante o mínimo necessário. Já em Majd e Yaghmaee (2006), Santos e Mateus (2009) e Xueshun et al. (2009) existe o controle de admissão, mas apenas para a garantia de banda. Banda e atraso são garantidos em Kulkarni, Sharma e Mishra (2012) e Maia (2006), porém apenas a garantia desses dois parâmetros também não satisfazem as necessidades de QoS de aplicações de vídeo e voz, pois a qualidade destas dependem

também dos parâmetros *jitter* e perdas, como foi mostrado nas recomendações de QoS do capítulo 2 (Tabela 1).

4.3 Experimento 3 - Falha de Enlace

4.3.1 Descrição do Cenário

Com o objetivo de avaliar o comportamento do sistema de TE diante de uma falha de enlace, neste cenário as três aplicações listadas na Tabela 6 iniciam, no instante 2 s, suas transmissões na rede ilustrada na Figura 25. O vídeo *Soccer* é uma filmagem, de um jogo de futebol, retirada de Technische Universität München e Institute for Data Processing (2011), que também é a fonte do vídeo *Oktoberfest* já mencionado na descrição do experimento 1. Uma falha no enlace $0 \rightarrow 1$ ocorre no instante 9 s, tornando o enlace indisponível até o fim da simulação.

Além das três aplicações que solicitam os recursos apresentado na Tabela 7, cinco aplicações iniciadas no instante 1 s compõem o tráfego de fundo com o protocolo de transferência de arquivo (*File Transfer Protocol* - FTP). As aplicações FTPs não solicitam recursos ao sistema de TE e portanto são consideradas como não prioritárias por este. O tráfego gerado pelo protocolo FTP não possui taxa fixa, mas se comporta como uma aplicação oportunista utilizando toda a banda que encontrar disponível. O controle da taxa é feita através do mecanismo de controle de congestionamento do protocolo de transporte TCP que aumenta gradativamente a taxa enquanto a transmissão ocorre sem perdas. Quando perdas são detectadas pelo TCP, a taxa é reduzida automaticamente. O resultado é uma rede na qual os enlaces utilizados pelas aplicações FTPs estão sempre no limite do congestionamento.

Tabela 6 – Perfil das aplicações do experimento3.

Aplicação	Ícone	Vazão Média / Pico	Descrição
Vídeo1		2.3 Mbps / 3.0 Mbps	Taxa de bit variável vídeo "Soccer"em HD
Vídeo2		1.0 Mbps / 1.4 Mbps	Taxa de bit variável vídeo "Oktoberfest"em SD
CBR		0.6 Mbps / 0.6 Mbps	Taxa de bit constante

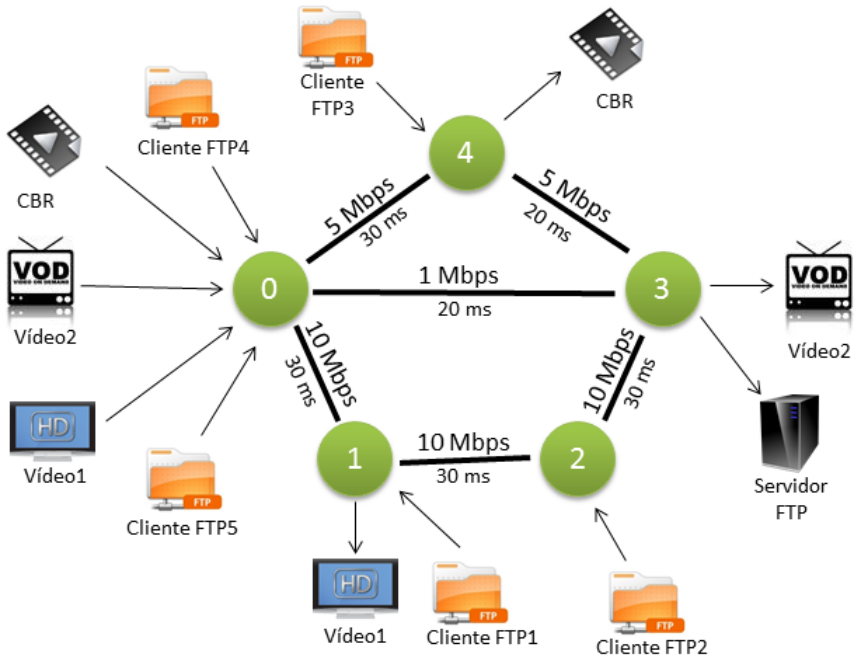


Figura 25 – Cenário do experimento 3.

Tabela 7 – Requisitos de QoS do experimento 3.

Aplicação	Icone	Banda Mínima	Atraso Máximo	Início	Término
Vídeo1		3.0 Mbps	150 ms	2 s	16 s
Vídeo2		1.5 Mbps	150 ms	2 s	16 s
CBR		0.7 Mbps	150 ms	2 s	20 s

4.3.2 Resultado

O experimento foi realizado com e sem o sistema de TE proposto. Inicialmente apenas a alocação da rota do Vídeo2 diferiu do menor caminho, entretanto após a falha de enlace a configuração de rotas dos dois sistemas ficaram completamente diferentes, como mostra a Tabela 8.

Sem o sistema de TE apenas a aplicação Vídeo1, afetada diretamente pela falha do enlace $0 \rightarrow 1$, foi realocada para o novo menor caminho $0 \rightarrow 3 \rightarrow 2 \rightarrow 1$. Já o sistema de TE decidiu realocar não somente a aplicação diretamente afetada pela falha, mas também realocou a aplicação CBR em busca de uma melhor distribuição dos recursos disponíveis dado o novo estado da rede. O impacto dessas decisões do sistema nos parâmetros de QoS vazão, atraso e perdas, é avaliado nas seções seguintes.

Tabela 8 – Relação de rotas alocadas pelos sistemas com e sem TE para o experimento 3.

Aplicação	Antes da falha		Após falha	
	Rota - semTE	Rota - comTE	Rota - semTE	Rota - comTE
Vídeo1	$0 \rightarrow 1$	$0 \rightarrow 1$	$0 \rightarrow 3 \rightarrow 2 \rightarrow 1$	$0 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$
Vídeo2	$0 \rightarrow 3$	$0 \rightarrow 4 \rightarrow 3$	$0 \rightarrow 3$	$0 \rightarrow 4 \rightarrow 3$
CBR	$0 \rightarrow 4$	$0 \rightarrow 4$	$0 \rightarrow 4$	$0 \rightarrow 3 \rightarrow 4$
*FTP1	$1 \rightarrow 0 \rightarrow 3$		$1 \rightarrow 2 \rightarrow 3$	
*FTP2	$2 \rightarrow 3$		$2 \rightarrow 3$	
*FTP3	$4 \rightarrow 3$		$4 \rightarrow 3$	
*FTP4	$0 \rightarrow 3$		$0 \rightarrow 3$	
*FTP5	$0 \rightarrow 3$		$0 \rightarrow 3$	

*As rotas das aplicações não prioritárias não diferem no sistema com ou sem TE, pois são sempre determinadas pelo OSPF.

4.3.2.1 Vazão

Os resultados de vazão instantânea para cada uma das três aplicações prioritárias nos cenários com e sem TE estão representados na Figura 26. Observa-se que antes da falha a vazão das aplicações Vídeo1 e CBR são iguais para os dois cenários. Já a aplicação Vídeo2 no cenário sem TE apresenta inicialmente uma vazão inferior que aumenta gradativamente até alcançar e estabilizar no mesmo nível da vazão dessa mesma aplicação no cenário com TE. Tal comportamento se deve ao fato da aplicação Vídeo2, ao começar sua transmissão pelo caminho $0 \rightarrow 3$ no cenário sem TE, encontrar a rede já ocupada pelas aplicações FTPs. Portanto a disputa inicial pelo enlace congestionado reduz a vazão do Vídeo2, no entanto ao detectar o congestionamento as aplicações FTPs reduzem suas taxas e assim a aplicação Vídeo2 domina o enlace. No cenário com TE tal comportamento não acontece não só pelo sistema de TE ter alocado o Vídeo2 em uma rota diferente, mas também porque o sistema de TE confere prioridade na transmissão dos pacotes das aplicações prioritárias e portanto não depende do mecanismo de controle de congestionamento do TCP para garantir sua vazão desde o início da transmissão.

No momento da falha, percebe-se uma perturbação na vazão das três aplicações do cenário com TE, mas que logo é restaurada para os níveis anteriores a falha. A aplicação Vídeo1 sofre a maior perturbação por estar alocada no enlace que sofre a falha, a aplicação CBR sofre uma perturbação por ter sido realocada em outra rota e a aplicação Vídeo2 sofre uma pequena perturbação por receber em sua rota o fluxo da aplicação Vídeo1.

Já no cenário sem TE, a vazão das aplicações Vídeo1 e Vídeo2 é reduzida no momento da falha e permanece assim até o fim da simulação, pois ambas passam a disputar uma rota com capacidade inferior a necessária. A aplicação CBR não sofre qualquer

perturbação já que sua rota não sofre com a queda e nem recebe novo fluxo.

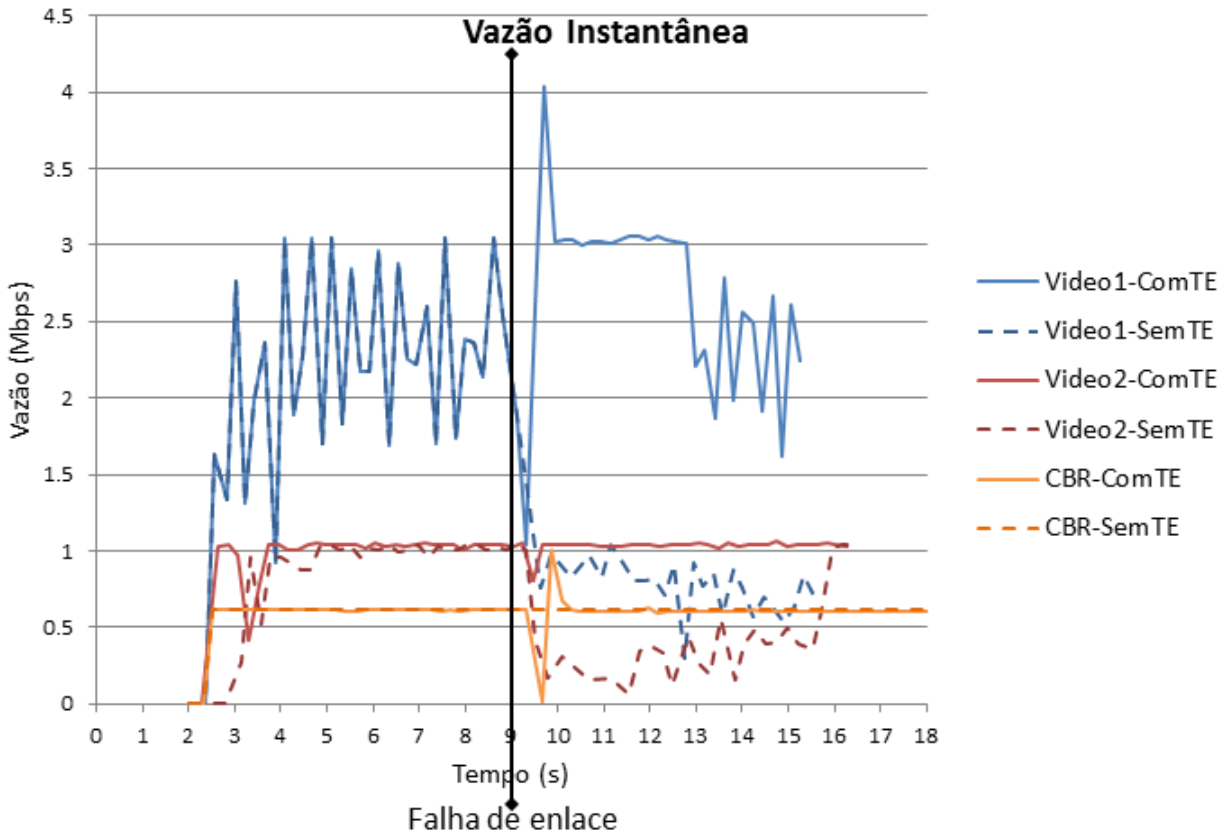


Figura 26 – Vazão instantânea das aplicações Vídeo1, Vídeo2 e CBR do terceiro experimento.

4.3.2.2 Atraso

Os atrasos instantâneos obtidos para as três aplicações prioritárias em ambos cenários simulados neste experimento são apresentados na Figura 27. As aplicações Vídeo1 e CBR apresentam o mesmo comportamento de atraso para ambos cenários antes da falha, pois suas rotas coincidem no cenário com e sem TE, e também não há tráfego de fundo nessas rotas e portanto as condições são exatamente as mesmas. Já a aplicação Vídeo2 apresenta atraso muito superior no cenário sem TE por ter iniciado sua transmissão em uma rota já congestionada pelo tráfego de fundo FTP.

No cenário com TE as três aplicações prioritárias sofrem variação do atraso no momento da falha. O Vídeo1 é o mais afetado pois no momento da falha ele é temporariamente redirecionado para a menor rota $0 \rightarrow 3 \rightarrow 2 \rightarrow 1$ que não dispõe dos recursos necessários e portanto o tempo de fila começa a crescer até que o sistema de TE encontra uma nova solução satisfatória e redireciona o Vídeo1 para a rota $0 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$, reduzindo e estabiliziando o atraso. O Vídeo2 sofre um pequeno aumento no atraso, pois após a falha passa a compartilhar parte de sua rota com o Vídeo1 e suas demandas combinadas praticamente usam toda a capacidade que a rota dispõe. A aplicação CBR é

realocada de sua rota original, com o intuito de liberar recursos para os Vídeos 1 e 2, para uma rota com maior tempo de propagação, o que explica o aumento em seu atraso.

No cenário sem TE a aplicação CBR não sofre perturbação no atraso, já que não houve qualquer alteração nas condições de sua rota inicial. Já o Vídeo1 é redirecionado para o menor caminho $0 \rightarrow 3 \rightarrow 2 \rightarrow 1$, no qual já se encontrava alocado o Vídeo2. Com isso o enlace $0 \rightarrow 3$ fica congestionado e o atraso do Vídeo1 cresce rapidamente saturando nos níveis máximos que o enlace permite. O atraso do Vídeo2 não se altera porque já se encontrava saturado no tempo limite de espera em fila. Portanto qualquer aumento de demanda nessa situação de congestionamento contribui para o aumento de perdas, como será mostrado na seção de análise de perdas a seguir.

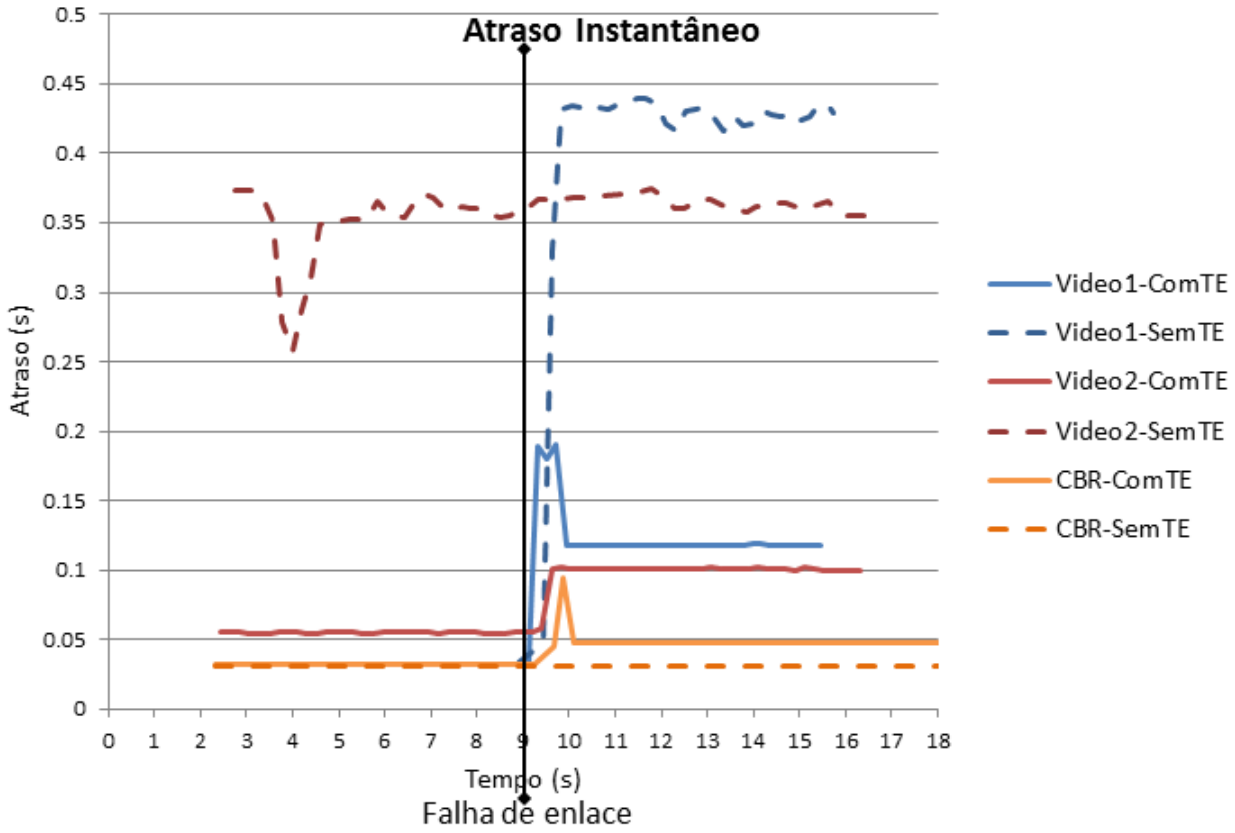


Figura 27 – Atraso instantâneo das aplicações Vídeo1, Vídeo2 e CBR do terceiro experimento.

4.3.2.3 Perda

As perdas resultantes da simulação dos cenários com e sem TE são apresentadas na Figura 28. As aplicações que não tiveram perda alguma durante toda a simulação foram omitidas do gráfico. Para as demais, as perdas foram calculadas em porcentagem com média móvel de um segundo.

Antes da falha apenas uma aplicação apresenta perdas, o Vídeo2 no cenário sem TE. Isso porque no cenário sem TE quando o Vídeo2 é alocado na rota $0 \rightarrow 3$, ela já está

em 100

No momento da falha, no cenário sem TE, o fluxo Vídeo1 é realocado no novo menor caminho que compartilha o enlace $0 \rightarrow 3$ com a rota do Vídeo2. Esse enlace que já estava congestionado passa então a causar perdas acima dos 70

Já no cenário com TE, o Vídeo1 perde alguns pacotes no momento da falha e é temporariamente redirecionado para o novo menor caminho, enquanto o sistema de TE procura por uma solução ótima. Entretanto o menor caminho não possui capacidade suficiente para atender a demanda desta aplicação que acaba por congestionar o enlace $0 \rightarrow 3$. Assim que o sistema de TE atua, realocando o fluxo Vídeo1 em sua nova rota ótima, as perdas para esta aplicação retornam a zero e assim permanece até o fim da simulação.

A solução ótima encontrada pelo sistema de TE não envolve apenas a realocação do Vídeo1, mas exige também a realocação da aplicação CBR para que recursos sejam liberados na nova rota do Vídeo1. Dessa forma a aplicação CBR é realocada para a rota $0 \rightarrow 3 \rightarrow 4$ que embora possua capacidade suficiente para atender a demanda desta aplicação, encontra-se congestionada pelos pacotes do Vídeo1 logo após a falha, gerando assim as perdas observadas para a aplicação CBR. Mas com a solução ótima aplicada, o enlace logo se estabiliza eliminando as perdas da aplicação CBR. O Vídeo2 não sofre perda alguma durante a simulação.

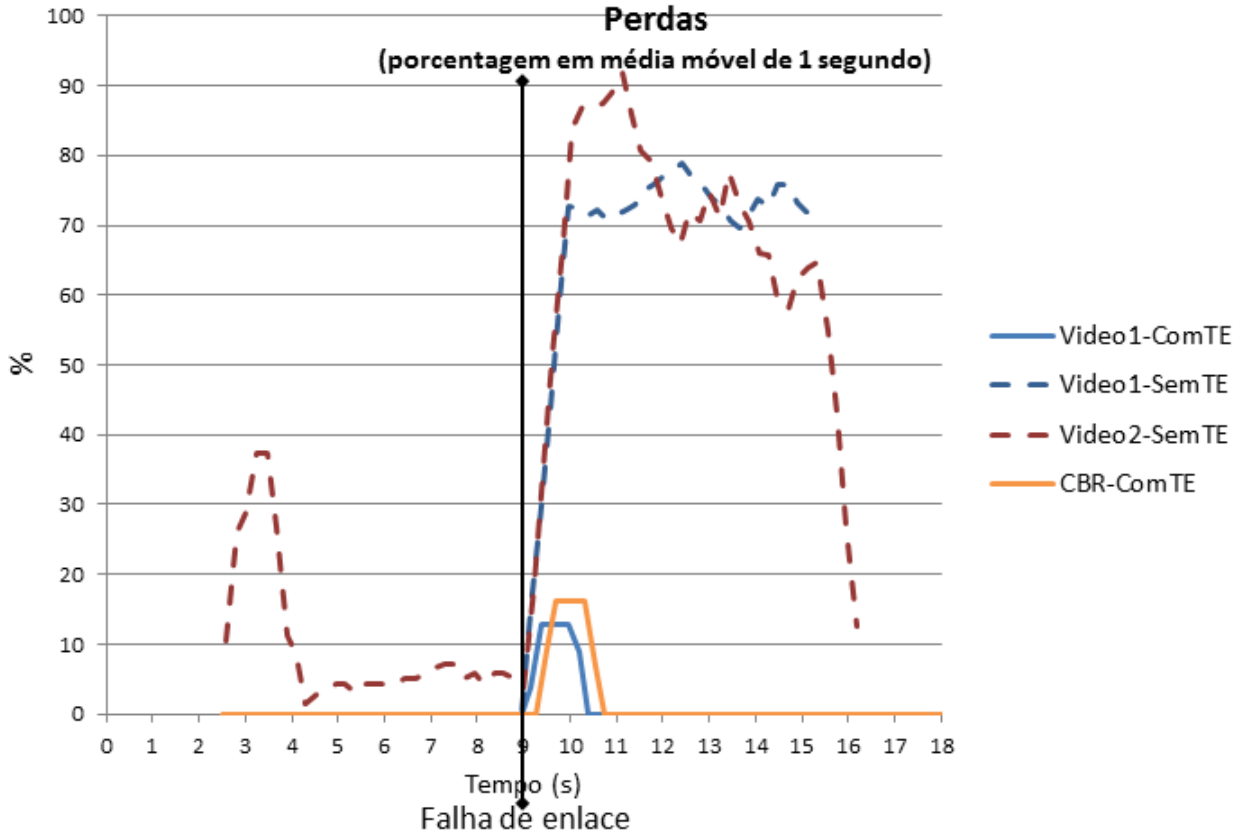


Figura 28 – Porcentagem de perdas das aplicações Vídeo1, Vídeo2 e CBR do terceiro experimento.

4.3.3 Análise dos Resultados

O objetivo desse experimento é avaliar o sistema de TE proposto em uma situação de falha na rede. O sistema se mostrou capaz de reagir à falha e se reconfigurar, buscando a melhor utilização dos recursos disponíveis para continuar atendendo a QoS das aplicações admitidas. A inclusão de um tráfego de fundo no cenário permitiu verificar que a transmissão de tráfego não gerenciado não interfere nos recursos alocados pelo sistema de TE. Os resultados para o cenário de comparação, sem o sistema de TE, mostram que a solução do menor caminho não é adequada, justificando portanto as decisões do sistema de TE por caminhos alternativos.

Dos trabalhos relacionados apenas Maia (2006) se propôs a tratar o problema de recuperação de falhas. Em sua proposta, o sistema de TE realoca a aplicação atingida pela falha para uma rota LSP alternativa, que foi previamente configurada durante a alocação dessa aplicação. Já o sistema proposto nesse trabalho aciona o otimizador para encontrar um novo caminho, para só então realocar a aplicação. Claramente o sistema com rotas *backup* possui um tempo de reação menor, porém em um ambiente complexo e dinâmico, que é a rede convergente, dificilmente um caminho alternativo calculado em um outro momento continuará com a mesma disponibilidade de recursos. O que reduz

drasticamente a chance do caminho alternativo atender as necessidades da aplicação. Essa estratégia pode ainda gerar congestionamento no caminho alternativo, atrapalhando a QoS de outras aplicações. Outra desvantagem é que no momento que se calcula uma rota *backup*, nem sempre é possível encontrar uma rota completamente diferente, i.e pode ocorrer da rota *backup* possuir um ou mais enlaces em comum com a rota principal. E como não se sabe qual enlace irá falhar, existe a possibilidade da solução alternativa também ser afetada pela falha. Portanto, mesmo que uma nova busca por um caminho ótimo tenha um tempo de reação maior, possui a vantagem de garantir que o novo caminho irá manter a QoS necessária.

O tempo médio de execução do otimizador, para os três cenários apresentados, foi de 416 ms. Todos os experimentos foram realizados em um computador com duas CPUs Intel Xeon 6-Core E5645 e 32 GB de memória RAM DDR3-1333.

CAPÍTULO 5

CONCLUSÕES

A proposta deste trabalho foi o desenvolvimento de um Sistema de Engenharia de Tráfego autogerenciável, para operação em uma rede IP. A rede IP foi escolhida por ser compatível com uma grande variedade de dispositivos multimídia e também por ser uma tecnologia aberta a adaptações. O objetivo principal foi o de obter uma rede convergente e aderente ao SLA, ou seja, capaz de garantir QoS a aplicações diversas como serviços de voz, imagem, áudio, dados e vídeo, além de buscar maior eficiência e robustez na operação e manutenção da rede. Os benefícios imediatos são a redução dos custos de operação, administração e manutenção da infraestrutura de rede e a oferta de novos serviços integrados e interativos.

Junto com os benefícios, esta rede convergente traz porém uma demanda muito dinâmica e complexa, tornando o trabalho do gerente de rede ineficiente. Por esse motivo foi proposto que o sistema seja autogerenciável, minimizando a necessidade da interferência humana em seu funcionamento. Buscou-se nos conceitos da computação autonômica as características necessárias para se alcançar a autogerência eficiente, dentre elas principalmente o autoconhecimento, a autoconfiguração, a auto-otimização e a autocura. Para a obtenção dessas características foram utilizados mecanismos da engenharia de tráfego, como o monitoramento contínuo e atualização do estado da rede por meio da tecnologia *Link State*, a configuração de rotas explícitas através do MPLS, a modelagem do tráfego (*traffic shaping*) com o gerenciamento de fila CBQ e a busca por alocações ótimas de caminhos por meio de um algoritmo genético.

A otimização por meio da modelagem de tráfego com filas prioritárias é geralmente criticada por não respeitar o princípio da equidade, o qual estabelece que todos os dados devem receber o mesmo tratamento independentemente de seu tipo. Porém, ao contrá-

rio de vários outros trabalhos de TE que implementam diferentes classes de prioridade baseada no tipo de serviço (voz, vídeo, dados, etc.), o sistema de TE apresentado neste trabalho não fere o princípio da equidade, pois não faz qualquer distinção quanto ao tipo de dado, mas procura atender as necessidades de QoS que lhes são solicitadas. Em outras palavras, as aplicações possuem direitos iguais a solicitarem recursos independente de seus respectivos conteúdos. A única razão do uso de duas classes no CBQ é a preocupação com a compatibilidade do sistema com a rede IP. Em uma situação ideal todas aplicações solicitariam recursos e portanto todas seriam igualmente "prioritárias". Porém forçar todas as aplicações de uma rede real a solicitarem recursos ao sistema seria inviável, para isso foi criada a classe "não-prioritária". A própria aplicação se classifica como não prioritária no momento em que escolhe abrir mão de seu direito de solicitar recursos.

Para a garantia da QoS foram ainda desenvolvidos:

- uma interface para que cada aplicação especifique sua demanda de recursos através da definição de quatro parâmetros: a banda mínima, o atraso máximo, a variação do atraso (*jitter*) máxima, e a máxima taxa de perdas;
- um mecanismo de controle de admissão que permita somente a alocação daquelas aplicações cujas demandas podem ser atendidas pela rede em seu estado corrente;
- um gerenciador dos fluxos alocados que acompanha ao longo de suas respectivas transmissões a QoS alcançada, acionando a busca por soluções alternativas quando detectadas falhas de enlace ou QoS abaixo do nível acordado.

Optou-se pela utilização do simulador de redes ns-2, largamente usado em pesquisas na área de redes, para a implementação e experimentação do sistema proposto. Embora várias das tecnologias adotadas nesse trabalho, como MPLS, CBQ, OSPF, estejam disponíveis no ns-2, parte da arquitetura proposta não tinha como ser implementada diretamente no simulador, mais especificamente os módulos de comportamento autonômico, algoritmo genético e controle de admissão. Foi então desenvolvido um ambiente de simulação em C++ que se utiliza do ns-2, possibilitando a simulação de redes autonômicas. Apesar da experimentação e teste do sistema ter sido a necessidade primária para o desenvolvimento do ambiente de simulação, considerou-se desde o início de seu desenvolvimento que tal ambiente poderia ser útil para trabalhos futuros e portanto este foi projetado e implementado de forma modularizada e reutilizável. O resultado é um ambiente que permite a troca de módulos inteiros, como por exemplo a troca do GA por um outro algoritmo de otimização, sem que toda a estrutura do ambiente tenha de ser refeita.

O ambiente implementa ainda uma interface simplificada com o usuário, possibilitando ao pesquisador que não domina o ns-2 a configurar seus próprios cenários de experimentação, sem ter que escrever uma única linha de código do ns-2. Isto inclusive já

foi testado no grupo de pesquisa, no qual alunos de iniciação científica puderam trabalhar em módulos como o de medição e otimização e verificar o resultado de suas alterações sem ter contato direto com o ns-2.

Para testar o sistema de engenharia de tráfego foram configurados três cenários de simulação. Embora não tenha sido possível reproduzir os experimentos dos trabalhos de outros autores, devido a não disponibilidade das estruturas específicas de suas implementações, foi possível realizar comparações conceituais entre as diversas propostas para cada cenário. O primeiro experimento mostrou que o sistema é capaz de otimizar a alocação dos recursos disponíveis para atender a uma demanda próxima da capacidade total da rede. Em comparação com as estratégias de outros trabalhos, foram ressaltados pontos que indicam que o sistema proposto é capaz de alcançar uma maior eficiência de utilização da rede. Em contrapartida é esperado que tenha um tempo de resposta de alocação mais lento, devido a sua maior complexidade computacional, com exceção do trabalho de Santos e Mateus (2009), que poderia alcançar uma maior eficiência ao custo de um tempo de resposta ainda mais lento.

Com o segundo experimento, mostrou-se que, diante de uma demanda maior do que a capacidade máxima da rede, é necessária a atuação de um mecanismo de controle de admissão que aceite apenas aquelas aplicações que puderem ser atendidas dentro de seus requisitos de QoS. Embora mecanismos semelhantes estejam presentes em alguns dos trabalhos relacionados, o sistema proposto é o único capaz de garantir todos os parâmetros de QoS necessários, segundo as recomendações de QoS para serviços multimídia.

O terceiro e último cenário simulou uma situação de falha na qual um enlace da rede tem seu funcionamento interrompido abruptamente. Nessa situação o sistema de TE autogerenciável foi capaz de reagir e recuperar a QoS necessária para todas as aplicações afetadas, embora não tenha sido capaz de evitar uma perturbação temporária no momento da falha. Uma análise comparativa da estratégia proposta neste trabalho com aquela proposta por Maia (2006), mostra que esta útlima pode não recuperar a QoS necessária e ainda perturbar a QoS de outras aplicações não afetadas diretamente pela falha, embora possua um tempo de reação menor.

O sistema proposto demonstrou autoconhecimento ao gerenciar seus recursos disponíveis com eficiência, sem prometer o que não poderia cumprir. Demonstrou autootimização ao encontrar soluções que maximizam a demanda admitida e autoconfiguração ao atuar e implementar suas próprias soluções sem a interferência humana. Mostrou ainda um grau de autocura ao reagir à falha de enlace, recuperando e mantendo a QoS nos padrões acordados. Considera-se portanto alcançado o objetivo principal deste trabalho, que é a implementação e avaliação de um sistema autogerenciável como solução de engenharia de tráfego para a garantia de QoS e maximização da eficiência de utilização da rede. Para se alcançar esse objetivo foi necessário cumprir os objetivos específicos, como a implemen-

tação do ambiente de simulação, do mecanismo de admissão, do módulo de medição, do módulo de otimização e do gerenciador de fluxos. Buscou-se ainda a viabilização proposta ao utilizar tecnologias já existentes como o MPLS, OSPF e CBQ, além de concentrar a complexidade do sistema nos roteadores de borda e de implementar o sistema de TE de forma paralela ao tradicional modelo *best effort*, garantindo a compatibilidade com as inúmeras aplicações da rede IP.

Entretanto, para a implementação do sistema proposto em uma rede real, falta ainda superar algumas limitações da versão atual. É necessário expandir o ambiente de simulação para que seja possível realizar experimentos com o sistema de TE instalado em mais de um roteador de entrada do domínio. Também é necessário realizar experimentos que relacione a convergência e o tempo de resposta do otimizador. Na análise comparativa deste trabalho com outros relacionados, concluiu-se que o tempo de resposta possa ser uma possivel desvantagem do algoritmo quando é necessário alocar apenas uma aplicação.

Algumas sugestões de trabalhos futuros são investigar e propor uma melhoria para o otimizador, para que este funcione com diferentes níveis de tempo de resposta, podendo oferecer uma solução viável em tempo rápido e uma solução ótima em tempo médio, e com isso atender melhor as diversas necessidades de QoS. Pode-se também desenvolver um mecanismo que confira a propriedade de autoproteção ao sistema, que não foi considerada neste trabalho. Outra possibilidade é a de criar um sistema cliente para gerenciar a QoS na última milha do sistema de comunicação. Tal sistema poderia estimar e adaptar a necessidade de QoS de suas aplicações, conforme os recursos disponíveis no núcleo da rede, através da negociação com o sistema de TE proposto, e com isso buscar a maximização da qualidade de experiência (*Quality of Experience* - QoE) do usuário. Também é necessário realizar mais experimentos em diferentes topologias, uma vez que nesse trabalho foram realizados experimentos apenas em redes pequenas com o objetivo de mostrar que o sistema proposto funciona pelo menos nos cenários apresentados.

REFERÊNCIAS

ANDRADE, A. V. *Provisionamento de Qualidade de Serviço em Redes MPLS utilizando Algoritmos Bio-inspirados em um Ambiente de Tráfego Auto-Similar*. Tese (Doutorado em Engenharia Elétrica) — Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, 2008. Citado 6 vezes nas páginas 27, 29, 34, 47, 48 e 59.

ANDRADE, A. V. et al. Analysis of selection and crossover methods used by genetic algorithm-based heuristic to solve the lsp allocation problem in mpls networks under capacity constraints. *EngOpt 2008 - International Conference on Engineering Optimization*, 2008. 2008. Citado na página 26.

ANDRADE, A. V.; ERRICO, L. de; ASSIS, L. Estudo comparativo entre algoritmos imunológicos e algoritmo genético aplicados ao problema de estabelecimento de rotas com restrição de capacidade utilizando o protocolo mpls. *Simpósio Brasileiro de Pesquisa Operacional*, 2009. 2009. Citado na página 34.

AWDUCHE, D. et al. *RFC3272 - Overview and Principles of Internet Traffic Engineering*. [S.l.], 2002. Disponível em: <http://tools.ietf.org/html/rfc3272>. Acesso em: 6.8.2013. Citado na página 20.

AWDUCHE, D. et al. *RFC2702 - Requirements for Traffic Engineering Over MPLS*. [S.l.], 1999. Disponível em: <http://www.ietf.org/rfc/rfc2702.txt>. Acesso em: 6.8.2013. Citado 2 vezes nas páginas 15 e 19.

CHAKRABORTY, M.; CHAKRABORTY, U. K. An analysis of linear ranking and binary tournament selection in genetic algorithms. *International Conference on Information, Communications and Signal Processing*, 1997. v. 1, p. 407–411, 1997. Citado na página 26.

CHEN, Y.; FARLEY, T.; YE, N. Qos requirements of network applications on the internet. *IOS Press: Information Knowledge Systems Management*, 2004. 2004. Citado na página 17.

CISCO. *Quality of Services Overview*. [S.l.], 2011. Disponível em: <http://www-.cisco.com/en/US/docs/ios/qos/configuration/guide/qos_overview.pdf>. Acesso em: 12.9.2013. Citado na página 19.

COSTA, L. H. M. K.; DUARTE, O. C. M. B. A scalable qos-based routing mechanism for supporting multimedia applications. *International Conference on Multimedia Computing and Systems - ICMCS*, 1999. v. 2, p. 346–351, 1999. Citado na página 28.

DENKO, M. K.; YANG, L. T.; ZHANG, Y. *Autonomic Computing and Networking*. [S.l.]: Springer, 2009. Citado na página 25.

DéSOBLIN, G.; PAPINI, H. Sla management: a key differentiator for service providers. *Telecommunications Review - 3rd Quarter*, 2001. 2001. Citado na página 15.

FLOYD, S.; JACOBSON, V. Random early detection gateways for congestion avoidance. *IEEE/ACM. Transactions on Networking*, 1993. v. 1, n. 4, 1993. Citado na página 22.

FLOYD, S.; JACOBSON, V. Link-sharing and resource management models for packet networks. *IEEE/ACM. Transactions on Networking*, 1995. v. 3, n. 4, 1995. Citado na página 23.

GARCIA, A. L.; GARCIA, B.; FRIEDMANN, C. V. Resolução de um problema de equilíbrio de trabalho e distribuição de agentes de serviço. *Simpósio Brasileiro de Pesquisa Operacional*, 2000. Viçosa, n. 32, p. 923–935, Novembro 2000. Citado na página 27.

GHEIN, L. D. *MPLS Fundamentals*. [S.l.]: Cisco Press, 2007. Citado na página 21.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. [S.l.]: Addison Wesley, 1989. Citado na página 27.

HOLLAND, J. H. *Adaptation in natural and artificial systems*. [S.l.]: University of Michigan Press, 1989. Citado 2 vezes nas páginas 25 e 27.

HORN, P.; IBM. *Autonomic computing: IBM's perspective on the state of information technology*. [S.l.], 2001. Citado 2 vezes nas páginas 15 e 24.

ITU-T. *Recommendation ITU-T E.800 - Definitions of terms related to quality of service*. [S.l.], 2008. Disponível em: <http://www.itu.int/rec/T-REC-E.800-200809-I>. Acesso em: 22.8.2013. Citado na página 17.

KAR, K.; KODIALAM, M.; LAKSHMAN, T. Minimum interference routing of bandwidth guarenteed tunnels wth mpls traffic engineering applications. *IEEE Journal on Selected Areas in Communications*, 2000. v. 18, n. 12, p. 2566–2579, 2000. Citado na página 28.

KEIMEL, C. et al. Visual quality of current coding technologies at high definition iptv bitrates. *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2010. p. 390–393, 2010. Citado na página 52.

KULKARNI, S.; SHARMA, R.; MISHRA, I. New qos routing algorithm for mpls networks using delay and bandwidth constraints. *International Journal of Information and Communication Technology Research*, 2012. v. 2, n. 4, 2012. Citado 3 vezes nas páginas 28, 58 e 63.

LEWIS, C.; PICKAVANCE, S. *Implementing Quality of Service Over Cisco MPLS VPNs*. [S.l.]: Cisco Press, 2006. Citado na página 18.

MACEDO, R. J. de A. A vision on autonomic distributed systems. *II Workshop de Sistemas Distribuídos Autonômicos - XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2012. p. 31–34, 2012. Citado na página 25.

MAIA, N. A. *Engenharia de Tráfego em Domínio MPLS Utilizando Técnicas de Inteligência Computacional*. Tese (Doutorado em Engenharia Elétrica) — Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, 2006. Citado 4 vezes nas páginas 29, 63, 71 e 75.

MAJD, N. E.; YAGHMAEE, M. H. A fuzzy algorithm for qos-based routing in mpls network. *Asia-Pacific Conference on Communications*, 2006. p. 1–5, 2006. Citado 3 vezes nas páginas 27, 58 e 63.

MCKENNEY, P. Stochastic fairness queueing. *Proceedings of INFOCOM*, 1990. 1990. Citado na página 23.

MOY, J. *RFC1131 - OSPF Version 2*. [S.l.], 1991. Disponível em: <http://www.ietf-.org/rfc/rfc1247.txt?number=1247%7C>. Acesso em: 03.12.2013. Citado na página 20.

NASSIF, A. T. *Redes da Próxima geração: aspectos técnicos, econômicos e cenários de migração*. Tese (Dissertação de Mestrado) — Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, 2004. Citado na página 14.

NS2. *The Network Simulator - ns*. [S.l.], 2011. Disponível em: <http://nsnam.isi-.edu/nsnam/index.php/Main_Page>. Acesso em: 13.10.2013. Citado na página 42.

ORFANUS, D. et al. Performance of wireless network simulators: a case study. *PM2HW2N: Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2008. p. 59–66, 2008. Citado na página 42.

PANT, R.; SANGUANKOTCHAKORN, T. A heuristic approach for bandwidth constraint path selection in mpls based networks. *International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON)*, 2010. p. 909–913, 2010. Citado 3 vezes nas páginas 28, 58 e 63.

PAUL, S. *Digital Video Distribution in Broadband, Television, Mobile and Converged Networks*. [S.l.]: John Wiley & Sons, 2011. 235-236 p. Citado na página 18.

PETERSON, L. L.; DAVIE, B. S. *Computer Networks: A Systems Approach*. [S.l.]: Elsevier Science, 2003. Citado na página 20.

SANTOS, F. A.; MATEUS, G. R. Otimização multi-objetivo aplicada à alocação dinâmica de rotas lsp em redes mpls. *27º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2009. p. 349–362, 2009. Citado 4 vezes nas páginas 30, 59, 63 e 75.

SRINIVAS, M.; PATNAIK, L. M. Genetic algorithms: a survey. *IEEE: Computer*, 1994. v. 27, n. 6, p. 17–26, 1994. Citado na página 26.

Technische Universität München; Institute for Data Processing. *TUM LDV Multi Format Test Set*. 2011. Disponível em: <http://www.ldv.ei.tum.de/videolab>. Citado 2 vezes nas páginas 52 e 64.

THOMAS, B.; GRAY, E. *RFC3037 - LDP Applicability*. [S.l.], 2001. Disponível em: <http://www.ietf.org/rfc/rfc3037.txt>. Acesso em: 21.8.2013. Citado na página 21.

TRUSZKOWSKI, W. et al. *Autonomous and Autonomic Systems - with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*. [S.l.]: Springer, 2009. Citado na página 25.

VASCONCELOS, J. A. et al. Improvements in genetic algorithms. *Transactions on Magnetics*, 1995. n. 37, 1995. Citado na página 26.

WANG, Z.; CROWCROFT, J. Quality of service routing for supporting multimedia applications. *IEEE JSAC*, 1996. p. 1288–1294, 1996. Citado 2 vezes nas páginas 25 e 28.

XUESHUN, W. et al. An improved multiple objectives optimization of qos routing algorithm base on genetic algorithm. *5th International Conference on Wireless Communications, Networking and Mobile Computing*, 2009. p. 1–4, 2009. Citado 3 vezes nas páginas 30, 58 e 63.

YEN, J. Y. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 1970. v. 27, p. 526–530, 1970. Citado na página 45.

YEN, J. Y. Finding the k shortest loopless paths in a network. *Management Science*, 1971. v. 17, p. 712–716, 1971. Citado na página 45.