Algoritmo adaptativo para balanceamento de carga para tráfego tipo melhor esforço em redes IP/MPLS

M. D. Cavalcante, T. M. R. Lima, V. L. da Costa, M. F. Magalhães, R. S. Mendes
DCA - FEEC - UNICAMP - Caixa Postal 6101
Campinas, SP, Brasil, CEP 13083-970
Email: {madaca,tulius,vinicius,mauricio,rafael}@dca.fee.unicamp.br

Sumário

A má utilização dos recursos disponíveis de uma rede de computadores pode levar ao congestionamento de elementos dessa rede. Uma forma de melhorar a utilização de recursos é controlar o caminho seguido por um fluxo de pacotes, rotulando-os, em uma técnica conhecida como MPLS (*Multiprotocol Label Switching*). Neste trabalho, é proposto um algoritmo para balanceamento de carga entre LSPs (*Label Switched Paths*) configurados entre os nós de ingresso e de egresso de uma nuvem MPLS. São apresentados dois algoritmos, sendo um encontrado na literatura e um proposto. O balanceamento de carga é visto como um problema de otimização, em que se quer minimizar o valor da função de custo a partir de seu gradiente. Como função de custo são utilizados o atraso médio por pacote, o número médio de pacotes por LSP e a taxa de perda de pacotes; como parâmetro, o fluxo de pacotes por LSP. Para calcular o gradiente é utilizado o Método da Secante. Resultados são apresentados e comparados.
Palavras-chave: Engenharia de Tráfego, MPLS, Balanceamento de Tráfego, *Qualidade de Serviço, Internet*.

Abstract

The unbalanced utilization of a network resources can result in network congestion. One way to improve the usage of a network resources is to control the trajectory of packet flows by adding a label on packet headers. This technique is called MPLS (Multiprotocol Label Switching). This report presents two load balancing algorithms (one proposed) that improve network utilization by distributing traffic among LSPs (Label Switched Paths) already configured between an ingress-egress pair of nodes in an MPLS domain. The load balancing problem is presented as an optimization problem. The cost functions chosen are mean packet delay, mean number of packets on the observed network and packet loss rate; the parameter of interest is the packet flow per LSP. The cost function gradient estimation is based on the Secant Method. Results are presented and commented.
Keywords: Traffic Engineering, MPLS, Load Balancing, *Quality of Service, Internet*.

1 Introdução

Congestionamento é um dos problemas mais significativos em um contexto IP aparecendo na rede basicamente porque os recursos da rede são insuficientes ou porque são utilizados inadequadamente para acomodar a carga oferecida, fazendo com que determinadas partes da rede estejam sub-utilizadas, enquanto outras estão sobrecarregadas (Awduche *et al.*, 1999; Awduche, 1999). Quase sempre, congestionamentos resultam em degradação da qualidade de serviço para usuários finais.

Engenharia de tráfego (*Traffic Engineering* ou TE) é o aspecto das redes IP que está ligado à otimização do desempenho da rede (Awduche *et al.*, 2001). TE engloba tecnologia e princípios científicos, para medir, modelar, caracterizar e controlar o tráfego Internet, a fim de facilitar a operacionalidade da rede, ao mesmo tempo em que é otimizada a utilização dos recursos disponíveis e que são minimizados congestionamentos; é uma função que controla a resposta da rede a demandas de tráfego e outros estímulos, como falhas. Um dos objetivos da TE é prover garantia de qualidade de serviço (*Quality of Service* ou QoS) aos usuários finais dos serviços prestados (Xiao e Ni, 1999). A engenharia de tráfego engloba quatro problemas básicos: o controle de admissão de novas conexões; o roteamento de pacotes, dadas algumas restrições; o re-roteamento de conexões já estabelecidas; e o planejamento dos recursos da rede (Girish *et al.*, 2000).

MPLS (*Multiprotocol Label Switching*) ou mais genericamente GMPLS (*Generalized Multiprotocol Label Switching*) (Awduche e Rekhter, 2001; Banerjee *et al.*, 2001) é uma tecnologia de uso crescente que permite que redes IP sejam submetidas à engenharia de tráfego, ao permitir o desvio do tráfego do "menor caminho", com isso permitindo melhor uso da infra-estrutura da rede. A partir do MPLS, define-se uma nova forma de encaminhamento de pacotes em redes IP. A idéia básica é a possibilidade de agrupamento de pacotes em fluxos, semelhante ao que fazem as redes ATM (*Asynchronous Transfer Mode*) e Frame-Relay (Girish *et al.*, 2000). Uma vez rotulados, o encaminhamento de pacotes é feito baseado apenas no rótulo, independente do cabeçalho IP. A classificação de pacotes em fluxos é feita por meio de filtros que examinam os campos do cabeçalho IP, como endereço de fonte, endereço de destino, bits de tipo de serviço etc. Ao mapear esses fluxos em LSPs (*Label Switched Paths*) é possível ao provedor de serviços Internet executar a função de engenharia de tráfego (Kodialam e Lakshman, 2000). Uma característica importante de redes que utilizam MPLS é a possibilidade de configuração de LSPs por explicitação do caminho. Isto significa que, quando um LSP está sendo configurado, é possível especificar todos os roteadores intermediários entre os nós de ingresso e de egresso (Kodialam e Lakshman, 2000; Suri *et al.*, 2000).

A capacidade do MPLS para engenharia de tráfego pode ser vista em vários níveis. Um exemplo dessa divisão em níveis pode ser:

- Alocação de LSPs para fluxos conhecidos *a priori* (contratuais, por exemplo), via algoritmos *offline*. Esse nível está ligado à instalação de caminhos de longa duração, a exemplo das VPNs (*Virtual Private Networks*).

- Alocação de LSPs para fluxos com retrições de tempo severas e de duração de serviço curta, como, por exemplo, um cliente que quer estabelecer em pouco tempo uma videoconferência. Nesse caso, devem ser utilizados algoritmos *online*.

- Balanceamento de carga entre LSPs paralelos para fluxos do tipo melhor esforço em LSPs configurados a partir de algoritmos *offline* ou *online*. Podem ser utilizados, nesse caso, caminhos já configurados para um dos níveis superiores, ou caminhos estabelecidos exclusivamente para o tráfego tipo melhor esforço. A motivação para esse tipo de algoritmo é que, apesar de hoje em dia haver várias classes de serviço como áudio e vídeo, a maioria do tráfego Internet é do tipo melhor esforço.

Em um nível mais alto, deve-se alocar todos os LSPs na infra-estrutura da rede, visando fazer um uso de alguma forma otimizado dos recursos da rede, ao mesmo tempo em que são garantidos parâmetros de QoS como, por exemplo, atraso máximo, variação de atraso máxima e taxa de descarte de pacotes. Nesse caso, são considerados conhecidos todos os túneis ou LSPs a serem roteados, os recursos disponíveis e requisitos de tolerância a falhas. *No nível intermediário*, o roteamento baseia-se em informações de estado da rede propagadas por protocolos de roteamento. Os algoritmos para esse nível têm um compromisso com a otimalidade e o tempo de execução. *O nível mais baixo* da arquitetura de TE apresentada, por sua vez, está associado ao tráfego melhor esforço. O objetivo desse nível de TE é preencher a banda residual de enlaces após alocado todo o tráfego prioritário ou com exigências de QoS. Esse preenchimento deve ser feito através do balanceamento de carga entre LSPs paralelos. Por esse procedimento, o nó de ingresso decide como o tráfego será dividido entre os LSPs, levando em conta parâmetros como variação de atraso e perdas de pacotes em cada LSP. Alguns exemplos de algoritmos que podem ser associados a um dos níveis da arquitetura de TE apresentada podem ser encontados em Xiao *et al.* (2000), Liu *et al.* (2000), Jüttner *et al.* (2000), Kodialam e Lakshman (2000), Suri *et al.* (2000); Widjaja e Elwalid (2000); e Elwalid *et al.* (2001).

No contexto deste trabalho, será focalizada atenção no nível mais baixo da arquitetura apresentada. A motivação para este tipo de algoritmo é que, apesar de hoje em dia haver muitos tipos de classes de serviço como audio e vídeo, o tráfego Internet é formado basicamente da classe melhor esforço. E mesmo se no futuro requisitos de QoS aumentarem, o tráfego melhor esforço ainda existirá, de forma que seu impacto na rede não pode ser descartado.

A divisão do trabalho foi feita conforme apresentado a seguir. A seção 2 apresenta as características comuns dos dois algoritmos para balanceamento de carga entre LSPs configurados entre pares ingresso-egresso de uma nuvem MPLS apresentados neste trabalho. Os algoritmos são formulados como um problema de otimização resolvido a partir do cálculo do gradiente da função de custo escolhida. Dentre os algoritmos apresentados, um foi proposto e o outro encontra-se descrito em Widjaja e Elwalid (2000) e Elwalid *et al.* (2001). As seções 3 e 4 apresentam as características particulares desses dois algoritmos. A seção 5 apresenta os resultados de simulação para uma topologia apresentada em Elwalid *et al.* (2001). Por fim, a seção 6 apresenta as conclusões do trabalho.

2 MATE: algoritmos de balanceamento para TE em redes MPLS

Conforme mencionado, este trabalho focaliza o nível mais baixo da arquitetura de TE considerada. Dois algoritmos serão apresentados e comparados. Esses algoritmos apresentam características comuns, podendo ser agrupados em uma ferramenta de engenharia de tráfego, chamada de MATE (*MPLS Adaptive Traffic Engineering*).

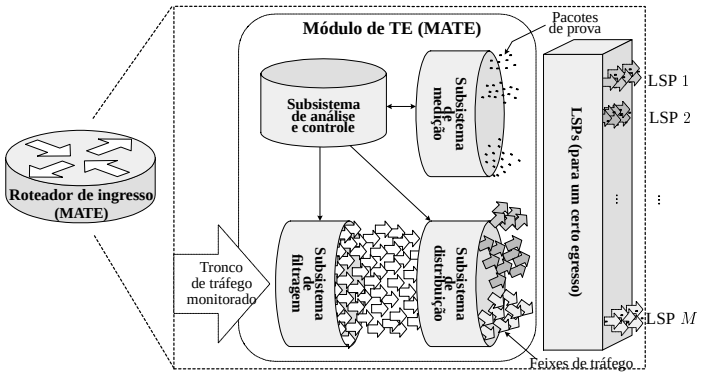


Figura 1: Roteador de ingresso (subpartes necessárias à execução dos algoritmos MATE)

O primeiro destes algoritmos é chamado de El-MATE (Elwalid MATE)[1] e é baseado na descrição do algoritmo apresentada em Widjaja e Elwalid (2000) e Elwalid *et al.* (2001), com algumas modificações que visam diminuir o tráfego de pacotes de controle e medição (pacotes de prova). O outro algoritmo é chamado de AS-MATE (*Adaptive Shifting MATE*). Este é baseado na descrição do El-MATE e do algoritmo de Gallager apresentado em Gallager (1977) e Cassandras *et al.* (1990).

A principal característica dos algoritmos MATE para o terceiro nível da arquitetura de TE apresentada é evitar congestionamentos na rede, balanceando a carga entre LSPs existentes entre os nós de ingresso e de egresso de um domínio MPLS. Esses algoritmos assumem que os caminhos entre os nós de ingresso e de egresso são configurados usando um método convencional, como o CR-LDP (*Constraint Routing Label Distribution Protocol*) ou RSVP-TE (*Resource reServation Protocol for Traffic Engineering*) e devem responder adaptativamente a mudanças no estado da rede para uma situação de carga quase-estacionária. As mudanças no estado da rede são detectadas por um monitoramento periódico do atraso dos pacotes de prova. Dentre as vantagens desses algoritmos citam-se: são distribuídos, no sentido de que cada nó de ingresso executa uma instância do algoritmo; são adaptativos, por ajustarem a configuração de carga de acordo com o estado corrente da rede; os protocolos são fim-a-fim (apenas os nós de ingresso e de egresso participam do processo); e as decisões de roteamento são baseadas em medições. Ao todo, três funções de custo são definidas: atraso médio por pacote, número médio de pacotes por LSP, e taxa de perda de pacotes (em conjunto com uma das funções anteriores). O parâmetro a ser variado é o tráfego ao longo de cada LSP. Basicamente, o problema de associar pacotes a um determinado LSP é definido como um problema de otimização, em que a função de custo deve assumir um valor menor possível.

Para executar as funções exigidas pelos algoritmos MATE, o roteador de ingresso deve conter um módulo de engenharia de tráfego composto por quatro subsistemas principais: um subsistema de *filtragem*, um subsistema de *distribuição*, um subsistema de *medição* e um subsistema de *análise e controle*, conforme ilustra a Figura 1.

O *subsistema de filtragem* divide o tronco de tráfego para um dado par ingresso-egresso igualmente entre N feixes menores. O número de feixes, N, determina a mínima quantidade de tráfego que pode ser desviada de um LSP para outro. Em Widjaja e Elwalid (2000), é feita uma

[1]Na referência original, esse algoritmo é chamado simplesmente de MATE. No contexto deste trabalho, será chamado de El-MATE, para diferenciá-lo do conjunto de algoritmos para engenharia de tráfego, MATE

discussão sobre diferentes formas de se definir um feixe. O *subsistema de distribuição* mapeia os N feixes de tráfego entre os LSPs existentes até o nó de egresso, levando em conta que a ordem dos pacotes deve estar preservada, ao chegarem ao nó de egresso. Essa distribuição de feixes entre os LSPs permite o balanceamento de carga, objetivo final do algoritmo. O *subsistema de medição* determina o estado de cada LSP. A medição é feita *enviando pacotes de prova* (Figura 1) periodicamente ao longo de cada LSP. A freqüência desses pacotes de prova deve ser baixa, em relação à taxa de pacotes da fonte monitorada (cinco por cento ou menos, segundo Widjaja e Elwalid (2000)), para não alterar de forma significativa as medidas de congestionamento da rede. O envio de pacotes de prova deve ser dependente da classe dos pacotes de mensagem que se está monitorando. Para tanto, os pacotes de prova devem conter o mesmo tipo de cabeçalho que os pacotes monitorados (Elwalid *et al.*, 2001). O *subsistema de análise e controle* é responsável pelo controle dos demais subsistemas e execução de decisões de roteamento. Esse sistema, por exemplo, determina o número de feixes (N) em que se vai dividir o tronco de tráfego monitorado, o número de feixes a serem encaminhados pelo subsistema de distribuição em cada um dos LSPs para o egresso de interesse, e a freqüência de pacotes de prova a serem enviados pelo subsistema de medição e as informações neles contidas. Os algoritmos de balanceamento de carga são executados nesse subsistema.

Os algoritmos MATE utilizam a informação de taxa de variação da função de custo definida em relação ao parâmetro de interesse e a taxa de perda de pacotes, para determinar o número de feixes em cada LSP. Sejam: m o índice do LSP em observação (esse índice muitas vezes será mantido implícito); a_k, o instante de saída do k-ésimo pacote de prova do nó de ingresso, com destino ao nó de egresso; e d_k, o instante de chegada do k-ésimo pacote de prova ao nó de egresso. A medição do atraso do k-ésimo pacote de prova, r_k, é feita da seguinte forma: o nó de ingresso adiciona a informação a_k (processo chamado *timestamping*) ao pacote de prova e o envia ao nó de egresso; ao chegar ao nó de egresso, este calcula o atraso do k-ésimo pacote de prova usando a seguinte equação: $r_k = d_k - a_k$. Essa forma de calcular o atraso de pacotes é diferente da referência original. Em Widjaja e Elwalid (2000) e Elwalid *et al.* (2001), os autores sugerem que o nó de egresso apenas adicione a informação d_k ao cabeçalho do k-ésimo pacote de prova, mandando-o de volta ao nó de ingresso. O nó de ingresso, então, calcula o atraso de cada pacote e atualiza as estatísticas das medições. A estratégia adotada neste trabalho têm as vantagens de *diminuir o fluxo de pacotes de prova no sentido oposto ao tráfego monitorado*, e de manter o controle do tamanho das amostras (ou, equivalentemente, do período de observação utilizado) no *subsistema de controle* do nó de ingresso, o que centraliza todos os parâmetros variáveis do algoritmo no nó de ingresso.

Durante o processo de medição do estado da rede, algum tempo depois do início do envio de pacotes de prova, o nó de ingresso envia um pacote de prova com a função especial de terminar o processo de medição. Após o recebimento deste pacote, o nó de egresso atualiza as estatísticas das medições feitas e envia ao nó de ingresso os resultados obtidos. Nas simulações apresentadas na seção 5, foram utilizadas fontes TCP para esses pacotes de prova especiais.

O nó de egresso pode atualizar as variáveis de estado de cada LSP a partir das seguintes

equações:

$$\begin{aligned} K &= k_{perdido} + k_{\text{recebido}}; \\ \overline{r} &= \frac{1}{K}\sum_{k=1}^{K} r_k; \text{ e} \\ p\,[perda] &= \frac{k_{\text{perdido}}}{K}. \end{aligned} \tag{1}$$

Nessas equações, k_{perdido} é o número estimado de pacotes de prova perdidos, k_{recebido} é o número estimado de pacotes de prova recebidos, K representa o número total estimado de pacotes de prova enviados pelo nó de ingresso durante o último período de amostragem, $\overline{r}$ é o atraso médio por pacote estimado, e $p\,[perda]$ é a taxa estimada de perda de pacotes. Note-se que a equação (1) leva em consideração K valores de atraso, ou seja, ela inclui os pacotes perdidos também.

Para calcular a taxa de perda de pacotes de prova, acrescentam-se números de seqüência aos cabeçalhos dos pacotes de prova. Dessa forma, o número de pacotes perdidos pode ser estimado, observando-se a quebra nessa seqüência. As perdas podem ser incorporadas na função de custo de duas maneiras. A primeira possibilidade é usar o produto da função de custo e a taxa de perda de pacotes de prova como uma nova função de custo. Essa alternativa pode ser usada quando a freqüência de perda de pacotes não está muito próxima a zero ou a um, e a freqüência de pacotes de prova permite boas medições. A segunda maneira é a de associar um atraso suficientemente grande a cada pacote perdido. Essa alternativa deve ser usada, quando a freqüência dos pacotes de prova for muito pequena, em relação ao fluxo de pacotes monitorado.

Nas próximas seções, serão apresentados os dois algoritmos comparados neste trabalho.

3 Algoritmo El-MATE

Esta seção descreve o algoritmo El-MATE, baseado em Widjaja e Elwalid (2000) e Elwalid *et al.* (2001) e a descrição dos algoritmos MATE apresentada na seção anterior.

Para o El-MATE, a função de custo a ser minimizada é o atraso médio por pacote, considerando todos os LSPs disponíveis. Para minimizá-la, esse algoritmo desvia uma quantidade fixa de feixes de tráfego, n, do LSP de maior derivada da função de custo, para o LSP de menor derivada. Essa variável n é crucial para o desempenho do algoritmo, devendo ser escolhida com cautela. Em Widjaja e Elwalid (2000), é mencionada a importância do uso de um n variável. Uma forma possível de se definir o valor de n apropriadamente a cada iteração do algoritmo de balanceamento de carga pode ser encontrada na seção 4, para o algoritmo AS-MATE (*Adaptive Shifting MATE*). Em Elwalid *et al.* (2001) é mostrado que, quando N é suficientemente grande e n é suficientemente pequeno, o algoritmo é convergente e estável.

Para esse algoritmo, como n é constante, para garantir que desvios de tráfego parem eventualmente de ocorrer e que o fluxo por LSP convirja, o algoritmo pára de desviar feixes de tráfego, quando há a deteção de que o último desvio de tráfego não foi bem sucedido (não trouxe vantagens ao estado da rede). Portanto, o comportamento do algoritmo é caracterizado por fases de operação, conforme ilustra a Figura 2.

Dentre as fases do algoritmo, duas formam a função de engenharia de tráfego propriamente dita: a fase de *balanceamento de carga* (Fase 2) e a fase de *monitoramento* (Fase 3). Na

Fase 1: Divide o tráfego igualmente entre os LSPs

1. Inicializam-se as variáveis de roteamento: $\xi_m^{(0)}(j)$, para todos os LSPs m, fazendo $\xi_{m_0}^{(0)}(j) = N$, e $\xi_m^{(0)}(j) = 0$, para $m \neq m_0$.

2. Para cada LSP $m \neq m_0$,

2.1. Desviam-se $\Delta\xi_m = \left(\frac{N}{M}\right)$ feixes do LSP m_0 para o LSP m;

2.2. Estimam-se J_{m_0} e $\frac{dJ_{m_0}}{d\xi_{m_0}}$, e J_m e $\frac{dJ_m}{d\xi_m}$.

Fase 2: Equaliza as funções de custo

(A cada iteração do algoritmo)

1. Determinam-se os LSPs com maior e menor valores de $\frac{d\widehat{J}_m}{d\xi_m}$.

2. Desvia(m)-se n feixe(s) de tráfego do LSP com maior valor de $\frac{d\widehat{J}_m}{d\xi_m}$, para o LSP de menor valor.

3. Estimam-se J_m e $\frac{dJ_m}{d\xi_m}$, após o desvio de tráfego, para esses dois LSPs.

4. Se a soma dos valores de $\widehat{J}_m$ depois do último desvio de tráfego é maior do que o valor antes do desvio de tráfego (o último desvio fez a função de custo aumentar), para cada LSP:

4.1. Guardam-se os valores de atraso médio por pacote *antes* e *depois* de cada desvio de feixes.

4.2. Passa-se à **Fase 3**.

Fase 3: Monitora o estado da rede

A cada iteração, verifica-se a ocorrência de uma mudança persistente e de tamanho considerável no estado da rede, a partir do teste expresso em Widjaja e Elwalid (2000). Se a mudança for detectada, o algoritmo passa à **Fase 4**.

Fase 4: Ajusta as medições para voltar à Fase 2

1. Alteram-se os valores estimados de derivada, de forma a:

1.1. Dobrar o valor da derivada dos LSPs em que foi verificado um aumento do atraso médio por pacote (desde que o LSP não tenha passado no teste da Fase 3).

1.2. Diminuir pela metade os valores das derivadas dos LSPs em que foi verificada uma diminuição do atraso médio por pacote (desde que não tenha passado no teste da Fase 3).

2. Retorna-se à **Fase 2**.

Figura 2: Algoritmo El-MATE: detalhamento das fases

fase de balanceamento de carga, o algoritmo tenta igualar as medidas de congestionamento, até que o atraso médio total, para todos os LSPs, não possa mais ser diminuído. Quando as medidas de congestionamento estão aproximadamente iguais, o algoritmo passa à fase de monitoramento. Nessa fase, o roteador de ingresso envia continuamente pacotes de prova ao nó de egresso. Nessa etapa, não devem ser feitos desvios de tráfego da fonte monitorada, entre os LSPs estabelecidos. Se uma mudança persistente e de amplitude considerável no estado da rede é detectada, conforme teste encontrado em Wijaja e Elwalid (2000), o algoritmo passa à fase de balanceamento de carga e todo o processo se repete. As outras duas fases do El-MATE têm a função de inicializar e atualizar dados. A Fase 1 inicializa as medições de atraso, incluindo a estimativa da taxa de variação da função de custo do sistema; e a Fase 4 ajusta apropriadamente as medidas de congestionamento, para voltar à segunda fase, após detectada uma mudança no estado da rede, na fase de monitoramento.

Sejam, agora, $\xi_m(j)$, ou simplesmente ξ_m, o número de feixes de tráfego destinados a j e que atravessam o LSP m (essas variáveis serão chamadas de agora em diante de *variáveis de roteamento*); e J_m a função de custo do LSP m. De acordo com o procedimento descrito por Widjaja e Elwalid (2000), o nó de ingresso pode estimar J_m e $\frac{dJ_m}{d\xi_m}$, para a (n)-ésima iteração do

algoritmo, a partir das seguintes expressões:

$$\widehat{J}_m^{(n)} = \frac{1}{K}\sum_{k=1}^{K}[r_k]_m \text{ ; e} \quad (2)$$

$$\frac{d\widehat{J}_m^{(n)}}{d\xi_m} = \frac{\Delta\widehat{J}_m^{(n)}}{\Delta\xi_m} = \frac{\widehat{J}_m^{(n)} - \widehat{J}_m^{(n-1)}}{\Delta\xi_m}. \quad (3)$$

Na equação (2), $[r_k]_m$ representa o atraso do k-ésimo pacote de prova a atravessar o LSP m. O método de aproximação de $\frac{d\widehat{J}_m^{(n)}}{d\xi_m}$ da equação (3) será chamado daqui por diante de Método da Secante. Note-se que, no caso de alguma diferença de sincronismo entre os relógios dos nós de ingresso e de egresso, a estimativa de gradiente pelo Método da Secante não será modificada. Isto implica que o algoritmo *não necessita de sincronismo entre os relógios* dos nós de ingresso e de egresso.

Ao iniciar o algoritmo (começo da Fase 1), considera-se que todo o tronco de tráfego de interesse é roteado através do menor caminho convencional. Seja esse caminho, por exemplo, o LPS m_0, e seja M o número de LSPs entre o par ingresso-egresso em que se está executando o algoritmo de balanceamento de carga (M varia tipicamente entre 2 e 5). A cada passo consecutivo, o algoritmo move $\left(\frac{N}{M}\right)$ feixes do total do tráfego do LSP m_0 para outro LSP. Após esse rearranjo de feixes, outro conjunto de pacotes de prova é enviado para fazer a nova medida de atraso, através da equação (2) e os valores de $\frac{d\widehat{J}_m}{d\xi_m}$ através da equação (3). O processo se repete, e os feixes são transferidos passo-a-passo para os outros LSPs, de forma que, no final da Fase 1, aproximadamente a mesma quantidade de feixes de tráfego atravessam cada um dos LSPs entre os nós de ingresso e de egresso, e as estimativas da função de custo são conhecidas para todo LSP m. O algoritmo passa à segunda fase. Na segunda fase, o algoritmo determina quais os LSPs com maior e menor gradientes da função de custo e transfere n feixes de tráfego do LSP de maior gradiente (e, portanto, menos satisfatório) para o LSP de menor gradiente (mais satisfatório). Após essa transferência de feixes, novos pacotes de prova devem ser enviados nesses LSPs (de maior e de menor derivadas) e nova estimação de atraso médio e de gradiente da função de custo devem ser realizadas, através das equações (2) e (3). Após a segunda medição, devem ser novamente determinados os caminhos com maior e menor gradientes da função de custo, e toda a fase se repete, até que o atraso conjunto dos LSPs de menor e maior derivadas aumente, após o desvio de tráfego, o que significa que a última transferência de feixes não levou a um resultado positivo, ou seja, à diminuição do atraso total de pacotes. O algoritmo passa, então, à terceira fase. A Figura 2 ilustra todo o procedimento acima descrito.

4 Algoritmo AS-MATE

Essa seção apresenta o algoritmo proposto neste trabalho. Maiores detalhes podem ser encontrados em Cavalcante (2001). Para esse algoritmo, a função de custo adotada é o número médio de pacotes do fluxo de interesse por LSP, que é, pela Lei de Little, equivalente a otimizar o atraso médio de pacotes, objetivo do El-MATE. A função de custo foi alterada para ficar de acordo com o algoritmo de Gallager descrito em Gallager (1977) e Cassandras *et al.* (1990). Contudo, simulações utilizando a mesma forma de calcular a derivada do El-MATE também foram feitas. Os resultados são semelhantes.

Sejam m, $\overline{r}_m$ e ξ_m conforme apresentadas anteriormente. Sejam, ainda, f_m, o fluxo esperado entre os nós de ingresso e de egresso, ao longo do LSP m; e J_m, o número médio de pacotes presentes no LSP m (não importa em qual *buffer*). Pela lei de Little, J_m, pode ser considerada uma função do fluxo de pacotes no LSP m. Nesse caso, $J_m = f_m \overline{r}_m$ (Cassandras e Lafortune, 1999).

Durante o processo de otimização, o nó de ingresso precisa decrementar o número de feixes no LSP cujo valor $\frac{dJ_m}{d\xi_m}$ é pequeno e incrementar o número de feixes em que $\frac{dJ_m}{d\xi_m}$ é grande, conforme procedimento descrito em Gallager (1977) e Cassandras *et al.* (1990). Ao final de cada iteração do algoritmo, quando o nó de ingresso recebe as estimativas de J_m, para cada LSP m, o nó de ingresso pode determinar as seguintes variáveis:

$$m_{\min} = \arg\min_m \frac{dJ_m}{d\xi_m}; \tag{4}$$

$$A_i = \min_m \frac{dJ_m}{d\xi_m}; \tag{5}$$

$$a_m(j) = \frac{dJ_m}{d\xi_m} - A_i; \text{ e} \tag{6}$$

$$\Delta\xi_m^{(n)}(j) = \min\left[\xi_m^{(n)}(j), \eta a_m(j)\right], \tag{7}$$

onde $m_{\min}$ é o LSP com menor valor de gradiente da função de custo; A_i é o menor valor de J_m, $\forall m$; e $a_m(j)$ é uma variável auxiliar para o cálculo do número de feixes para o egresso j que devem ser desviados do LSP m na (n)-ésima iteração do algoritmo, $\Delta\xi_m^{(n)}(j)$. Para atualizar as variáveis de roteamento, as seguintes equações podem ser utilizadas:

$$\xi_m^{(n)}(j) = \begin{cases} \xi_m^{(n-1)}(j) - \Delta\xi_m^{(n)}(j), & \text{se } m \neq m_{\min}, \\ \xi_m^{(n-1)}(j) + \sum\limits_{m \neq m_{\min}} \Delta\xi_m^{(n)}(j), & \text{se } m = m_{\min}, \end{cases} \tag{8}$$

O parâmetro η que aparece na equação (7) é definido de acordo com algumas considerações. São elas:

- η deve ser variável, para adaptar o algoritmo a diferentes intensidades do tráfego de interesse. De acordo com Cassandras *et al.* (1990), um valor constante de η não garante um bom comportamento do algoritmo para faixas largas do tráfego de interesse. Uma maneira de solucionar o problema de usar o η constante é substituí-lo por termos de segunda derivada, conforme apresentado em Bertsekas *et al.* (1984).

- O valor de η deve garantir que $\Delta\xi_m(j)$ fique dentro do intervalo $0 \leq \Delta\xi_m(j) \leq \Delta_{\max}$, onde $\Delta_{\max}$ é o máximo número de feixes que podem ser transferidos entre LSPs em apenas uma iteração do algoritmo. Esse limite para o valor de $\Delta\xi_m(j)$ é desejável, porque evita grandes oscilações de tráfego e melhora as condições de estabilidade do algoritmo.

- Se $\Delta\xi_m(j)$ não ficar no intervalo desejado, o valor de η deve ser diminuído. O novo valor de η é determinado através da seguinte equação:

$$\eta = \min_m \left[\frac{\Delta_{\max}}{a_m(j)}\right]. \tag{9}$$

Fase 1: Divide o tráfego igualmente entre os LSPs (Igual à Fase 1 do algoritmo El-MATE)
Fase 2: Equaliza as funções de custo (A cada período de observação, para todos os M LSPs) **1.** Estimam-se $\frac{d\widehat{J}_m}{d\xi_m}$, $\forall m$. **2.** Calculam-se A_i, m_{min}, $a_m\,(j)$, $\Delta\xi_m\,(j)$, a partir das equações de (4) a (7). **3.** Se $\Delta\xi_{m_{\min}}^{(n)}\,(j) = 0$: **3.1.** Se $\Delta\phi_{m_{\min}}^{(n-1)}\,(j) \neq 0$, guarda-se o valor de atraso médio calculado, para todo LSP m. **3.2.** Se uma mudança no estado da rede é detectada, recalcula η. **4.** Se $\Delta\xi_m\,(j) > \Delta_{\max}$, recalcula η. **5.** Atualizam-se as variáveis de roteamento $\xi_m^{(n)}\,(j)$, a partir da equação (8).

Figura 3: Algoritmo AS-MATE: detalhamento das fases

- Após detectar que nenhum feixe foi transferido na última iteração do algoritmo, o nó de ingresso deve guardar o valor do atraso médio por pacote para cada LSP em uma variável especial, para tornar possível detectar se ocorrer uma mudança considerável no estado da rede. Nas iterações seguintes à deteção de que não houve desvio de feixes ($\Delta\xi_{m_{\min}} = 0$), o algoritmo deve recalcular o valor de η, se uma diferença percental no valor do atraso médio medido ultrapassar um certo limiar e o número de feixes a serem desviados for novamente zero.

O AS-MATE, assim como o El-MATE, utiliza o Método da Secante, para calcular o gradiente da função de custo. Neste caso, a seguinte aproximação é utilizada:

$$\frac{d\widehat{J}_m^{(n)}}{d\xi_m} = \left(\frac{\xi_m^{(n)}}{\Delta\xi_m^{(n)}}\right)\overline{r}_m^{(n)} - \left(\frac{\xi_m^{(n)} - \Delta\xi_m^{(n)}}{\Delta\xi_m^{(n)}}\right)\overline{r}_m^{(n-1)},$$

sendo $\overline{r}_m^{(n)}$ calculado através da equação (1).

Por estimar o gradiente da função de custo a partir do Método da Secante, o algoritmo AS-MATE necessita de uma fase de inicialização das medições da função de custo e de uma estimativa de seu gradiente, da mesma forma que o El-MATE. Após a execução dessa primeira etapa, o algoritmo passa à fase de balanceamento de carga propriamente dita, onde são calculados os desvios de feixes de tráfego entre os M LSPs estabelecidos entre ingresso e egresso. Nesta fase, enquanto é detectada a convergência do roteamento (o desvio de tráfego calculado para a iteração seguinte do algoritmo é nulo), é feito um monitoramento da rede. Nesse período, se é detectada uma mudança no estado da rede de um percentual superior a um certo limiar, o algoritmo recalcula o valor de η. As fases do algoritmo estão apresentadas na Figura 3, do ponto de vista do nó de ingresso. Através dessa figura, pode-se perceber que, uma vez na Fase 2, o algoritmo não volta a executar a Fase 1.

A Fase 1, de inicialização, é idêntica à Fase 1 do El-MATE. A Fase 2, por outro lado, difere da Fase 2 do El-MATE, porque no El-MATE o desvio de tráfego é constante e igual a $\Delta\xi_m = n$, durante toda a execução do algoritmo, enquanto no AS-MATE, o algoritmo *calcula* o número de feixes a ser desviado. Além disso, os desvios de tráfego para o El-MATE acorrem apenas do LSP com maior derivada da função de custo para o de menor derivada, enquanto para o AS-MATE ocorre de todos os LSPs para o de menor derivada.

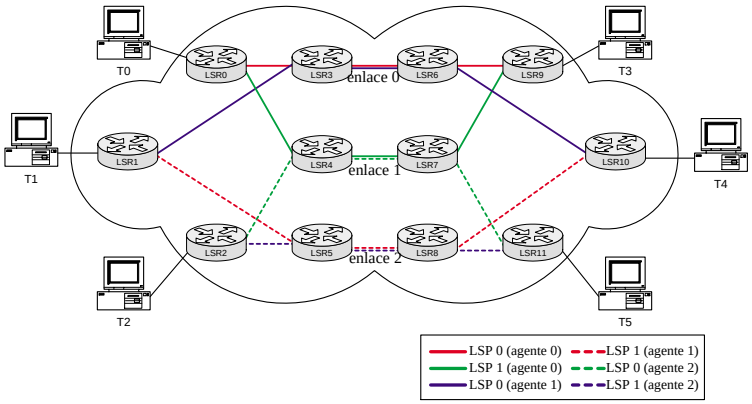


Figura 4: Topologia utilizada

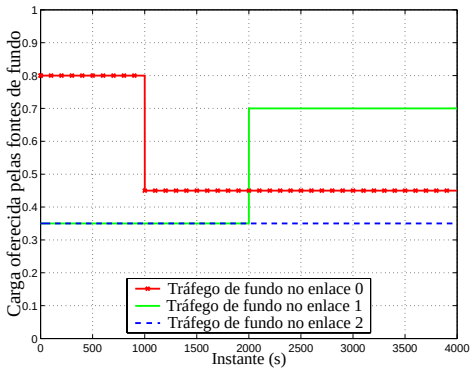


Figura 5: Tráfego de fundo utilizado

5 Resultados de simulação

Esta seção apresenta os resultados de simulações usando o simulador de redes NS (*Network Simulator*), versão 2.1b6, para os algoritmos discutidos anteriormente. Uma descrição mais detalhada sobre as simulações pode ser encontrada em Lima (2002).

A topologia simulada tem uma estrutura simétrica, com três roteadores de ingresso ligados a três roteadores de egresso, cada um dos pares executando o algoritmo de balanceamento de carga. Essa topologia foi apresentada em Elwalid *et al.* (2001) e visa estudar o comportamento dos algoritmos ao executarem balanceamento de carga em paralelo, ou seja, no caso em que agentes de TE possuem enlaces comuns. A Figura 4 ilustra a topologia utilizada. Em Elwalid *et al.* (2001), não são definidas as capacidades dos enlaces, nem as características das fontes utilizadas. No contexto desse trabalho, foram feitas as seguintes considerações: as fontes de tráfego principais são exponenciais, isto é, os intervalos de tempo entre transmissões de pacotes são considerados variáveis aleatórias exponencialmente distribuídas; todos os enlaces da rede têm a mesma capacidade (8,64 Mb); e todos os pacotes presentes nas simulações têm o mesmo tamanho (512 bytes). As fontes exponenciais foram utilizadas para permitir a comparação com os resultados apresentados em Elwalid *et al.* (2001). Resultados para diferentes topologias, distribuições de tráfego (por exemplo, Pareto) e tamanho de pacotes podem ser encontradas em Cavalcante (2001). Para esse conjunto de simulações, foram considerados 40 feixes de tráfego no total.

Ao todo, serão apresentadas duas simulações: uma com nível de tráfego correspondendo a

100% da capacidade dos enlaces 0, 1 e 2 e a outra com nível de tráfego correspondendo a 85% da capacidade dos enlaces 0, 1 e 2. Os resultados apresentados em Elwalid *et al.* (2001) são semelhantes aos do primeiro caso aqui estudado. Foi utilizada uma relação entre o número de pacotes de prova e o número de pacotes da fonte principal por volta de 0,05. Como referência, foram utilizados resultados para uma distribuição equitativa dos feixes de tráfego.

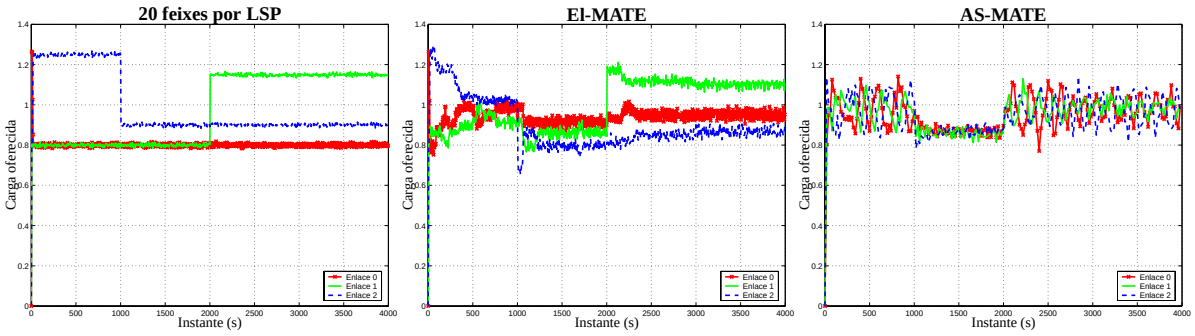
As fontes de tráfego de fundo estão conectadas aos enlaces 0, 1 e 2 (Figura 4). Um gráfico das fontes de fundo utilizadas encontra-se na Figura 5. Essas fontes são CBR. Há, neste exemplo, três fontes principais: uma de T0 a T3, uma de T1 a T4 e outra de T2 a T5 e cada nó de ingresso deve distribuir o tráfego entre dois caminhos (Figura 4), o que favorece o aparecimento de oscilações nas curvas de resultados dos algoritmos para essa topologia.

Para evitar uma resposta oscilatória do algoritmo El-MATE, segundo Elwalid *et al.* (2001), pode-se acrescentar um intervalo de tempo aleatório e pequeno (não é especificado o quanto), entre as fases de monitoramento da rede e de balanceamento de carga. Nas simulações aqui apresentadas, esse tempo foi considerado uma variável aleatória exponencialmente distribuída, com média 1,0 s. Nenhum mecanismo para evitar oscilações foi utilizado para o AS-MATE.

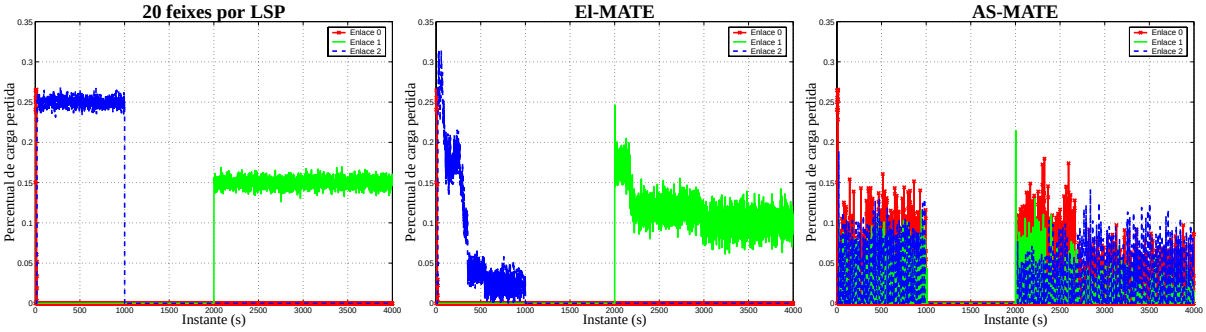
Os resultados obtidos para o primeiro caso estudado (maior carga) podem ser encontrados na Figura 6. Esses gráficos mostram os resultados de carga oferecida para os enlaces de possível congestionamento, que são os enlaces 0, 1 e 2 (Figura 4). Por essa figura, pode-se perceber que o algoritmo El-MATE, teve uma boa resposta até o instante $t = 1000,0$ s, mas não conseguiu chegar a níveis semelhantes de utilização após as modificações das fontes de fundo que ocorrem em $t = 1000,0$ s e em $t = 2000,0$ s. Os resultados de atraso médio por pacote para cada algoritmo, podem ser encontrados na Figura 6(c). Esses resultados não são apresentados em Elwalid *et al.* (2001).

Em relação ao algoritmo AS-MATE, para esse exemplo, chegou-se a resultados extremamente oscilatórios. Nos gráficos apresentados na Figura 6 foi incluído apenas um ponto dos dados a cada 20 obtidos, para que se pudesse visualizar melhor o resultado. Como conseqüência desse processo de retirada de pontos, as oscilações dos gráficos ficaram com a amplitude reduzida, a menos do gráfico de carga oferecida. Para o El-MATE foram retirados 4 em cada 5 pontos. Para esse exemplo, o algoritmo El-MATE não conseguiu diminuir a taxa de perda de pacotes, mantendo um nível médio por volta de 10%. O AS-MATE, manteve seu comportamento oscilatório, mas sempre atuando. Em relação ao atraso, pode-se perceber que o algoritmo El-MATE não conseguiu diminuir o atraso por pacote para os agentes 0 e 2, e o algoritmo AS-MATE teve o comportamento oscilatório, mencionado anteriormente.

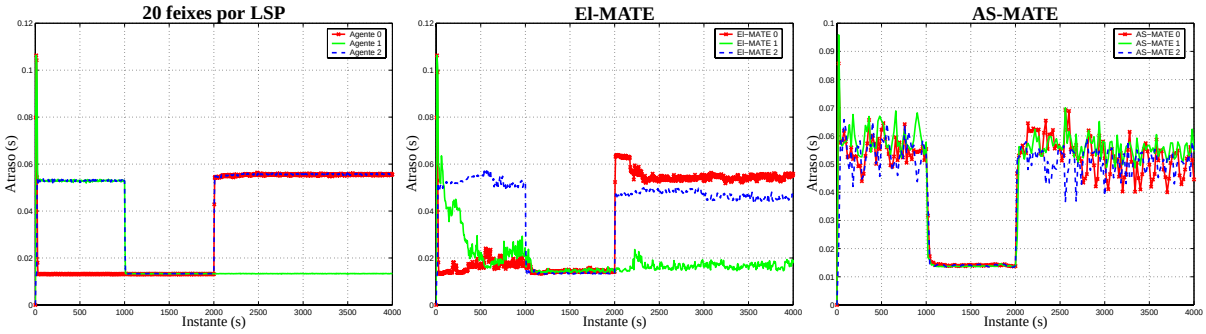
Os resultados para o segundo exemplo em estudo, quando a carga na rede é diminuída, estão ilustrados na Figura 7. Analisando os diversos gráficos, pode-se perceber que o AS-MATE não apresentou oscilações e obteve respostas rápidas para a queda da taxa de utilização dos enlaces de possível congestionamento. Nesse caso, o algoritmo El-MATE obteve resultados piores praticamente não percebendo a movimentação das fontes de fundo após $t = 2000,0$ s. Por isso, o algoritmo não conseguiu diminuir a taxa de perda de pacotes que ficou por volta de 20% para o enlace 1.



(a) Carga oferecida em relação à capacidade do enlace

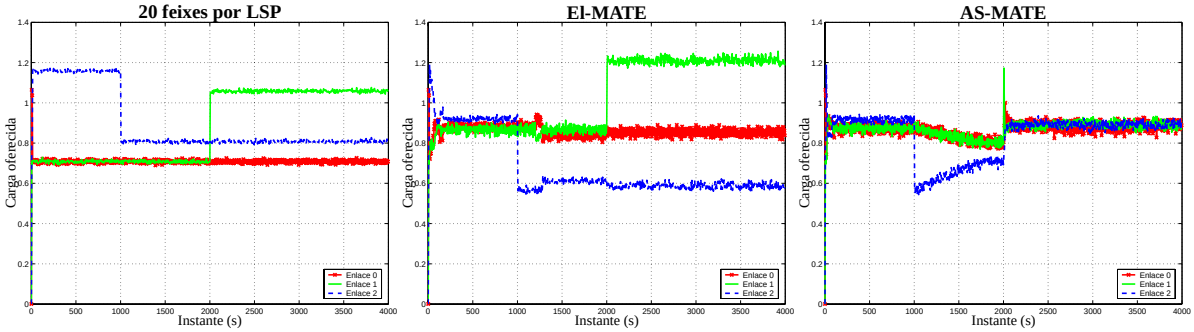


(b) Taxa de pacotes perdidos em relação à capacidade do enlace

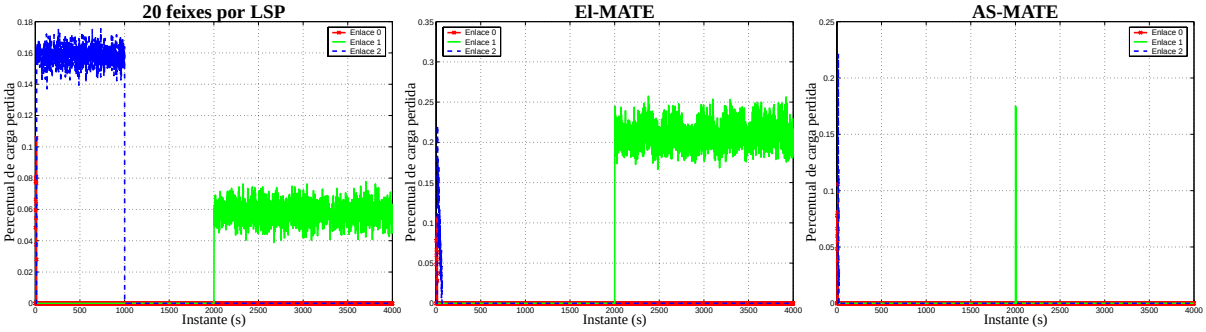


(c) Atraso médio por pacote

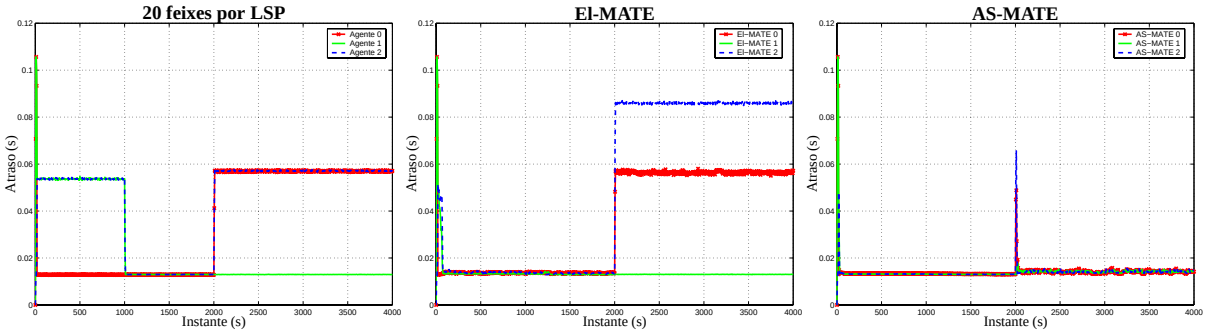
Figura 6: Exemplo 1: carga crítica



(a) Carga oferecida em relação à capacidade do enlace



(b) Taxa de pacotes perdidos em relação à capacidade do enlace



(c) Atraso médio por pacote

Figura 7: Exemplo 2: carga mais baixa

6 Conclusões

Neste trabalho foi proposto um algoritmo para balanceamento de carga entre LSPs configurados entre nós de ingresso e de egresso em uma nuvem MPLS. A arquitetura de engenharia de tráfego apresentada foi dividida em três níveis. Os algoritmos apresentados neste trabalho se adequam ao terceiro nível dessa arquitetura.

Os resultados obtidos mostraram que os algoritmos conseguem diminuir o atraso médio de pacotes e a taxa de perda de pacotes associada, em relação ao caso em que o mesmo número de feixes é mantido ao longo de cada LSP. Quanto à implementação, o algoritmo El-MATE mostrou-se de difícil configuração. Por outro lado, o AS-MATE mostrou um comportamento oscilatório, para uma situação de carga muito elevada na rede.

Em relação às oscilações apresentadas pelo algoritmo AS-MATE, algumas estratégias podem ser adotadas. Por exemplo, pode haver algum tipo de coordenação entre as atuações dos algoritmos, conforme sugestão apresentada em Elwalid *et al.* (2001). Outra forma é definir alguma função do número de feixes que auxilie o algoritmo a detectar um comportamento repetitivo diminuindo o valor do parâmetro η, que determina o número de feixes desviados a cada iteração. Dessa forma, o algoritmo diminuiria a amplitude da oscilação, até um ponto de convergência do roteamento.

Os algoritmos apresentados neste trabalho utilizam o Método da Secante para o cálculo do gradiente da função de custo. Outros técnicas para cálculo de gradientes poderiam ser utilizadas. Uma forma alternativa seria usar o método de Análise de Perturbação, conforme Cassandras *et al.* (1990) e Cavalcante (2001). Para compor uma ferramenta de TE completa, algoritmos para o primeiro e o segundo níveis da arquitetura de TE apresentada podem ser propostos.

Referências

AWDUCHE, D., MALCOLM, J., AGOGBUA, J., O'DELL, M., MCMANUS, J. *Requirements for traffic engineering over MPLS.* RFC 2702, Sept. 1999.

AWDUCHE, D., REKHTER, Y. Multiprotocol lambda switching: *Combining MPLS Traffic Engineering Control with Optical Crossconnets*. *IEEE Communications Magazine*, p. 111–116, Mar. 2001.

AWDUCHE, D. O. MPLS and traffic engineering in IP networks. *IEEE Communications Magazine*, p. 42–47, Dec. 1999.

AWDUCHE, D. O., CHIU, A., ELWALID, A., WIDJAJA, I., XIAO, X. *A framework for Internet traffic engineering.* (draft-ietf-tewg-framework-04.txt), Apr. 2001.

BANERJEE, A., DRAKE, J., LANG, J. P., TURNER, B., KOMPELLA, K., YAKOV, R. Generalized multiprotocol label switching: An overview of routing and management enhancements. *IEEE Communications Magazine*, p. 144–150, Jan. 2001.

BERTSEKAS, D. P., GAFNI, E. M., GALLAGER, R. G. Second derivative algorithms for minimum delay distributed routing in networks. *IEEE Transactions on Communication*, v. COM-32, nº 8, p. 73–85, Aug. 1984.

CASSANDRAS, C. G., ABIDI, M. V., TOWSLEY, D. Distributed routing with on-line marginal delay estimation. *IEEE Transactions on Communications*, v. 38, nº 3, p. 348–359, Mar. 1990.

CASSANDRAS, C. G., LAFORTUNE, S. *Introduction to discrete event systems*. Kluwer Academic Publishers. 1999.

CAVALCANTE, M. D. *Algoritmos de balanceamento de carga para tráfego tipo melhor esforço em redes IP/MPLS*. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Computação e Automação da Universidade Estadual de Campinas. Campinas-SP, Brasil : UNICAMP, 2001.

ELWALID, A., JIN, C., LOW, S., WIDJAJA, I. MATE: MPLS adaptive traffic engineering. In: INFOCOM, Anchorage, Alaska : IEEE, TCCC, 2001.

GALLAGER, R. G. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communication*, v. COM-25, nº 1, p. 73–85, Jan. 1977.

GIRISH, M. K., ZHOU, B., HU, J.-Q. Formulation of the traffic engineering problems in MPLS based IP networks. In: Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC'00), Antibes-Juan les Pins, France : IEEE, Communications Society and Computer Society, 2000.

JÜTTNER, A., SZVIATOVSZKI, B., SZENTESI, R., ORINCSAY, D., HARMATOS, J. On-demand optimization of label switched paths in MPLS networks. In: IEEE ICCCN, 2000.

KODIALAM, M., LAKSHMAN, T. V. Minimum interference routing with applications to MPLS traffic engineering. In: Proceedings of IEEE, Infocom, 2000.

LIMA, T. M. R. *Plataforma para simulação de algoritmos de balanceamento de carga em redes MPLS*. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Computação e Automação da Universidade Estadual de Campinas. Campinas-SP, Brasil : UNICAMP, 2002 (em andamento).

LIU, Z., SUN, Y., XUE, X. A static routing algorithm used in the Internet traffic engineering. *IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS)*, p. 231–234, Dec. 2000.

SURI, S., WALDVOGEL, M., WARKHEDE, P. R. *Profile-based routing:* a new framework for MPLS traffic engineering. 2000.

WIDJAJA, I., ELWALID, A. *MATE: MPLS adaptive traffic engineering*. (draft-widjaja-mpls-mate-00.txt), Aug.. 2000.

XIAO, X., HANNAN, A., BAILEY, B., CARTER, S., NI, L. M. Traffic engineering with MPLS in the internet. *IEEE Network magazine*, p. 28–33, Mar. 2000.

XIAO, X., NI, L. Internet QoS: A big picture. *IEEE Network Magazine*, p. 8–18, Mar./Apr. 1999.