FuDyLBA: Um Esquema de Engenharia de Tráfego para Balanceamento de Carga em Redes MPLS Baseado em Lógica Difusa*

Joaquim Celestino Júnior[1], Pablo Rocha Ximenes Ponte[1], Antonio Clécio Fontelles Tomaz[1], Ana Luíza Bessa de Paula Barros Diniz[1]

[1]Laboratório de Redes de Comunicação e Segurança da Informação (LARCES)
Universidade Estadual do Ceará (UECE)
Av. Parajana, 1700 – Itaperi – 60.720-020 – Fortaleza – CE – Brasil
{celestino,pablo,clecio,analuiza}@larces.uece.br

Abstract. *This paper describes a new traffic engineering scheme based on a load balance reactive method for congestion control in MPLS networks that makes use of local search and fuzzy logic techniques. It is an algorithm based on the "First-Improve Dynamic Load Balance Algorithm" (FID) and it works with linguistic values while performing load balance decisions. Computer simulation experiments have shown a better traffic distribution over the links of a network if compared to the behavior of the original version of the algorithm under the same circumstances.*

Resumo. *Este artigo descreve um novo esquema de engenharia de tráfego baseado em um método reativo de balanceamento de carga para controle de congestionamento em redes MPLS utilizando técnicas de busca local e lógica difusa. Trata-se de um algoritmo baseado no "First-Improve Dynamic Load Balance Algorithm" (FID) que utiliza grandezas lingüísticas para o cálculo das decisões de balanceamento. Experimentos obtidos por simulações computacionais apontam que nosso método é capaz de uma distribuição mais equânime do tráfego pelos enlaces de uma rede quando o comparamos com o comportamento da versão original do algoritmo, sob as mesmas circunstâncias.*

1. lntrodução

Uma das aplicações mais interessantes do *Multi-Protocol Label Switching* (MPLS) para redes baseadas em IP é a Engenharia de Tráfego, ou *Traffic Engineering* (TE) [Awduche et al. 2002]. O Principal objetivo da TE é otimizar o desempenho de uma rede através da utilização eficiente de seus recursos. No contexto do MPLS isto é operacionalizado através dos *Label Switched Paths* (LSP) que se tratam de caminhos estabelecidos através dos enlaces de uma rede onde rótulos são utilizados como parâmetros para as decisões de comutação. A característica mais importante dos LSPs para a TE é a possibilidade do estabelecimento de rotas explícitas através de uma rede,

** Trabalho parcialmente financiado pela Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) e pela HP do Brasil.*

propiciando o controle efetivo do direcionamento dos diversos fluxos componentes do tráfego de uma rede.

A principal abordagem utilizada pelo *framework* MPLS em termos de engenharia de tráfego é o chamado roteamento baseado em restrições, ou *Constraint Based Routing* (CBR). Neste artigo, é apresentado um esquema alternativo de engenharia de tráfego baseado em lógica difusa inspirado no *First-Improve DyLBA* (FID), um algoritmo de balanceamento de carga proposto em [Salvadori et al. 2002]. O FID é um algoritmo baseado em busca local onde a principal idéia é re-rotear, de forma eficiente, os LSPs dos enlaces mais congestionados na rede de forma que o tráfego seja balanceado entre os diversos enlaces, permitindo um melhor uso de recursos. Enquanto que o CBR é baseado em um mecanismo preventivo, o FID age apenas quando o congestionamento é detectado na rede, sendo assim baseado em um mecanismo reativo. O congestionamento, em uma rede MPLS, pode ser detectado basicamente de duas formas: ou a capacidade de um determinado link está próxima de sua capacidade total, ou uma nova requisição por um LSP não pôde ser executada por falta de recursos. O FID trabalha na primeira hipótese, monitorando a intensidade de utilização de cada enlace da rede. Porém, o principal problema encontrado no FID é o fato de não levar em conta a intensidade do congestionamento em cada enlace para definir quantos LSPs irá re-rotear, o que faz, em certos casos, que várias iterações sejam necessárias até o efetivo descongestionamento do enlace. Nossa proposta adiciona um controlador fuzzy ao algoritmo, onde a entrada é mensurada a partir da taxa de utilização de cada enlace e a saída será utilizada como uma medida de intensidade com a qual o algoritmo deve diminuir a carga do enlace em questão.

Este trabalho é organizado da seguinte forma: na seção 2 é feita uma breve discussão sobre as técnicas de engenharia de tráfego em redes MPLS; na seção 3 o contexto de nosso modelo, bem como suas motivações, é definido; na seção 4 apresentamos o algoritmo e algumas considerações a seu respeito; na seção 5 mostramos os procedimentos utilizados nos experimento, bem como os resultados obtidos; e na seção 6 fazemos uma breve conclusão do trabalho.

2. Engenharia de Tráfego em Redes MPLS

Os provedores de serviços de rede lutam diariamente para minimizar o congestionamento em seus sistemas. Reduzir o congestionamento, para redes baseadas em comutação de pacotes, é, também, diminuir retardos de transmissão, ou *delays*, o que cria as condições básicas para uma melhor garantia de qualidade de serviços (QoS) além de diminuir a carga de tráfego nos roteadores. Já em redes baseadas em comutação de circuitos, reduzir o congestionamento significa aumentar a largura de banda disponível em cada enlace, de forma que futuras requisições por conexões possam ser aceitas [Salvadori et al. 2002].

De acordo com a RFC 3272, *"Overview and Principles of Internet Traffic Engineering"*, os mecanismos de gerenciamento de congestionamento para Engenharia de Tráfego podem ser caracterizados, de acordo com a sua política, pelos seguintes critérios: pelo tempo de resposta, pelo caráter reativo ou preventivo; e pelo fato de ser baseado na geração de fluxo ou em sua demanda.

A maioria das propostas de mecanismos de gerenciamento de TE são preventivas, ou seja, acomodam caminhos pela rede de forma a prevenir situações de congestionamento. Os dois mecanismos mais mencionados na literatura são o *Constraint Based Routing* (CBR) e o *Traffic Splitting* (partição de tráfego) [Salvadori et al. 2002]. O CBR teve suas origens no velho problema de roteamento para Qualidade de Serviço em redes baseadas em IP e diz respeito à alocação de LSPs baseando-se em diversos tipos de restrições como largura de banda disponível, atraso máximo, etc. O mecanismo de *Traffic Splitting* funciona distribuindo porções de um fluxo, ou particionando o tráfego por LSPs paralelos entre o par de nós ingresso-egresso.

Um dos esquemas de CBR mais mencionados é o chamado MIRA *(Minimum Interference Routing Algorithm)*, ou Algoritmo de Roteamento de Interferência Mínima [Kar et al. 2000]. Ele é baseado em um algoritmo *online* de escolha heurística dinâmica de caminho. A idéia principal é tirar vantagem do conhecimento prévio do par de nós ingresso-egresso para que seja evitado o estabelecimento de rotas que passem por enlaces que possam interferir com futuras novas alocações de caminhos. Tais enlaces críticos são identificados pelo MIRA como sendo aqueles que, se ficarem altamente utilizados, poderão impedir que futuras requisições por conexão sejam satisfeitas entre os mesmos nós ingresso-egresso. O principal ponto fraco deste mecanismo é a complexidade computacional causada pelo cálculo do fluxo máximo entre os nós comunicantes, requisito para a identificação dos enlaces críticos, e a utilização desbalanceada da rede. Como demonstrado em [Wang et. al. 2002], o MIRA não é capaz de estimar gargalos em enlaces que sejam críticos para clusters de nós de rede. Além disso, ele não leva em conta a carga de tráfego atual em suas decisões de roteamento. Imaginemos um cenário onde um par origem-destino é conectado por duas ou mais rotas, cada uma com a mesma largura de banda residual. Quando uma nova requisição por um LSP for gerada, uma destas rotas será escolhida para satisfazer a requisição. Após este procedimento, todos os enlaces pertencentes às rotas que não foram escolhidas tornam-se críticos, de acordo com a definição supramencionada. Isso significa que todas as futuras requisições de novos LSPs entre o mesmo par ingresso-egresso serão roteadas pelo mesmo caminho, enquanto as outras rotas continuam livres de carga, o que causa um uso desbalanceado dos recursos da rede. Ainda pior, em cenários onde LSPs são alocados e desalocados dinamicamente, este esquema pode causar a criação de rotas por caminhos ineficientes e bloqueios futuros para certas rotas. Essa gama de problemas é comum a todos os mecanismos CBR propostos na literatura e é relacionada diretamente com seus comportamentos implicitamente preventivos [Salvadori et al. 2002]. Dos poucos modelos de controle de congestionamento (gerenciamento reativo) para redes MPLS propostos na literatura temos o *Fast Acting Traffic Engineering* (FATE) proposto em [Holness e Phillips 2000], que estende o *Constraint Routing Label Distribution Protocol* (CR-LDP), um protocolo de sinalização baseado em CBR para distribuição de rótulos em redes MPLS. O FATE incrementa o CR-LDP da seguinte forma: quando determinados LSPs experimentam perda de pacotes, eles são re-roteados para caminhos menos congestionados, remediando, assim, a condição de congestionamento. Outro esquema reativo encontrado na literatura foi proposto por [Jüttner et al. 2000] que é baseado no re-roteamento otimizado de um LSP já estabelecido quando um novo LSP não pôde ser alocado na rede por falta de recursos. A idéia baseia-se no fato de que em altos níveis de

utilização a alocação sob demanda de LSPs baseados em CBR pode sofrer falhas. A otimização na escolha da nova rota para o LSP que mudará seu caminho utiliza técnicas de Programação Linear Inteira em uma adaptação do protocolo *Constrained Shortest Path First* (CSPF). Mais uma vez, trata-se de uma solução acessória que serve de coadjuvante às técnicas de CBR. Por último, o FID é um algoritmo baseado em uma busca local otimizada onde, para cada enlace considerado congestionado em uma determinada rede, é procurado o primeiro LSP que, se re-roteado, é capaz de melhorar a capacidade do enlace em questão sem prejudicar a capacidade geral da rede. O FID é um aprimoramento do *Dynamic Load Balance Algorithm* (DyLBA) [Battiti et al. 2002], onde a principal distinção está na profundidade da busca, que no caso do DyLBA só encerra quando acha o melhor LSP a ser re-roteado, ao invés de parar na primeira opção viável.

Não fomos capazes de encontrar nenhuma proposta de mecanismo com técnicas de lógica *fuzzy* para o balanceamento de carga em redes MPLS, o que nos fez comparar nosso modelo apenas com o FID original.

3. Motivações e Definição do Problema

Não escolhemos um modelo preventivo de gerência de congestionamento como base para nosso esquema devido às deficiências já demonstradas que se relacionam, essencialmente, com o caráter passivo que é dado na abordagem de atuação. A escolha do FID se deu, principalmente, pelo fato de este ser essencialmente reativo, de forma distinta das outras propostas mencionadas, que funcionam apenas como anexos que suavizam os problemas essenciais dos modelos preventivos. Outro fator que nos impulsionou a escolhê-lo foi sua técnica de busca local rápida baseada em enlace, o que nos possibilitou desenvolver um controlador difuso também baseado em enlace de reposta mais rápida. Ademais, um grande problema dos atuais algoritmos reativos é a sua forma de mensurar a rede, baseando-se em um limite, onde apenas acima ou abaixo dele o algoritmo é disparado. Em especial no caso do FID, onde apenas um LSP é re-roteado por iteração, isso pode significar que várias iterações do algoritmo por completo poderão ser executadas até o efetivo balanceamento da rede. Então, torna-se essencial a incorporação de variáveis lingüísticas aos algoritmos reativos de balanceamento de carga, possibilitando uma resposta mais adequada do algoritmo a cada situação medida. Isso faz nossa proposta deveras conveniente.

Uma especificação do problema, semelhante ao mencionado em [Salvadori et al. 2002], foi definida. A rede considerada consiste de n roteadores. Um subconjunto de roteadores que formem o par ingresso-egresso entre os quais conexões podem ser potencialmente estabelecidas é definido. Cada requisição de conexão é feita ao roteador de ingresso ou a central de gerenciamento da rede (no caso de gerência centralizada) o qual determina a rota explícita para o LSP de acordo com a topologia atual e com os recursos disponíveis da rede. Para realizar o algoritmo de balanceamento é necessário que cada roteador na rede (ou a central de gerência) saiba a topologia atual da rede e a largura de banda residual em cada enlace. Para tanto, cada roteador na rede MPLS precisa ter suporte a algum protocolo de roteamento baseado em estado do link com suporte a notificações de largura de banda residual por enlace (*residual bandwidth advertisements*). A largura de banda residual para um determinado enlace de rede

deverá ser calculada pela diferença entre a sua capacidade total e a soma do consumo de largura de banda de todos os LSPs que passem por ele.

Nosso objetivo é aumentar a largura de banda residual de cada enlace na rede a fim de distribuir a carga da rede o mais eqüitativamente possível por todos os enlaces, de forma que haja o mínimo possível de enlaces ociosos, ou de enlaces sobrecarregados.

4. Um Algoritmo Fuzzificado de Engenharia de Tráfego para Balanceamento de Carga

A meta principal de nosso algoritmo é balancear o tráfego de uma rede MPLS dinamicamente. Se reduzirmos este problema ao aumento da largura de banda residual de cada enlace da rede, teremos um objetivo secundário que é maximizar a banda residual de cada enlace de rede, mas com a restrição de que a operação a ser realizada para a maximização local não comprometa a qualidade global do balanceamento.

4.1. O Algoritmo FID

Como nosso trabalho baseou-se essencialmente no algoritmo FID, é importante que nos detenhamos brevemente para explicá-lo um pouco mais a fundo.

O FID concentra sua atuação no enlace. Primeiramente, ele cria uma lista de todos os enlaces que são considerados congestionados, de acordo com um parâmetro de entrada que representa o mínimo de largura de banda residual que um enlace deve ter para não ser selecionado, da forma FID(X), onde X é este parâmetro. Em seguida, ele realiza uma busca local em cada enlace da lista de enlaces congestionados, tentando encontrar um LSP a ser re-roteado, a fim de livrar o enlace da carga deste LSP. Para encontrar este LSP, o FID examina todos os LSPs que passem pelo enlace em questão. Para cada LSP examinado, são feitos alguns procedimentos. Primeiro, a contribuição do LSP, em termos de consumo de banda, é retirada da rede. É calculado, então, um caminho alternativo para o LSP, que não percorra o enlace a ser examinado. O novo caminho terá sua largura de banda residual medida e comparada com a largura de banda residual do caminho atual do LSP. Se for maior, o LSP é re-roteado para o novo caminho e a busca para, iniciando-se o exame do enlace seguinte. Se for menor, o próximo LSP será examinado. Em qualquer um dos casos, a iteração termina restaurando a contribuição de consumo de banda do LSP à rede. Caso todos os LSPs do enlace sejam examinados sem sucesso, o próximo enlace é, então, examinado.

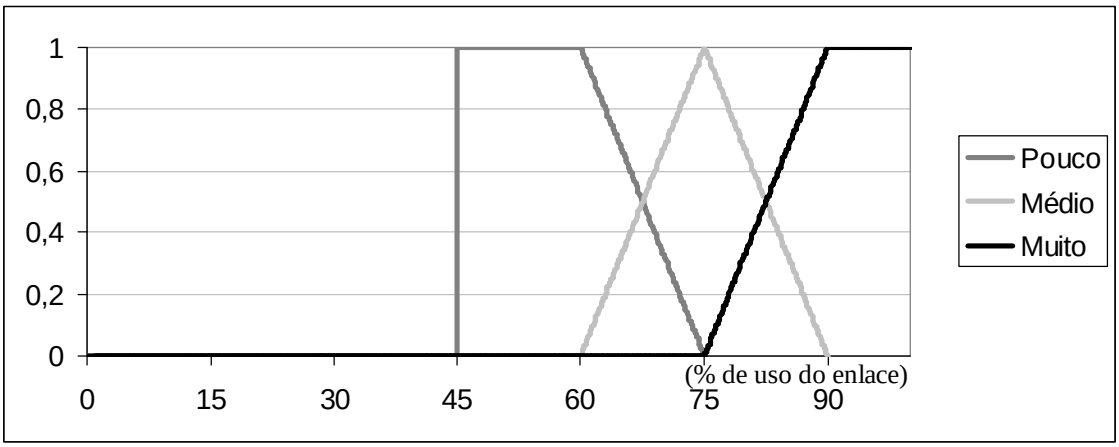

Figura 4.1. Funções de pertinência de entrada utilizadas pelo controlador fuzzy na identificação lingüística do congestionamento de um enlace.

4.2. Visão Geral do FuDyLBA

Em nosso método, o núcleo do algoritmo FID foi modificado de forma que trabalhasse com níveis de intensidade, criando uma função de descarga de enlace. Essa função recebe como entrada uma intensidade que serve de parâmetro para o quanto um determinado enlace deverá ser liberado de carga. Tal função examina cada LSP que percorre o enlace, calculando um caminho alternativo para esse LSP que não passe pelo enlace em questão. Depois, compara o novo caminho com o caminho atual, sem a contribuição do LSP a ser examinado. Se o novo caminho oferecer uma largura de banda residual inferior a do caminho atual, o próximo LSP é examinado. Caso a largura de banda residual do novo caminho seja maior, o LSP é re-roteado e a busca local reinicia de volta ao primeiro LSP que percorre o enlace, diferente do mecanismo encontrado no FID, que para definitivamente a busca e passa a examinar um enlace seguinte. Nossa função apenas parará após ter atingido um percentual de descarga para aquele enlace, igual à intensidade requisitada, ou após ter chegado ao último LSP sem que nenhum outro fosse capaz de ser re-roteado.

Para calcular a intensidade adequada de descarga para cada enlace, foi projetado um controlador *fuzzy* que tem, como entrada, a intensidade de utilização do enlace e, como saída, a intensidade da função de descarga para aquele enlace.

Para compreender o nível de carga em cada enlace, foram definidos três valores lingüísticos de entrada: pouco congestionado, medianamente congestionado e muito congestionado. Cada um desses valores é definido por sua respectiva função de pertinência, como podemos ver na figura 4.1, nomeadas pertinências de entrada. Para definir lingüisticamente a intensidade de atuação da função de descarga, foram, também, definidas três funções não contínuas para as pertinências de saída como podemos ver na figura 4.2, das quais: descongestionar pouco, descongestionar medianamente e descongestionar muito.

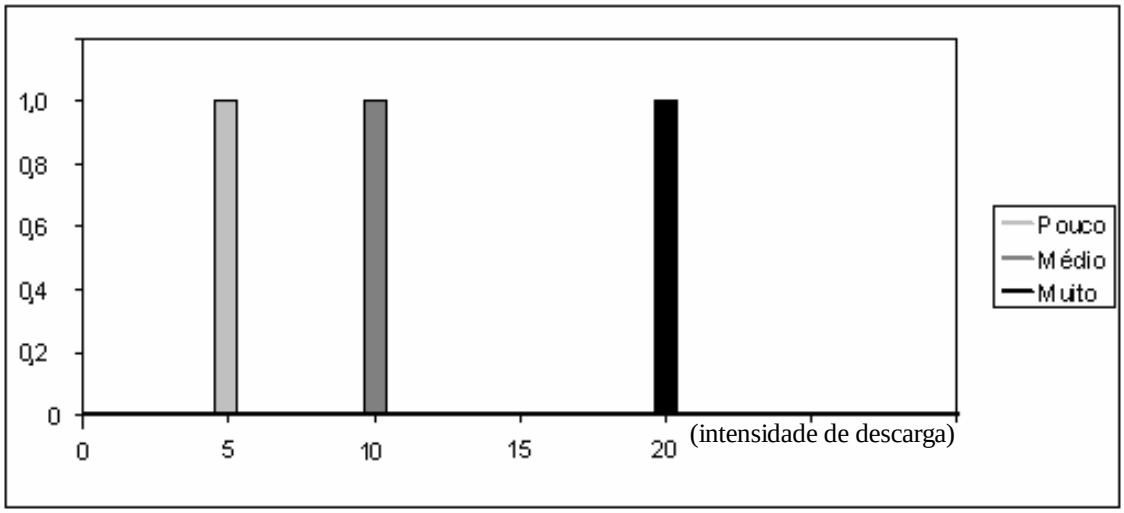

Figura 4.2. Funções de pertinência de saída utilizadas pelo controlador *fuzzy* na identificação lingüística da intensidade de descarga de um enlace

Cada enlace da rede é medido em relação à sua proporção de utilização de largura de banda. Quando uma determinada medida se enquadra em qualquer dos valores lingüísticos de entrada, o enlace é considerado congestionado e a função de descarga é executada recebendo com entrada a saída do processo de defuzzificação do controlador *fuzzy*, utilizando-se a técnica de centro de gravidade.

Intitulamos nosso algoritmo de FuDyLBA, pois trata-se de uma versão modificada por fuzzificação do algoritmo *First-Improve* DyLBA (FID), ou apenas, *Fuzzified DyLBA* (FuDyLBA).

4.3. O Pseudo-Código

O pseudocódigo de nosso algoritmo é demonstrado na figura 4.3. Definamos seus entes e significados. É possível se identificar 3 blocos, onde o primeiro demonstra a estrutura do algoritmo e o segundo e terceiro blocos constituem o procedimento fuzzificado de descarga de enlace, representado, no primeiro bloco, pela função *unload*. No primeiro bloco, temos o corpo geral do algoritmo. Neste bloco a variável *LinkSet* recebe o conjunto de enlaces da rede e, em seguida, é executado um laço que chama a função *unload* para cada elemento da variável *LinkSet*. Em cada iteração deste laço, o enlace a ser examinado é registrado na variável *link* que servirá de parâmetro para a função *unload*. O laço apenas terminará quando todos os enlaces forem examinados. A função *unload* é um misto do segundo e terceiro blocos. De fato, essa função trata-se, primeiramente, de um controlador *fuzzy*, ilustrado no segundo bloco, que, baseando-se na porcentagem de utilização do enlace, calcula a intensidade de descarga adequada, registrando-a na variável *level*. Em seguida, a função de descarga é executada para este enlace, tendo como entradas as variáveis *level* (intensidade de descarga) e *link* (enlace a ser descarregado). No terceiro bloco, encontramos o pseudocódigo da função de descarga. Nesta função, primeiramente, iniciamos a variável *change* com valor zero. Ela servirá para guardar a variação de carga do enlace (definido pela variável *link*), resultante

de cada iteração do laço seguinte. A partir daí, o primeiro laço é iniciado e só terminará quando a variável *change* for maior ou igual ao valor da variável *level* (que guarda a intensidade com a qual o enlace deve ser descarregado). Dentro deste primeiro laço, começamos inicializando as variáveis *LSPFound*, com valor falso, e *LSPSet*, com o conjunto de LSPs que passam pelo enlace registrado em *link*,que está sendo examinado durante a iteração do primeiro laço. Em seguida, um novo laço é iniciado. Este segundo laço terminará apenas quando a variável *LSPFound* tiver valor verdadeiro ou quando não houver nenhum outro elemento na variável *LSPSet* a ser examinado. Dentro dele, inicialmente, tomamos um elemento da variável *LSPSet* e o registramos na variável *LSPi*. Este será o LSP a ser examinado durante a iteração. Em seguida, retiramos a contribuição no consumo de banda, dada pelo LSP registrado em LSPi, da rede MPLS, através da função RemovePartialLoad. Sem essa contribuição podemos calcular a banda residual no caminho percorrido por LSPi e a guardamos em CurrResBw (banda residual do caminho atual). Através da função FindAlternativePath, é procurado um caminho alternativo para LSPi que não inclua o enlace guardado na variável link e o registramos em A_LSP. Se o caminho for achado, para ele será calculada, também, a quantidade de banda residual que registraremos em altResBw (banda residual do caminho alternativo). Em seguida, altResBw é comparada com o valor de CurrResBw. Caso seja maior, fazemos LSPFound receber o valor verdadeiro e incrementamos a variável change com a razão do consumo de banda do LSP definido por LSPi, pela capacidade total do enlace definido por link, ou seja, o quanto, em termos percentuais, foi liberado da capacidade do enlace naquela iteração. Continuando, restauramos a contribuição em termos de largura de banda dado pelo LSP definido por *LSPi*, à rede MPLS e, se *LSPFound* for igual a verdadeiro, re-roteamos *LSPi* para a nova rota definida por *A_LSP*.

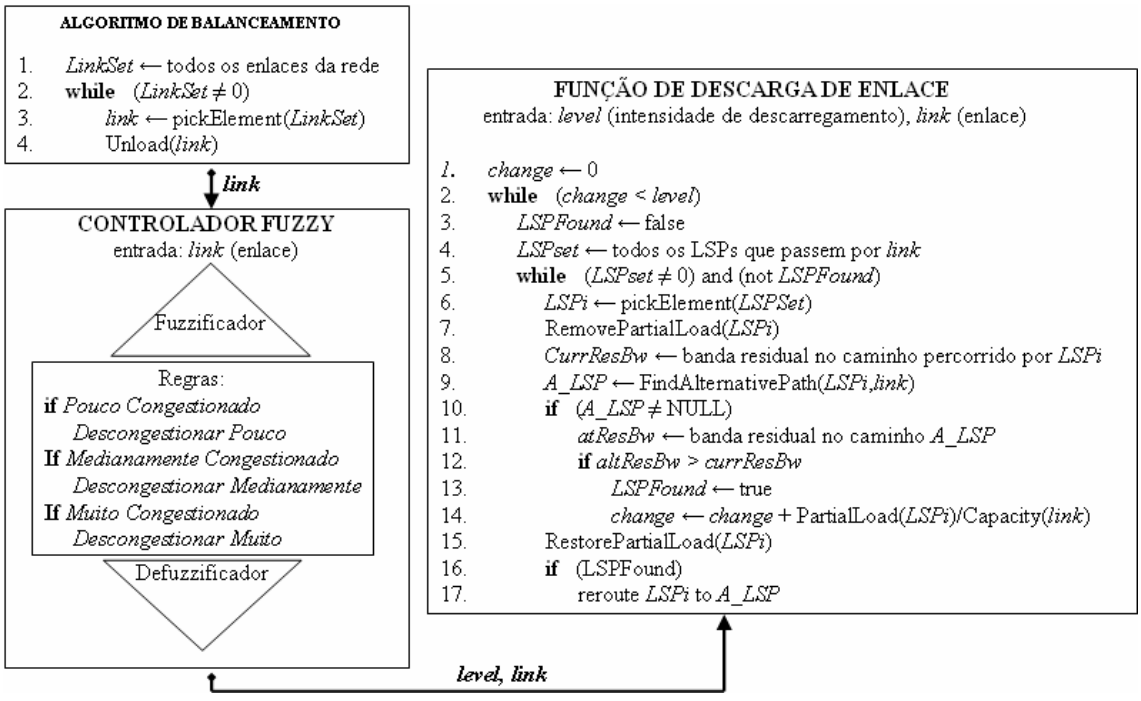

Figura 4.3. Pseudocódigo do algoritmo de balanceamento

4.4. Considerações sobre o Algoritmo

Certas observações se fazem necessárias a respeito do FuDyLBA. A granularidade dos LSPs, considerada como a relação do consumo de banda de cada LSP com as capacidades dos enlaces da rede, desempenha um papel fundamental na atuação de nosso algoritmo. Uma granularidade mais baixa faz com que o algoritmo comporte-se mais assemelhadamente com sua versão não fuzzificada. Isso fica claro, se analisarmos mais atentamente a função de descarga, que para de executar assim atinge a intensidade requisitada. Caso um LSP que passe pelo enlace que está sendo descarregado tenha um consumo de banda proporcionalmente grande à capacidade total do enlace, ou seja, baixa granularidade, re-roteá-lo significaria, possivelmente, atingir, ou até ultrapassar demasiadamente, a intensidade de descarga. Se isso for generalizado para todos os LSPs da rede, a função, no limite, irá re-rotear apenas um LSP por enlace, semelhante ao comportamento do algoritmo original.

Outra consideração diz respeito à complexidade computacional de nossa proposta comparada ao FID. Ela, sem dúvida, irá ser superior, devido a dois fatores: o custo computacional do controlador fuzzy e o número maior de iterações executadas por enlace, durante a busca local, pois não apenas um LSP é re-roteado, mas quantos forem necessários para a descarga do enlace na intensidade desejada. Essa diferença, porém, pode ser consideravelmente reduzida. Com relação ao controlador, poderíamos adotar uma abordagem pré-computada, muito comum em controladores *fuzzy*, que reduziria seu custo computacional a zero. Com relação ao número de iterações no procedimento de busca, ele tende a equiparar-se com o número de iterações na busca local do FID, de acordo com a granularidade dos LSPs: granularidades mais baixas causam um número de iterações mais próximo nos dois algoritmos. Esse comportamento justifica-se pelo mesmo motivo apresentado na observação anterior, quando comentamos os efeitos da granularidade. Com isso, somos levados à conclusão de que, se toda a funcionalidade adicional dada pelas técnicas de lógica *fuzzy* for perdida, devido a uma granularidade extremamente baixa, e o controlador *fuzzy* utilizar uma abordagem pré-computada, o FuDyLBA tende a possuir uma complexidade computacional idêntica à do FID.

Uma terceira observação pode ser feita a respeito das funções de pertinência de entrada e saída. Apesar de, até a presente data, termos trabalhado e estudado baseando-nos nas funções apresentadas neste artigo, isso não implica que tais funções sejam inerentes ao nosso modelo. De fato, elas são apenas um caso particular de nosso controlador fuzzy. Na verdade, nosso algoritmo trabalha com pertinências variáveis, de forma que suas futuras simulações e possíveis implementações terão como entrada, na inicialização, um parâmetro que servirá para a construção das funções de pertinência dinamicamente, do mesmo modo que o FID possui um parâmetro de entrada, na forma FID(X), onde X é o percentual de banda residual, onde, abaixo dele, um determinado enlace é considerado congestionado.

5. Simulações e Resultados

O objetivo das simulações foi, principalmente, mensurar a melhoria obtida após a inserção das técnicas de lógica fuzzy ao algoritmo reativo original, pois o FID já foi

extensivamente mensurado e comparado com outras técnicas de engenharia de tráfego em [Salvadori et al. 2002] de forma que sua funcionalidade já é bem conhecida.

5.1. Os Experimentos

Para realizar os experimentos, a topologia definida na figura 5.1 foi utilizada, onde as linhas finas tracejadas representam enlaces com capacidade de 1024 unidades, as linhas finas não tracejadas representam enlaces com capacidade de 2048 unidades e a linha grossa representa um enlace com capacidade de 3072 unidades. A requisições por LSPs estão limitadas apenas entre o par de LSRs 1 e 9, sempre originadas no LSR 1.

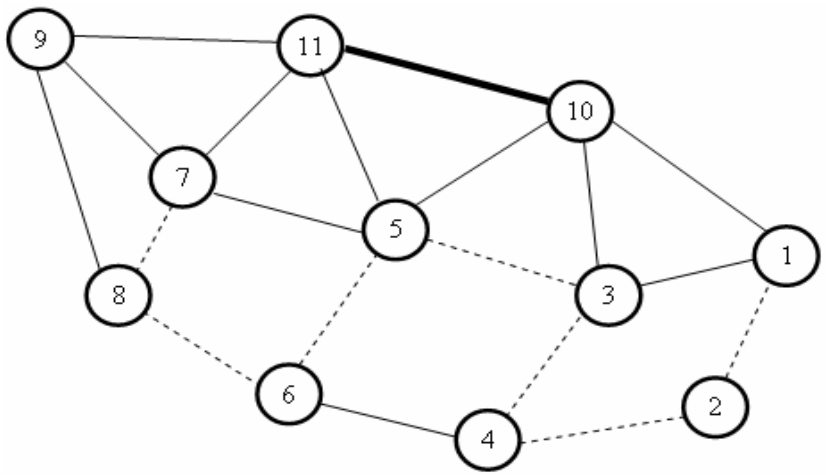

Figura 5.1 – Topologia de Rede Utilizada nos Experimentos

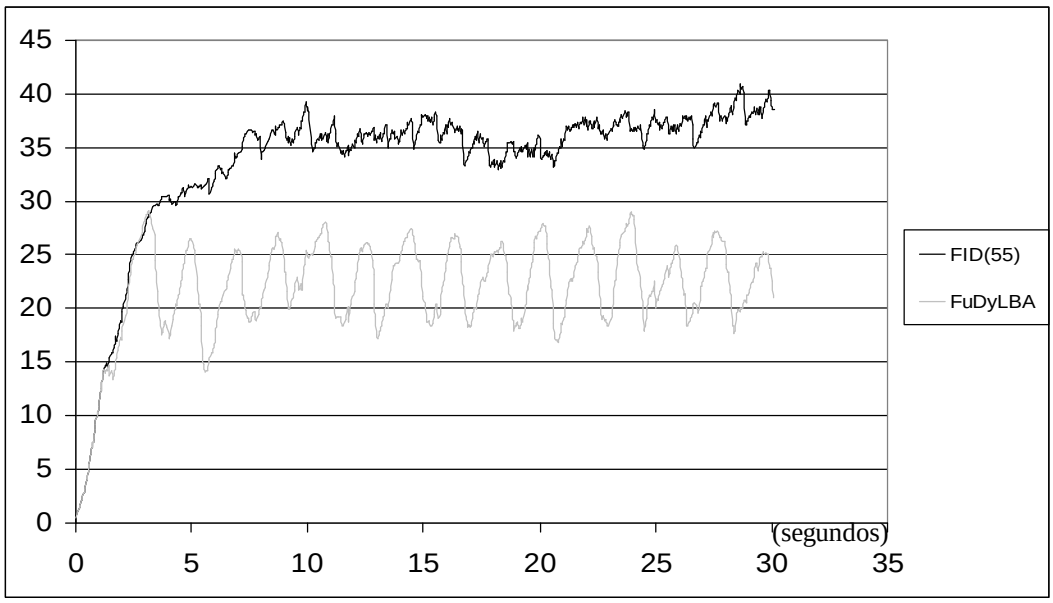

Figura 5.2 – Desvios Padrão das Porcentagens de utilização de cada enlace da rede, ao longo do tempo, para o primeiro experimento.

Foram realizados dois experimentos, ambos com 30 segundos de duração, reproduzindo exatamente as mesmas condições para os dois algoritmos (FID e FuDyLBA). Adotamos o FID(55) para as simulações para que nenhuma vantagem

estivesse presente, de forma que ambos FID e FuDyLBA disparassem seus mecanismos com a mesma sensibilidade. O primeiro experimento utilizou alta granularidade de LSPs na carga da rede, enquanto que o segundo utilizou uma granularidade mais baixa. Os LSPs variaram ciclicamente em consumo de banda, caminho e duração, ao passo que foram sendo introduzidos a cada 0,05 segundos. Foram introduzidos 600 LSPs para os dois experimentos. Em ambos os experimentos, os caminhos utilizados para a carga da rede, foram 1-3-10-11-9 e 1-10-11-9.

No primeiro experimento, o consumo de banda de cada LSP variou entre 8 e 32 unidades e sua respectiva duração variou entre 3 e 8 segundos.

No segundo experimento, o consumo de banda de cada LSP variou entre 32 e 128 unidades com as mesmas durações utilizadas para o primeiro experimento.

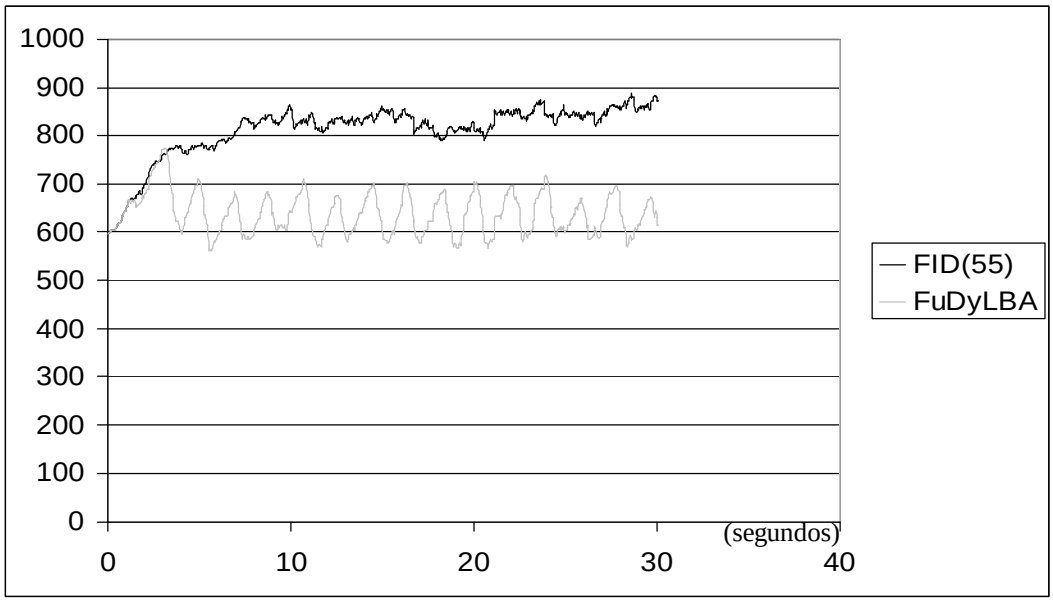

Figura 5.3 – Desvios Padrão das Larguras de Banda Residuais para cada enlace da rede, ao longo do tempo, para o primeiro experimento.

5.2. Resultados

A técnica que utilizamos para mensurar a qualidade de balanceamento para cada algoritmo foi o cálculo do desvio padrão, para cada instante de tempo, tanto para as porcentagens de utilização de cada enlace da rede, como para suas respectivas larguras de banda residuais. O desvio padrão é uma medida de dispersão que se adequada perfeitamente para quantificação da qualidade de balanceamento. Pelo desvio padrão, é mensurado o quão afastadas entre si estão as amostras, que em nosso caso são as percentagens de utilização dos enlaces e suas respectivas larguras de banda residuais. Um desvio padrão elevado significa, em última instância, uma pior distribuição do tráfego pelos enlaces da rede, o que significa um balanceamento de menor qualidade.

Na figura 5.2, podemos ver uma comparação das curvas de desvios padrão entre o FID(55) e FuDyLBA, para o primeiro experimento, em termos de porcentagem de

utilização do enlace. Na figura 5.3, temos o mesmo experimento, visto pela comparação dos desvios padrão das larguras de banda residuais. O segundo experimento é retratado pela figura 5.4, para as porcentagens de utilização dos enlaces, e pela figura 5.5 para as larguras de banda residuais.

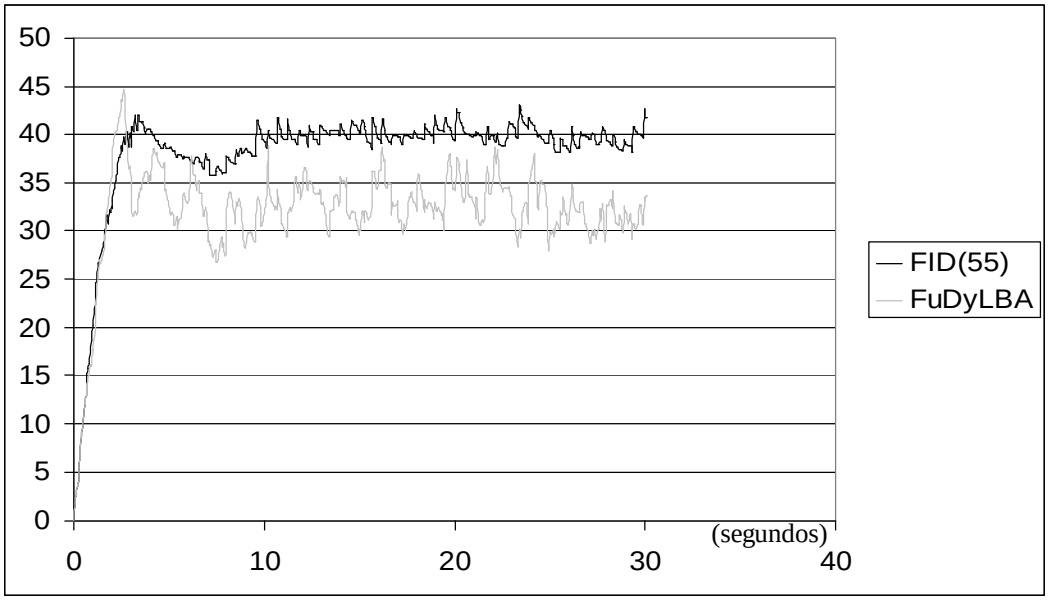

Figura 5.4 – Desvios Padrão das Porcentagens de utilização de cada enlace da rede, ao longo do tempo, para o segundo experimento.

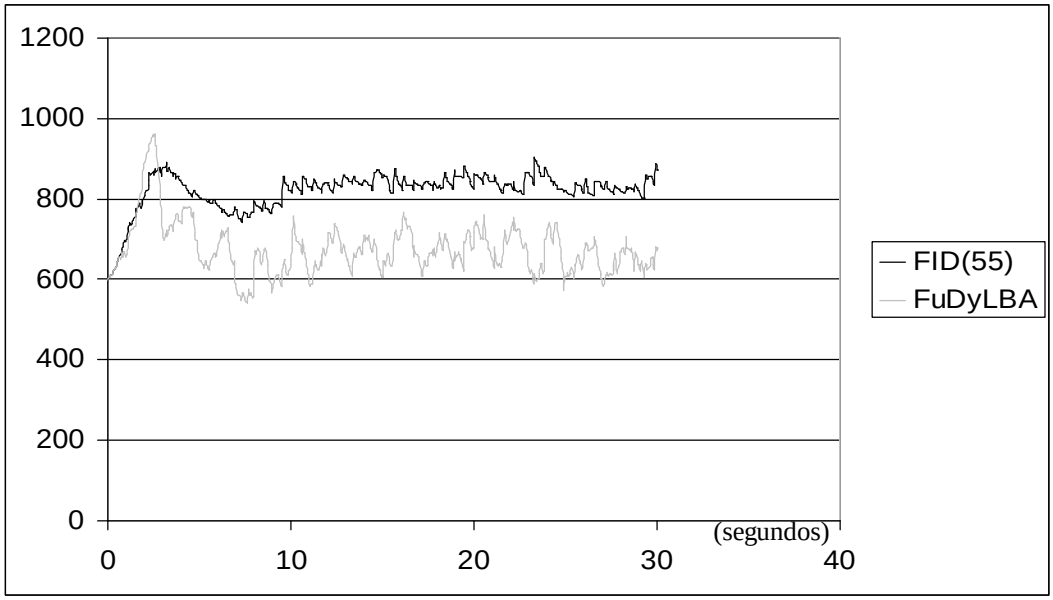

Figura 5.5 – Desvios Padrão das Larguras de Banda Residuais para cada enlace da rede, ao longo do tempo, para o segundo experimento.

Pelos gráficos apresentados, podemos observar que, no primeiro experimento, o FuDyLBA efetua um balanceamento bem melhor e mais rápido que o FID, além de

permanecer quase que pela totalidade do experimento com a rede bem melhor balanceada. O segundo experimento teve a intenção de demonstrar como o efeito da granularidade pode influenciar negativamente o FuDyLBA. Nele, é possível se notar uma maior aproximação das duas curvas, mostrando que em situações de menor granularidade, o FuDyLBA tende a comportar-se mais próximo do FID, o que corrobora nossa previsão teórica a respeito dos efeitos da granularidade em nosso algoritmo.

6. Conclusões

Neste artigo, demonstramos como a Lógica Difusa pode enriquecer as técnicas de engenharia de tráfego, especialmente no controle de congestionamento em algoritmos de balanceamento de carga reativos. Foi mostrado, também, o FuDyLBA, um algoritmo reativo de balanceamento de carga que demonstrou possuir desempenho superior ao de seu equivalente não fuzzificado, o FID, no tocante a balanceamento de carga em redes MPLS. Foi demonstrado, também, que características atenuantes dos efeitos das técnicas fuzzy, como a granularidade de LSPs na rede, fazem, na pior hipótese, que o FuDyLBA comporte-se como sua versão não fuzzificada, mas, em compensação, sua complexidade decai de forma que o acréscimo de complexidade gerado pelas técnicas de lógica difusa tende a desaparecer.

Referências

Awduche, D.; Chiu, A.; Elwalid, A.; Widjaja, I. e Xiao, X.. (2002) “Overview and Principles of Internet Traffic Engineering.”, IETF RFC 3272, http://www.faqs.org/rfcs/rfc3272.html, Maio

Awduche, D. O. e Jabbari, B. (2002) "Internet Traffic Engineering using Multi-Protocol Label Switching (MPLS)", Computer Networks, (40):p. 111–129, Setembro.

Battiti, R. e Salvadori, E. (2002) “A Load Balancing Scheme for Congestion Control in MPLS Networks. Technical report”, Universitá di Trento, Dipartimento di Informatica e Telecomunicazioni, Novembro.

Black, U. (2001) “MPLS and Label Switching Networks”, Prentice Hall

Chen, C.H. (1996) "Fuzzy Logic and Neural Network Handbook/the Handbook of Software for Engineers and Scientists", IEEE, Setembro

Holness, F. e Phillips, C. (2000) “Dynamic Congestion Control Mechanism for MPLS Networks.”, In: SPIE’s International Symposium on Voice, Video and Data Communications. Internet, Performance and Control Network ystems, p. 1001–1005, Boston - MA, Novembro

Jüttner, A.; Szviatovszki, B.; Szentesi, A.; Orincsay, D. e Harmatos, J. (2000) “On-demand Optimization of Label Switched Paths in MPLS Networks.”, In: Proceedings of IEEE International Conference on Computer Communications and Networks, p. 107–113, Las Vegas - Nevada, Outubro

Kar, K.; Kodialam, M. e Lakshman, T.V. (2000) "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications", IEEE Journal on Selected Areas in Communications, 18(12):p. 2566–2579, Dezembro.

Osborne, E. e Simha, A. (2003) “Traffic Engineering With MPLS”, Cisco Press

Salvadori, E;.Sabel, M e.Battiti, R. (2002) “ A Reactive Scheme For Traffic Engineering In Mpls Networks. Technical report”, Universitá di Trento, Dipartimento di Informatica e Telecomunicazioni, Dezembro.

Wang, B.; Su, X. e Chen, C.P.. (2002) “A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering”, In: Proceedings of ICC, volume 2, p.1001-1005, New York - USA