UNIVERSIDADE SALVADOR - UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO EM REDES DE COMPUTADORES
MESTRADO PROFISSIONALIZANTE EM REDES DE COMPUTADORES

MARCELO BURGOS MORGADE CORTIZO

MINIMIZAÇÃO DE FRAGMENTAÇÃO DE BANDA EM ALGORITMOS DE ROTEAMENTO PARA ENGENHARIA DE TRÁFEGO

Salvador
2004

MARCELO BURGOS MORGADE CORTIZO

MINIMIZAÇÃO DE FRAGMENTAÇÃO DE BANDA EM ALGORITMOS DE ROTEAMENTO PARA ENGENHARIA DE TRÁFEGO

Dissertação apresentada à Universidade Salvador, como parte das exigências do Curso de Mestrado Profissional em Redes de Computadores, área de concentração em Redes de Computadores, para obtenção do título de Mestre.

Orientador: *Prof. Dr. José Augusto Suruagy Monteiro*

Salvador
2004

Dedico este trabalho aos meus pais, meus irmãos e à Milena.

AGRADECIMENTOS

Agradeço a Deus pela força e discernimento que tem me dado para conquistar meus objetivos.

Aos meus pais e meus irmãos, pelo apoio moral, orientação, e exemplo de vida. À minha noiva Milena, por me ajudar em todos os sentidos possíveis nessa caminhada.

Ao meu orientador Prof. Suruagy, pelo afinco com que trabalhou na orientação técnica e metodológica deste trabalho, e pela grande oportunidade de poder trabalhar no NUPERC. Aos professores da banca de defesa.

A todos os colegas do NUPERC, pela companhia e pelas discussões de alto nível travadas durante os dias de trabalho que também são uma grande forma de orientação.

Ciência da Computação está tão relacionada aos computadores quanto a Astronomia aos telescópios, Biologia aos microscópios, ou Química aos beakers e tubos de ensaio. A Ciência não estuda ferramentas. Ela estuda como nós as utilizamos, e o que descobrimos com elas.

—EDSGER W. DIJKSTRA (1930–2002)

RESUMO

Fragmentação de banda pode ocorrer em redes orientadas a conexão como MPLS e ATM devido à ordem em que os pedidos de conexão são processados. O problema identificado neste trabalho causa a redução do número total de requisições que podem ser atendidas. Assumindo que o conjunto de requisitos de banda associados às conexões pode ser pré-determinado, este trabalho apresenta um método de baixa complexidade capaz de minimizar os efeitos do problema de fragmentação de banda. Este método é baseado na solução de equações diofantinas lineares e execuções do Algoritmo de Euclides Estendido, sendo capaz de computar uma medida de potencial de fragmentação para cada enlace da rede. Estas medidas são usadas para aumentar proporcionalmente o peso dos enlaces e, conseqüentemente, reduzir a probabilidade dos algoritmos de roteamento usarem estes enlaces de forma a causar fragmentação. Este método foi aplicado ao algoritmo MINHOP e ao algoritmo de interferência mínima de Su e Chen. Resultados obtidos através de simulações mostraram os benefícios desta solução genérica através do aumento na quantidade total de requisições atendidas.

Palavras-chave: Engenharia de Tráfego, Algoritmos de Roteamento, MPLS, Fragmentação de Banda.

ABSTRACT

Bandwidth fragmentation can occur in connection-oriented networks such as MPLS and ATM due to the order in which connection requests are processed. This problem detected by this work causes the reduction of the overall number of accepted requests. Assuming that the set of connection requirements are known in advance, this work presents a low complexity method to minimize the effects of the bandwidth fragmentation problem. This method is based on the solution of linear diophantine equations and executions of the Extended Euclidean Algorithm, computing a fragmentation potential measure for each link. These measures are used to proportionally increase the link weights therefore reducing the fragmentation caused by routing algorithms. This method was applied to MINHOP, and Su and Chen's modified MIRA algorithms. Simulation results showed the benefits of this generic solution by increasing the overall number of accepted requests.

Keywords: Traffic Engineering, Routing Algorithms, MPLS, Bandwidth Fragmentation.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

O sucesso das redes IP como solução fácil e de baixo custo se consolidou com sua utilização em larga escala no mundo inteiro. O enorme crescimento destas redes criou uma necessidade urgente de melhoria nas técnicas de controle sobre a real capacidade da rede e sobre sua eficiência no transporte dos fluxos de dados. Neste contexto, o processo de Engenharia de Tráfego (ET) [Awduche et al., 2002] surgiu como uma ferramenta eficiente para o controle do crescimento da rede. Os estudos sobre Engenharia de Tráfego em redes IP tomam como base as atividades de otimização e avaliação do desempenho de redes em operação, com o objetivo de torná-las mais eficientes e confiáveis. A otimização do desempenho do tráfego visa a melhoria dos indicadores de qualidade de serviço (*QoS*) nestas redes onde, muitas vezes, existe apenas o serviço de melhor esforço.

A Engenharia de Tráfego é um processo definido pela IETF (Internet Engineering Task Force) que passa por fases de coleta de dados, formulação de políticas e aplicação de ações corretivas ou preventivas de melhoria na distribuição do tráfego. Este processo é cíclico e envolve o modelo de ações definido na Figura 1.1. O objetivo geral deste processo é distribuir o tráfego de forma eficiente, evitando que uma região da rede esteja saturada enquanto outras estão sub-utilizadas. Assim se reduz a probabilidade de congestionamento e aumenta a percepção de qualidade que o usuário tem da rede.

Implementar o processo de Engenharia de Tráfego usando somente as funcionalidades providas pelo IP é muito difícil devido à sua carência de recursos avançados para controle de tráfego e roteamento explícito. Mas, com a especificação do *Multi-Protocol Label Switching* (MPLS) [Rosen et al., 2001], surgiram as ferramentas capazes de resolver estes problemas. O MPLS se encaixou perfeitamente nas necessidades da Engenharia de Tráfego devido à sua capacidade de poder criar um modelo sobreposto de circuitos virtuais numa rede orientada a datagrama como o IP. A Figura 1.2 mostra que o trabalho do

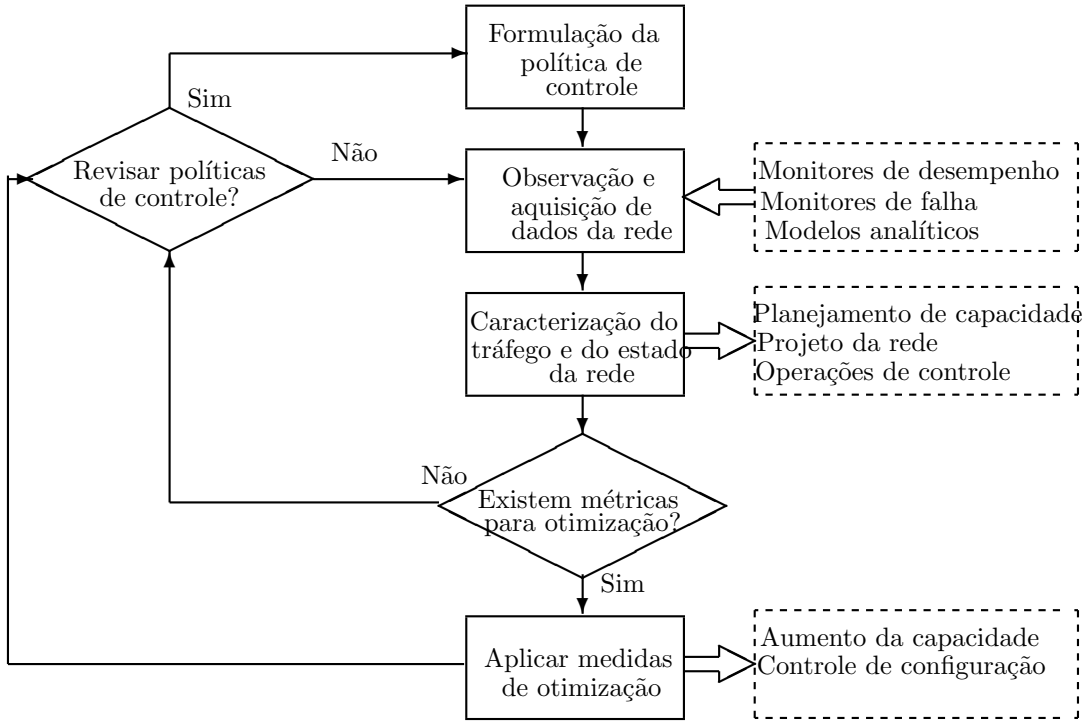

Figura 1.1. Modelo de processo de Engenharia de Tráfego [Awduche, 1999].

MPLS sobre uma rede IP é semelhante ao do ATM, ou seja, criar um rede orientada a conexões virtuais que servirá como base para o roteamento e gerenciamento dos datagramas IP. Uma das grandes vantagens do MPLS neste caso é que não é necessário trocar os roteadores IP por um outro hardware comutador. Os Roteadores de Comutação de Rótulos (*Label Switching Routers* — LSRs) podem ser o memos hardware dos roteadores IP rodando o software MPLS. Assim, o MPLS é capaz de criar caminhos comutados por rótulos ou *Label Switched Paths* (LSPs) sobre enlaces IP dinamicamente a um baixíssimo custo [Awduche, 1999].

A garantia de medidas de qualidade de serviço é um dos objetivos da Engenharia de Tráfego que é provido por estes caminhos virtuais através da associação de LSPs com valores de banda. Através de processos especiais de escalonamento nas filas dos enlaces, estes valores de banda podem ser reservados para um determinado LSP em cada enlace que compõe o seu caminho. Estes canais virtuais controlados são estabelecidos através da utilização de alguns protocolos de sinalização auxiliares do MPLS como o *Constraint-Routing Label Distribution Protocol* (CR-LDP) [Jamoussi et al., 2002] e o *Resource Reservation Protocol — Traffic Engineering* (RSVP-TE) [Awduche et al., 2001]. Assim, o papel central do MPLS na Engenharia de Tráfego é mapear os fluxos identifi-

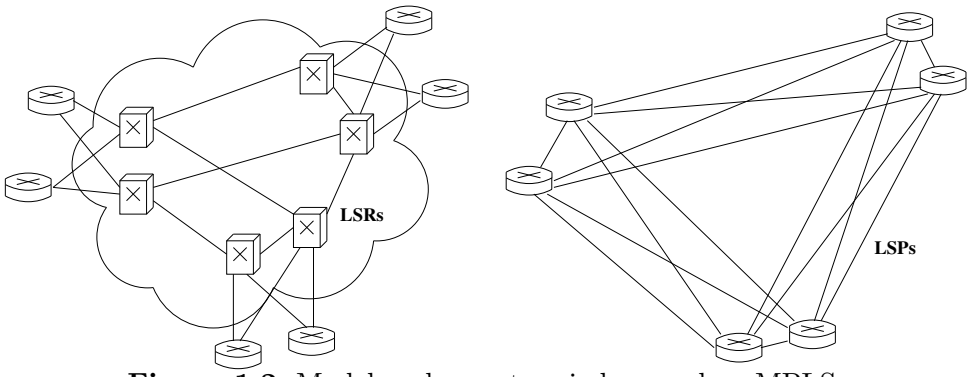

Figura 1.2. Modelo sobreposto criado usando o MPLS.

cados pelo estudo da rede de modo eficiente em LSPs, levando em conta a distribuição otimizada do tráfego e restrições pertinentes à garantia de *QoS*. O MPLS também deve ser responsável pela sinalização que estabelece e mantém estes LSPs, além de implementar a comutação dos pacotes rotulados.

Mas, antes do MPLS atuar, existe uma fase que consta da seleção dos enlaces e nós intermediários sobre os quais cada LSP passará e utilizará recursos. As diversas necessidades de *QoS* que podem estar ligadas a um fluxo de tráfego é um problema que aumenta muito a complexidade na busca de um caminho cujos recursos disponíveis estejam compatíveis com o que é requisitado. Avaliar se um determinado caminho suprirá as necessidades de vazão, atraso máximo, variação do atraso (*jitter*) e perdas de um fluxo é uma tarefa extremamente custosa, que envolve informações sobre a largura de banda dos enlaces, utilização e tamanho de buffers além da capacidade de processamento dos nós. Entre as soluções propostas para amenizar este problema, tornou-se popular a utilização do conceito de banda efetiva [Kelly, 1996]. A banda efetiva é um valor calculado ou aproximado através de métodos probabilísticos. Este valor de banda é capaz de sumarizar todas as necessidades de recursos da rede para provisão das medidas de *QoS* desejadas por uma fonte de tráfego. Assim, é possível usar como principal restrição ao estabelecimento de um LSP, a necessidade de que cada um dos enlaces utilizados pelo caminho tenha banda residual que satisfaça o valor de banda efetiva total associada aos fluxos de tráfego a serem roteados pelo LSP.

Apesar da banda efetiva ser utilizada para simplificar o problema, o processo que determina o conjunto de caminhos otimizado para um conjunto de LSPs numa rede é

conhecido por ser um problema polinomial não determinístico[1] [Girish et al., 2000]. Esta complexidade é um problema sério, visto que a chegada de requisições de modo dinâmico traz a necessidade de tempos de resposta muito curtos. A necessidade por respostas rápidas para a resolução deste problema tão complexo motivou o desenvolvimento de diversos algoritmos de roteamento que tentam produzir um conjunto de caminhos próximos do ótimo.

Estes algoritmos vêm sendo aperfeiçoados e são objetos de constante estudo. Eles seguem diferentes critérios de otimização na forma de alocar os recursos da rede com o objetivo de maximizar a quantidade de caminhos que podem ser estabelecidos pela rede. Este objetivo é alcançado através da arrumação dos caminhos de forma a evitar que regiões da rede estejam saturadas enquanto outras estão sub-utilizadas. Estes algoritmos utilizam informações sobre a topologia, o estado global da rede ou até dados sobre os possíveis pares de entrada ou saída. Estes dados são usados de forma a implementar métodos de busca de caminhos que sejam compatíveis com um conjunto de regras ou restrições pré-estabelecidas, as quais podem determinar a quantidade ou tipos de recursos necessários na rede para o perfeito encaminhamento dos fluxos. Assim, em contraste com a idéia básica de roteamento do IP, onde a rota mais curta é sempre a escolhida, no roteamento baseado em restrições (*Constraint Based Routing* — CBR) [Jamoussi et al., 2002] os caminhos escolhidos devem sempre ser compatíveis com as necessidades dos fluxos descritas nas regras. A banda efetiva associada aos fluxos que passarão por um caminho entre dois nós da rede é uma das restrições mais usadas no momento do roteamento para a criação de um novo LSP. Enlaces com capacidade residual menor que este valor de banda requisitado são descartados pelos algoritmos de roteamento baseado em restrições antes mesmo da busca começar.

Numa observação mais cuidadosa, este padrão de comportamento dos algoritmos de roteamento pode levar a um problema que foi identificado neste trabalho e denominado como Fragmentação de Banda. Foi identificado que este problema acontece à medida em que os enlaces vão chegando perto da sua saturação. À medida que os LSPs vão sendo roteados sobre um enlace, sua banda residual vai diminuindo até chegar

[1] São chamados de problemas NP-Completos. É uma classe de problemas para os quais não se conhece, ou foi provado que não existe solução em tempo menor que exponencial.

num ponto em que as novas requisições de LSPs não conseguem mais utilizá-lo pois têm demandas associadas quase sempre maiores que a banda restante no enlace. Neste momento, existe ainda uma quantidade de banda disponível nestes enlaces, mas que nunca é utilizada para o atendimento de novas requisições. Numa rede com muitos enlaces, o total dessa banda não utilizada (a banda fragmentada) acaba tornando-se um ponto de queda de desempenho dos algoritmos de roteamento. Se os algoritmos de roteamento conseguissem aproveitar toda a banda fragmentada que fica espalhada nos enlaces da rede, mais requisições de LSPs poderiam ser atendidas. Em redes densas cujas demandas de requisições são relativamente altas, este problema se torna ainda pior pois a quantidade de banda fragmentada tende a ser muito maior.

Na Figura 1.3 é possível observar as conseqüências do problema. Na topologia em questão, a capacidade residual dos enlaces mostra que a rede ainda suporta levar um fluxo máximo (*maxflow*) de 8 unidades [Ford & Fulkerson, 1956]. Mas, mesmo com este fluxo total disponível, não há nenhum caminho capaz de suportar o LSP que acaba de chegar com 5 unidades de demanda associadas. O roteamento não é possível por que este fluxo máximo de 8 unidades disponível passa por enlaces com banda fragmentada que não podem ser usados pelo novo LSP (os enlaces de capacidade residual 2 e 3). Se estas bandas residuais tivessem outra arrumação pelos enlaces, certamente este novo caminho poderia se estabelecido.

A fragmentação de banda pode ocorrer devido à combinação dos valores que vão sendo alocados e à ordem na qual as requisições chegam na rede. A alocação de certas combinações de valores de banda em um enlace pode deixar o mesmo com uma capacidade residual livre que dificilmente poderá ser utilizada pelo conjunto de demandas esperado pela rede. Isto acaba reduzindo a porcentagem total de banda utilizável para o estabelecimento de LSPs e conseqüentemente, acarreta uma diminuição na quantidade total de LSPs que podem ser suportados pela rede.

Uma possível forma de contornar este problema seria evitar as combinações de valores alocados que geram banda fragmentada. Este controle poderia ser feito utilizando informações sobre o conjunto de valores de demanda que serão alocados pelos LSPs e a banda residual dos enlaces. Mas, o momento de descarte dos enlaces com pouca

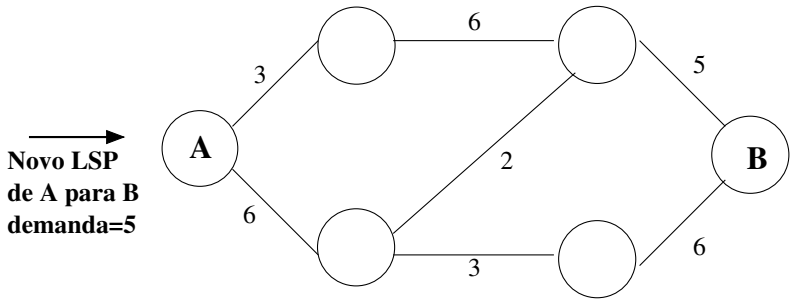

Figura 1.3. Roteamento prejudicado pela fragmentação.

capacidade é o único ponto em que as demandas de banda associadas aos LSPs são utilizadas pelos algoritmos de roteamento. Ao contrário de outras informações globais como topologia e estado dos enlaces, estes algoritmos não utilizam os dados globais sobre o conjunto de requisitos de banda para LSPs que chegaram ou que chegarão na rede. A não utilização destas informações sobre as possíveis demandas de banda acaba impedindo-os de contemplar o problema da fragmentação de banda.

Os objetivos deste trabalho são apresentar a definição do problema da fragmentação de banda que ocorre com a utilização dos algoritmos de roteamento propostos na literatura para Engenharia de Tráfego e propor um método capaz de diminuir os efeitos deste problema sem aumentar a complexidade do processo de roteamento. Como ferramenta para alcançar estes objetivos, os algoritmos de roteamento e o método proposto para redução da fragmentação foram implementados no simulador *Network Simulator* (ns2) [Fall & Varandhan, 2003] e foram utilizados em experimentos que validam os dados apresentados no trabalho.

No próximo capítulo são apresentados alguns algoritmos de roteamento comumente usados na Engenharia de Tráfego, os quais são atualmente alvo de constante estudo. No capítulo 3 o problema da fragmentação de banda e o método para minimização da fragmentação são apresentados em detalhes. No capítulo 4 são apresentados os resultados de experimentos que mostram a atuação do método de minimização da fragmentação em conjunto com os algoritmos de roteamento para Engenharia de Tráfego. Finalmente, o capítulo 5 apresenta as considerações finais, sumariza as contribuições deste trabalho e sugestões para trabalhos futuros.

CAPÍTULO 2

ALGORITMOS DE ROTEAMENTO PARA ENGENHARIA DE TRÁFEGO

Este capítulo apresenta um resumo sobre as características dos principais algoritmos de roteamento usados pela Engenharia de Tráfego. A seção 2.1 apresenta os requisitos para a prática da Engenharia de Tráfego usando alocação dinâmica de recursos ou o roteamento *online*. A seção 2.2 apresenta a solução geral para o problema do roteamento baseado em restrições, a partir da qual foram elaborados os algoritmos de roteamento. A seção 2.3 apresenta o funcionamento de algoritmos de roteamento baseados em critérios simples de otimização. A seção 2.4 apresenta o funcionamento de algoritmos de roteamento mais complexos baseados no conceito de interferência mínima. Finalmente, a seção 2.5 apresenta um breve comparativo entre os algoritmos de roteamento apresentados durante o capítulo.

2.1 ENGENHARIA DE TRÁFEGO E O ROTEAMENTO ONLINE

A utilização e padronização da Engenharia de Tráfego nas redes IP começou a ganhar uma presença maior após a especificação e implementação do MPLS. Com sua sofisticada capacidade de controle sobre o roteamento, o MPLS surgiu como solução alternativa ao ATM para constituição de um modelo sobreposto de circuitos nas redes IP, trazendo novas vantagens e soluções para os problemas encontrados anteriormente. O MPLS simplificou a aplicação do modelo, necessitando de menos elementos de rede, e podendo utilizar os mesmos equipamentos que trabalham no roteamento tradicional do IP com um novo software. Com isso, criou-se uma solução com custo de implantação e gerenciamento muito menor.

Um dos conceitos básicos da aplicação do MPLS para engenharia de tráfego é a de troncos de tráfego. Um tronco de tráfego, segundo [Awduche et al., 1999], é um

conjunto de fluxos de tráfego que seguem de um mesmo LSR de entrada (*ingress LSR*), para um mesmo LSR de saída (*egress LSR*). Os troncos podem ser encaminhados em qualquer LSP que ligue o LSR de entrada ao LSR de saída.

Desse modo, podem ser definidos três problemas fundamentais para a engenharia de tráfego usando o MPLS:

- Como mapear os pacotes na rede em Classes Equivalentes de Encaminhamento (*Forwarding Equivalence Classes* — FECs),
- Como mapear as FECs em troncos de tráfego e
- Como alocar LSPs para os troncos de tráfego.

Os algoritmos de roteamento são os componentes responsáveis pela resolução deste terceiro problema. O objetivo central destes algoritmos é otimizar o processo de alocação de banda nos enlaces, distribuindo o tráfego que passa pelos LSPs de modo a aumentar a capacidade de atendimento de novas demandas que possam chegar na rede. Porém, este processo de alocação otimizada é conhecido por apresentar uma complexidade muito alta devido às inúmeras possibilidades de roteamento dos caminhos pela rede.

Redes que usam roteamento *offline*, nas quais todos os LSPs são estabelecidos antes do início da sua operação, não apresentam exigências muito rígidas quanto ao tempo de resposta do processo. Este tipo de roteamento não precisa computar rotas em tempo real, sendo utilizado na maioria das vezes quando é necessário estimar as informações necessárias para a otimização ou quando se fazem buscas intensivas em espaços de solução multi-dimensionais.

Mas, nas redes que usam alocação dinâmica ou roteamento *online*, as requisições para estabelecimento de novos LSPs são feitas já com a rede em operação. Este tipo de roteamento deve computar rotas em tempo real e é utilizado quando a base de informação é o estado corrente da rede. O tempo de resposta e o custo computacional do processo de roteamento neste caso é um fator crítico, pois estas requisições podem ser disparadas por demandas de aplicações dos usuário ou operações de re-roteamento. Além disso, este processo não deve prejudicar as outras funções do dispositivo que o executa. Assim, aplicar a otimização completa em roteamento *online* não é eficiente o suficiente

para gerar soluções aceitáveis, sendo necessária a utilização de heurísticas capazes de gerar boas soluções em um tempo razoável.

Diversos algoritmos de roteamento vêm sendo propostos a fim de solucionar este problema. Diferentes abordagens também são utilizadas nestas propostas. Os algoritmos mais comuns utilizam o conceito de banda efetiva e outros critério de otimização para implementar um processo que atribui pesos aos enlaces que servem como entrada para a execução de variantes do algoritmo de *Dijkstra* [Dijkstra, 1959]. Alguns algoritmos usam explicitamente atraso e *jitter* como métrica nos seus processos de roteamento [Goel et al., 2001, Ma & Steenkiste, 1997]. Esta é uma abordagem cuja implementação acaba sendo um pouco complexa devido à natureza dinâmica de medidas como o espaço no buffer dos nós, necessárias para o processamento destes algoritmos. Algumas abordagens utilizam técnicas de otimização baseadas em conceitos de Inteligência Artificial como algoritmos genéticos [Riedl, 2002] e redes neurais [Stüzle & Hrycej, 2002]. Outras abordagens diferentes para o problema podem ser encontradas em [Bagula & Krzesinski, 2001, Riedl, 2003, Fortz & Rexford, 2002]. Estes algoritmos podem ser executados a partir de um agente centralizado de controle (um servidor de rotas), ou num esquema distribuído onde cada nó determina os caminhos para otimização a partir das informações que ele mesmo obtém sobre o estado da rede.

Estes algoritmos podem utilizar como entrada qualquer informação coletada pelo processo de Engenharia de Tráfego como a topologia da rede, conhecimento sobre os nós que geram tráfego, as características dos fluxos que são roteados, capacidade total e residual dos enlaces. Cada execução dos algoritmos atende a uma requisição de um caminho de um nó origem a um nó destino com uma ou mais medidas de *QoS* associadas que devem ser respeitadas. A saída do algoritmo pode ser somente a seqüência de enlaces descrevendo o caminho encontrado ou a rejeição da requisição. Assim, estes algoritmos ainda dependem de um mecanismo de admissão de chamadas e de processos que estabeleçam de fato os LSPs utilizando uma rota definida explicitamente. Neste ponto entram em cena os protocolos de sinalização do MPLS (CR-LDP ou RSVP-TE), que ficam responsáveis por estabelecer e manter os LSPs sobre os enlaces que formam os caminhos encontrados pelos algoritmos de roteamento.

Uma das medidas mais importantes para a medição do desempenho destes algoritmos é a quantidade de requisições bloqueadas ou a quantidade total de vazão dos fluxos nos LSPs que puderam ser estabelecidos. Algoritmos de roteamento eficientes devem atender o máximo possível de requisições, começando a bloquear os pedidos apenas quando a rede estiver quase que completamente saturada.

2.2 ROTEAMENTO BASEADO EM RESTRIÇÕES

Apesar da existência das diferentes abordagens, todos os algoritmos de roteamento para Engenharia de Tráfego partem do conceito de roteamento baseado em restrições. A idéia central do roteamento baseado em restrições é a de encontrar caminhos capazes de realmente garantir a QoS associada às requisições. Assim, decisões sobre o direcionamento do tráfego podem ser tomadas a partir de regras que garantam o desempenho do tráfego, ou regras administrativas definidas de acordo com as políticas da rede. Uma série de restrições de roteamento, identificadas em [Awduche et al., 1999] podem ser usadas no processo de estabelecimento dos LSPs. Estas regras atuam dentro dos algoritmos de roteamento, e podem excluir da busca os recursos da rede com capacidade insuficiente ou não compatíveis com as características associadas aos fluxos do futuro LSP.

A restrição básica e normalmente a mais usada nos algoritmos é a que impede que o total da banda necessária aos LSPs que estejam sendo criados exceda a capacidade residual de um dos enlaces. Essa capacidade residual é a diferença entre a capacidade total do enlace e a soma das bandas associadas a cada um dos LSPs já estabelecidos que utilizam o enlace[1]. Como os enlaces com capacidade insuficiente são excluídos antes da busca começar, garante-se que se um caminho entre o nó origem e o nó destino for encontrado, certamente será um caminho capaz de prover a banda necessária para o LSP a ser estabelecido.

Outras restrições podem envolver o número de saltos, excluindo a possibilidade dos algoritmos gerarem caminhos muito longos. Esta é uma forma de não aumentar muito

[1]Detalhes sobre a formulação do problema de roteamento baseado em restrições podem ser encontrados em [Girish et al., 2000].

o atraso experimentado pelos fluxos no LSP. Outras restrições relativas a atrasos e *jitter* podem envolver regras ligadas à utilização dos buffers nos nós da rede. Porém, por serem medidas mais dinâmicas, estas regras são mais difíceis de serem implementadas. É aqui que o conceito de banda efetiva pode ser utilizado de forma a simplificar as regras de roteamento.

Nas seções a seguir, é apresentado como alguns algoritmos de roteamento baseados no roteamento baseado em restrições trabalham em conjunto com o algoritmo de *Dijkstra* para encontrar caminhos compatíveis com as necessidades das requisições. Estes algoritmos normalmente usam diferentes critérios na definição dos pesos dos enlaces, o que modifica a ordem de alocação dos enlaces e, conseqüentemente, o desempenho de cada um deles em diferentes situações.

2.3 ALGORITMOS DE ROTEAMENTO SIMPLES

Estes algoritmos usam critérios bem simples para a definição dos pesos de seus enlaces, como o número de saltos do caminho ou a banda residual nos enlaces que serão utilizados. Devido a isso, apresentam uma complexidade computacional relativamente baixa, normalmente igual ou bem próxima à complexidade do algoritmo de *Dijkstra*. A baixa complexidade é uma vantagem interessante, mas essa simplicidade acaba custando com a diminuição do desempenho na avaliação dos resultados do roteamento em relação aos algoritmos mais sofisticados. Como são feitas poucas análises dos dados obtidos em relação às características dos fluxos e da topologia, decisões que melhorariam a distribuição do tráfego deixam de ser tomadas.

2.3.1 Minimum Hop Algorithm

O Algoritmo de Minimização de Saltos (*Minimum Hop Algorithm* — MINHOP) é a mais simples proposta de algoritmo de roteamento para Engenharia de Tráfego. A essência do MINHOP é a utilização do critério de escolha do caminho mais curto (*Shortest Path First* – SPF) em conjunto com o roteamento baseado em restrições. Assim, a rota escolhida do LSR de entrada para o LSR de saída é aquela que passa pelo

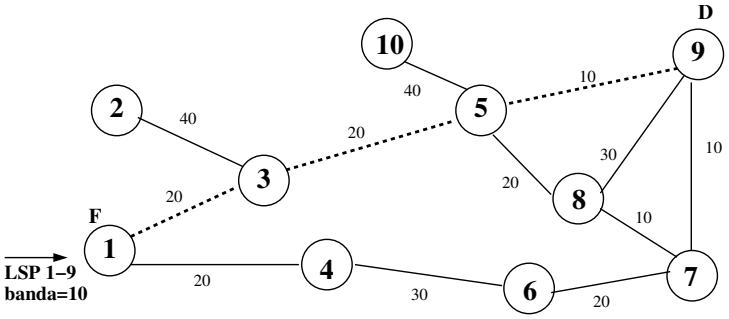

Figura 2.1. Caminho de 1 a 9 escolhido pelo MINHOP.

menor número de nós e enlaces que sejam compatíveis com as restrições do caminho. A Figura 2.1 ilustra a ação do MINHOP, numa rede em que chega uma requisição de caminho entre os nós 1 e 9 com 10 unidades de banda como requisito para alocação. Como todos os enlaces têm capacidade suficiente, o algoritmo faz a busca usando a rede completa e encontra o caminho 1–3–5–9, que é o caminho com menor número de saltos possível de se estabelecer.

A implementação desta abordagem é também muito simples. Ao chegar uma nova requisição, o algoritmo primeiro descarta os enlaces que não têm capacidade suficiente para ser alocada, e depois roda o algoritmo de *Dijkstra* com uma matriz de pesos onde todos os enlaces têm valor unitário. Uma variante do MINHOP é conhecido como *Widest-Shortest Path (WSP)*, que usa como critério de desempate (quando mais um de caminho de tamanho mínimo for encontrado) a escolha do caminho com mais banda disponível nos enlaces.

Usar a métrica de número de saltos é a visão clássica do roteamento IP e visa reduzir o atraso esperado para os fluxos devido à redução da quantidade de vezes em que os pacotes são enfileirados e encaminhados em cada nó. Usar o caminho mais curto também é a forma de alocar a menor quantidade de recursos possível para o estabelecimento de um LSP. Porém escolher sempre o menor caminho nem sempre é a melhor escolha à longo prazo. Esse comportamento significa que os caminhos entre um certo par de entrada e saída da rede usarão sempre os mesmos enlaces, fazendo com que certos enlaces sejam rapidamente consumidos.

Isso pode trazer uma série de problemas. Pares de nós que fazem requisições muito rapidamente podem monopolizar enlaces centrais na rede em detrimento de outros

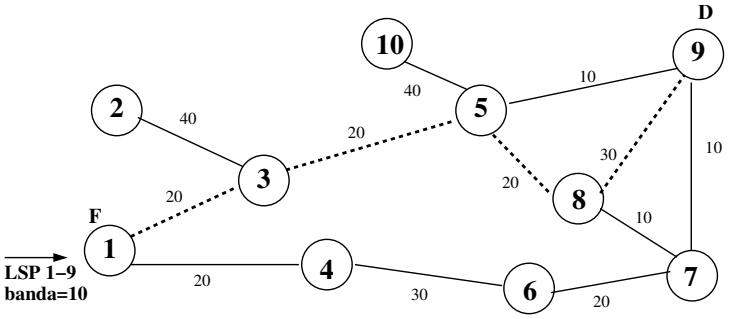

Figura 2.2. Caminho de 1 a 9 escolhido pelo SWP.

pares. A saturação prematura de certos enlaces causada pela escolha freqüente de um mesmo caminho curto, também pode fazer com que certas áreas da rede fiquem virtualmente desconectadas. Todos estes problemas acabam fazendo com que o MINHOP comece a bloquear requisições mais rapidamente que outros algoritmos. Assim, apesar de ser o algoritmo mais simples, o MINHOP é geralmente o que apresenta as menores curvas de desempenho em experimentos encontrados na literatura.

2.3.2 Shortest-Widest Path

O *Shortest-Widest Path* (SWP) é um outro algoritmo de roteamento também bastante simples. O principal critério do SWP é a busca ao caminho do nó de entrada ao nó de saída que possua a maior capacidade disponível no momento do roteamento. A capacidade de um caminho é a medida máxima de banda que pode ser alocada usando os enlaces que o formam. O enlace (ou os enlaces) com menor capacidade residual num caminho no momento do roteamento é o elemento limitador do máximo de vazão que pode ser provida pela futura conexão. Assim a capacidade de um determinado caminho é o menor valor de capacidade residual encontrado nos enlaces que ligam o nó de partida ao nó de chegada.

A Figura 2.2 ilustra a ação do SWP, na mesma situação ilustrada na Seção 2.3.1. O SWP escolhe o caminho 1–3–5–8–9, que tem a capacidade de suportar LSPs com demanda de até 20 unidades de banda. Apesar do caminho 1–3–5–9 apresentar menos saltos, o SWP não o escolhe por ter uma capacidade de apenas 10 unidades de banda.

A implementação do SWP é baseada numa pequena modificação no algoritmo de *Dijkstra*. No momento da execução do roteamento o peso de cada enlace é inicializado

com o valor inverso da respectiva capacidade residual. Isso faz com que enlaces com maior capacidade sejam mais facilmente escolhidos. A modificação do algoritmo foi feita justamente no passo que ajusta a distância percorrida pela busca até um certo nó. Em vez de somar os valores acumulados dos pesos até o nó, a distância do nó é definida como o maior valor de peso encontrado no caminho até ele. Assim, o algoritmo consegue encontrar o caminho cujo enlace de maior peso tem o menor valor entre os valores encontrados no demais caminhos. Como estes pesos são inversos das capacidades, o algoritmo acaba achando o caminho que tem a maior capacidade. Caso o algoritmo encontre dois caminhos com a mesma capacidade, aquele com o menor número de saltos é escolhido.

Esta abordagem utilizada pelo SWP apresenta um desempenho bem melhor em relação ao MINHOP. Esta melhoria pode ser creditada ao fato do critério de escolha do caminho ser baseado numa medida mais dinâmica. No MINHOP, o critério do menor número de saltos é uma medida estática, e por isso um conjunto de enlaces centrais eram sempre escolhidos até ficarem rapidamente saturados. O critério do caminho de maior capacidade é dinâmico, pois as capacidades residuais dos enlaces mudam a cada alocação. Devido a isso, o SWP consegue "espalhar" melhor os LSPs à medida em que as requisições vão sendo atendidas.

Porém, o SWP também tem os seus problemas. Usar sempre o critério de maior capacidade em redes mais complexas pode levar o algoritmo a encontrar caminhos com um número muito grande de saltos. Isso acaba aumentando muito o atraso experimentado pelo pacotes que utilizam o caminho.

2.3.3 Outros algoritmos

Outros algoritmos baseados em critérios simples para definição de pesos foram definidos, tentando ajustar os pontos que causam os problemas gerados pelos algoritmos definidos na seção anterior. Em [Ma & Steenkiste, 1997], resultados de experimentos mostram que a eficiência destes algoritmos não é muito superior, ficando sempre próximos do MINHOP e abaixo do SWP.

Um destes algoritmos é chamado de *Dynamic-Alternative Path*. Este algo-

ritmo tenta ajustar a questão da possibilidade da escolha de caminhos muito longos. Para evitar este problema, o algoritmo adiciona uma restrição ao número de saltos que os caminhos escolhidos podem conter. O caminho com maior capacidade é buscado inicialmente pelo *Dynamic-Alternative Path*, com um número máximo de saltos possível. Se nenhum dos caminhos possíveis respeitar a restrição do número máximo de saltos, o algoritmo busca um caminho com um salto a mais que tenha mais capacidade disponível. Caso ainda assim não sejam encontrados caminhos compatíveis com o número de saltos, o pedido de estabelecimento do LSP é rejeitado.

Um outro algoritmo simples é chamado de *Shortest-dist(P,1)*. Como no SWP, este algoritmo também usa o inverso das bandas residuais dos enlaces no momento do roteamento como valor dos pesos. Porém neste caso, o algoritmo de *Dijkstra* não é modificado. O objetivo é encontrar um caminho P composto por k enlaces com capacidade residual R_i que tenha a menor distância $dist(P,1)$ definida como:

$$dist(P,1) = \sum_{i=1}^{k} \frac{1}{R_i}.$$

Assim, o caminho escolhido é sempre aquele que tem a maior soma das capacidades residuais dos seu enlaces.

2.4 ALGORITMOS DE ROTEAMENTO COM INTERFERÊNCIA MÍNIMA

Esta classe de algoritmos aplica o conceito de interferência mínima entre pares de entrada e saída da rede, com o objetivo de maximizar a probabilidade de atendimento de futuras requisições. Diferente dos critérios usados nos algoritmos simples, para minimizar a interferência entre caminhos são necessários mais dados da topologia e um processamento mais apurado sobre as possibilidades de roteamento.

A interferência é uma medida que envolve pares de nós origem-destino. Num dado momento, o conjunto de enlaces que podem formar caminhos entre os nós de um par têm juntos uma determinada capacidade máxima de vazão capaz de encaminhar fluxos do nó de entrada para o nó de saída. Isso vale para todos os pares origem-destino da rede. Essa vazão total que pode ser obtida usando os diversos caminhos entre um par de nós é chamada de fluxo máximo ou *maxflow* [Ford & Fulkerson, 1956]. Quando um

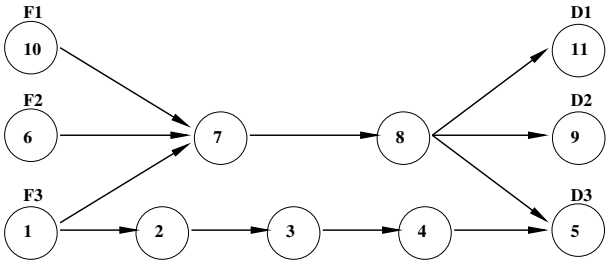

Figura 2.3. Exemplo de efeitos da interferência [Kodialam & Lakshman, 2000].

par A estabelece um LSP entre seus nós origem-destino, parte de seu *maxflow* restante é consumida devida à alocação de banda em alguns enlaces. Estes mesmos enlaces utilizados por A podem participar do conjunto de enlaces que determinam o *maxflow* do par B. Neste caso, B também tem seu fluxo máximo diminuído devido ao LSP estabelecido para A. Assim, diz-se que ocorreu interferência de A em B. O valor da interferência é a quantidade decrementada do *maxflow* de B devido ao roteamento do caminho para A. O objetivo principal de um algoritmo de roteamento com interferência mínima é escolher um caminho para um determinado par origem-destino que preserve a maior quantidade possível de fluxo máximo entre os outros pares da rede.

Nota-se que é necessário utilizar o conhecimento sobre todos os nós de borda da rede. Os algoritmos utilizam este conhecimento para calcular a interferência dos caminhos a fim de reduzir o número de novas requisições rejeitadas. Este conhecimento é adquirido nas fases iniciais do processo de Engenharia de Tráfego. O grafo mostrado na Figura 2.3 mostra claramente os efeitos da interferência. Neste grafo, cada aresta tem capacidade igual a 1 unidade. Os pares origem-destino estão marcados como $F_1 - D_1$, $F_2 - D_2$ e $F_3 - D_3$. Considera-se que chega uma nova requisição de caminho entre $F_3 - D_3$ com 1 unidade de banda associada. Algoritmos simples neste caso escolheriam o caminho mais curto (1-7-8-5). Mas alocar os enlaces deste caminho tornaria impossível rotear qualquer nova requisição entre $F_1 - D_1$ e $F_2 - D_2$. Estes dois pares seriam virtualmente desconectados do grafo por causa da interferência causada por $F_3 - D_3$. Tomar o caminho 1-2-3-4-5, mesmo sendo mais longo, deixaria o enlace 7-8 livre para ser utilizado pelos outros pares origem-destino.

Neste caso, define-se que 7-8 é um enlace crítico. Enlaces críticos são, num determinado momento, enlaces que devem ser preservados. Isto porque a sua utilização

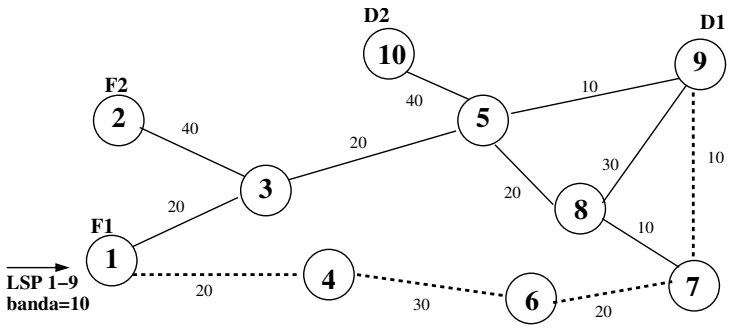

Figura 2.4. Caminho de 1 a 9 escolhido pelos algoritmos de interferência mínima.

no estabelecimento de um novo caminho resultará na diminuição do *maxflow* de um ou mais pares de entrada-saída. Os algoritmos de interferência mínima normalmente implementam processos capazes de determinar enlaces críticos que devem ser evitados para a formação de novos caminhos. Assim, podem buscar a maximização do menor valor de *maxflow* entre outros pares origem-destino, ou a maximização da soma ponderada do *maxflow* de cada par.

No cenário de exemplo mostrado na Figura 2.4, se forem considerados os pares origem-destino como 1-9 e 2-10, nota-se que a alocação dos enlaces 1-3 e 3-5 pelo MINHOP faz com que fluxo máximo do segundo par caia em 20 unidades de banda. Um algoritmo de interferência mínima normalmente escolheria o caminho 1-4-6-7-9 mostrado na Figura 2.4, que diminuiria o *maxflow* do par 2-10 em apenas 10 unidades de banda.

Os algoritmos mostrados a seguir baseiam-se nestes conceitos, mas implementam de formas diferentes o processo de determinação de enlaces críticos.

2.4.1 Minimum Interference Routing Algorithm — MIRA

A primeira proposta de algoritmo de roteamento com interferência mínima para Engenharia de Tráfego foi o *Minimum Interference Routing Algorithm* (MIRA) [Kodialam & Lakshman, 2000]. O MIRA utiliza o conceito de que o *maxflow* disponível entre um par origem-destino de uma rede está sempre associado a um conjunto de corte mínimo no grafo que representa a topologia [Cormem et al., 1990].

A primeira fase do MIRA consta em calcular o *maxflow* para cada par origem-destino conhecido da rede. É já neste ponto que surge o principal problema da abordagem. Os algoritmos para cálculo do fluxo máximo entre um par de nós numa rede apresentam

uma complexidade computacional relativamente elevada, a qual tende a aumentar ainda mais em topologias cujo grafo de representação tem muitas arestas de valência alta. Assim, calcular o maxflow para um conjunto de pares origem-destino pode ser uma tarefa bastante custosa. A especificação do MIRA recomenda a utilização do algoritmo de Goldberg-Tarjan [Goldberg & Tarjan, 1988] para o cálculo do conjunto de *maxflow*. Este algoritmo consegue reduzir a complexidade do processo de determinação do fluxo máximo, apesar desta etapa ainda continuar a ser a mais demorada do processo. O cálculo do *maxflow* permite descobrir o conjunto de corte mínimo associado ao par, e é neste conjunto de corte que estão os enlaces críticos. Cada vez que algum valor de banda é alocado em qualquer enlace no conjunto, o valor do fluxo máximo do par é reduzido. Assim o MIRA deve descobrir os conjuntos de corte mínimo de cada par origem-destino e evitar que a requisição que está sendo processada seja roteada por arestas que compõem o conjunto.

Descrevendo formalmente, o MIRA trabalha numa rede representada num grafo $G = (V, E)$ na qual existe um conjunto de pares origem destino $P(a, b)$. Para cada par $(s, d) \in P(a, b)$ é calculado o *maxflow* e então definido o conjunto de corte mínimo C_{sd} onde estão todos os enlaces críticos para o par em questão. Uma vez descobertos os conjuntos de enlaces críticos para cada par, o algoritmo deve calcular os pesos globais de cada enlace a fim de se encontrar o caminho usando o algoritmo de *Dijkstra*. Quanto mais conjuntos de corte um enlace está associado, maior deve ser seu peso. O MIRA ainda permite determinar ordens de importância para os pares origem-destino (α_{sd}), que serão os multiplicadores dos pesos. Assim o peso de cada enlace l em E pode ser determinado como:

$$w(l) = \sum_{(s,d):l \in C_{sd}} \alpha_{sd}, \qquad \forall l \in E. \tag{2.1}$$

Quando o peso de todos os enlaces são conhecidos, o algoritmo de *Dijkstra* é utilizado para encontrar o caminho. Assim, os enlaces que podem causar maior interferência são evitados devido ao seu alto peso. O número de saltos também é implicitamente controlado com a acumulação dos pesos que formam a distância total à medida que o caminho aumenta. Os resultados de experimentos mostrados em [Kodialam & Lakshman, 2000] mostram que o MIRA apresenta uma eficiência maior que

o MINHOP e o SWP, principalmente na diminuição do número de requisições bloqueadas.

2.4.2 Algoritmo de Su e Chen

O algoritmo[2] proposto em [Su & Chen, 2002] busca melhorar ainda mais os resultados conseguidos pelo MIRA. Um dos problemas detectados no MIRA foi durante a determinação da criticalidade dos enlaces. A criticalidade dos enlaces no MIRA exibe sempre valores discretos baseados no exame de cada par origem-destino separadamente. O argumento de SU&CHEN é que o valor da criticalidade também é afetada por efeitos ligados a um grupo de pares, não sendo bastante o cálculo de par em par. Assim, o MIRA pode tomar decisões de roteamento não tão precisas. Isso foi demonstrado em [Su & Chen, 2002], onde notou-se que o MIRA não tem um bom desempenho em topologias que abrigam nós concentradores e nós distribuidores.

A fase inicial de cálculo dos fluxos máximos também existe no SU&CHEN, assim como o problema da complexidade. Mas, em vez de utilizar o conjunto de corte mínimo para a determinação dos pesos, calcula-se a participação de cada enlace no *max-flow* do par. Assim, define-se f_l^{sd} como a participação do enlace l com capacidade residual $R(l)$ no fluxo máximo θ^{sd} entre o par (s, d). O cálculo dos pesos foi modificado de forma a utilizar esta participação como regra de escolha dos caminhos:

$$w(l) = \sum_{(s,d)\in P(a,b)} \frac{f_l^{sd}}{\theta^{sd} R(l)}, \qquad \forall l \in E. \tag{2.2}$$

Assim, o SU&CHEN consegue não apenas determinar enlaces críticos, mas também um valor de contribuição normalizado para as futuras requisições. Além disso, informações sobre a banda residual dos enlaces também são levadas em conta no algoritmo. Resultados de experimentos apresentados em [Su & Chen, 2002] mostram um ganho de desempenho em relação ao MIRA representado pelo número de requisições bloqueadas e fluxo máximo preservado.

[2]Este algoritmo não recebeu dos autores nenhum nome ou acrônimo específico. Para facilitar a leitura, este trabalho utiliza o termo "SU&CHEN" para se referir ao algoritmo em questão

2.4.3 Light Minimum Interference Routing — LMIR

O *Light Minimum Interference Routing* (LMIR) é um algoritmo proposto com a finalidade de reduzir a complexidade no processo de minimização da interferência durante o roteamento [Figueiredo et al., 2004, Figueiredo, 2003]. Como visto nas seções anteriores, o cálculo do *maxflow* nos diferentes pares origem-destino é o principal fator de aumento da complexidade dos algoritmos MIRA e SU&CHEN. O LMIR propõe uma abordagem de roteamento com interferência mínima que não utiliza algoritmos de *maxflow*.

O LMIR parte da idéia de que o caminho de menor capacidade entre um par origem-destino contém o enlace de menor capacidade, ou seja, o enlace crítico. Assim o algoritmo busca um certo conjunto de K caminhos com menor capacidade entre um par origem-destino e assim encontrar um certo conjunto de enlaces críticos. Como os piores caminhos são encontrados utilizando uma variação do algoritmo de *Dijkstra* (de modo semelhante ao SWP), a complexidade do processo de determinação dos enlaces críticos tem uma complexidade menor do que a dos algoritmos que calculam o *maxflow* explicitamente.

Após encontrar os K piores caminhos, todos os enlaces destes caminhos recebem pesos também de acordo com a equação 2.2, de modo que esses enlaces tenham sempre um peso mais alto para evitar de serem usados. Por fim, assim como nos outros algoritmos, estes pesos são usados como entrada para o algoritmo de *Dijktra*, que definirá o caminho a ser utilizado.

Resultados de experimentos mostrados em [Figueiredo et al., 2004], mostram que o LMIR tem um desempenho semelhante aos dos outros algoritmos de interferência mínima e apresenta uma complexidade computacional menor em relação aos mesmos.

2.5 COMPORTAMENTO DOS ALGORITMOS DE ROTEAMENTO

A análise do desempenho dos algoritmos de roteamento é objeto de constante estudo na literatura. Estes estudos visam determinar e comparar o comportamento dos algoritmos em ação analisando métricas como o número de requisições atendidas ou rejei-

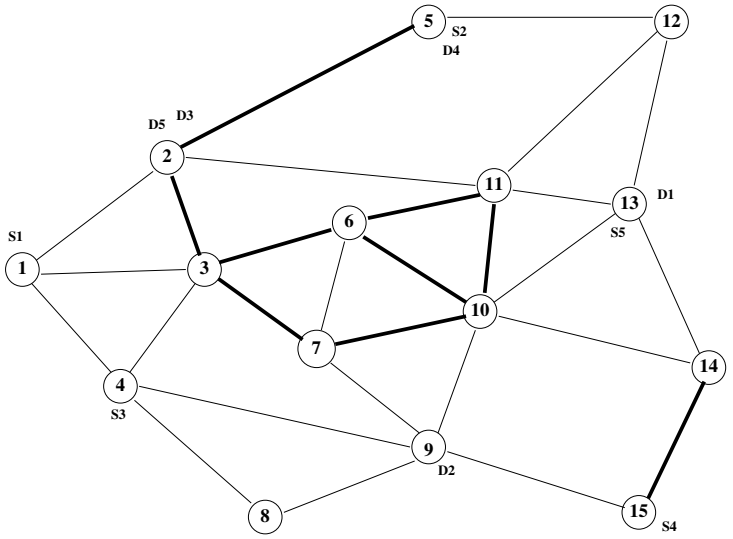

Figura 2.5. Topologia de referência usada em experimentos de algoritmos.

tadas, variação dos fluxos máximos e da interferência sofrida pelos pares origem-destino.

Os experimentos especificados nos estudos de [Kodialam & Lakshman, 2000] e [Su & Chen, 2002] muitas vezes são usados como referência para comparação dos diferentes algoritmos de roteamento. Estes experimentos usam a topologia definida na Figura 2.5. Neste grafo, os enlaces mais escuros têm capacidade inicial de 4800 unidades de banda, enquanto os enlaces mais claros têm capacidade inicial de 1200 unidades de banda. Na Figura também estão marcados os possíveis pares de origem-destino da rede em (S_1, D_1), (S_2, D_2), (S_2, D_3), (S_4, D_4) e (S_5, D_5). Estes são os pares usados pelos algoritmos de interferência mínima. Os pares também são usados para o cálculo do fluxo máximo total, e cálculo da interferência sofrida pelo roteamento.

As requisições de LSPs partem aleatoriamente dos pares marcados, com demandas de banda de 1 a 4 distribuídas uniformemente. O experimento consta em simular até 8.000 requisições para cada um dos diferentes algoritmos, de forma que eles atuam até a saturação total da rede.

A Figura 2.6 mostra uma comparação geral de todos os algoritmos mostrados neste capítulo. A figura mostra a variação do fluxo máximo total em função do estabelecimento de mais conexões. Como a manutenção do fluxo máximo aumenta a probabilidade de atendimento de novas requisições, os algoritmos que mantêm as curvas mais altas durante a alocação dos caminhos são os mais eficientes. Essa maior eficiência é comprovada observando a Figura 2.7, que mostra o crescimento no número de requisições bloqueadas.

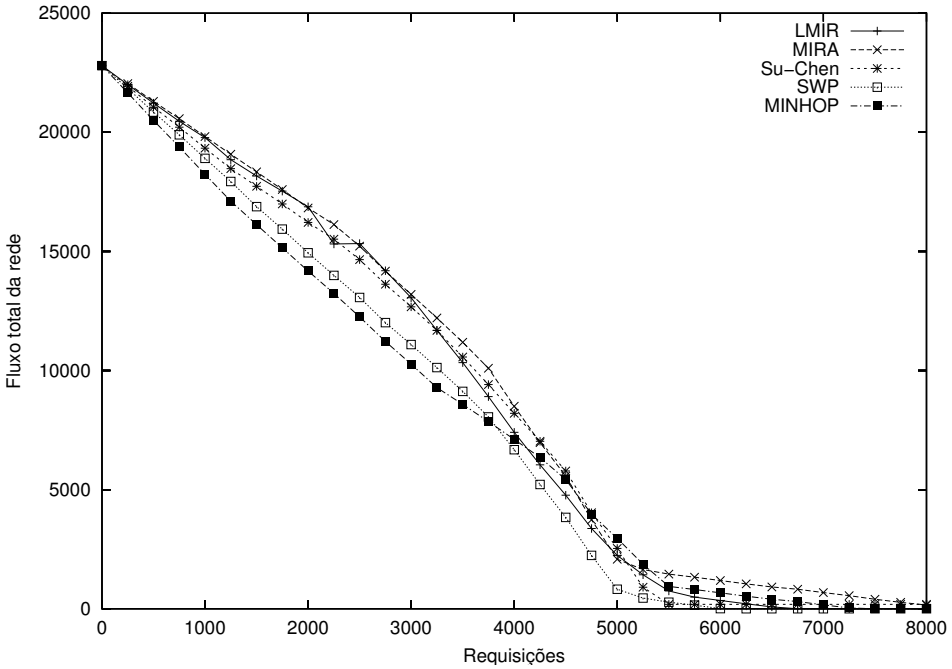

Figura 2.6. Comparação dos diferentes algoritmos [Figueiredo et al., 2004].

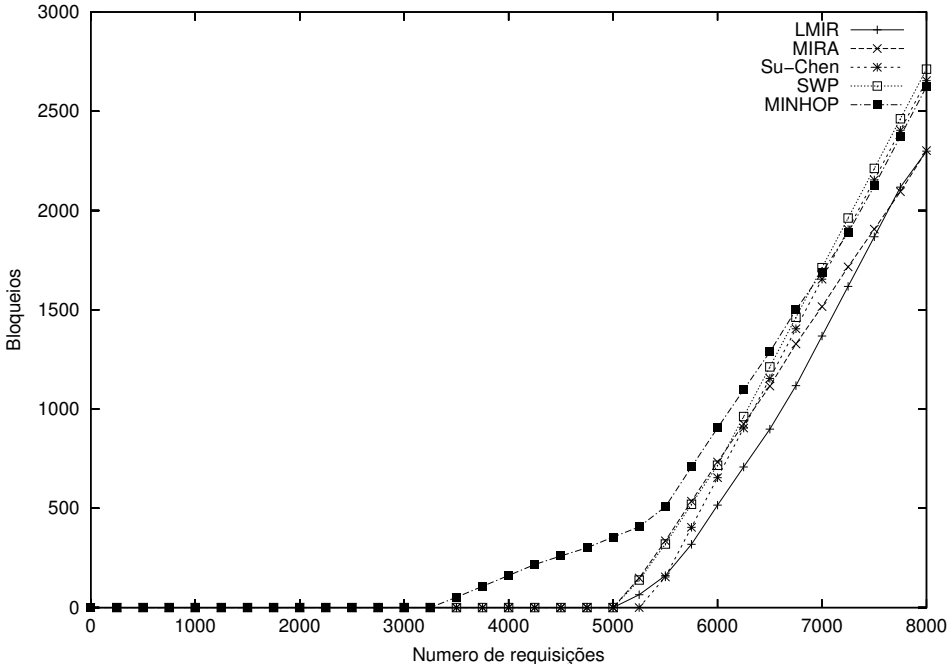

Figura 2.7. Número de Bloqueios [Figueiredo et al., 2004].

Os algoritmos que conseguem preservar mais o fluxo disponível apresentam um menor crescimento no número de bloqueios e assim têm uma maior taxa de requisições atendidas com sucesso.

Como era de se esperar, os algoritmos de roteamento com interferência mínima levam considerável vantagem em relação aos outros até a requisição 5000, quando a rede está quase completamente saturada e restam poucas opções de caminhos. O MINHOP mostra-se o mais ineficiente, pois diminui rapidamente o fluxo máximo total logo nas primeiras requisições.

Após a requisição 5000 as curvas de todos os algoritmos descem linearmente até o esgotamento total do fluxo máximo disponível. É importante observar que o fluxo é consumido totalmente porque as requisições simuladas no experimento têm demandas de banda muito pequenas, inclusive com a existência de requisições de uma única unidade de banda.

Mas, em reproduções deste experimento, nos quais o conjunto de valores destas requisições simuladas foram maiores e formaram outras possíveis combinações, os algoritmos não conseguiram aproveitar todo fluxo disponível na rede. Requisições passaram a ser permanentemente bloqueadas, mesmo quando havia fluxo disponível. A causa deste problema foi chamada neste trabalho de Fragmentação de Banda, e é especificada no capítulo seguinte.

CAPÍTULO 3

MINIMIZAÇÃO DA FRAGMENTAÇÃO DE BANDA

Neste capítulo, o problema da fragmentação de banda e seus efeitos são apresentados na seção 3.1. A seção 3.2 apresenta um método usado para detecção de potenciais de fragmentação em enlaces baseado na resolução de equações diofantinas lineares. A seção 3.3 apresenta o algoritmo MINFRAG, que utiliza o processo de detecção de potenciais de fragmentação a fim de reduzir os efeitos da fragmentação. Finalmente, a seção 3.4 apresenta uma análise da complexidade da aplicação do MINFRAG nos algoritmos de roteamento.

3.1 O PROBLEMA DA FRAGMENTAÇÃO DE BANDA

Os algoritmos mostrados nas Seções 2.3 e 2.4 mostram a constante evolução na busca por processos capazes de encontrar em pouco tempo, soluções de alto desempenho e baixa complexidade para o problema de roteamento em Engenharia de Tráfego. O problema da fragmentação de banda é uma questão não identificada no experimento de referência dos algoritmos de roteamento devido aos baixos valores das demandas de banda associadas aos LSPs e às possíveis combinações destes valores. Este capítulo apresenta o problema da fragmentação de banda, seus efeitos, e uma proposta de solução de baixa complexidade que pode ser facilmente inserida dentro dos algoritmos de roteamento.

O termo "Fragmentação de Banda" criado neste trabalho é uma analogia ao problema de fragmentação de memória, presente em sistemas operacionais que não implementam recursos de paginação [Stallings, 1998]. Nestes sistemas operacionais, a alocação e desalocação de faixas de memória contínua para os processos pode criar fragmentos de memória livre. A depender do tamanho dos processos e da ordem em que eles são criados, podem ser gerados fragmentos que nunca são grandes o bastante para serem alocados a novos processos. Esses fragmentos acabam não tendo utilidade prática dentro do

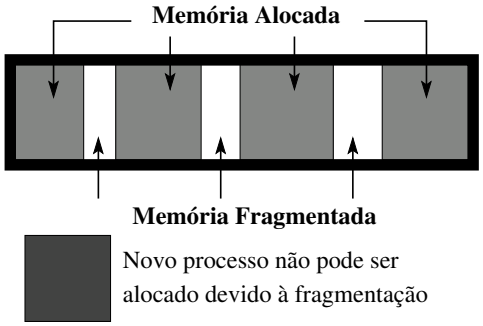

Figura 3.1. Alocação de processo negada devido à fragmentação da memória.

sistema. Isso causa uma grande degradação no desempenho do sistema. Os algoritmos de alocação tentam arrumar os novos processos de forma a diminuir o problema. Mas, quando o sistema está repleto destes fragmentos, a porcentagem de memória utilizável cai a um ponto em que o sistema começa a negar pedidos de alocação. Na situação representada na Figura 3.1, o sistema tem memória livre disponível para o processo, mas ela está tão fragmentada que não pode ser utilizada. Nesta situação, o sistema precisa rodar um algoritmo de desfragmentação da memória para rearrumar os processos na memória. Isso é um processo custoso pois interrompe a execução de todos os processos no sistema.

No contexto dos algoritmos de roteamento, a alocação de memória para os processos é uma atividade análoga à alocação de banda para os LSPs. Neste caso não há "faixas de banda" a serem alocadas, mas um conjunto de enlaces espalhados pela rede que contém os recursos a serem alocados. Mas em cada enlace, a depender da banda dos LSPs que são alocados e da ordem em que eles são estabelecidos, valores de banda residual que nunca são suficientes para serem usados por um LSP também podem ficar sem utilização. Numa rede densa, isso pode gerar uma grande quantidade de recursos que não podem ser utilizados. Assim, em certos momentos, pode existir um fluxo máximo relativamente alto entre os pares origem-destino, mas que não pode ser usado para estabelecer um LSP devido à banda residual em cada enlace ser muito pequena. O processo análogo ao da desfragmentação de memória, neste caso, seria o re-roteamento otimizado de alguns LSPs. Assim como nos sistemas operacionais, este processo de re-roteamento é lento, custoso, e causa uma interrupção momentânea no serviço.

Apesar destas semelhanças entre os dois problemas, os algoritmos têm uma ferramenta preciosa que falta aos sistemas operacionais. Graças aos ciclos do processo

de Engenharia de Tráfego que são anteriores à fase de roteamento, os algoritmos podem usar informações importantes que podem ajudar na minimização dos efeitos do problema. Uma informação importante pode ser obtida na fase inicial do processo de Engenharia de Tráfego através da coleta de dados sobre o tráfego de forma a caracterizar os fluxos que passarão pela rede. Nesta etapa, é possível determinar os tipos e demandas dos LSPs que estarão sendo requisitados para a rede, assim como se descobrem os possíveis nós origem-destino. Este conhecimento prévio sobre o conjunto de demandas de banda associadas aos LSP's que serão estabelecidos na rede durante um dos ciclos do processo de Engenharia de Tráfego é a ferramenta básica para estudar os efeitos do fenômeno da fragmentação de banda e suas possíveis soluções.

Para ilustrar de forma simples este problema, supõe-se que em uma rede corporativa, a fase inicial do processo de Engenharia de Tráfego estudou os fluxos de tráfego importantes e definiu que os LSP's a serem requisitados servirão para sessões de voz sobre IP usando o *codec* G.729 (8kbps) e sessões de vídeo em tempo real a uma taxa média de 28kbps. Neste caso, é possível observar que qualquer enlace com banda residual menor que 8kbps na rede será inútil para o estabelecimento de LSP's, pois não atenderá a mais nenhuma das possíveis demandas. Neste cenário, o somatório das bandas residuais de todos os enlaces com banda residual menor que 8kbps representa o total de banda fragmentada na rede. Este exemplo com LSP's de baixa granularidade é uma forma simples de observar o problema, porém ele pode ser detectado em qualquer rede onde o ciclo do processo de Engenharia de Tráfego coleta informações que permitem definir conjuntos de demandas associadas ao LSP's que serão estabelecidos.

Considerando a topologia simples mostrada na Figura 3.2 onde chegam requisições de 8kbps e 28kbps do nó 1 ao nó 3, e considerando que a primeira requisição é um LSP de 8kbps, é possível observar que todos os algoritmos de engenharia de tráfego descritos no capítulo 2 elegeriam o caminho 1-3 para criar o LSP. Isto aconteceria porque este é o caminho mais curto e com mais banda disponível e sem problemas de interferência. Esta escolha, deixaria o enlace 1-3 com uma banda residual de 22kbps.

Com este valor residual, o enlace suportará apenas mais dois LSP's de 8kbps e terminará com 6kbps de banda fragmentada que não pode ser usada. Além disso, se

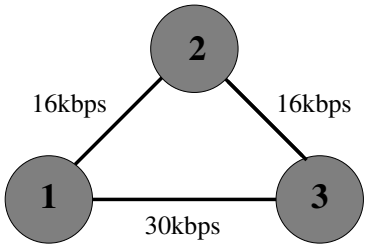

Figura 3.2. Cenário onde ocorre fragmentação de banda.

a segunda requisição for de um LSP de 28kbps, esta já será bloqueada. A rede passará a bloquear todas as requisições entre o par 1-3 quando aproximadamente 87% do fluxo máximo inicial entre estes nós for utilizado, além de ter a probabilidade de bloqueio aumentada devido à perda precoce da capacidade de estabelecer LSPs de 28kbps. Em redes mais densas, a porcentagem de fluxo máximo sem utilização por causa da fragmentação tende a ser ainda maior.

Porém, se a primeira requisição utilizasse o caminho 1-2-3, a banda residual dos enlaces 1-2 e 2-3 seria de 8kbps. Estes enlaces poderiam ainda suportar mais um LSP de 8kbps e não gerariam banda fragmentada. A combinação ideal, que causaria a menor fragmentação e maximizaria a utilização dos enlaces neste caso seria conseguir passar dois LSP's de 8kbps pelo caminho 1-2-3 e um LSP de 28kbps pelo caminho 1-3. Essa solução geraria apenas 2kbps de banda fragmentada. Assim a rede começaria a bloquear todas as requisições entre o par 1-3 somente quando aproximadamente 96% do fluxo máximo inicial entre estes nós for utilizado, diminuindo também a probabilidade de bloqueio precoce.

Como visto na Seção 2.5, o problema da fragmentação de banda não é visualizado no experimento de referência dos algoritmos de roteamento. Neste cenário com um número grande de requisições de baixos valores de demanda associados (inclusive com demandas de 1 unidade) atendidas com enlaces de alta capacidade, não há possibilidade de haver fluxo sem utilização por causa da fragmentação. Isso é visto na Figura 2.6, que mostra no final do experimento o fluxo ser lentamente consumido pelas requisições de uma unidade de banda. Porém, em muitos casos práticos, as requisições têm valores de banda maiores e em combinações como a do exemplo mostrado neste capítulo. Nestes casos, a fragmentação é um fenômeno que pode ser notado facilmente porque provoca a

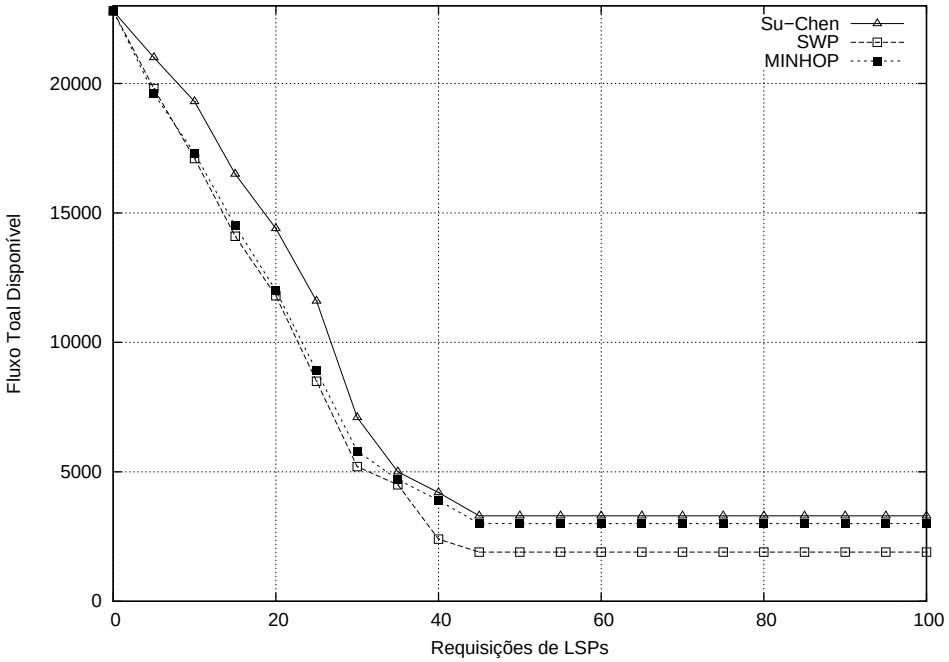

Figura 3.3. Reprodução do experimento assumindo requisições de 300 e 400.

diminuição da capacidade de utilização dos recursos da rede.

Para observar um exemplo deste fenômeno, basta simular o mesmo experimento de referência com outros valores possíveis de banda associada às requisições[1]. O gráfico da Figura 3.3 também mostra a variação do fluxo máximo total que foi observada reproduzindo o experimento de referência para alguns algoritmos, porém desta vez com as demandas das requisições assumindo os valores 300 e 400.

É possível observar que no início, as curvas dos algoritmos têm uma queda mais acentuada devido às requisições com valores bem maiores, porém neste ponto ainda são similares às curvas obtidas com o experimento original. A principal diferença está no final do gráfico. Enquanto no experimento original o fluxo é totalmente utilizado devido às requisições de uma unidade de banda, neste novo experimento observa-se que as curvas param de cair enquanto ainda há fluxo disponível. Isto indica que aproximadamente após a requisição 40, todas as novas requisições foram bloqueadas, e devido a isso o fluxo total parou de cair. Neste ponto, ainda há um fluxo total disponível de 2000 a 4000 unidades de banda entre os pares origem-destino que não pode ser utilizado porque está composto por enlaces com banda menor que 300. Neste momento o fluxo disponível está espalhado

[1]Os detalhes sobre a reprodução do experimento de referência e as implementações dos algoritmos de roteamento estão no capítulo 4.

em enlaces com banda fragmentada de modo similar ao exemplo mostrado na Figura 1.3.

Assim, fica claro que o conceito de banda fragmentada está ligado ao menor valor de requisição previsto para a rede. Pode se definir que numa rede em operação que possui um conjunto de E enlaces, onde cada enlace e possui banda residual b_e, e cujo processo de Engenharia de Tráfego determinou um conjunto S de possíveis valores de banda associados aos LSPs, a banda total fragmentada (β_{frag}) pode ser definida como:

$$\beta_{frag} = \sum_{e \in E} b_e, \qquad b_e < \min(S). \tag{3.1}$$

O conjunto de demandas S é um dado muito importante na visualização deste problema. Assim como a informação sobre possíveis pares origem-destino utilizada pelos algoritmos de interferência mínima pode ser obtida durante fase inicial do processo de Engenharia de Tráfego, a informação sobre as possíveis demandas associadas às requisições também pode ser determinada pelo estudo cuidadoso feito nesta fase. Este conjunto de demandas pode estar associado ao conjunto de aplicações que podem disparar requisições, pode estar ligado a serviços de linhas privadas virtuais com diferentes ofertas de banda, ou quaisquer outras políticas administrativas que determinem os valores das demandas. A depender dos valores contidos em S, a fragmentação pode ocorrer ou não. Quando, por exemplo, os valores neste conjunto são todos múltiplos de um mesmo número, a ordem de chegada das requisições não fará diferença na banda fragmentada gerada.

Assim, é interessante desenvolver um processo que, tendo como entrada o conjunto S, seja capaz de orientar os algoritmos de roteamento de forma a manter o valor de β_{frag} sempre o menor possível. Dessa maneira, os algoritmos podem ter um desempenho ainda melhor nas situações onde pode ocorrer a fragmentação, pois serão capazes de utilizar o máximo possível do fluxo disponível na rede e assim podem atender a um número maior de requisições. A seção a seguir, propõe um método capaz de identificar situações onde pode ocorrer fragmentação que será a base para o desenvolvimento de um processo de minimização da fragmentação total que pode ser inserido nos algoritmos de roteamento para Engenharia de Tráfego.

3.2 DETECÇÃO DO POTENCIAL DE FRAGMENTAÇÃO EM ENLACES

O potencial de fragmentação de banda em um enlace está ligado diretamente à sua banda residual b_e e ao conjunto S de valores de banda que podem ser alocados. Um enlace tem um potencial de fragmentação relativamente alto quando existem poucas ou nenhuma combinação dos valores de S cuja soma seja igual ou um pouco menor que b_e. Quando existem muitas combinações com soma igual a b_e, o potencial de fragmentação é baixo. Assim, aplicar um método de minimização da fragmentação total num algoritmo de roteamento pode ser visto como um processo de escolha de caminhos de modo que o conjunto de enlaces e bandas remanescentes tenham o menor potencial de fragmentação possível após o estabelecimento do LSP. Assim, é necessário desenvolver um método que defina o potencial de fragmentação de um enlace, dado um valor de banda residual e um conjunto de possíveis requisições.

Um caso de fragmentação de banda semelhante ao mostrado no início deste capítulo pode ser utilizado para formular o problema. Na rede em questão, requisições de 8 e 28kbps devem ser atendidas usando a banda residual dos enlaces. Num dado momento, uma requisição de 8kbps deve ser atendida. Antes do roteamento desta requisição acontecer, é possível analisar cada enlace a fim de descobrir se a alocação de mais 8kbps deixará a banda residual com um grande potencial de fragmentação. Supondo que antes do atendimento da requisição, um determinado enlace tem um valor de 38kbps de banda residual. Deseja-se então saber se ainda é possível esgotar a banda deste enlace caso sejam alocados mais 8kbps para o novo LSP. A alocação destes 8kbps deixaria o enlace com 30kbps de banda residual, podendo assim definir esta instância do problema como a descoberta da possibilidade de utilizar totalmente estes 30kbps consumindo a banda somente em partes de 8 e 28kbps. Expressando em forma matemática, o problema seria descobrir se existe uma ou mais soluções para a equação $8x + 28y = 30$ com x e $y \in N$.

Formulando o problema de forma genérica para qualquer enlace e conjunto de demandas:

Dados $a_1, a_2, \ldots, a_n \in N$ e $b \in N$

$$\exists x_1, x_2, \ldots, x_n \in N \quad | \quad a_1x_1 + a_2x_2 + \ldots + a_nx_n = b. \tag{3.2}$$

O cálculo de soluções inteiras para sistemas de equações lineares com uma ou mais incógnitas é um problema matemático clássico estudado pela Álgebra na Teoria dos Números. Os problemas semelhantes aos apresentados na equação 3.2 são conhecidos como equações diofantinas lineares [Burton, 1976].

Equações diofantinas lineares podem não ter solução. Um exemplo simples que mostra esta possibilidade é a equação $2x + 4y = 7$. Este é um caso em que não existem combinações de valores inteiros para x e y que tornem a equação verdadeira. Aplicando esse fato ao problema de fragmentação de banda, é possível afirmar que enlaces com banda residual igual a 7 estão fadados a deixar banda fragmentada caso a rede aceite apenas LSP's de tamanho 2 e 4. Assim, o problema da descoberta do potencial de fragmentação de banda em um enlace pode ser definido como o problema de determinação da existência de pelo menos uma solução dentro do conjunto de números naturais para uma determinada equação diofantina linear. Se a equação tem solução, o potencial é muito baixo ou zero. Se a equação não tem solução, o valor do potencial é alto[2].

O matemático Diofante, que propôs este tipo de problema, estudou diversos métodos para resolução destas equações dentro do conjunto de números inteiros. Para equações diofantinas lineares com duas incógnitas no formato $ax + by = c$, a condição de existência de solução para a equação é baseada na Identidade de Bezout, que mostra que a equação comporta soluções inteiras se e somente se o máximo divisor comum entre a e b divide c. Ou seja:

Dados $a, b, c \in Z$:

$$\{\exists x, y \in Z \quad | \quad ax + by = c\} \Longleftrightarrow c \bmod \text{mdc}(a, b) = 0. \tag{3.3}$$

Para fazer a analogia com o problema do roteamento, deve ser feita uma adaptação importante no enunciado. Como x e y devem representar quantidades de

[2]A determinação de um valor exato para o potencial de fragmentação não é necessária usando a abordagem de minimização proposta neste trabalho. O que importa é o desnível entre os diferentes potenciais, uma vez que estes serão usados como multiplicadores dos pesos dos enlaces no meio do processo de roteamento (Seção 3.3).

LSP's a serem estabelecidos, somente soluções em N devem ser consideradas. Assim como também são admitidos somente $a, b, c \in N$ uma vez que representam valores de largura de banda. Com essas modificações, a Identidade de Bezout pode somente certificar que não existe solução para uma determinada equação. Nas situações em que se define que existe uma ou mais soluções em Z, é preciso desenvolver um método para encontrar a solução geral e verificar se ela admite pelo menos uma solução onde x e y são naturais.

A determinação da solução geral de uma equação diofantina linear também é um problema antigo na Matemática e envolve uma extensão do famoso Algoritmo de Euclides, usado para determinar o máximo divisor comum entre dois números. O algoritmo 1 mostra o pseudo-código do Euclides Estendido, usado para encontrar uma solução específica para a equação. O algoritmo de Euclides Estendido original recebe como entrada dois números inteiros. Porém aqui também assume-se que os valores de a e $b \in N$ pois refletem valores de demanda de banda que são sempre positivos.

Algoritmo 1 Euclides Estendido

Dados $a, b \in N$ onde $a > b$

1: $m = a$, $n = b$, $r = 1$
2: Definir *passos* como uma pilha
3: **enquanto** $r \neq 0$ **faça**
4: $\quad r = m \bmod n$
5: $\quad$ Empilhar "$r = m - n(m \text{ div } n)$" em *passos*
6: $\quad m = n$
7: $\quad n = r$
8: **fim enquanto**
9: $MDC = m$
10: Desempilhar um elemento em *passos* e descartar
11: Desempilhar um elemento em *passos* e guardar em eq_1
12: **enquanto** houver elemento em *passos* **faça**
13: $\quad$ Desempilhar elemento em *passos* e guardar em eq_2
14: $\quad$ Tomar eq_1 como "$r_1 = \alpha_1 - \beta_1(\mu_1)$" e eq_2 como "$r_2 = \alpha_2 - \beta_2(\mu_2)$".
15: $\quad \alpha_n = -\alpha_2\mu_1$, $\beta_n = \beta_2$, $\mu_n = -(\mu_1\mu_2 + \frac{\alpha_1}{\alpha_1})$
16: $\quad$ Definir eq_1 como "$r_1 = \alpha_n - \beta_n(\mu_n)$"
17: **fim enquanto**
18: $x_0 = \alpha_1/a$
19: $y_0 = -\mu_1$

Os passos de 1 a 9 do Euclides Estendido são os mesmos do Euclides original, com exceção do passo 5. O objetivo dessa fase é o mesmo objetivo do Euclides original, ou seja, encontrar o mdc entre a e b. O passo 5 serve para criar uma pilha com o histórico

de todas as operações feitas para encontrar o mdc entre os dois números, com os valores que foram sendo tomados por r, m e n. Essa pilha é usada a partir do passo 10, de forma a reverter o processo dos passos iniciais para, no fim, encontrar uma equação no formato $r_n = \alpha_n - \beta_n(\mu_n)$ e assim determinar os valores x_0 e y_0. Assim, os valores importantes que devem ser computados pelo Euclides Estendido são o x_0, y_0 e o MDC.

Em uma equação diofantina linear na forma $ax+bx = c$, para a qual o Euclides Estendido encontrou um valor de MDC entre a e b que divide c, pode ser definindo:

$$w = \frac{c}{\text{mdc}(a,b)}. \tag{3.4}$$

Como o $\text{mdc}(a,b)$ divide c, w será sempre um número inteiro. Este valor é usado em conjunto com as outras saídas do Euclides Estendido para definir uma solução geral:

Dado $d = \text{mdc}(a,b)$,

$$\begin{array}{ll} x = x_0 w + (b/d)t & \text{e} \\ y = y_0 w - (a/d)t, & t \in Z. \end{array} \tag{3.5}$$

Estas equações definem a solução geral da equação diofantina, bastando variar os valores de t para que novas soluções específicas sejam geradas. Mas, como esta solução geral admite valores em Z, não pode ser aplicada diretamente ao problema de fragmentação de banda. Os valores x e y devem ser ambos maiores ou iguais a zero pois refletem quantidades de LSPs. Porém, com esta solução geral em Z é possível verificar se existem soluções em N. Para isto, basta estudar as soluções geradas com a variação do t e se em algum ponto é gerada uma solução específica em N. Um maneira simples de verificar a existência de soluções em N é descobrir o maior valor que t pode assumir sem que o y gerado seja negativo. Este t máximo se encontra definindo:

$$t_{max} = \left\lfloor \frac{y_0 w}{b/d} \right\rfloor. \tag{3.6}$$

Se este valor de t_{max} encontrado para y e aplicado à solução geral de x não puder gerar um resultado não negativo, então nenhum outro valor de t conseguirá. Se o valor de t for aumentado acima de t_{max} a fim de encontrar um x não negativo, acaba-se obtendo um y negativo e invalidando a solução. Neste caso, conclui-se que não existe

solução em N e portanto, o enlace sendo analisado terá um alto potencial de fragmentação caso fique com um valor de banda residual igual a c. Quando existe solução em N, entende-se que existe ainda a probabilidade de utilizar todo o enlace sem perigo eminente de ocorrer fragmentação no futuro.

O valor final do potencial de fragmentação não precisa ser uma medida rígida para ser aplicado no processo de minimização proposto neste trabalho. É possível, por exemplo, definir o potencial como zero quando existe pelo menos uma solução para a equação em N. Se $c \bmod \text{mdc}(a,b) \neq 0$, o potencial de fragmentação pode ser tomado de forma geral como $c \bmod \text{mdc}(a,b)$. Se $c \bmod \text{mdc}(a,b) = 0$ e a equação diofantina não tiver solução em N, o potencial de fragmentação pode ser tomado de forma geral como $\min(a \bmod c, b \bmod c)$. Os valores de potencial calculados dessa maneira já são suficientes para encontrar as diferenças necessárias para o processo de minimização da fragmentação. Outras formas de refinamento do valor de potencial podem ser utilizadas. Por exemplo, o algoritmo de Euclides poderia ser rodado novamente para definir o maior valor abaixo de c que não gera fragmentação e assim determinar um potencial mais preciso. É possível definir potenciais um pouco maiores que zero quando existem poucas soluções em N para a equação diofantina. Mas, estas ações aumentariam a complexidade total do processo e não contribuiriam significativamente na melhoria dos resultados.

O método mostrado acima encontra soluções para equações de apenas duas incógnitas. Mas, a existência de soluções para equações diofantinas com mais incógnitas também pode ser determinada. Como é normal com os problemas algébricos, a existência de solução para uma equação diofantina com n variáveis ($n > 2$) pode depender de apenas $n-1$ destas variáveis [Ikenaga, 2003]. Numa equação com 3 incógnitas no formato $ax + by + cz = d$ é possível fatorar as duas primeiras incógnitas:

$$(a,b)\left(\frac{a}{(a,b)}x + \frac{b}{(a,b)}y\right) + cz = d. \tag{3.7}$$

Desta fatoração é possível definir o novo termo w:

$$w = \left(\frac{a}{(a,b)} x + \frac{b}{(a,b)} y \right). \tag{3.8}$$

E este termo pode então reduzir a equação para:

$$(a,b)w + cz = d. \tag{3.9}$$

A solução da equação 3.9 pode ser verificada com o Euclides Estendido. A equação da solução geral de w deve então ser expandida para os valores originais de x e y, encontrando-se uma nova equação diofantina com duas incógnitas.

O crescimento do número de incógnitas aumenta também a complexidade e o tempo total de definição do potencial de fragmentação. Mas, nem sempre é preciso adicionar todos os possíveis valores de LSPs na equação. Valores de demanda que são múltiplos de outros possíveis valores podem ser descartados pois não farão diferença no problema geral da fragmentação, assim como LSPs que aparecem com pouca frequência. De maneira geral, os valores mais importantes são os menores de demanda não múltiplos entre si. Fica clara a importância do estudo do conjunto de demandas. Este estudo deve definir uma equação diofantina linear modelo para cálculo dos potenciais de fragmentação, ou pode até mesmo decidir que a minimização da fragmentação não poderá ajudar muito no processo. Este estudo deve ser embutido dentro das primeiras fases do ciclo do processo de Engenharia de Tráfego e será a entrada principal para o método de minimização da fragmentação de banda.

Assim, é possível definir potenciais de fragmentação para todos os enlaces da rede usando a resolução das equações diofantinas lineares. A próxima seção mostra como a determinação destes potenciais podem ser usadas na aplicação de mudanças nos pesos dos enlaces, constituindo o processo de minimização da fragmentação.

3.3 O ALGORITMO MINFRAG

Como visto na seção anterior, os estudo das demandas e posterior definição de uma equação diofantina linear modelo são os pré-requisitos para a aplicação do processo de minimização da fragmentação. O processo de minimização da fragmentação (MINFRAG) proposto neste trabalho consiste no cálculo do potencial de fragmentação para cada enlace, e posterior utilização de cada valor calculado como fator de multiplicação de pesos pré-calculados que serão utilizados como entrada para o algoritmo de *Dijkstra*.

Como visto no Capítulo 2, muitos algoritmos de roteamento têm uma fase inicial que consta da aplicação de certos métodos que definem um conjunto de pesos para todos os enlaces da rede. Estes pesos são calculados levando em conta diferentes critérios, a fim de que possam ser utilizados numa fase final que é a definição final do caminho pelo algoritmo de *Dijkstra*. A proposta do MINFRAG é atuar como uma fase intermediária do processo de roteamento, alterando os pesos calculados por um algoritmo de roteamento quando o potencial de fragmentação de alguns enlaces for relativamente alto.

Usando uma equação diofantina linear modelo reduzida para duas incógnitas, o processo completo de minimização da fragmentação pode ser definido como:

O passo 1 do processo é a utilização de um dos algoritmos de roteamento para a determinação de um conjunto inicial de pesos. Por exemplo, um algoritmo de interferência mínima pode ser utilizado neste ponto para encontrar um conjunto de pesos inicial associado às criticalidades dos enlaces. No caso de MINHOP, todos estes pesos iniciais seriam zero.

O passo 2 consiste do cálculo do somatório dos pesos iniciais que serão o fator de multiplicação dos potenciais de fragmentação. Um valor alto é importante para que os enlaces com possibilidade de fragmentação tenham seus pesos explodidos e assim tenham menor possibilidade de serem utilizados.

O laço iniciado no passo 3 é usado para processar os potenciais de todos os enlaces com banda suficiente para serem utilizados no novo LSP. Para cada um dos enlaces, calcula-se o valor de banda residual (res) caso fosse feita a alocação da demanda d. Se o res é maior que 0 e menor que o menor valor de LSP, observa-se claramente que

Algoritmo 2 MINFRAG

Dados $a, b \in N(a > b)$ como os valores de LSP's e d como a nova demanda a rotear

1: Calcular $peso_n$ para os n enlaces a depender do critério de um algoritmo de roteamento
2: $M = \sum peso_i$
3: **para todo** enlace e na rede no qual $banda_e \geq d$ **faça**
4: $\rho = 0$, $res = d - banda_e$
5: **se** $res < b$ **e** $res > 0$ **então**
6: $\rho = res$
7: **senão**
8: Encontrar $d = \text{mdc}(a, b), x_0, y_0$ chamando Euclides Estendido
9: **se** $res \bmod d \neq 0$ **então**
10: $\rho = (res \bmod d)$
11: **senão**
12: $t_{max} = \left\lfloor \frac{y_0 w}{b/d} \right\rfloor$
13: **se** $(x_0 w + (b/d)t_{max}) < 0$ **então**
14: $\rho = \min(a \bmod c, b \bmod c)$
15: **fim se**
16: **fim se**
17: **fim se**
18: $peso_e = peso_e + M\rho$
19: **fim para**
20: Executar o algoritmo de Dijkstra usando $peso_n$ para cada enlace n

a utilização do enlace gerará banda fragmentada. A variável ρ é a que guarda o valor potencial de fragmentação. Caso a fragmentação não seja evidente, usa-se o Euclides Estendido para encontrar os dados para resolução da equação diofantina modelo. Neste ponto é importante notar que se a equação tiver n incógnitas (com $n > 2$), os passos de 8 a 16 devem ser rodados $n - 1$ vezes para a redução da equação usando o método de fatoração definido nas equações 3.7, 3.8 e 3.9. A determinação do potencial neste caso ocorre nos passos 9 até 16, seguindo as regras definidas na seção 3.2 para a resolução das equações em N usando o t_{max}.

No passo 18, o potencial ρ encontrado para o enlace é multiplicado ao somatório dos pesos e incrementado no peso correspondente do enlace que foi calculado inicialmente pelo algoritmo de roteamento. Assim, observa-se que enlaces com potencial ρ iguais a zero não têm seus pesos modificados. Enlaces com potenciais de fragmentação maiores que zero têm seus pesos bastante aumentados de formar a evitar ao máximo a sua escolha como integrante do novo caminho. O fator de multiplicação M pode ser

ajustado com outros valores (menores ou maiores que o somatório dos pesos iniciais) com o objetivo de determinar uma maior ou menor interferência da minimização da fragmentação nos critérios iniciais de escolha dos algoritmos de roteamento. A preservação destes critérios iniciais dos algoritmos de roteamento é maior quando os enlaces têm muita banda e conseqüentemente têm menores probabilidades de fragmentação devido a uma grande possibilidade de combinações de LSPs. À medida em que esta banda residual começa a diminuir, os potenciais calculados passam aos poucos a modificar algumas decisões que seriam corretas segundo o critério dos algoritmos, mas que acabariam gerando mais banda fragmentada.

Enfim, no passo 20 é executado o algoritmo de *Dijkstra* sendo passada a matriz de pesos modificada pelo processo de minimização da fragmentação. O algoritmo usará a matriz para encontrar o caminho usando os enlaces sugeridos pelo algoritmo de roteamento, porém evitando aqueles que tiveram o peso aumentado devido à ação do processo de minimização da fragmentação. Mesmo se não houver caminhos alternativos com potencial de fragmentação igual a zero, o roteamento não deixa de ser efetuado. Neste caso os prejuízos com a fragmentação continuam. A aplicação contínua deste processo tende a alocar os recursos dos enlaces de modo a que toda a sua banda possa ser consumida.

3.4 COMPLEXIDADE DO ALGORITMO MINFRAG

Definir o aumento na complexidade computacional do roteamento que acontece com a aplicação do processo de minimização da fragmentação de banda é uma etapa importante devido à necessidade de um baixo tempo de resposta no estabelecimento das conexões durante a operação da rede.

Para o processo de minimização não interferir significativamente no tempo de resposta do roteamento, basta que a sua complexidade seja menor que a complexidade total do algoritmo de roteamento que está sendo usado. A complexidade dos algoritmos de roteamento simples mostrados na seção 2.3 é praticamente a mesma do algoritmo de *Dijkstra*, já que quase todo os processos são baseados apenas nele. Definindo que os algoritmos trabalham na rede como um grafo de V vértices e E arestas, pode se afirmar que

a complexidade do algoritmo de *Dijkstra* é $O(V^2 + E)$ [Dijkstra, 1959]. Os algoritmos de roteamento com interferência mínima têm na etapa de cálculo do *maxflow*, o maior fator de aumento na complexidade. A depender do algoritmo usado e da relação entre o número de arestas e o número de nós essa complexidade varia, mantendo-se porém sempre acima da complexidade do algoritmo de *Dijkstra*. A implementação do algoritmo de *maxflow* de *Edmonds-Karp* apresenta complexidade de $O(V.E^2)$ [Cormem et al., 1990]. Em redes com uma maior proporção de enlaces em relação aos nós (como a Internet), percebe-se que esta complexidade é bem maior que a do algoritmo de *Dijkstra*. O algoritmo de Goldberg, que é uma implementação mais eficiente no tempo de cálculo do *maxflow*, apresenta uma complexidade $O(\min(V^{\frac{2}{3}}, E^{\frac{1}{2}}).E.\log(\frac{V^2}{E}).\log(U))$ numa rede com capacidades no intervalo $[1, U]$ [Goldberg & Tarjan, 1988]. Esta complexidade ainda supera a do algoritmo de *Dijkstra*. O algoritmo LMIR é um caso particular que não executa cálculo de *maxflow*, usando um processo de busca de caminhos de menor capacidade que apresenta complexidade $O(V^2)$[Figueiredo et al., 2004].

A complexidade do MINFRAG está ligada ao cálculo do potencial de fragmentação para cada enlace E da rede. O algoritmo de cálculo do potencial de fragmentação de um enlace tem a mesma complexidade do algoritmo de Euclides Estendido. Em Euclides, para $b \leq a$, o número máximo de divisões que ocorre é $O(\log(b))$ [Cormem et al., 1990]. Quando há n incógnitas na equação diofantina, no pior caso, o algoritmo de Euclides deve ser rodado $n - 1$ vezes para decidir se existe ou não solução para a equação. Assim, pode ser definida a complexidade total do processo como $O(E.(n-1).\log(b))$. Comparando a complexidade do processo de minimização da fragmentação com a complexidade total dos algoritmos de roteamento, é possível perceber que o MINFRAG não prejudica consideravelmente o tempo total do processo de roteamento. Isso possibilita a real aplicação do processo de minimização da fragmentação em esquemas de roteamento online.

O capítulo seguinte apresenta os resultados de experimentos que mostram os efeitos da aplicação deste método de baixa complexidade durante a execução dos algoritmos de roteamento.

CAPÍTULO 4

RESULTADOS EXPERIMENTAIS

Este capítulo apresenta resultados de experimentos realizados usando a implementação dos algoritmos de roteamento e processos mostrados nos capítulos anteriores. A seção 4.1 apresenta os objetivos gerais da realização destes experimentos. A seção 4.2 apresenta todos os detalhes referentes à implementação dos algoritmos de roteamento e do processo de minimização da fragmentação usando a linguagem de *script* interpretada pelo simulador. Finalmente, a seção 4.3 mostra os detalhes referentes aos experimentos realizados, além de apresentar e analisar os resultados obtidos nestes experimentos.

4.1 OBJETIVOS DOS EXPERIMENTOS

Os efeitos práticos da aplicação do processo de minimização da fragmentação podem ser demonstrados através da simulação de requisições de LSPs numa rede que são roteadas através dos algoritmos mostrados no capítulo 2 com e sem a aplicação do MINFRAG.

O objetivo destes experimentos é recolher informações sobre o desempenho do roteamento das sucessivas requisições e assim determinar quais as melhores opções de utilização. As medidas de interesse normalmente são a utilização do fluxo máximo, número de requisições atendidas e bloqueadas, ou também a quantidade de interferência sofrida por pares de origem-destino a cada requisição.

A curva de utilização de fluxo máximo mostra a velocidade com que um algoritmo consome os recursos utilizáveis da rede para fazer o atendimento das requisições. Para a determinação dos valores desta medida, são calculados os valores de *maxflow* para um conjunto pré-determinado de pares origem-destino. O fluxo total é o somatório dos valores de *maxflow* de cada um destes pares. Os algoritmos de interferência mínima introduziram a leitura dessa medida a fim de mostrar como a determinação e preservação

de enlaces críticos pode aumentar a capacidade de atendimento de mais requisições, uma vez que os fluxos máximos entre os pares origem-destino são preservados o máximo possível e assim os recursos são consumidos mais lentamente. Porém, a leitura da curva de utilização do fluxo máximo deve ser interpretada observando também outras medidas. Quando a rede tem recursos de sobra, a utilização acentuada do fluxo é prejudicial. Mas, à medida em que os recursos vão ficando escassos, algumas requisições começam a ser bloqueadas. Quando uma requisição é bloqueada o algoritmo não consome nada do fluxo. Assim, quando a taxa de requisições bloqueadas começa a aumentar, a curva de utilização começa a suavizar até atingir um limite no qual torna-se horizontal.

As figuras 2.6 e 3.3 mostram curvas de utilização de fluxos em diferentes situações. Nas duas figuras, observa-se que os algoritmos de interferência mínima conseguem preservar mais o fluxo até que num determinado ponto as curvas se encontram e mudam de comportamento. É neste ponto em que o bloqueio de requisições começa a modificar a taxa de utilização do fluxo máximo. Foi argumentado no capítulo 3 que a diferença nos finais entre as curvas das duas figuras ocorre devido aos efeitos da fragmentação. Na Figura 2.6 fica claro que o fluxo deixa de ser utilizado devido aos bloqueios decorrentes à saturação quase completa da rede. Porém, foi observado na Figura 3.3 que a curva de utilização fica horizontal antes da saturação. Os motivos da parada da curva continuam sendo os bloqueios, porém neste caso eles ocorrem antes da saturação, devido ao problema da fragmentação de banda. Assim, uma das formas de avaliação do sucesso do método de minimização da fragmentação seria obter neste mesmo experimento, curvas de utilização de fluxo com um comportamento semelhante na parte inicial, mas que apresentem o padrão horizontal de constantes bloqueios bastante próximo do ponto de saturação da rede.

O número total de requisições atendidas e bloqueadas também é uma medida muito importante, pois mostra a capacidade do algoritmo arrumar corretamente os caminhos de modo a conseguir obter o máximo possível de êxito nas tentativas de estabelecimento de LSPs. Além do número total, é interessante que o algoritmo atrase o máximo possível o momento em que ocorre o primeiro bloqueio. Mais uma vez, os algoritmos de roteamento com interferência mínima são mais eficientes nesta medida devido

à preservação dos fluxos. O aumento do número total de requisições atendidas também é uma medida de referência para o processo de minimização da fragmentação. Uma vez que o MINFRAG tenta fazer com que toda a banda dos enlaces possa ser utilizada, sobram mais recursos capazes de serem utilizados para o estabelecimento de mais LSPs.

As seções a seguir mostram alguns dados obtidos para estas e outras medidas a partir de simulações do processo de minimização da fragmentação. Estes dados mostram os efeitos que a aplicação do processo pode exercer nos valores obtidos com a simulação dos algoritmos sem a minimização da fragmentação.

4.2 IMPLEMENTAÇÃO DOS ALGORITMOS NO SIMULADOR NS2

Os experimentos foram feitos utilizando o simulador *Network Simulator 2* (*ns2*) [Fall & Varandhan, 2003]. O *ns2* é um simulador conhecido pela precisão e largamente utilizado em experimentos científicos dos mais variados na área das redes de computadores. A interface amigável para a criação de scripts de simulação através de uma versão orientada a objetos da linguagem Tcl também é outro grande atrativo para a utilização do simulador.

Como o suporte ao MPLS da versão utilizada do *ns2* ainda não é completo, foi utilizado em conjunto o pacote *MPLS Network Simulator* (*mns*) [Ahn, 2001] que disponibiliza funções do MPLS mais completas, como a implementação do CR-LDP. Porém, esta versão não implementa quase nenhum dos algoritmos de roteamento mostrados no capítulo 2. Assim, apesar das simulações dos algoritmos de roteamento não envolverem aspectos de um sistema de filas, a utilização do *ns2* com o *mns* permite a utilização de recursos de programação mais simples e também possibilita uma contribuição importante através da implementação e disponibilização dos algoritmos de roteamento que faltam nos pacotes originais, além do processo de minimização da fragmentação de banda embutido nestes algoritmos.

4.2.1 Algoritmos de roteamento

A criação de extensões para o pacote *mns* envolve a adição de métodos às classes dos objetos usados pela linguagem de simulação. O estudo destes objetos e das estruturas usadas no pacote original possibilitou a implementação dos algoritmos *Min-Hop Algorithm* (MHA), a versão do algoritmo de interferência mínima de [Su & Chen, 2002] (Su-Chen) e o *Widest-Shortest Path* (WSP).

A implementação do algoritmo de *Dijkstra* feita em Tcl foi a base para a implementação de todos os algoritmos de roteamento. O pacote *mns* disponibiliza uma estrutura de dados que pode ser usada para montar uma matriz de n x n nós, que contém as bandas residuais nos enlaces que ligam cada um dos nós aos seus respectivos vizinhos. Cada um dos algoritmos trabalha sobre esta matriz a fim de produzir uma segunda matriz com as mesmas dimensões, que deverá guardar os pesos dos enlaces. A implementação do MINHOP, por exemplo, simplesmente verifica em cada célula da matriz inicial se existe enlace com banda residual suficiente entre o nó da linha e o nó da coluna. Caso exista, a célula equivalente na matriz de pesos é marcada com o número 1. Caso contrário é marcada com o número -1. Já a implementação do Su-Chen usa a matriz inicial para rodar o algoritmo de *maxflow* em cada par de entrada e saída e por fim gerar a matriz de pesos onde cada célula recebe um valor de acordo com a fórmula 2.2, ou recebe -1 caso não haja enlace entre o nó da linha e o nó da coluna. Um método adicionado na classe do objeto simulador do *ns2* permite definir quais serão os pares origem-destino usados pelos cálculos dos algoritmos. A ação do WSP é semelhante, porém com as células da matriz de pesos recebendo o inverso da capacidade residual do enlace. A fase final de todas as implementações é a execução do algoritmo de *Dijkstra* com o objetivo de encontrar o caminho final.

A implementação do algoritmo de *Dijkstra* usa a matriz calculada pelos algoritmos (descartando as células com valor -1) para computar o caminho de menor distância de acordo com os pesos. Alguns algoritmos, como o WSP, usam uma outra opção de implementação que computa o caminho cujo enlace de maior peso tem o menor valor.

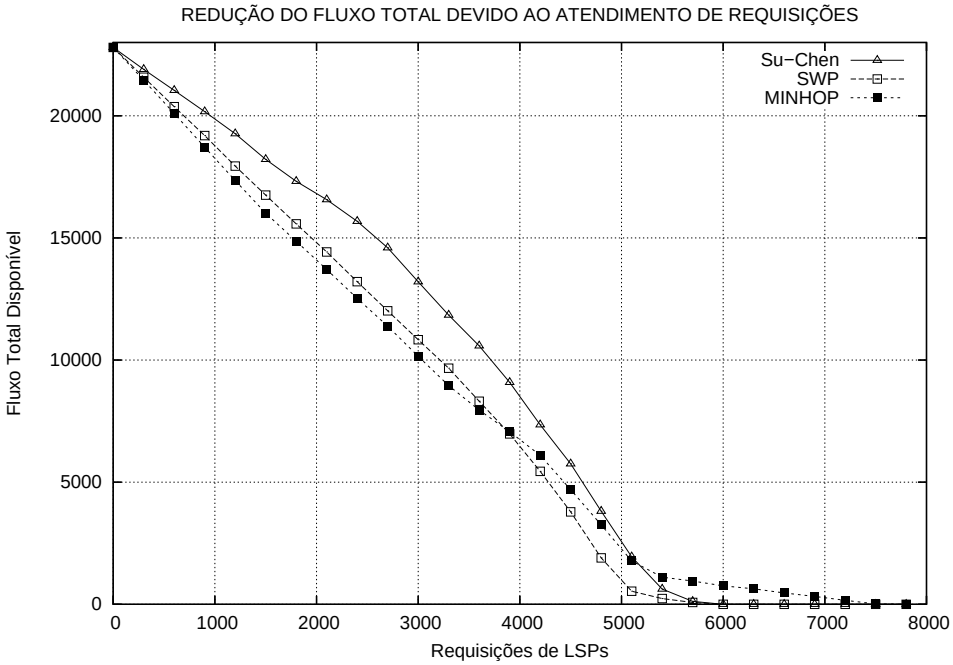

Figura 4.1. Gráfico de fluxo gerado pelos algoritmos implementados no *ns2*.

4.2.2 Validação dos resultados

Com o objetivo de validar as implementações dos algoritmos de roteamento, o experimento de referência especificado em [Su & Chen, 2002] foi reproduzido. A mesma topologia foi reconstituída usando *scripts* em Tcl para o *ns2*. A simulação de 8000 requisições foi feita usando as mesmas especificações do experimento de referência. Os algoritmos implementados (MINHOP, WSP e Su-Chen) foram usados para atender as requisições e os resultados foram coletados a fim de serem comparados com o experimento original. Estes resultados geraram o gráfico mostrado na Figura 4.1. Estas curvas refletem exatamente os comportamentos obtidos no experimento original em [Su & Chen, 2002] que também foram obtidos em [Figueiredo et al., 2004] como mostra a Figura 2.6. Mais detalhes sobre estes resultados obtidos encontram-se na Tabela 4.1

Essa validação é importante pois garante que o experimento seguinte realmente mostre os reais efeitos da fragmentação sobre os algoritmos implementados. O resultado deste experimento (usando requisições de 300 e 400 unidades de banda) foi mostrado na Figura 3.3, na qual se observa a impossibilidade de utilizar todo o fluxo disponível devido à ocorrência da fragmentação.

Com esta comparação também será possível validar os efeitos da aplicação do

Tabela 4.1. Resultados da reprodução do experimento de Referência

Algoritmo	Bloqueios	Início dos Bloqueios
MINHOP	2696	após 3600 requisições
WSP	2763	após 5100 requisições
Su-Chen	2524	após 5400 requisições

método de minimização da fragmentação implementado com o objetivo de melhorar o desempenho dos algoritmos nas situações em que a banda residual fica sem utilização.

4.2.3 Implementação do MINFRAG

O primeiro passo da implementação do MINFRAG foi a adição de um método à classe do objeto simulador que permite definir quais serão os tamanhos dos LSPs usados para construir a equação diofantina modelo que determina as possibilidades de fragmentação nos enlaces.

Como mostrado no capítulo 3, o MINFRAG não é um algoritmo de roteamento à parte. Para possibilitar sua utilização na simulação, as chamadas aos algoritmos de roteamento implementados foram modificadas de forma a receber como parâmetro de entrada a opção de utilização ou não da minimização da fragmentação durante o processamento. No fim da implementação de cada algoritmo de roteamento, esta opção é checada para que o MINFRAG possa entrar em ação. Assim como mostra o passo 1 no Algoritmo 2, o MINFRAG receberá a matriz de pesos calculada pelo algoritmo de roteamento como uma das entradas.

O MINFRAG checa as possibilidades de fragmentação para cada enlace e modifica os pesos de acordo com esta possibilidade. Quando a banda residual que ficaria após o roteamento é maior que zero e menor que o menor tamanho de requisição declarado, a alta possibilidade de fragmentação é rapidamente encontrada. Caso contrário, é preciso checar essa probabilidade através do Euclides Estendido.

O Euclides Estendido foi implementado segundo o pseudo-código mostrado no Algoritmo 1. A implementação usada nos experimentos permite encontrar possíveis soluções inteiras para equações diofantinas lineares com até três incógnitas . A implementação deste algoritmo e dos algoritmos de roteamento na linguagem Tcl encontra-se no Apêndice A.

O MINFRAG usa a solução geral do Euclides Estendido para verificar através do t_{max} (equação 3.6) a existência de possíveis soluções no conjunto dos números naturais. Assim, cada um dos pesos é modificado ou mantido de acordo com a probabilidade de fragmentação. Depois desse processamento, o controle do processamento volta ao algoritmo de roteamento que executa sua tarefa final: a execução do algoritmo de *Dijkstra*.

Assim, é possível construir experimentos que comparam a ação dos mesmos algoritmos usando ou não o método de minimização da fragmentação de modo a mostrar os possíveis ganhos de desempenho que a aplicação do MINFRAG pode trazer em diferentes casos de utilização.

4.3 EXPERIMENTOS

Os experimentos realizados visaram comparar o desempenho dos algoritmos de roteamento (Su-Chen e MINHOP) em cenários onde o conjunto de requisições provoca fragmentação de banda, utilizando ou não o processo de minimização da fragmentação.

Assim como feito nos experimentos de validação das implementações mostrados na seção 4.2.2, a configuração partiu do experimento de referência usado na comparação dos algoritmos de roteamento. A topologia usada é mesma mostrada na Figura 2.5, também com os enlaces escuros contendo 4800 unidades de banda e os enlaces claros contendo 1200 unidades de banda. Os mesmos pares origem-destino (S_1, D_1), (S_2, D_2), (S_2, D_3), (S_4, D_4) e (S_5, D_5) também foram usados como referência para as requisições e para o cálculo do fluxo total disponível. A distribuição das requisições entre estes pares também foi aleatória.

A principal diferença em relação ao experimento de referência foi o conjunto de demandas associadas aos LSPs. Como visto na seção 2.5, no experimento de referência as requisições assumem valores relativamente muito pequenos em relação à banda inicial dos enlaces. Quando se assume esse conjunto de valores (1, 2, 3 e 4) para as demandas, não existe a possibilidade de ocorrer fragmentação.

Para observar os efeitos da fragmentação e do processo de minimização, novos conjuntos de demandas foram definidos com valores maiores em relação à banda inicial

e em combinações que podem gerar altos índices de fragmentação. Um conjunto de requisições foi definido com demandas de 300 e 400, e um outro com requisições de 300, 400 e 500. Assim, foi possível testar a aplicação da implementação do processo usando equações diofantinas lineares de modelo com 2 e 3 incógnitas. Estas demandas são distribuídas uniformemente entre os cinco pares origem-destino. Com esses valores relativamente maiores para o tamanho das demandas, o número total de requisições do experimento também foi reduzido pois a rede se esgota mais rapidamente.

Um dos objetivos destes experimentos foi observar a curva de utilização do fluxo máximo, de modo a descobrir se o processo de redução da fragmentação possibilitaria aos algoritmos consumir todo o fluxo disponível. Isso leva a outros pontos de observação importantes como a observação do número de requisições bloqueadas e momento de início dessas requisições usando ou não o processo de minimização da fragmentação.

A medida apresentada como a banda fragmentada total β_{frag} na equação 3.1 também deve ser observada com o objetivo de analisar a eficiência bruta da aplicação do MINFRAG na redução da fragmentação causada pelos algoritmos de roteamento.

Os experimentos foram implementados na versão 2.1b6 do *ns*. É nesta versão que funciona o *mns* v2.0, a qual inclui os pacotes para simular redes MPLS suportando o CR-LDP e roteamento explícito através de ER-LSPs (*Explicit Routed LSPs*). Os algoritmos de roteamento implementados podem encontrar os caminhos e estes caminhos são usados para estabelecer estes ER-LSPs.As simulações foram executadas numa máquina rodando o sistema operacional Linux versão 2.4.18-14.

4.3.1 Experimento com 2 incógnitas

O experimento no qual as requisições assumem as demandas 300 e 400 permitem usar o processo de minimização da fragmentação de banda através de uma equação diofantina linear modelo com duas incógnitas. Para cada enlace a ser analisado no momento do roteamento, o processamento do MINFRAG será: "Dada a banda residual b_e do enlace e e um LSP associado a uma demanda d que deve ser estabelecido, definir se existe alguma solução em N para a equação diofantina $300x + 400y = (b_e - d)$."

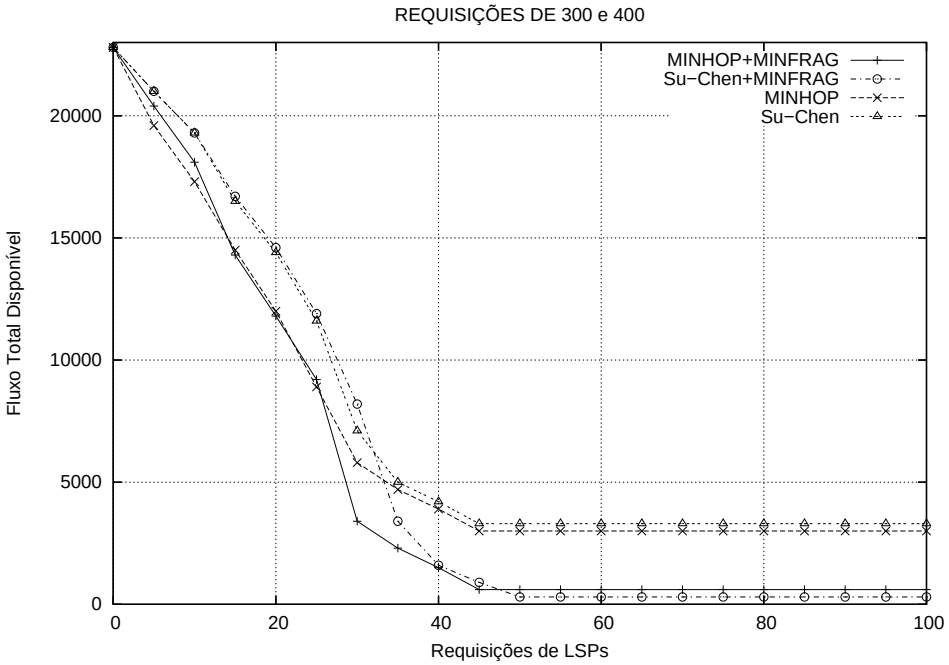

Figura 4.2. Redução do fluxo.

A curva mostrada na Figura 4.2 mostra a redução sofrida no fluxo máximo total entre os pares origem-destino à medida em que as requisições chegam na rede.

A análise desta figura permite algumas observações interessantes. O primeiro ponto importante a se destacar é a similaridade que se nota na parte inicial do gráfico entre curvas dos algoritmos originais e suas versões modificadas com o MINFRAG. Este comportamento é importante pois mostra que o MINFRAG não perturba muito os critérios de escolha iniciais dos algoritmos de roteamento quando há muita banda disponível e menores probabilidades de se detectar altos índices de fragmentação nos enlaces. Estas menores probabilidades fazem com que nas primeiras requisições o MINFRAG modifique muito pouco as matrizes de pesos construídas pelos algoritmos de roteamento. Assim, fica claro que não é apenas o MINFRAG que determinará o desempenho total do processo de roteamento. A melhor forma de comparar o desempenho do processo de minimização da fragmentação seria comparar os resultados do algoritmo inicial com a sua versão modificada pelo MINFRAG.

A atuação do MINFRAG começa a ser realmente notada aproximadamente na chegada da trigésima requisição. Neste ponto as curvas dos algoritmos originais e suas versões com o MINFRAG começam a se separar. Nos algoritmos originais, a queda

Tabela 4.2. Comparativo da utilização do fluxo disponível na rede

Algoritmo	**Fluxo Utilizado**	**Aproveitamento**
MINHOP	19800	86%
Su-Chen	19500	85%
MINHOP+MINFRAG	22200	97%
Su-Chen+MINFRAG	22500	98%

das curvas começa a ser suavizada e torna-se horizontal com um fluxo restante entre 3000 e 4000. Como argumentado no capítulo 3, esta mudança nas curvas ocorre devido ao crescimento na taxa de bloqueios que faz com que a taxa de utilização dos fluxos também caia até que nenhuma nova requisição possa ser atendida. Mas, observa-se que esta suavização da queda começa mais tarde nas versões dos algoritmos que usam o MINFRAG. Além disso, observa-se claramente que o gráfico assume o comportamento horizontal num nível em que o fluxo disponível é quase zero. Este comportamento encaixa-se perfeitamente com a ação executada pelo MINFRAG. Devido ao fato de evitar ao máximo que os enlaces gerem banda fragmentada, o MINFRAG faz com que haja mais opções de roteamento quando a banda está escassa em toda a rede. Isso faz com que quase todo o fluxo disponível possa ser utilizado. A Tabela 4.2 mostra o quanto cada algoritmo conseguiu utilizar do fluxo para estabelecer as conexões. Assim como visto no gráfico, observa-se um aproveitamento maior do fluxo disponível nas versões dos algoritmos que usam o MINFRAG. Uma conseqüência direta desse maior aproveitamento é notada ao se observar os dados relativos às requisições bloqueadas.

A curva mostrada na Figura 4.3 mostra a quantidade de bloqueios registrada à medida em que as requisições chegam na rede. Ao observar este novo gráfico deve ser analisada não só a quantidade total de bloqueios que ocorreram mas também o momento em que o primeiro bloqueio ocorre. Quanto mais tempo demora para que a rede bloqueie a primeira requisição, melhor o desempenho do algoritmo.

O comportamento das curvas de bloqueio reflete o comportamento das curvas de utilização do fluxo. Este gráfico mostra que a versão original do MINHOP é o primeiro algoritmo que começa a bloquear requisições. Como o Su-Chen consegue preservar mais o fluxo disponível no início do experimento, acaba demorando mais um pouco para bloquear a primeira requisição. Neste experimento os algoritmos originais tiveram o mesmo número total de bloqueios.

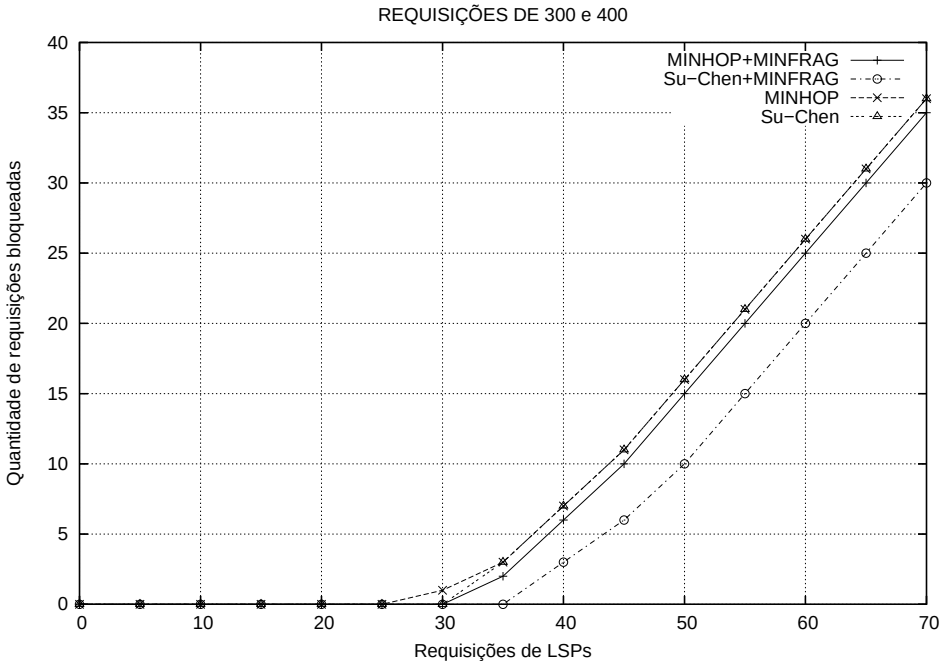

Figura 4.3. Requisições bloqueadas.

Tabela 4.3. Comparativo de bloqueios após 70 requisições

Algoritmo	**Total de Bloqueios**	**Primeiro bloqueio**
MINHOP	41	após 30 requisições
Su-Chen	36	após 35 requisições
MINHOP+MINFRAG	35	após 35 requisições
Su-Chen+MINFRAG	30	após 40 requisições

É possível perceber que o maior aproveitamento do fluxo promovido pela utilização do MINFRAG faz com que os bloqueios demorem mais a acontecer, e determina uma menor taxa final de requisições bloqueadas. O MINHOP com o MINFRAG apresentou um número total de bloqueios um pouco menor que os algoritmos originais. Já o Su-Chen com o MINFRAG conseguiu atrasar bem o primeiro bloqueio, além de determinar um número total de bloqueios menor que todos os outros algoritmos. A Tabela 4.3 mostra os números obtidos das medidas de bloqueio.

Uma outra curva interessante a ser analisada é a que mostra o crescimento total da banda fragmentada nos enlaces da rede. A Figura 4.4 mostra o comportamento da banda fragmentada usando os diferentes algoritmos.

Analisar o gráfico de banda fragmentada é uma forma de avaliar diretamente o desempenho do MINFRAG em seu objetivo central. A Figura 4.4 mostra que o processo cumpriu de modo eficiente este objetivo. Os algoritmos originais não levam em conta o

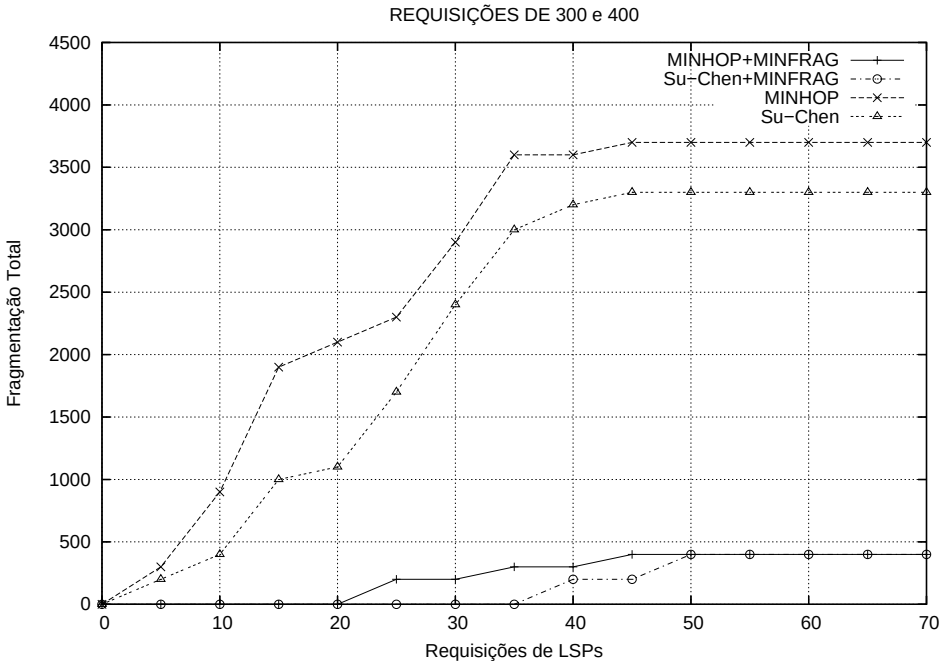

Figura 4.4. Banda Fragmentada.

fenômeno da fragmentação, e por isso esta medida cresce desde as primeiras requisições. Como visto nos gráficos anteriores, este crescimento precoce da fragmentação acaba diminuindo as possibilidades de aproveitamento completo do fluxo, o que acaba gerando um número de bloqueios maiores. O MINHOP original apresentou a maior taxa de fragmentação seguido logo abaixo pelo Su-Chen original. Na Tabela 4.4 observa-se que este total é de 3700 e 3300 unidades de banda que fica sem utilização pelos enlaces da rede.

A análise dos resultados das versões dos algoritmos que usam o MINFRAG mostra que o processo de minimização da fragmentação realmente cumpre o seu papel. Estas duas versões apresentam um total de banda fragmentada de apenas 400 unidades de banda. Pela curva apresentada na Figura 4.4, é possível observar que os algoritmos com o MINFRAG ainda geram um pouco de fragmentação no meio do experimento. Esta banda fragmentada foi gerada apenas em situações nas quais não existiu nenhuma alternativa de caminho tal que o roteamento de uma requisição fosse feito sem deixar enlaces com alguma fragmentação. Como nestes casos não há nenhuma maneira de se evitar a fragmentação, o LSP é estabelecido no caminho que gera a menor quantidade possível de banda fragmentada.

Tabela 4.4. Comparativo do total de banda fragmentada

Algoritmo	**Fragmentação Total**
MINHOP	3700
Su-Chen	3300
MINHOP+MINFRAG	400
Su-Chen+MINFRAG	400

4.3.2 Experimento com 3 incógnitas

O experimento no qual as requisições assumem as demandas 300, 400 e 500 permitem usar o processo de minimização da fragmentação de banda através de uma equação diofantina linear modelo com três incógnitas. Para cada enlace a ser analisado no momento do roteamento, o processamento do MINFRAG será: "Dada a banda residual b_e do enlace e e um LSP associado a uma demanda d que deve ser estabelecido, definir se existe alguma solução em N para a equação diofantina $300x + 400y + 500z = (b_e - d)$".

As medidas usadas para avaliar os resultados deste experimento foram as mesmas do experimento anterior. As observações feitas sobre estas medidas e os dados obtidos no experimento anterior também podem se aplicar a este experimento. Observando-se o gráfico apresentado na Figura 4.5, observa-se que as curvas apresentam um padrão parecido com o encontrado no experimento anterior. Mas, com a maior possibilidade de combinações usando 300, 400 e 500, a possibilidade de se obter banda fragmentada nos enlaces diminui um pouco naturalmente. Devido a isso, observa-se uma leve aproximação entre curvas dos algoritmos originais e das versões que usam o MINFRAG.

Esta pequena diferença também reflete em mudanças no gráfico que mostra o comportamento do número de requisições bloqueadas neste experimento. Na Figura 4.6, o gráfico de requisições bloqueadas também mostra uma aproximação das curvas. Ainda assim, as versões dos algoritmos que usam o MINFRAG conseguiram superar as respectivas versões originais tanto no número total de bloqueios quanto no momento do primeiro bloqueio.

A Figura 4.7 mostra o gráfico de banda fragmentada deste experimento. O maior número de combinações favoreceu os algoritmos em sua versão original, fazendo com que a sua geração de banda fragmentada fosse menor. Mas, a utilização do MINFRAG ainda assim conseguiu reduzir significativamente os valores dessa medida.

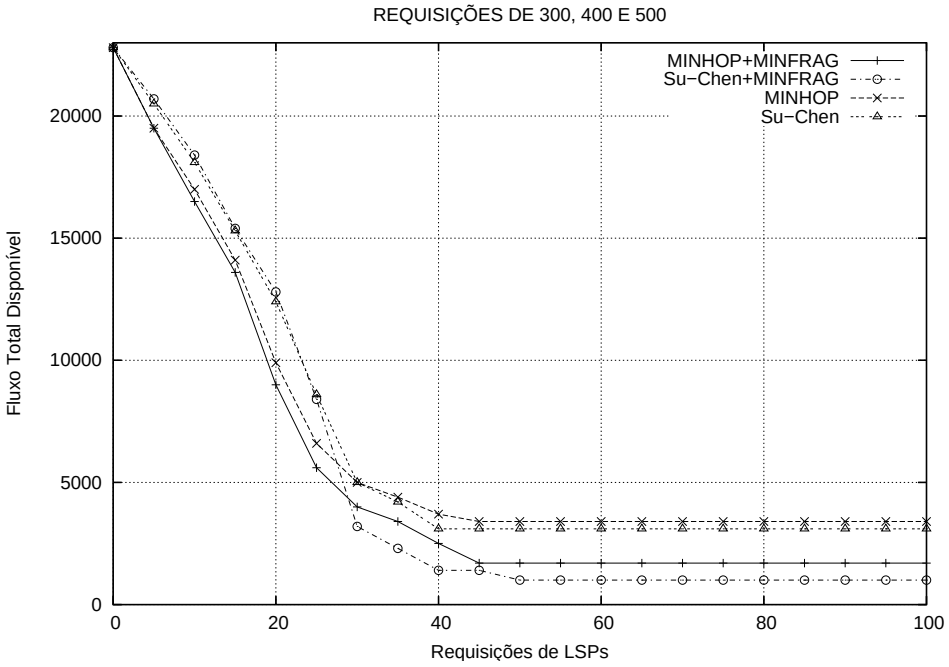

Figura 4.5. Redução do fluxo.

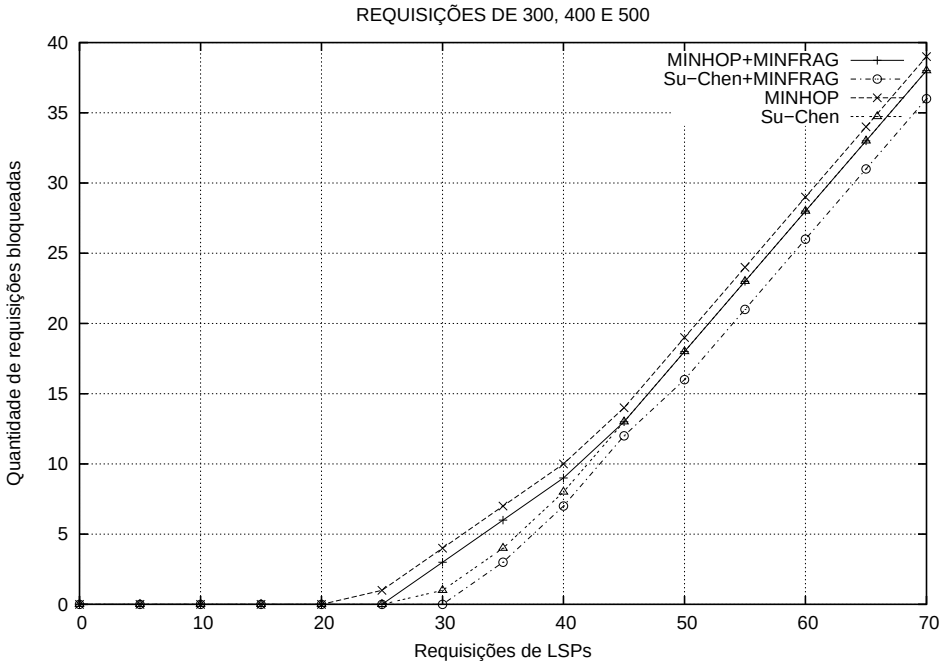

Figura 4.6. Requisições Bloqueadas.

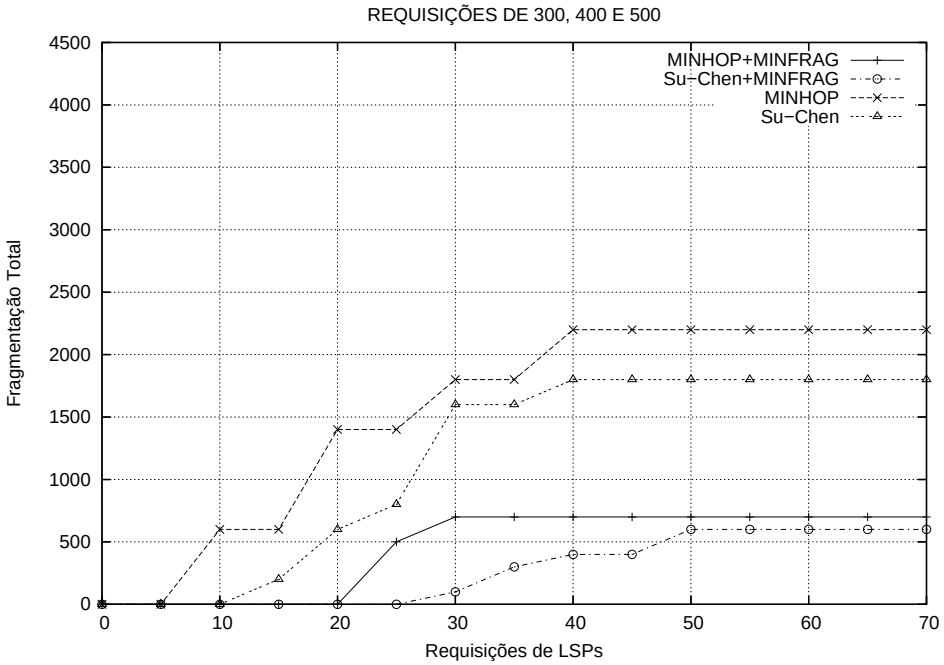

Figura 4.7. Banda Fragmentada.

Tabela 4.5. Comparativo geral (experimento de 3 incógnitas)

Algoritmo	**Fluxo Aproveitado**	**Bloqueios**	**Frag. Total**
MINHOP	85%	39	2200
Su-Chen	86%	38	1800
MINHOP+MINFRAG	92%	38	700
Su-Chen+MINFRAG	95%	36	600

A Tabela 4.5 mostra um comparativo geral dos resultados obtidos do experimento usando três incógnitas.

4.3.3 Avaliação Geral do Resultados

Estes experimentos serviram para demonstrar os resultados das ações do processo de minimização da fragmentação num cenário tipicamente utilizado para o teste dos algoritmos de roteamento.

Os resultados mostraram que o MINFRAG realmente consegue melhorar o desempenho dos algoritmos de roteamento quando o conjunto de demandas usado na rede é propício à geração de banda fragmentada. A capacidade de reduzir o fenômeno da fragmentação possibilitou uma maior utilização dos recursos da rede por parte dos algoritmos. A conseqüência mais importante dessa ação foi observada como a redução do número total de requisições bloqueadas e uma maior quantidade de atendimentos com

sucesso até o primeiro bloqueio.

Estes resultados também mostraram que a utilização de bons algoritmos de roteamento também é importante para um bom desempenho final. Foi observado que o MINFRAG atua pouco durante a fase em que a rede tem abundância de banda, devido às baixas probabilidades de fragmentação. Durante esta fase, o critério mais forte para escolha dos enlaces é o que está definido pelo algoritmo de roteamento. Assim, esse algoritmo ainda deve ter um bom critério de escolha que evite outras formas de esgotamento precoce dos recursos da rede. Nos resultados dos dois experimentos, por exemplo, observou-se que o melhor desempenho foi alcançado ao se combinar o MINFRAG com o algoritmo Su-Chen, que é reconhecido por apresentar resultados superiores em relação aos outros algoritmos. Percebe-se claramente que, neste caso, o MINFRAG conseguiu realçar ainda mais o já bom desempenho do algoritmo Su-Chen.

Assim, pode se concluir que a aplicação do processo de minimização da fragmentação é um recurso que em muitas situações é capaz de melhorar o desempenho dos algoritmos de roteamento a um baixo custo computacional.

CAPÍTULO 5

CONSIDERAÇÕES FINAIS

Este capítulo apresenta as considerações finais e algumas conclusões a respeito deste trabalho. A seção 5.1 apresenta um resumo de todo o trabalho realizado. A seção 5.2 apresenta as principais contribuições do trabalho. Finalmente, a seção 5.3 apresenta algumas sugestões para trabalhos futuros.

5.1 TRABALHO REALIZADO

Este trabalho apresentou o problema da Fragmentação de Banda em redes orientadas a conexão e mostrou como este problema pode diminuir a capacidade de utilização dos recursos disponíveis na rede. O processo de Engenharia de Tráfego e seu processo de roteamento *online*, tecnologias que apresentam um alto crescimento no mercado, foram utilizados para a demonstração do problema e foram as ferramentas usadas para possibilitar a definição do processo de minimização da fragmentação apresentado neste trabalho.

Os algoritmos de roteamento usados na Engenharia de Tráfego foram estudados mais profundamente pois foram identificados como um dos fatores chave da ocorrência ou não de fragmentação. Afinal, são estes algoritmos que acabam definindo como as combinações de demandas são alocadas nos diferentes enlaces. Os algoritmos usam critérios diretos como número de saltos ou a minimização da interferência e não levam em conta o problema da fragmentação. O método chamado neste trabalho como MINFRAG mostrou ser possível combinar o respeito a estes critérios com a tentativa de minimizar a quantidade de banda fragmentada.

Por se basear num processo de complexidade relativamente baixa (o Algoritmo de Euclides Estendido), mostrou-se que o MINFRAG tem uma implementação perfeitamente viável já que não interfere significativamente nos baixos tempos de resposta es-

perados quando se executa o roteamento *online*. Porém, assim como nos algoritmos de interferência mínima, a correta utilização do processo de minimização depende de um estudo bem feito na fase de planejamento da Engenharia de Tráfego a fim de definir uma equação diofantina modelo eficiente.

Os experimentos mostraram a eficiência do processo de minimização da fragmentação usando uma topologia utilizada nos testes dos algoritmos de roteamento. A diminuição do número de bloqueios e o atraso do momento do primeiro bloqueio mostrou que o MINFRAG pode trazer benefícios importantes no aumento do desempenho do processo de roteamento de requisições de LSPs.

5.2 CONTRIBUIÇÕES

Uma primeira contribuição interessante foi a implementação dos algoritmos de roteamento MINHOP, WSP e Su-Chen dentro do pacote *mns* para o *ns2*. Estes algoritmos foram implementados como um novo módulo do *mns* e disponibilizados de modo a poder serem utilizados em outros trabalhos correlatos. A utilização destas implementações no simulador é bem simples. Para os algoritmos MINHOP e WSP, chamadas aos algoritmos retornam a rota escolhida, dado um par origem-destino. Para usar o Su-Chen, é preciso configurar o simulador com os possíveis pares de origem-destino antes do início do roteamento. Essa configuração também é simples, sendo feita através de novas chamadas adicionadas ao simulador. Essas rotas encontradas pelo algoritmos podem ser passadas para uma chamada do simulador que permite estabelecer um LSP recebendo uma rota explicitamente.

A conceituação do problema da fragmentação de banda e o processo de minimização da fragmentação foram as contribuições mais importantes do trabalho. Os experimentos mostraram que este processo de minimização pode aumentar o desempenho do roteamento com baixo custo computacional, o que torna possível sua implementação em situações práticas.

A implementação do processo de minimização da fragmentação de banda no *ns* e *mns* também foram contribuições importantes. Através desta implementação é que

foi possível obter os resultados que mostraram o ganho de desempenho que pode se obter com a utilização do MINFRAG. A utilização do MINFRAG no simulador também é bem simples. É preciso configurar o simulador com os possíveis valores de demandas que serão usados para formar a equação diofantina modelo antes do início do roteamento. Essa configuração também é feita através de chamadas adicionadas ao simulador. Após a entrada destes dados, os algoritmos de roteamento implementados podem ser chamados com uma opção que ativa a minimização da fragmentação.

5.3 TRABALHOS FUTUROS

A continuação deste estudo pode envolver a especificação de novos processos que facilitem ainda mais a utilização prática do método de minimização da fragmentação.

O estudo da rede para a determinação do conjunto de demandas que estarão chegando com as requisições, pode ser uma etapa difícil no processo de Engenharia de Tráfego. Uma das formas de facilitar a utilização do MINFRAG neste caso seria desenvolver um processo de estudo estatístico capaz de determinar aproximadamente este conjunto de demandas através da observação das primeiras requisições que chegam na rede. Como visto nos resultados dos experimentos, o MINFRAG evita modificar muito o comportamento dos algoritmos de roteamento durante o início da operação da rede, quando os enlaces ainda têm um volume relativamente alto de banda residual. Nesta fase inicial, este estudo estatístico poderia determinar o conjunto de demandas aproximado e assim definiria a equação diofantina modelo dinamicamente. Esta seria uma solução alternativa, já que este estudo pode ser feito nos ciclos anteriores do processo de Engenharia de Tráfego.

Outros trabalhos futuros podem envolver a otimização do processo através da determinação de novos algoritmos capazes de detectar a probabilidade de fragmentação num enlace com desempenho ainda melhor em termos de complexidade computacional e capacidade de maior aproveitamento da banda disponível na rede.

REFERÊNCIAS BIBLIOGRÁFICAS

[Ahn, 2001] Ahn, G. (2001). MPLS Network Simulator. `http://flower.ce.cnu.ac.kr/~fog1/mns/`.

[Ameur et al., 2002] Ameur, W. B., Bourquia, N., Gourdin, E., & Tolla, P. (2002). Optimal routing for efficient Internet networks. *ECUMN*, (pp. 10–17).

[Avallone et al., 2003] Avallone, S., Esposito, M., Pescapè, A., Romano, S. P., & Ventre, G. (2003). An Experimental Analisys of DiffServ-MPLS Interoperability. *Proceedings of the 10th International Conference on Telecommunication*, (pp. 281–287).

[Awduche, 1999] Awduche, D. (1999). MPLS and Traffic Engineering in IP networks. *IEEE Communications Magazine*, 37(12), 42–47.

[Awduche et al., 2002] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., & Xiao, X. (2002). *Overview and principles of Internet Traffic Engineering – RFC3272*. Relatório técnico, IETF.

[Awduche et al., 2001] Awduche, D., Gan, D.-H., Li, T., Swallow, G., & Srinivasan, V. (2001). *Extensions to RSVP for Traffic Engineering – RFC3209*. Relatório técnico, IETF.

[Awduche et al., 1999] Awduche, D., Malcom, J., Agogbua, J., O'Dell, M., & McManus, J. (1999). *Requirements for Traffic Engineering over MPLS – RFC2702*. Relatório técnico, IETF.

[Bagula & Krzesinski, 2001] Bagula, A. B. & Krzesinski, A. E. (2001). Traffic Engineering Label Switched Paths in IP Networks using a Pre-Planned Flow Optimization Model. *MASCOTS*, (pp. 70–77).

[Blake et al., 1998] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., & Weiss, W. (1998). *An Architecture for Differentiated Services – RFC2475*. Relatório técnico, IETF.

[Burton, 1976] Burton, D. M. (1976). *Elementary Number Theory*. Allyn Bacon.

[Carpenter & Nichols, 2002] Carpenter, B. E. & Nichols, K. (2002). Diferentiated Services in the Internet. *Proceedings of the IEEE*, (pp. 1479–1494).

[Cetin, 2002] Cetin, R. (2002). *Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base for Fast Reroute – Internet Draft*. Relatório técnico, IETF.

[Chang & Thomas, 2001] Chang, C.-S. & Thomas, J. A. (2001). *Effective Bandwidth in High Speed Digital Networks*. Relatório técnico, Transaction Design, Inc.

[Cisco,] Cisco. Advanced Topics in MPLS-TE Deployment. `http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/mwglp_wp.htm`.

[Cormem et al., 1990] Cormem, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to Algorithms*. MIT Press and McGraw Hill.

[Cortizo & Monteiro, 2004] Cortizo, M. B. M. & Monteiro, J. A. S. (2004). *Estudo do Gerenciamento de Capacidades*. Relatório técnico, UNIFACS - CPqD.

[Davie & Rekhter, 2000] Davie, B. & Rekhter, Y. (2000). *MPLS Technology and Applications*. Morgan Kaufmann Publishers.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1, 269–271.

[Durham et al., 2000] Durham, D., Boyle, J., Cohen, R., Herzog, S., & Sastry, R. R. A. (2000). *The COPS (Common Open Policy Service) Protocol – RFC2748*. Relatório técnico, IETF.

[Ehrensberger & Messmer,] Ehrensberger, J. & Messmer, B. CAN PC: Computer Aided Network Planning Cockpit. *Swiss Priority Programme for Information and Communication Structures*, (pp. 42–44).

[Fall & Varandhan, 2003] Fall, K. & Varandhan, K. (2003). ns Notes and Documentation. `http://www.isi.edu/nsnam/ns/doc/index.html`.

[Faucheur, 2004] Faucheur, F. L. (2004). *Protocol extensions for support of Differentiated-Service-aware MPLS Traffic Engineering*. Relatório técnico, IETF.

[Faucheur et al., 2002] Faucheur, F. L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., & Heinanen, J. (2002). *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services - RFC 3270*. Relatório técnico, IETF.

[Faucheur & Lai, 2003] Faucheur, F. L. & Lai, W. (2003). *Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering – RFC3564*. Relatório técnico, IETF.

[Figueiredo, 2003] Figueiredo, G. B. (2003). Algoritmos de Roteamento com Interferência Mínima. Tese de Mestrado, Universidade Estadual de Campinas.

[Figueiredo et al., 2004] Figueiredo, G. B., da Fonseca, N. L. S., & Monteiro, J. A. S. (2004). Light Minimum Interference Routing. *The IEEE International Conference on Communications (ICC)*.

[Floyd & Jacobson, 1993] Floyd, S. & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), 397–413.

[Ford & Fulkerson, 1956] Ford, L. R. & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8, 399–404.

[Fortz & Rexford, 2002] Fortz, B. & Rexford, J. (2002). Traffic Engineering with Traditional IP Routing Protocols. *IEEE Communications Magazine*, (pp. 118–124).

[Girish et al., 2000] Girish, M. K., Zhou, B., & Hu, J.-Q. (2000). Formulation of the Traffic Engineering Problem. *IEEE ISCC*, (pp. 214–219).

[Goel et al., 2001] Goel, A., Ramakrishnan, K., Kataria, D., & Logothetis, D. (2001). Efficient Computation of Delay-sensitive Routes from One Source to All Destinations. *IEEE IFOCOM*, (pp. 854–858).

[Goldberg & Tarjan, 1988] Goldberg, A. V. & Tarjan, R. E. (1988). A new approach to the maximum flow problem. *Journal ACM*, 35, 921–940.

[Gomes et al., 2002] Gomes, D. G., Agoulmine, N., & de Souza, J. N. (2002). IP Bandwidth Allocation Management using Agents and Neural Network Approach. *IEEE Workshop on IP Operations and Management*, (pp. 8–12).

[Ikenaga, 2003] Ikenaga, B. (2003). Notes on Elementary Number Theory. `http://marauder.millersville.edu/~bikenaga/numth/numnote.html`.

[Jamoussi et al., 2002] Jamoussi, B., Andersson, L., Callon, R., Dantu, R., Wu, L., Doolan, P., Worster, T., Feldman, N., Girish, A. F. M., Gray, E., Heinanen, J., Kilty, T., & Malis, A. (2002). *Constraint-Based LSP Setup using LDP – RFC3212*. Relatório técnico, IETF.

[Jütner et al., 2000] Jütner, A., Szviatovszki, B., Áron Szentesi, Orinesay, D., & Harmatos, J. (2000). On-demand optimization of label switched paths in MPLS networks. *ICCCN*, (pp. 107–113).

[Karbowski, 2002] Karbowski, A. (2002). Decentralized Traffic Control in Data Networks - A Methodological View. *Proceedings of the International Conference on Parallel Computing in Eletrical Engineering*.

[Kelly, 1996] Kelly, F. (1996). Notes on effective bandwidths. In *Stochastic Networks: Theory and Applications* (pp. 141–168).

[Kodialam & Lakshman, 2001] Kodialam, M. & Lakshman, T. V. (2001). Dynamic Routing of Lacally Restorable Bandwidth Garanteed Tunnels using Aggregated Link Usage Information. *IEEE INFOCOM*, (pp. 376–385).

[Kodialam & Lakshman, 2000] Kodialam, M. S. & Lakshman, T. V. (2000). Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *INFOCOM (2)* (pp. 884–893).

[Kompella, 2002] Kompella, K. (2002). *LSP Hierarchy with Generalized MPLS TE – Internet Draft*. Relatório técnico.

[Kompella, 2003] Kompella, K. (2003). *A Traffic Engineering MIB – Internet Draft*. Relatório técnico, IETF.

[Lai et al., 2003a] Lai, W., Tibbs, R., & den Berghe, S. V. (2003a). *A Framework for Internet Traffic Engineering Measurement*. Relatório técnico, IETF.

[Lai et al., 2003b] Lai, W. S., Tibbs, R., & den Berghe, S. V. (2003b). *A Framework for Internet Traffic Engineering Measurement – Internet Draft*. Relatório técnico, IETF.

[Ma & Steenkiste, 1997] Ma, Q. & Steenkiste, P. (1997). Quality-of-Service Routing for Traffic with Performance Guarantees. *International Workshop on Quality of Service*.

[Mitra et al., 1998] Mitra, D., Morrison, J. A., & Ramakrishnan, K. (1998). Refined approximations for design and optimization of wide-area, multi-service broadband networks. *Fourth Informs Telecommunications Conference*, (pp. 646–651).

[Mitra & Ramakrishnan, 1999] Mitra, D. & Ramakrishnan, K. (1999). A case study of Multiservice, Multipriority Traffic Engineering design for data networks. *Global Telecommunications Conference*, (pp. 1077–1083).

[OPNET, 2001] OPNET (2001). *Simulation Based Analysis of MPLS Traffic Engineering - A Case Study*. Relatório técnico, OPNET Technologies Inc.

[P.Aukia et al., 2000] P.Aukia, Kodialam, M., Koppol, P. V., Lakshman, T. V., Sarin, H., & Suter, B. (2000). RATES: A Server for MPLS Traffic Engineering. *IEEE Network Magazine*.

[Ramakrishnan & Floyd, 1999] Ramakrishnan, K. & Floyd, S. (1999). *A Proposal to add Explicit Congestion Notification (ECN) to IP – RFC2481*. Relatório técnico, IETF.

[Riedl, 2002] Riedl, A. (2002). A Hybrid Genetic Algorithm for Routing Optimization in IP Networks utilizing Bandwidth and Delay Metrics. *IEEE Workshop on IP Operations and Management*, (pp. 166–170).

[Riedl, 2003] Riedl, A. (2003). Optimized Routing Adaptation in IP Networks Utilizing OSPF and MPLS. *IEEE International Conference on Communications*, (pp. 1754–1758).

[Rosen et al., 2001] Rosen, E., Viswanatan, A., & Callon, R. (2001). *Multiprotocol Label Switching Architeture*. Relatório técnico, IETF.

[Rouhana & Horlait, 2001] Rouhana, N. & Horlait, E. (2001). Differentiated Services and Integrated Services use of MPLS. *IEEE INFOCOM*, (pp. 376–385).

[Schimdt & Kuritsubu, 2001] Schimdt, J. & Kuritsubu, J. (2001). *Fundamentals of Capacity Management*. Relatório técnico, Transaction Design, Inc.

[Srinivasan et al., 2003] Srinivasan, C., Viswanathan, A., & Nadeau, T. (2003). *Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base – Internet Draft*. Relatório técnico, IETF.

[Stallings, 1998] Stallings, W. (1998). *Operating systems (3rd ed.): internals and design principles*. Prentice-Hall, Inc.

[Stüzle & Hrycej, 2002] Stüzle, E. A. & Hrycej, T. (2002). Estimating Multivariate Conditional Distributions via Neural Networks and Global Optimization. *World Congress On Computational Inteligence*, (pp. 1767–1772).

[Su & Chen, 2002] Su, B. W. X. & Chen, C. P. (2002). A new bandwidth guaranteed routing algorithm for MPLS Traffic Engineering. In *Proceedings of IEEE International Conference on Communications* (pp. 1001–1005).

[Wang & Wang, 1999] Wang, Y. & Wang, Z. (1999). Explicit Routing Algorithms for Internet Traffic Engineering. *IEEE ICCCN*, (pp. 582–588).

[Weissten, 2004a] Weissten, E. W. (2004a). Diophantine Equation. MathWorld – `http://mathworld.wolfram.com/DiophantineEquation.html`.

[Weissten, 2004b] Weissten, E. W. (2004b). Euclidean Algorithm. MathWorld – `http://mathworld.wolfram.com/EuclideanAlgorithm.html`.

[Yarushkina, 2002] Yarushkina, N. G. (2002). Genetic Algorithms for Engineering Optimization: Theory and Practice. *Proceedings of the IEEE International Conference on Artificial Intelligence Systems*.

APÊNDICE A

CÓDIGO FONTE

A.1 MÉTODOS ADICIONADOS À CLASSE SIMULATOR DO NS

```
# Registra um nó origem-destino no simulador
Simulator instproc add-IE-pair { src dst } {
  $self instvar IEnodes_

  lappend IEnodes_ [join "$src $dst " "@"]
}


# Retorna o número de pares origem-destino
Simulator instproc number-IE-pairs { } {
  $self instvar IEnodes_
  return [llength $IEnodes_]
}

# Retorna uma lista com os pares origem-destino
Simulator instproc get-IE-pair { idx } {
  $self instvar IEnodes_
  if { $idx > [llength $IEnodes_] } {
    puts "No such IE pair index: $idx"
    exit -1
  }
  return [split [lindex $IEnodes_ $idx] "@"]
}

# Adiciona uma incógnita para equação diofantina modelo usada pelo MINFRAG
Simulator instproc add-lsp-req-size { size } {
  $self instvar lsp_sizes_
  if {! [info exist lsp_sizes_]} {
    set lsp_sizes_ ""
  }
  set s [parse-bw $size]
  set found 0
  for {set i 0} {$i < [llength $lsp_sizes_]} {incr i 1} {
    if { $s < [lindex $lsp_sizes_ $i] } {
       set lsp_sizes_ [expr int([linsert $lsp_sizes_ $i $s])]
       return $size
    }
  }
  lappend lsp_sizes_ [expr int($s)]
  return $size
```

```
}

# Remove uma incógnita para a equação diofantina modelo
Simulator instproc remove-lsp-req-size { size } {
  $self instvar lsp_sizes_
  set lsp_sizes_ [lremove $lsp_sizes_ [parse-bw $size]
}
```

A.2 MÉTODOS ADICIONADOS À CLASSE MPLSNODE DO MNS

```
# Algoritmo de Roteamento de Su-Chen com opção de minimizar a fragmentação
MplsNode instproc minimum-interferece-route {dst rbw no_frag} {
  global ns peso_ TE_DB
  set src $self
  set bw [parse-bw $rbw]
  set IEnodes [$ns set IEnodes_]
  if { [llength IEnodes] == 0 } {
    puts "SuChen Error: No ingress-egress pairs setted."
    return -1
  }

  set mypair [join "$src $dst" "@"]
  set pairidx [lsearch $IEnodes $mypair]

  if { $pairidx == -1 } {
    puts "MIRA Error: The pair ([$src id] [$dst id]) was not added"
    exit
  }
  set allnodes [$ns set linked_mplsnodes_]
  set len [llength $allnodes]
  if { $len == 0 } {
    puts "SuChen Error: No LSRs created"
    return -1
  }

  for {set i 0} {$i < $len} {incr i 1} {
    for {set j 0} {$j < $len} {incr j 1} {
       set peso_($i,$j) 0
    }
  }

  update-te-db

  set totalflow 0
  for {set i 0} {$i < [llength $IEnodes]} { incr i 1} {
    set pair [split [lindex $IEnodes $i] "@"]
```

```
    set ingress [lindex $pair 0]
    set egress [lindex $pair 1]
    set maxflow($i) [max-flow $ingress $egress 1 1]

    set totalflow [expr $totalflow + $maxflow($i)]
  }
  total-flow-calculed $totalflow

  if { $no_frag == 1 } {
    set max [expr $len * $len]
    for {set i 0} {$i < $len} {incr i 1} {
       for {set j 0} {$j < $len} {incr j 1} {
          set frag [frag-prob $i $j $rbw]
          set peso_($i,$j) [expr $peso_($i,$j) + ($max * $frag)]
       }
    }
  }

  set route [dijkstra $src $dst $bw]
  return $route

}


# Algoritmo MINHOP com opção de minimizar a fragmentação
MplsNode instproc min-hop-route { dst rbw calc_totalflow} {
  global ns peso_ TE_DB
  set src $self
  set bw [parse-bw $rbw]

  update-te-db

  set allnodes [$ns set linked_mplsnodes_]
  set len [llength $allnodes]

  if { $calc_totalflow == 1 } {
    set IEnodes [$ns set IEnodes_]
    set totalflow 0
    for {set i 0} {$i < [llength $IEnodes]} { incr i 1} {
      set pair [split [lindex $IEnodes $i] "@"]
      set ingress [lindex $pair 0]
      set egress [lindex $pair 1]

      set maxflow($i) [max-flow $ingress $egress 0 1]

      set totalflow [expr $totalflow + $maxflow($i)]
    }
    total-flow-calculed $totalflow
```

```
  }

  for {set i 0} {$i < $len} {incr i 1} {
    for {set j 0} {$j < $len} {incr j 1} {
       set peso_($i,$j) 1
    }
  }

  set route [dijkstra $src $dst $bw]
  return $route
}

# Algoritmo WSP com opção de minimizar a fragmentação
MplsNode instproc
          widest-shortest-path-route {src dst rbw calc_totalflow} {
  global ns peso_ TE_DB
  set bw [parse-bw $rbw]

  update-te-db

  if { $calc_totalflow == 1 } {
    set IEnodes [$ns set IEnodes_]
    set totalflow 0
    for {set i 0} {$i < [llength $IEnodes]} { incr i 1} {
      set pair [split [lindex $IEnodes $i] "@"]
      set ingress [lindex $pair 0]
      set egress [lindex $pair 1]


      set maxflow($i) [max-flow $ingress $egress 0]

      set totalflow [expr $totalflow + $maxflow($i)]
    }
    total-flow-calculed $totalflow
  }

  set srcid [$src id]
  set dstid [$dst id]
  set allnodes [$ns set linked_mplsnodes_]
  set len [llength $allnodes]

  foreach no_ $allnodes {
    set pi_([$no_ id]) -1
    set capacidade([$no_ id]) -1
    set hops([$no_ id]) 0x7fffffff
  }

  set capacidade($srcid) 0x7fffffff
```

```
set hops($srcid) 0

set Q_ $allnodes

set S_ ""
while { $Q_ != "" } {
  set qlen [llength $Q_]
  set widest 0
  set u_ -1
  foreach no_ $Q_ {
    set i [$no_ id]
    if {$capacidade($i) >= $widest} {
      set widest $capacidade($i)
      set u_ $no_
    }
  }

  if {$u_ == -1} {
    break
  }

  set Q_ [lremove $Q_ $u_]
  lappend S_ $u_
  set u_id [$u_ id]

  foreach no_ [$u_ set neighbor_] {
    set v_id [$no_ id]
    set Cuv $TE_DB($u_id,$v_id)
    if {$bw <= $Cuv} {
      set capaV [minimo $capacidade($u_id) $Cuv]
      if { ($capacidade($v_id) < $capaV) ||
   ($capacidade($v_id) == $capaV &&
           $hops($v_id) >= [expr $hops($u_id) + 1] )} {
        set hops($v_id) [expr $hops($u_id) + 1]
        set capacidade($v_id) $capaV
        set pi_($v_id) $u_
      }
    } else {

    }
  }
}

set i [$dst id]
set route "$i"
while {$i != [$src id]} {
  if {$pi_($i) == -1} {
    return -1
```

```
    }
    set i [$pi_($i) id]
    set route [linsert $route 0 $i]
  }
  return $route
}
```

A.3 ROTINAS AUXILIARES

```
# Rotinas utilitárias
proc minimo { i_ j_ } {
  return [expr ($i_< $j_)?$i_:$j_]
}

proc maximo { i_ j_ } {
  return [expr ($i_> $j_)?$i_:$j_]
}

proc lremove { lista elem } {
  set pos [lsearch $lista $elem]
  if {$pos == -1} {
    return $lista
  }

  set last_ [expr [llength $lista] -1]
  if { $last_ <= 0 } {
    return ""
  } elseif {$pos == 0} {
    return [lrange $lista 1 $last_]
  } elseif {$pos == $last_} {
    return [lrange $lista 0 [expr $last_ - 1]]
  } else {
    return
    [lreplace $lista $pos [expr $pos + 1] [lindex $lista [expr $pos + 1]]]
  }
}

# CallBacks usadas nos experimentos para coleta de dados
proc total-flow-calculed { tf } {
}


# Algoritmo de Dijkstra
proc dijkstra {src dst bw} {
  global ns peso_ TE_DB
```

```
set srcid [$src id]
set dstid [$dst id]
set allnodes [$ns set linked_mplsnodes_]
set len [llength $allnodes]

foreach no_ $allnodes {
  set pi_([$no_ id]) -1
  set d_([$no_ id]) 0x7fffffff
  set hops([$no_ id]) 0x7fffffff
}
set d_($srcid) 0
set hops($srcid) 0

set Q_ $allnodes

set S_ ""
while { $Q_ != "" } {
  set qlen [llength $Q_]
  set min 0x7ffffffe
  set u_ -1
  foreach no_ $Q_ {
    set i [$no_ id]
    if {$d_($i) <= $min} {
      set min $d_($i)
      set u_ $no_
    }
  }

  if {$u_ == -1} {
    break
  }

  set Q_ [lremove $Q_ $u_]
  lappend S_ $u_
  set u_id [$u_ id]
  foreach no_ [$u_ set neighbor_] {
    set v_id [$no_ id]
    set Cuv $TE_DB($u_id,$v_id)
    if {$bw <= $Cuv} {
      set distancia [expr $d_($u_id) + $peso_($u_id,$v_id)]
      if { ($d_($v_id) > $distancia) ||
   ($d_($v_id) == $distancia &&
           $hops($v_id) >= [expr $hops($u_id) + 1] )} {
        set hops($v_id) [expr $hops($u_id) + 1]
        set d_($v_id) $distancia
        set pi_($v_id) $u_
      }
```

```
      } else {
      }
    }
  }


  set i [$dst id]
  set route "$i"
  while {$i != [$src id]} {
    if {$pi_($i) == -1} {
      return -1
    }
    set i [$pi_($i) id]
    set route [linsert $route 0 $i]
  }
  return $route
}


# Algoritmo de Ford-Fulkerson usado para cálculo do max-flow
proc max-flow { src dst computa_pesos unidade} {
  global ns peso_ TE_DB


  set srcid [$src id]
  set dstid [$dst id]
  set allnodes [$ns set linked_mplsnodes_]
  set len [llength $allnodes]
  for {set i 0} {$i < $len} {incr i 1} {
    for {set j 0} {$j < $len} {incr j 1} {
      set Gfluxo($i,$j) 0
    }
  }

  while {1} {
    for {set i 0} {$i < $len} {incr i 1} {
      set nodeMark($i) -1
      set nodeScanned($i) -1
    }
    set nodeMark($srcid) 0
    set i 0
    set aumento -1
    while {$i < $len && $aumento < 0} {
      if {  $nodeMark($i) != -1 && $nodeScanned($i) == -1} {
        set noi [lindex $allnodes $i]
        set vizinhos [$noi set neighbor_]
        foreach vizinho $vizinhos {
          set j [$vizinho id]
```

```
      if {$nodeMark($j) == -1 } {
        set deltai [lindex $nodeMark($i) 2]
        set cij $TE_DB($i,$j)
        if {$cij < $unidade} {
          set cij 0
        } else {
          set cij [expr $cij - (int($cij) % int($unidade))]
        }
        if { $Gfluxo($i,$j) < $cij } {
          set deltaij [expr $cij - $Gfluxo($i,$j)]
          set deltaj 0
          if {$i == $srcid} {
            set deltaj $deltaij
          } else {
            set deltaj [minimo $deltaij $deltai]
          }
          set nodeMark($j) "+ $i $deltaj"
          if {$j == $dstid} {
            set aumento $deltaj
          }
        } else {
          if { $Gfluxo($j,$i) > 0} {
            set deltaj [minimo $Gfluxo($j,$i) $deltai]
            set nodeMark($j) "- $i $deltaj"

            if {$j == $dstid} {
                set aumento $deltaj
            }
          }
        }
      }
    }
    set nodeScanned($i) 0
    set i 0
  } else {
    incr i 1
  }
}

if {$aumento > -1} {
  set no2 $dstid
  while {$no2 != $srcid} {
    set no1 [lindex $nodeMark($no2) 1]

    set sig [lindex $nodeMark($no2) 0]

    if { $sig == "+" } {
      set Gfluxo($no1,$no2) [expr $Gfluxo($no1,$no2) + $aumento]
```

```
        } elseif { $sig == "-"} {
          set Gfluxo($no2,$no1) [expr $Gfluxo($no2,$no1) - $aumento]
        } else {
          puts "Marca inválida: $nodeMark($no2)"
        }
        set no2 $no1
      }
    } else {
      set vizinhos [$src set neighbor_]
      set maxflow 0

      foreach no $vizinhos {
        set bw [expr $Gfluxo($srcid,[$no id])]
        set maxflow [expr $maxflow + $bw]
      }

      if {$maxflow > 0 && $computa_pesos == 1} {
        for {set i 0} {$i < $len} {incr i 1} {
          set noi [lindex $allnodes $i]
          set vizinhos [$noi set neighbor_]
          foreach noj $vizinhos {
            set j [$noj id]
            set fluxoij [expr $Gfluxo($i,$j)]
            set cij $TE_DB($i,$j)
            if {$maxflow > 0 && $cij > 0} {
              set peso_($i,$j) [expr $peso_($i,$j)
                 + ( double($fluxoij) / ($maxflow * $cij) )]
            }
          }
        }
      } else {
      }
      return $maxflow
    }
  }
}


# Constroi a base de dados com o estado da rede para roteamento
proc update-te-db {} {
  global ns TE_DB

  set allnodes [$ns set linked_mplsnodes_]
  set len [llength $allnodes]

  for { set i 0 } { $i < $len } { incr i 1 } {
    set noi [lindex $allnodes $i]
    set vizinhos [$noi set neighbor_]
```

```
    for { set j 0 } { $j < $len } { incr j 1 } {
      if {$i == $j} {
        set TE_DB($i,$j) 0
      } else {
        set noj [lindex $allnodes $j]
        set n [lsearch $vizinhos $noj]
        set bw -1
        if { $n >= 0 } {
          set bw [[$noi get-link-ptr $noj] get-usable-bw]
        }
        set TE_DB($i,$j) $bw
      }
    }
  }
}
```