Otimização de Alocação de Rotas e Comprimentos de Onda em redes WDM

H. Crispim, M.A. Cutrim, P. Garcia, R. Machado, Eduardo T. L. Pastor, H. Abdalla Jr e A.J.M. Soares

Resumo — Este trabalho apresenta o desenvolvimento de um *software* de simulação para a otimização de recursos de uma rede óptica WDM. As métricas a serem consideradas no processo de otimização são a saturação e a alocação de rotas e de comprimentos de onda. Apresenta-se uma visualização gráfica dos cenários obtidos e a mensuração quantitativa dos resultados.

***Palavras-Chave* — Rede óptica WDM, Rede OMEGA, Otimização, Saturação de rotas e comprimentos de onda.**

***Abstract* — This work presents the elaboration of a simulation software for the optimization of an optical WDM network. The overloading, the routes and wavelengths allocation are the metrics to be consider for the optimization process. A graphical visualization of the obtain scenarios and the quantitative results are shown.**

***Index Terms*— WDM Optical Network, OMEGA Network, Optimization, Overloading of the routes and wavelengths.**

I. Introdução

As redes ópticas com base na tecnologia WDM (*wavelength division multiplexing*) e no protocolo IP permitem o uso eficiente de recursos, por meio da integração de serviços e transporte. É possível conseguir uma maior eficiência destas redes otimizando a alocação de rotas e comprimentos de onda das requisições de conexão. Tal otimização é possível com a análise de uma rede óptica real. Nesse sentido, o plano de controle da rede OMEGA, instalado no CPqD em Campinas, foi emulado no Laboratório de Comunicações da UnB (LabCom). Essa é uma rede óptica transparente composta de cinco nós, com capacidades equivalentes, ligados entre si por pares de fibras, formando a topologia física apresentada na Fig.1 [1].

Cada um desses nós possui: um dispositivo *add/drop* local; um módulo de comutadores ópticos; um módulo de controle; amplificadores ópticos; e *transponders*. Além disso, eles possuem três portas, cada uma com fibras de entrada e de saída, para a comunicação com os nós adjacentes. Os comutadores ópticos, compostos por oito entradas e oito saídas, são responsáveis por redirecionar um dos oito comprimentos de onda de uma das quatro entradas para uma das saídas (três nós adjacentes e o *add/drop* local).

H. Abdalla Jr., *et al*, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília-DF, Brasil, e-mail: abdalla@ene.unb.br. Este trabalho foi parcialmente financiado pela ANATEL, CNPq, CAPES e CPqD.

O plano de controle dessa rede apresenta uma arquitetura distribuída. Cada nó possui um computador com três placas FastEthernet (100 Mbps) e duas placas controladoras dos comutadores ópticos que constituem o módulo de controle, conforme mostrado na Fig.2. A rede possui, ainda, um esquema de proteção e restauração do tipo 1:N [2].

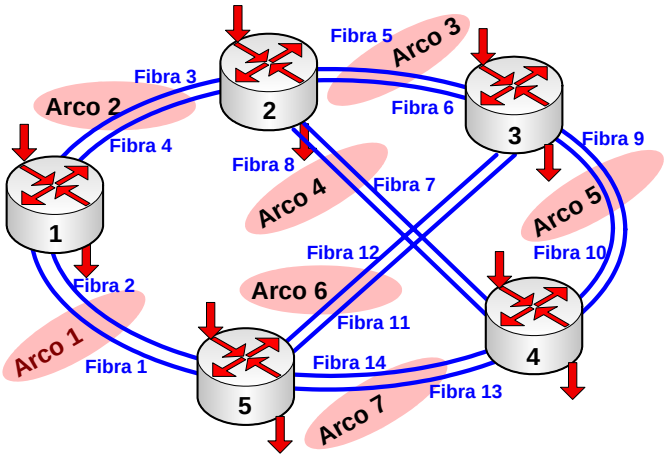


Fig. 1. Topologia física da rede OMEGA.

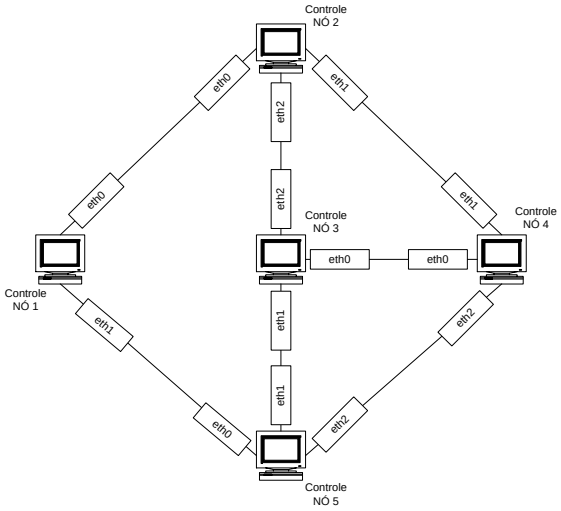


Fig. 2. Plano de controle da rede OMEGA.

II. Emulação da Rede OMEGA

A. Simulação do plano de controle

O plano de controle da rede OMEGA foi emulado por meio de cinco microcomputadores com configurações

semelhantes. Em cada um deles, representando um dos nós ópticos da rede, instalou-se o sistema operacional Linux Red Hat 7.03. A configuração física da rede de simulação inclui três placas Ethernet 10/100 Mbps por máquina, possibilitando uma topologia similar à apresentada pela rede OMEGA. A Fig. 3 ilustra a rede emulada com endereços IP (192.168.0.0/24), não-roteáveis. Em cada uma das máquinas foi instalado um cliente do programa ZEBRA 0.93, um aplicativo livre que gerencia protocolos de roteamento TCP/IP, como BGP-4, RIPv1, RIPv2 e OSPFv2 [3-6].

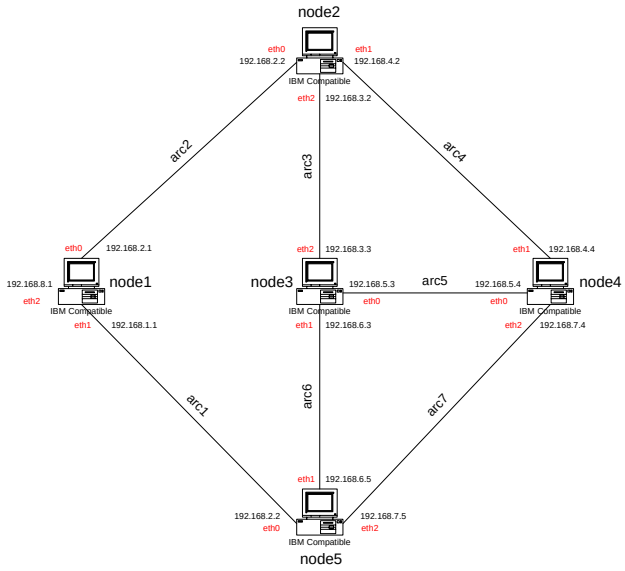


Fig. 3. Plano de Controle emulado no Laboratório da UnB

B. Simulação do plano de transporte

Para que os protocolos de controle pudessem operar, foi necessário o carregamento de cada ponto da rede de controle com um arquivo de topologia da rede física OMEGA. Nesse arquivo, são fornecidas informações acerca da quantidade de nós, ligação das fibras e esquema de proteção adotado. Assim, a criação de *lightpthts* pôde ser feita manualmente ou por um algoritmo RWA associado a um Dijsktra.

III. PROJETO DO SISTEMA DE OTIMIZAÇÃO

O objetivo é desenvolver um sistema de otimização para a alocação de recursos na rede óptica transparente OMEGA, quando esta se apresentar saturada. Para tal, foi criada uma interface de usuário gráfica (GUI) e implementado um algoritmo de otimização que deverá fornecer uma nova configuração para a rede. O processo de funcionamento é o seguinte: o administrador da rede, ao verificar um estado de saturação de recursos, no qual um pedido de rota não pôde ser atendido devido à falta de rotas e/ou de comprimentos de onda disponíveis, deve solicitar uma ação de otimização. Nesse instante, o sistema de controle faz um *backup* do estado corrente, que corresponde às rotas atendidas e suas respectivas configurações, e requisita os serviços do sistema de otimização. Este, por sua vez, recebe o arquivo com as rotas atendidas, interpreta-o e aplica o algoritmo de otimização implementado. Para finalizar o processo, o sistema de otimização deve formatar um arquivo com os dados da otimização, que será repassado pelo administrador para o sistema de controle. Este deve levantar a rede com a nova configuração das rotas e, dado que uma otimização foi realmente alcançada, o pedido previamente negado deve ser atendido.

Devido a algumas dificuldades encontradas na simulação da rede, foi criado um algoritmo de saturação para a verificação da lógica implementada. Assim, todo o processo acontece na interface gráfica. O usuário do sistema de otimização, após abrir o programa, deve requisitar os serviços do algoritmo de saturação, que irá gerar o arquivo saturado.

A eficiência do algoritmo de otimização foi verificada com a criação de um algoritmo de comparação de topologias. A partir da interface gráfica, um arquivo de texto é criado mostrando uma comparação da alocação de recursos antes e depois da otimização.

O sistema também poderia ser implementado e estendido a outras redes ópticas que realizam alocações de *lightpaths* a cada pedido requisitado.

IV. DIAGRAMA DE CASOS DE USO

Partindo da visão geral do sistema, usando modelamento UML (Unified Modeling Language), cinco casos de uso distintos podem ser identificados [7-8]: executar backup; converter o banco de dados do Linux para o Windows; *otimizar a configuração*; converter o banco de dados do Windows para o Linux; e por fim, executar *restore*, Fig.4.

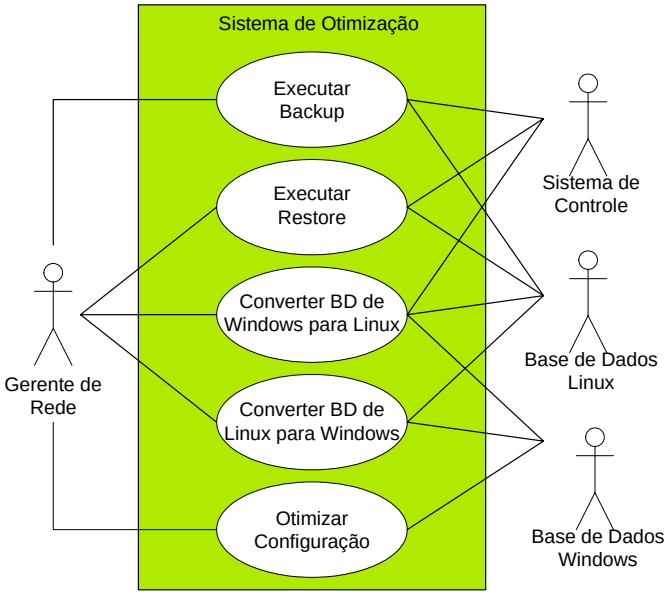


Fig. 4. Diagrama de casos de uso do sistema de otimização

Entre os casos de uso identificados, o processo de otimização é o que se pretende implementar. Os demais casos envolvem processos físicos, com poucos elementos lógicos. No que se refere ao caso de uso *otimizar configuração*, incluem-se os atores (gerente da rede e base de dados Windows); a visão geral (todas as ações são realizadas no

sistema operacional Windows), e a finalidade (receber estado saturado e devolver estado otimizado).

O caso se inicia quando o gerente da rede executa o programa de otimização e carrega o arquivo de *backup* com a configuração da rede saturada. Utilizando a interface gráfica gerada pelo programa, o gerente faz a requisição de otimização. Como resposta, o programa gera um novo arquivo com a configuração otimizada da rede. A Tabela 1 mostra a seqüência de eventos do sistema de otimização. O diagrama de classes do *software* de otimização é apresentado na Fig.5. Este *software* é composto, basicamente, por quatro módulos, Satur, Otimiz, Visual e Mensur, descritos a seguir.

Satur: módulo responsável por gerar um arquivo contendo um conjunto de requisições que satura os recursos da rede, além de um outro arquivo com as rotas das requisições que foram negadas (bloqueadas) a partir da saturação.

Este módulo procura reproduzir exatamente o mesmo cenário que seria obtido em um ambiente real de solicitações e liberações de conexões entre os diferentes nós, após um tempo razoável de operação da rede. Para isso, é gerada uma nova semente de aleatoriedade para o sorteio de uma das 104 possíveis rotas (na rede OMEGA) por linha de arquivo. Logo é sorteado um dos oito comprimentos de onda possíveis por rota. Após isso, o programa entra em um laço incondicional e só termina depois de ocorrido um certo número de negações de requisições devido à indisponibilidade de recursos.

Se a saturação já tiver sido atingida, efetua-se a gravação do arquivo de saturação e do arquivo com o conjunto de requisições bloqueadas, fecha-se ambos os arquivos e encerra-se o programa. Caso contrário, limpa-se novamente as variáveis correspondentes aos campos de uma linha do arquivo e sorteia-se aleatoriamente uma nova rota (requisição de conexão) para tentativa de alocação [10].

Otimiz: módulo responsável pela otimização propriamente dita, gerando, com base no arquivo saturado fornecido pelo Satur, um arquivo otimizado contendo o mesmo conjunto de requisições, porém com a utilização de uma menor quantidade de recursos. Este módulo foi desenvolvido em C++ [9].

O processo de otimização tem inicio com a transferência do conteúdo do arquivo saturado para um objeto de uma classe definida do módulo. Neste, é procurado a linha que contenha a rota de maior comprimento, em termos do número de enlaces (ou número de saltos), já que tal rota corresponderia à requisição que tem a maior chance de ter sido alocada de forma não-ótima. Assim, realizando primeiro a re-alocação das requisições cujas rotas são maiores, espera-se obter um aumento no desempenho da otimização.

Em seguida, é aberto um arquivo para escrita, e toda a rede é disponibilizada marcando-se como desocupados todos os enlaces da rede, para todos os comprimentos de onda. Procede-se, então, à re-alocação da rota e do comprimento de onda associados à linha que foi detectada como sendo a de maior comprimento. Tal re-alocação é obtida por meio da implementação de um algoritmo de RWA que determina, primeiramente, a rota com que a requisição deve ser atendida e, após isso, tenta alocar para essa rota algum comprimento de onda que esteja disponível ao longo de todos os enlaces desse caminho.

Tabela 1. Seqüência de eventos para otimizar a configuração.

Ação do ator	Resposta do sistema
Executa o programa de otimização	Mostra as opções do programa em um menu gráfico
Indica o arquivo de *backup* que deve ser otimizado	Interpreta o arquivo e mostra graficamente o estado atual das conexões entre nós
Requisita a otimização do estado de conexões	Implementa o algoritmo de otimização e gera um arquivo com as configurações otimizadas, mostrando graficamente o novo estado de conexões
Fecha o programa de otimização	

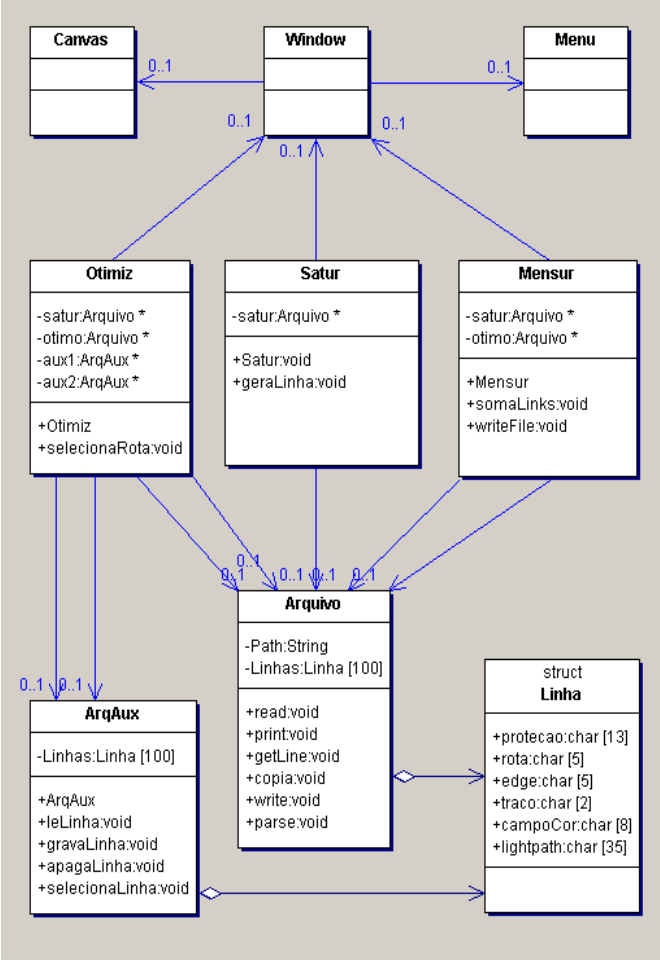
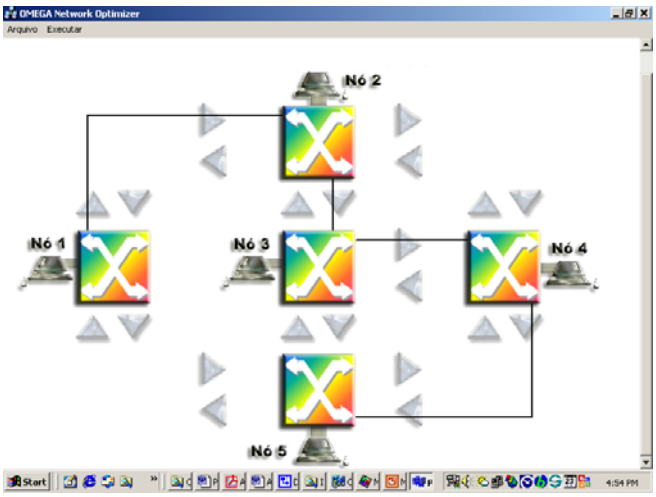
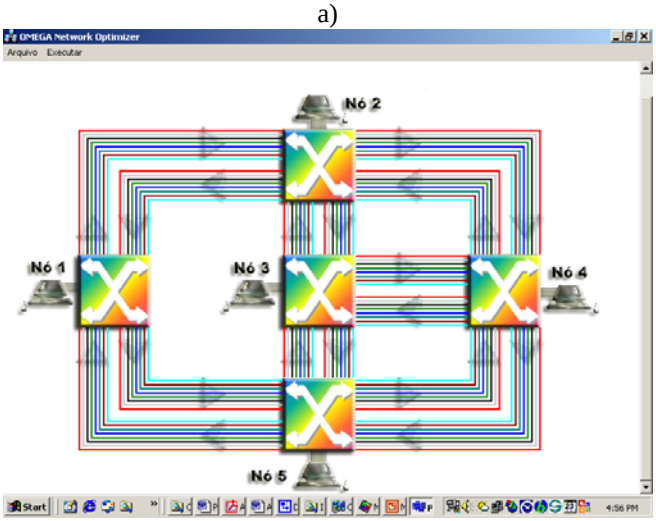


Fig. 5. Diagrama de classes do *software* de otimização.



a)



b)

Fig. 6. a) Representação da rota 1 → 2 → 3 → 4 → 5 com o comprimento de onda 3. b) Todos os seus comprimentos sendo utilizados.

A determinação das rotas é feita com base em um **algoritmo de roteamento alternado** no qual tem-se que o conjunto de *K* rotas candidatas especificado para cada par de nós *P* corresponde, na verdade, ao conjunto de todas as possíveis rotas existentes entre esse par de nós. Isso só foi possível devido à relativa simplicidade da rede em questão (104 rotas). Em uma realidade mais complexa, rotas candidatas corresponderiam apenas a um subconjunto de todas as rotas possíveis. Essas rotas candidatas foram calculadas a partir do algoritmo de menor caminho de Dijkstra, computado a *priori* (*offline*) para cada um dos pares origem-destino de nós [6][10].

A escolha dos comprimentos de onda, por sua vez, é feita com base no **algoritmo de seleção do comprimento de onda de ordem fixa**, em que os mesmos são testados em ordem crescente: a primeira alocação com o comprimento de onda 1, depois com o 2 e assim por diante [6][10].

Não existindo comprimentos de onda livres ao longo dos enlaces de qualquer uma das rotas que unem os *edges*, a requisição em questão será negada por indisponibilidade de recursos. Apesar de estar prevista no código do programa, não é esperado que tal condição aconteça, pois isso significaria que o algoritmo de otimização piorou a alocação das requisições, uma vez que não foi possível atender a uma requisição que, anterior à otimização, era efetivamente atendida.

Assim, a cada iteração do programa é determinada e re-alocada, no arquivo de otimização, a linha do arquivo saturado cuja rota associada possui o maior comprimento. Desse modo, após iterações equivalentes ao número de linhas do arquivo saturado, encontra-se no arquivo otimizado o mesmo conjunto de requisições inicialmente existente, porém com as rotas e comprimentos de onda associados re-alocados de acordo com o algoritmo de RWA e de modo a minimizar a utilização dos recursos da rede.

Visual: módulo responsável por possibilitar a visualização das rotas e comprimentos de onda associados de um arquivo que contenha um conjunto de requisições qualquer. Este módulo foi desenvolvido em *C++ Builder*.

Um arquivo de rotas, seja ele de otimização, saturação ou um arquivo de topologia da rede OMEGA, deve ser apresentado na tela com seus respectivos comprimentos de onda, Fig.6. Foram representados os comprimentos de cada arco com 16 linhas, 8 por fibra (TX e RX).

Para acessar os vários dados e informações, implementou-se um *menu* formado pelas funções exigidas pelo projeto: ***Abrir arquivo de topologia***, que abre um arquivo de topologia originário das requisições atendidas na rede OMEGA; ***Salvar topologia otimizada***, o qual salva o objeto final do processo de otimização, que é o novo arquivo de topologia; ***Salvar imagens***, guarda a imagem das rotas para eventuais testes e referências. Esse *menu* apresenta os itens denominados topologia original e topologia otimizada; ***Otimizar***, chama a rotina de otimização desenvolvida e apresenta a topologia resultante.

Mensur: módulo responsável por mensurar, tendo por base os arquivos saturado e otimizado, as características de utilização dos recursos da rede, bem como a porcentagem da otimização alcançada, além de verificar se as requisições bloqueadas podem, ou não, ser atendidas na nova configuração da rede resultante da otimização.

O algoritmo deste módulo procede à contagem da quantidade de recursos (pares enlace/comprimento de onda) utilizados por cada um dos arquivos de leitura, saturado e otimizado, quanto às porcentagens de tais quantidades frente ao total de recursos da rede como um todo, além de uma medida da porcentagem alcançada pela otimização específica em questão [11].

V. Testes Experimentais e Análise dos Resultados

A partir dos quatro módulos integrados em uma interface gráfica única, procedeu-se à realização de diversos testes. Os resultados, por módulo, serão expostos e analisados a seguir.

A. *Módulo Satur*

No atendimento a um pedido de conexão entre um par

origem-destino de nós, deve ser levado em consideração todo o histórico relativo aos pedidos de conexão realizados anteriormente. Nesse sentido, é necessário considerar o estado geral de ocupação ou disponibilidade das rotas e comprimentos de onda da rede como um todo. Dessa forma, eventualmente o atendimento pode ocorrer em uma rota não-ótima, visto que as melhores rotas e comprimentos de onda podem estar momentaneamente ocupados.

Assim, a rede apresenta uma clara tendência a atingir um estado de saturação não-ótimo. O módulo de saturação tenta simular esse mesmo ambiente por meio do sorteio e da tentativa de atendimento de um conjunto aleatório de requisições. Essa aleatoriedade é obtida com relação a todos os fatores passíveis de sorteio quando é feito um pedido de conexão. São sorteados os campos proteção, rota, *edge*, campoCor (relativo ao comprimento de onda) e *lightpath* (rota seguida pela requisição) [10-11].

Foi observado que à medida que se aumenta o valor do grau de saturação (expressado em número de requisições bloqueadas), obtém-se um número maior de requisições aceitas. Isso, apesar de ser uma tendência, não é necessariamente uma regra, devido ao caráter aleatório do sorteio das requisições. Assim, tem-se uma aproximação de um limite que corresponde à utilização completa de todos os recursos da rede. Uma outra característica perceptível é que se a rede está se tornando progressivamente mais saturada, as últimas conexões aceitas tendem a apresentar um número de saltos menor, uma vez que requisições desse tipo demandam menor quantidade de recursos.

B. Módulo Otimiz

Em todos os casos de otimização, foi possível notar dois aspectos. Primeiro, o número de requisições permaneceu igual, com os mesmos nós de origem e destino em relação aos arquivos saturados, indicando que, efetivamente, todos os pedidos de conexão existentes continuaram sendo mantidos, ocorrendo apenas uma re-alocação das rotas e dos comprimentos de onda correspondentes. Segundo, da seqüência de arquivos otimizados resultantes conseguiu-se alcançar um razoável grau de otimização, percebido pelo fato de a maioria das requisições otimizadas ter sido atendida utilizando-se caminhos constituídos por apenas um enlace.

C. Módulo Visual

Por meio deste modulo, é possível se ter uma idéia mais clara do grau de otimização alcançado. É possível perceber também quando se está, gradativamente, atingindo um estado de saturação completa da rede.

Pode-se notar uma alocação mais ordenada para o caso dos arquivos otimizados, quando comparada à alocação existente nos arquivos saturados, devido ao caráter aleatório do surgimento e conseqüente atendimento das requisições de conexão. No caso do arquivo otimizado, tem-se que a ordenação é uma conseqüência do caráter inerente ao algoritmo de ordem fixa utilizado para a escolha do comprimento de onda para uma requisição a ser atendida. A Fig.7 apresenta visualmente o estado da rede com o arquivo saturado (quando uma requisição foi bloqueada por falta de recursos), e com o arquivo otimizado.

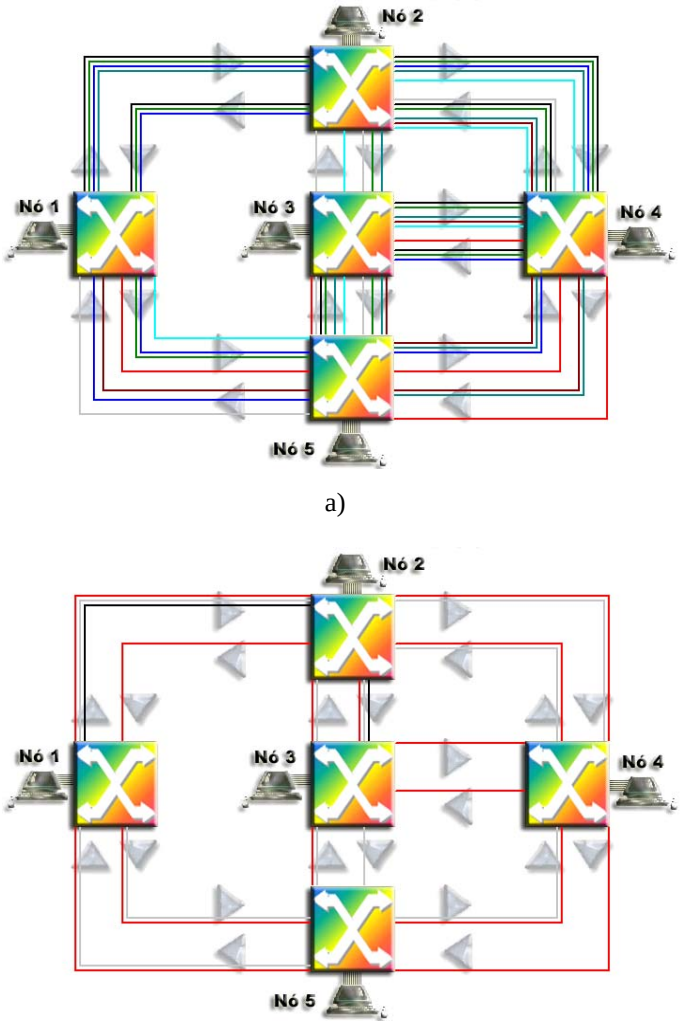


Fig. 7. Estado da rede: a) com o arquivo saturado com uma solicitação bloqueada, e b) com o arquivo otimizado.

D. Módulo Mensur

Este módulo permite uma comparação mais objetiva entre os cenários representados pelos arquivos saturados e otimizados correspondentes e verifica, ainda, a possibilidade de aceitação, na nova alocação das requisições obtida, daquelas rotas que haviam sido inicialmente bloqueadas na saturação.

A Fig.8 ilustra um exemplo do arquivo de saída desse módulo com grau de saturação correspondente a uma requisição bloqueada. O índice porcentagem da otimização alcançada corresponde à quantidade de pares enlace-comprimento de onda economizados devido à otimização, com relação à quantidade utilizada anteriormente.

Com base nos testes, foi possível estimar a porcentagem média da otimização alcançada, para uma execução qualquer do algoritmo de otimização, como sendo de 40,5%. Uma estimativa mais exata, porém, deveria considerar uma

quantidade bem mais elevada de testes. Contudo, mais do que a precisão desse valor, revela-se mais importante o caráter qualitativo de tal medida.

O grau de otimização alcançado pelo módulo Otimiz mostrou-se bastante satisfatório. Em um ambiente real, entretanto, tal grau de distribuição do conjunto de requisições de uma forma totalmente aleatória dificilmente seria alcançado, o que levaria a uma porcentagem de otimização menor. No entanto, para um ambiente de requisições e liberações de conexões reais, tem-se que o grau de aleatoriedade, em um dado momento, depende tanto da freqüência de solicitações e liberações de conexão quanto do tempo total de funcionamento da rede.

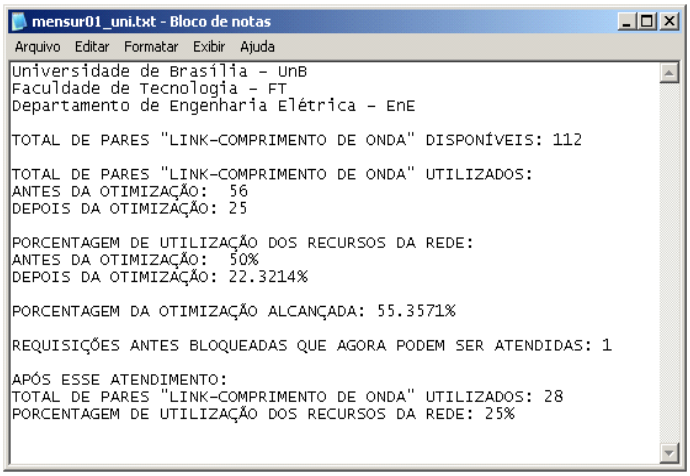


Fig. 8. Arquivo de saída do módulo Mensur com grau de saturação correspondente a uma requisição bloqueada.

VI. CONCLUSÕES

A escolha da rede OMEGA como base para a criação do projeto de otimização fundamentou-se principalmente na viabilidade de acesso ao seu sistema de controle, o que permitia a realização de testes práticos quanto ao funcionamento e à qualidade do programa desenvolvido. Um problema, porém, foi a limitação na quantidade de métricas a serem utilizadas como parâmetro de otimização. Na prática, as únicas métricas utilizadas foram a quantidade de comprimentos de onda permitidas pela rede e o número de saltos, já que todos os enlaces tinham o mesmo custo e distância.

Apesar disso, o conjunto de módulos desenvolvido permitiu simular de forma fidedigna um ambiente real de uma rede óptica WDM, com relação a algumas das suas características de funcionamento. A obtenção de um conjunto de requisições saturado (Satur), a respectiva otimização da alocação (Otimiz) e os módulos de visualização de resultados (Visual e Mensur) se traduzem em um retrato qualitativo e quantitativo a respeito da utilização de recursos, bem como do grau de otimização alcançado, além de ser verificada a possibilidade de atendimento a alguns dos pedidos que foram inicialmente bloqueados.

A conversão da interface gráfica de um sistema estático para um sistema de monitoramento em tempo real permitiria que qualquer modificação dinâmica na rede pudesse ser prontamente observada. Essa alteração não oferece um alto grau de dificuldade, devido à forma como o algoritmo foi implementado. Porém, a sua execução tem como pré-requisito a integração do sistema.

Finalmente, o desenvolvimento de um sistema de gerenciamento e de monitoramento do tráfego da rede permitiria a obtenção de mais métricas, como o histórico de tráfego na rede e o custo atribuído a cada enlace, para o desenvolvimento de novos esquemas de otimização. Além disso, essas novas características alterariam o funcionamento do algoritmo de escolha de melhor rota, o Dijkstra. No entanto, é importante ressaltar que tal modificação não implicaria na perda de validade do sistema de otimização projetado, uma vez que o resultado produzido pelo Dijkstra continuaria fornecendo como resposta a melhor rota.

AGRADECIMENTOS

Os autores agradecem ao CPqD pelas informações relativas à rede óptica OMEGA.

REFERÊNCIAS

[1] A.C. Sachs e S. Rossi. *Demonstração Experimental do Funcionamento de uma Rede Óptica com Sistema Remoto de Gerência para Aprovisionamento Dinâmico de Banda*, CPqD, Campinas, 2002.

[2] A.C. Sachs. *Sistema de Proteção e Restauração*, Demonstração de Protótipo de Laboratório, CPqD, Campinas, 2002.

[3] R. Ramaswami and K.N. Sivarajan, "Optical routing and wavelength assignment in all-optical networks", IEEE/ACM Trans. Networking, vol. 3, pp. 489, Oct. 1995.

[4] J. Hayes. *Fiber Optics Technician's Manual*, 2nd edition, Delmar Publishers, Albany, New York, 2001.

[5] R. Ramaswami e K. Sivarajan. *Optical Networks, A Pratical Perspective*, Academic Press, San Diego, CA, 1998.

[6] C.S.R Murthy e M. Gurusamy. *WDM Optical Networks: Concepts, Design, and Algorithms*, Prentice-Hall, Upper Saddle River, 2002.

[7] J.D. Furlan. *Modelagem de Objetos Através da UML*, Makron Books, São Paulo, 1998.

[8] http://www.omg.org/docs/formal/03-03-01.pdf. *Object Managemente Group: OMG Unified Modeling Language Specification*. Acessado em 25/09/2003.

[9] H.M. Deittel. *C++: Como Programar* , Bookman, Porto Alegre, 2001.

[10] A. Birman, "Computing approximate blocking probabilities for a class of all-optical networks", IEEE J. Select. Areas Commun. , vol. 14, pp. 852, June 1996.

[11] R.A. Barry and P.A.Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers", IEEE J. Select. Areas Commun., vol. 14, pp. 858, June 1996.