Escalonamento de canais em redes OBS usando informações topológicas

Gustavo B. Figueiredo[1], **Nelson L. S. da Fonseca**[1]

[1]Instituto de Computacao – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6176 – 13.084 -971 – Campinas – SP – Brazil

{gustavo,nfonseca}@ic.unicamp.br

Abstract. *This paper presents a novel channel scheduling algorithm for optical burst switching networks called Least Reusable Channel (LRC). This algorithm uses information about the network topology and about the potential routes the traffic can follow . Results obtained via simulations show that the LRC algorithm produces good performance when compared to others proposed existing algorithms.*

Resumo. *Este artigo apresenta um novo algoritmo de escalonamento de canais para redes OBS, denominado LRC (Least Reusable Channel). O algoritmo usa informações relativas à topologia da rede e as rotas por onde as rajadas trafegarão para determinar o canal que será alocado. Resultados derivados via simulação mostram o bom desempenho do algoritmo quando comparado a outros apresentados na literatura.*

1. Introdução

Redes ópticas de comutação de rajadas (OBS - *Optical Burst Switching*) são consideradas como uma das principais alternativas para implementação de uma Internet puramente óptica baseada em multiplexação de comprimentos de onda (WDM = *Wavelength Division Multiplexing*).

Em redes OBS, o processo de reserva de recursos é feito em uma via. Um pacote de controle é enviado da origem ao destino para realizar a reserva dos recursos para uma rajada. Nenhuma mensagem adicional confirmando a reserva é enviada ao nó de origem. Se no momento da transmissão da rajada os recursos não estiverem disponíveis em algum nó entre o nó de origem e o nó de destino, a rajada é sumariamente descartada. Obviamente, a reserva em uma via diminui substancialmente o *overhead*, permitindo que redes OBS tenham flexibilidade suficiente para lidar com as flutuações do tráfego.

O principal protocolo de reserva de recursos utilizado em redes OBS é o JET (*Just-Enough-Time*) [Qiao and Yoo 2000]. No JET, um pacote de controle é enviado para reservar um canal de saída por um período de tempo correspondente à duração da rajada. O instante de início da reserva é o instante esperado da chegada da rajada no nó. Dado que as rajadas e os pacotes são transmitidos em canais separados (e com velocidades diferentes), uma rajada poderia chegar a um nó i antes do pacote de controle a ela associado. Para que isso não aconteça, as rajadas só são transmitidas após decorrido um período de tempo suficientemente grande que permita a chegada do pacote de controle ao destino antes da respectiva rajada.

Dado que as rajadas não chegam imediatamente uma após a outra, a ocupação dos canais pode alternar entre períodos de reserva e de disponibilidade. Um desafio em redes OBS é o projeto de algoritmos de escalonamento rápidos que façam uso eficiente dos intervalos sem reserva ao alocá-los às novas rajadas. Um algoritmo de escalonamento ideal deve ser capaz de processar o pacote de controle suficientemente rápido de forma que um canal seja escolhido antes que a rajada chegue ao nó. Além disso, deve-se manter informações sobre os intervalos sem reserva, fazendo alocação das rajadas nesses intervalos, garantindo-se, assim, o aumento da utilização da rede.

Este artigo apresenta um algoritmo de escalonamento denominado LRC (*Least Reusable Channel*). O LRC faz uso de uma nova abordagem para a determinação do canal que será reservado para a transmissão da rajada. O algoritmo considera o conceito de reutilização de canais através de uma função que determina o canal com menor chance de ser utilizado por futuras rajadas. Para tal, o LRC usa informações sobre a posição de cada nó em relação a um certo destino (informações topológicas) para determinar a probabilidade de utilização dos canais.

Diferentemente dos demais algoritmos de escalonamento de canais apresentados na literatura, o LRC usa uma estratégia adaptativa para determinação do canal que acomodará a nova rajada, possibilitando o atendimento mais justo de rotas com diferentes tamanhos. Resultados numéricos mostram que o LRC produz elevada utilização da largura de banda e menor probabilidade de bloqueio quando comparado com outros algoritmos na literatura.

O resto do artigo está organizado como segue: A Seção 2 apresenta os algoritmos de escalonamento de canais existentes na literatura, a Seção 3 apresenta o conceito de escalonamento usando informações topológicas. A Seção 4 apresenta o algoritmo LRC propriamente dito. A Seção 5 mostra os exemplos numéricos e por fim a Seção 6 apresenta as conclusões.

2. Algoritmos de escalonamento de canal

Em uma rede OBS, é possível que um pacote de controle chegue T unidades de tempo antes da respectiva rajada. Isto acontece devido ao uso do tempo de ajuste, que é o tempo compreendido entre a transmissão do pacote de controle e da rajada a ele associada. Neste caso, a reserva dos recursos não se dá no tempo t correspondente à chegada do pacote de controle e sim no tempo $t' = t + T$ que é o instante de tempo em que a rajada chega. Se l for considerado o tamanho da rajada (em unidades de tempo), a reserva é feita até o instante $t'' = t' + l$.

Devido ao uso de diferentes tempos de ajustes para rajadas pertencentes à diferentes pares origem-destino, é possível que as rajadas não cheguem aos nós da rede na mesma ordem dos seus pacotes de controle. Assim, a utilização dos canais é fragmentada por períodos de reserva e períodos sem reserva denominados *void*. Inicialmente, cada um dos W canais de um enlace pode ser pensado como um intervalo de tempo aberto do zero ao infinito positivo. Cada *void* I_j pode ser modelado como um par ordenado (s_j, e_j) onde s_j e e_j correspondem, respectivamente, ao início e ao final do *void* I_j, com $e_j > s_j$. Um intervalo *void* é considerado viável a uma rajada de dados $b = (t', t'')$ se, e somente se, $s_j \leq t'$ e $e_j \geq t''$.

Quando um intervalo *void* de um canal I_j é alocado a uma requisição r, dois

novos *voids* são criados. Um deles é o *void* anterior (denotado por a_j) que corresponde ao intervalo formado entre o início do *void* original (s_j) e o início da reserva, (s_j, t'). O segundo é o *void* posterior (denotado por p_j) e corresponde ao intervalo compreendido entre o instante final da requisição e o final do *void* original, (t'', e_j).

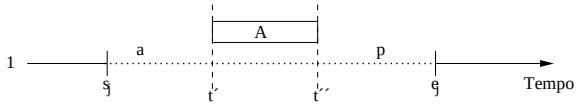


Figura 1. Criação de novos intervalos

A Figura 1 ilustra o processo de criação dos *voids* anterior e posterior. Um pacote de controle para a rajada de dados (A) solicita reserva de recursos entre os instantes de tempo t' e t'', para uma reserva no intervalo *void* I_j, compreendido entre os instantes s_j e e_j. Após a alocação dos recursos para a rajada A, os intervalos a e p são criados.

Estes novos *voids* podem ser usados para acomodar novas reservas. Dessa forma, um dos desafios nas redes OBS é a escolha adequada dos canais que acomodarão as reservas. O desempenho de um algoritmo de escalonamento de canais é, usualmente, medido em função da probabilidade de bloqueio por ele apresentada. Assim obviamente, quanto menor a probabilidade de bloqueio das rajadas, melhor o algoritmo de escalonamento.

Vários algoritmos de escalonamento (ou seleção) de canal tem sido propostos na literatura [Xiong et al. 2000, Turner 1999, Murty and Gurusamy 2002, Yu et al. 2004]. Turner [Turner 1999] propôs um algoritmo de escalonamento de canal denominado *Horizon* (também conhecido como *Latest Available Unused Channel* - LAUC, [Xiong et al. 1999]). Neste algoritmo, o escalonador armazena, para cada canal, apenas as informações sobre o tempo depois do qual não existem reservas (horizonte). O escalonador atribui a nova rajada considerando o horizonte mais próximo do início da reserva. A idéia por trás do LAUC é minimizar o intervalo *void* entre o horizonte do canal e o tempo de início do novo período de reserva. Este algoritmo é relativamente simples e tem complexidade computacional da ordem de $O(logW)$ para um enlace com W canais. Entretanto, ele produz baixa utilização da rede e altas taxas de perda já que todos os intervalos *void* são descartados.

Em [Xiong et al. 1999], foram propostas variações do LAUC. Tais variações possuem as mesmas informações do que o LAUC e também as mesmas limitações, ou seja, produzem baixa utilização da rede e altas taxas de perdas. A versão mais simples do LAUC é o *First Fit*, que busca por canais em uma ordem fixa e pré-estabelecida ou em *round-robin*, e seleciona o primeiro canal viável.

Para aumentar a utilização da rede e diminuir a taxa de perdas, Xiong et al. [Xiong et al. 2000] propuseram um algoritmo denominado LAUC-VF (*Latest Available Unused Channel with Void Filling*). O algoritmo armazena informações sobre todos os *voids*, inclusive aquele gerado pelo horizonte e o infinito positivo. Após a chegada de uma reserva no tempo t, o algoritmo procura acomodar a mesma em um intervalo *void* longo o suficiente, cujo horizonte seja o mais próximo.

Tais modificações fazem com que o LAUC-VF em relação ao LAUC apresente maior utilização da rede e menor taxa de perdas. Sua complexidade computacional é de ($O(WlogM)$, onde M representa o número de reservas em todos os canais) [Chen et al.

2004]. Uma adaptação simples do LAUC-VF é o algoritmo *RANDOM*. Neste algoritmo, os *voids* viáveis são selecionados. Depois, um deles é aleatóriamente escolhido.

Xu et al. [Xu et al. 2004, Xu et al. 2003] propuseram um conjunto de algoritmos usando diversos critérios para selecionar um canal para uma rajada. Os algoritmos usam técnicas geométricas para identificar *voids* viáveis. Os principais algoritmos apresentados foram MIN-SV, MIN-EV e *Best fit*. MIN-SV tenta, ao inserir uma rajada em um *void* viável, minimizar a diferença entre o tempo de início da nova reserva e o tempo do inicio do intervalo *void*. Conceitualmente, o algoritmo MIN-SV é igual ao LAUC-VF. Contudo, o MIN-SV usa uma árvore de busca binária balanceada, o que lhe permite uma grande redução da complexidade computacional ($O(logM)$, onde M representa o número de reservas em todos os canais).

O algoritmo MIN-EV, tenta minimizar o *void* gerado entre o final da nova reserva e uma reserva já existente. O algoritmo *Best Fit* tenta minimizar o tamanho total do início e do final dos *voids* gerados pela nova reserva. Experimentos de simulação mostraram que o desempenho do algoritmo MIN-SV é superior quando comparado com o MIN-EV e o *Best Fit*.

3. Escalonamento de canais em redes OBS usando informações topológicas

A minimização da probabilidade de bloqueio pode ser alcançada através da maximização das chances de acomodação de rajadas futuras. Em outras palavras, ao realizar a reserva dos recursos para uma rajada r_i, o algoritmo de escalonamento deve fazê-lo de forma que a requisição r_{i+1} tenha o máximo de chances de ser acomodada.

Se o escalonamento dos canais for feito sem levar em consideração as futuras requisições, a rede pode experimentar o caso extremo em que a alocação dos recursos para uma determinada requisição pode bloquear várias requisições posteriores.

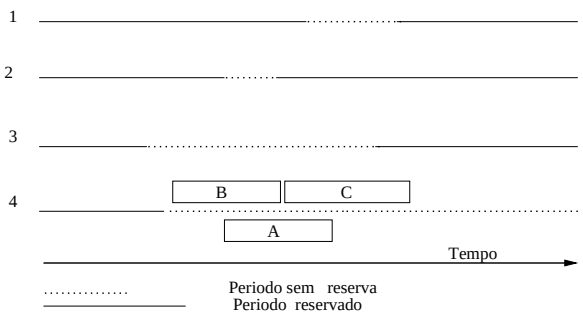


Figura 2. Problema no escalonamento de canais

A Figura 2 ilustra o problema causado por política de escalonamento que não leva em consideração as requisições futuras de reserva de canal. Considere, por exemplo, que as requisições de reserva de canal cheguem na ordem A, B, C, ou seja, primeiro chega o pacote de controle correspondente à rajada A; em seguida chega o pacote de controle correspondente à rajada B, e por fim o pacote de controle correspondente à rajada C. Os únicos canais capazes de acomodar as rajadas correspondentes às reservas A, B e C são os canais 3 e 4. É fácil notar que se o canal 4 for usado para acomodar a reserva A, a reserva B pode usar o canal 3 mas nenhum *void* pode ser utilizado pela reserva C. Entretanto,

se o canal 3 for usado para acomodar a reserva A, ambas as reservas (B e C) podem ser acomodadas com sucesso.

Dessa forma, ao fazer a alocação de um canal a uma requisição, dois fatores importantes devem ser levados em consideração pelo o algoritmo de escalonamento. O primeiro fator é a capacidade que os *voids* anterior e posterior têm de acomodar novas requisições. O segundo fator é a probabilidade de que as novas requisições venham a usar esses novos *voids*.

3.1. Vida útil de um intervalo *void*

Define-se a vida útil de um intervalo como sua capacidade de acomodar rajadas futuras. Dessa forma, o algoritmo deve levar em consideração a capacidade dos intervalos restantes de acomodar novas requisições, já que à medida que o tempo passa, o intervalo tem sua capacidade progressivamente diminuída, como ilustrado na Figura 3.

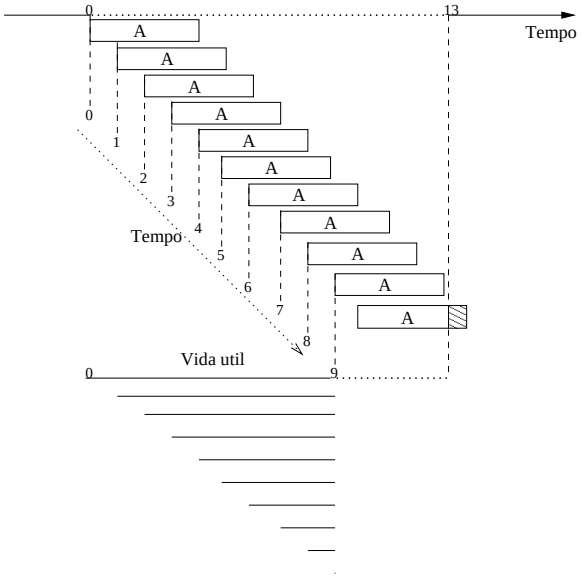


Figura 3. Tempo de vida útil de um intervalo

Sejam s_j , e_j, r_i e r_f o início e o final do *void* e da r-ésima requisição, respectivamente e β o tamanho médio das rajadas (em unidades de tempo). O *void* anterior só é capaz de receber uma rajada de tamanho β até o instante $r_i - \beta + 1$. Assim, o tempo de vida útil do intervalo a_j pode ser calculado como:

$$\upsilon(a_j) = r_i - \beta + 1 - s_j = r_i - (s_j + \beta) + 1 \quad (1)$$

de modo análogo, o tempo de vida útil do intervalo p_j pode ser calculado como:

$$\upsilon(p_j) = e_j - (r_f + \beta) + 1 \quad (2)$$

3.2. Inversão da ordem de chegada das requisições

Para identificar as chances de re-utilização de cada *void*, o algoritmo avalia as chances de que as próximas requisições tenham tempo de início anterior ou posterior ao tempo

de início da requisição sendo processada, ou seja, as chances de haver uma inversão na ordem de chegadas de pacotes de controle e rajadas.

A Figura 2 ilustra a situação de inversão na ordem de chegada de pacotes de controle e de rajadas. O instante de chegada do pacote de controle da reserva B é posterior ao da reserva A, já que as reservas chegam na ordem A, B, C. Entretanto, o instante de chegada da rajada B é anterior ao instante de chegada da rajada A.

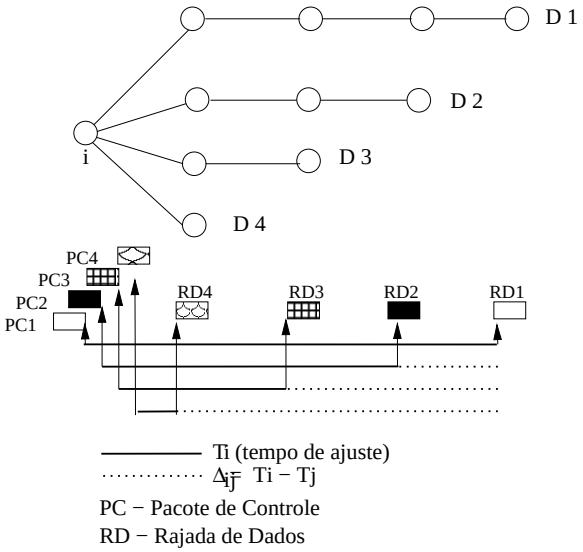


Figura 4. Inversão da ordem de chegada das requisições

A inversão na ordem de chegada dos pacotes de controle e rajadas acontece quando os pacotes de controle e as rajadas chegam a um nó i intermediário de várias rotas com diferentes distâncias em relação aos destinos. Os tempos de ajuste (T_i^j) dos pacotes de controle que chegam ao nó i são funções dos atrasos de propagação dos enlaces no caminho entre i e o destino j, bem como do tempo de processamento nos nós no caminho entre i e j. Quanto mais perto de seu destino está o nó i, menor o intervalo de tempo entre a chegada do pacote de controle e a chegada da rajada correspondente. Dado que um nó i pode ser um nó intermediário na rota de vários pares origem-destino (e consequentemente ter distâncias diferentes para todos os pares), pacotes com diferentes tempos de ajuste chegam ao nó i, o que pode ocasionar a inversão mencionada anteriormente.

A Figura 4 ilustra outra situação de inversão. Nesta figura, observa-se uma topologia com o nó i intermediário nas rotas entre os pares origem-destino (S_1, D_1), (S_2, D_2), (S_3, D_3), (S_4, D_4). Quando um pacote de controle com destino D_j chega ao nó i, o tempo de ajuste é igual a T_i^j. Se pacotes de controle destinados aos destinos $D1$ e $D2$ chegarem simultaneamente ao nó i no instante de tempo t, a rajada destinada ao nó 1 chegará ao nó i no instante de tempo $t' = t + T_i^1$, enquanto que a rajada destinada ao nó 2 chegará ao nó intermediário i no instante

$$t'' = t + T_i^2 = t + T_i^1 - \Delta_{12}$$

havendo assim a inversão na ordem em que chegam os pacotes de controle e suas rajadas correspondentes. Assim, para que haja a mencionada inversão dos eventos, é necessário

que o segundo pacote de controle contenha um tempo de ajuste menor do que o tempo de ajuste do primeiro pacote de controle.

Para que o algoritmo de escalonamento infira sobre a possibilidade de inversão na ordem de duas requisições consecutivas e suas rajadas correspondentes, é necessário que todo nó i retenha a memória do tempo de ajuste T_i^j para cada destino j. Com isso, dado que chegou uma reserva para o destino j no tempo t_j e com tempo de ajuste T_i^j, a probabilidade de que a próxima rajada chegue antes da rajada destinada ao nó j pode ser calculada.

3.2.1. Probabilidade de Inversão

Pelo que foi discutido anteriormente, define-se probabilidade de inversão como a probabilidade de que um pacote de controle destinado ao nó k chegue no intervalo

$$[t_j; t_j + \Delta_{jk}]$$

dado que um pacote de controle destinado ao nó j chegou no instante de tempo t_j.

A diferença, no nó i, entre os tempos de ajustes do par (S_j, D_j) e os tempos de ajuste do par (S_k, D_k) ($\Delta_{jk} = T_i^j - T_i^k$), corresponde, intuitivamente, ao intervalo de tempo de fronteira entre a inversão e a não inversão dos eventos de chegada dos pacotes de controle e suas respectivas rajadas. Considere a chegada de dois pacotes de controle pc_1 e pc_2 nos instantes de tempo t_j e t_k destinados respectivamente aos nós j e k. Para que haja inversão dos eventos de chegada das rajadas destinadas aos nós j e k, duas condições devem estar satisfeitas. A primeira condição é que o pacote de controle pc_2 (destinado ao nó k) deve conter um tempo de ajuste inferior àquele contido no pacote de controle pc_1 (destinado ao nó i). A segunda condição é que o pacote de controle pc_2 chegue ao nó i dentro do intervalo de tempo

$$[t_j; t_j + \Delta_{jk}]$$

que é o intervalo de tempo compreendido entre a chegada do primeiro pacote de controle mais a diferença entre os tempos de ajuste T_i^j e T_i^k.

Em outras palavras, se o pacote de controle destinado ao nó k chegar $\Delta_{jk} + \epsilon$ unidades de tempo após o pacote de controle destinado ao nó j, não haverá inversão dos eventos e a rajada destinada ao nó k chegará após a rajada destinada ao nó j.

Obviamente, os instantes de chegada dos pacotes de controle em cada nó são função do atraso de propagação e processamento, além do tempo necessário para a montagem das rajadas, o que pode variar dependendo do algoritmo de montagem de rajadas em questão. Assim, o algoritmo de montagem de rajadas utilizado na rede deve ser levado em consideração para o cálculo da probabilidade de não haver inversão.

Se o algoritmo utilizado é baseado em janelas de tempo, o intervalo entre a chegada das rajadas pertencentes ao par origem-destino (S_i, D_i) é fixo com valor W_i (Figura 5). Seja t_j o instante de chegada do último pacote de controle pertencente ao par (S_j, D_j) e N o número de pares origem-destino. A probabilidade de inversão é calculada tomando-se o número médio de fontes em que seus pacotes de controle podem chegar

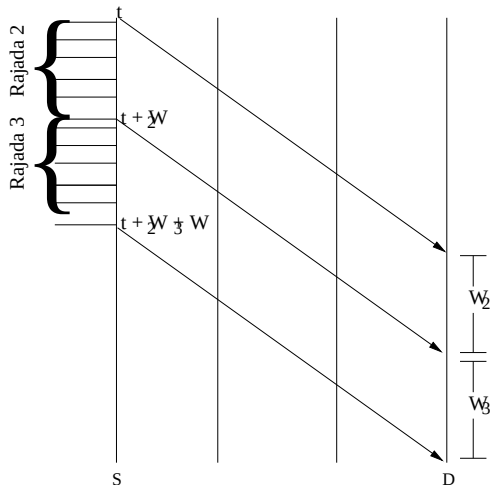


Figura 5. Tempo entre chegada de rajadas (algoritmo baseado em janelas de tempo)

dentro do intervalo $[t_j; t_j + \Delta_{jk}]$ e pode ser calculada como:

$$P_I = \frac{1}{N} \sum_{k=1}^{N} X_k \tag{3}$$

onde,

$$X_k = \begin{cases} 1 & \text{, se } t_j \leq t_k + W_k \leq t_j + \Delta_{jk}; \\ 0 & \text{, caso contrário} \end{cases}$$

Se o algoritmo de montagem de rajadas utilizado for baseado em volume de tráfego, uma rajada só é montada e o pacote de controle liberado após a chegada de n pacotes. Assim, a probabilidade de inversão equivale à probabilidade de que n pacotes com tamanho médio igual a β, de forma que $n = S/\beta$, onde S é o limiar para a criação da rajada) cheguem ao nó de origem dentro de um intervalo de tamanho igual a $[t_k; t_j + \Delta_{jk}]$, onde t_k é o instante de chegada do último pacote de controle do par (S_k, D_k).

Seja t_j o tempo de chegada do último pacote de controle destinado ao nó j, *assumindo que o processo de chegadas de pacotes segue a distribuição de Poisson*, a probabilidade de que um pacote de controle destinado ao nó k chegue no intervalo $[t_k; t_j + \Delta_{jk}]$ é dada pela equação [Papoulis 2002]:

$$P_I = \sum_{k=1}^{N} \frac{(\lambda_k((t_j - t_k) + \Delta_{jk}))^{n-1} e^{-\lambda_k((t_j - t_k) + \Delta_{jk})}}{(n-1)!} \tag{4}$$

onde λ_k corresponde à taxa média de chegada de pacotes entre o par (S_k, Dk).

4. O algoritmo *Least Reusable Channel* (LRC)

Ao adotar uma estratégia fixa para a acomodação de reservas, como por exemplo minimização de *void* anterior ou posterior, o algoritmo de escalonamento ignora o padrão

de chegadas de reservas futuras, o que limita a utilização dos *voids* anterior e/ou posterior resultantes. Note que todos os algoritmos descritos na seção 2 não consideram reservas futuras nas suas alocações.

Como mencionado, tais algoritmos de escalonamento de canais usam estratégias simples e fixas para a alocação dos canais. Os algoritmos LAUC, LAUC-VF e MIN-SV, tentam minimizar o *void* anterior (a_j). A idéia por trás de tal estratégia é que os *voids* posteriores tenham tamanho maior e sejam usados para acomodar as requisições que chegam no futuro. Já a estratégia adotada pelo algoritmo MIN-EV, é a de usar o *void* anterior para acomodar futuras requisições que cheguem dentro de um intervalo de tempo curto. Para ter mais sucesso na sua estratégia de acomodação de rajadas, o *void* anterior deve possuir maior tamanho, o que faz com que o algoritmo minimize o *void* posterior.

Se uma estratégia fixa é utilizada, os resultados podem ser insatisfatórios. Ao minimizar somente o *void* anterior o algoritmo minimiza as chances de que requisições futuras sejam acomodadas em intervalos de tempo anteriores aos da requisição precedente. De modo análogo, quando o *void* posterior é minimizado, as requisições que demandam intervalos de tempo posteriores podem ser prejudicadas. Além disso, os algoritmos que fazem uso de tal tipo de estratégia não levam em consideração a capacidade dos outros *voids* de acomodar novas requisições.

Diferentemente dos algoritmos discutidos na seção 2, o algoritmo LRC não usa uma estratégia fixa. Seu critério de seleção depende da potencial reusabilidade por futuras requisições dos *voids* anterior e posterior criados após a alocação da requisição corrente. Para inferir sobre tal reusabilidade, o algoritmo usa o padrão de chegadas das requisições futuras em relação a requisição sendo processada, que por sua vez depende da topologia em questão e do tamanho dos *voids* anterior e posterior. Isto lhe permite acomodar a rajada no *void* com menos chances de ser reutilizado, deixando livres os *voids* com maiores chances de serem utilizados por reservas futuras.

A idéia do LRC baseia-se no fato de que a minimização da probabilidade de bloqueio pode ser alcançada através da maximização das chances de que a próxima requisição de reserva de recursos seja atendida. Assim, no momento da reserva de recursos para a requisição i, o canal a ser escolhido deve ser aquele que dentre os que atendam os requisitos da i-ésima requisição, tenha a menor possibilidade de contemplar a requisição $i + 1$, ou seja, aquele com a menor chance de ser reutilizado.

Especificamente, o LRC faz uso de uma estratégia adaptativa, que se ajusta à topologia em uso, já que probabilidade de inversão em um dado nó, é uma função da sua posição relativa nas rotas em que atua como nó intermediário. O algoritmo LRC usa informações sobre a posição dos nós nas rotas para os vários destinos, para tomar decisões sobre qual estratégia adotar, ou seja, se o *void* a ser minimizado será o anterior ou o posterior.

Além disso, o tempo de vida útil dos intervalos é levado em consideração. A idéia é que para que um canal esteja disponível para acomodar rajadas no futuro, é preciso que os *voids* não utilizados tenham boas chances de acomodar novas requisições. Assim, os *voids* escolhidos devem ser os menos capazes de acomodar novas requisições. Para tal, uma função de reutilização de canais é utilizada. Dado que com a alocação de um canal w para a requisição r, dois novos *voids* (a_j e p_j) são criados, a reutilização do canal w

usado para acomodar a requisição r pode ser escrita como:

$$\varphi(w) = P_I.\upsilon(a_j) + (1 - P_I).\upsilon(p_j) \quad (5)$$

onde P_I é a probabilidade de inversão, ou seja, a probabilidade de que a rajada $r + 1$ chegue no novo *void* a_j e $\upsilon()$ é a vida útil do intervalo.

Assim, ao receber uma requisição de reserva de recursos, o algoritmo LRC primeiro determina quais os *voids* são capazes de acomodar a requisição. O algoritmo então calcula as chances de que uma nova requisição demande uma reserva para um intervalo de tempo anterior ao intervalo de tempo requerido pela reserva atual (através do cálculo da probabilidade de inversão). De posse desse valor, é possível saber se o *void* a ser minimizado será o *void* anterior ou o posterior. Em outras palavras, se a probabilidade de inversão for alta é melhor que um canal que minimize o *void* posterior seja escolhido, caso contrário, um canal que minimize o *void* anterior pode ser escolhido.

Após isso, para cada um dos canais, o algoritmo determina sua função de reutilização de acordo com a Equação 5, que corresponde às chances do canal de acomodar requisições no futuro caso seja escolhido para a requisição atual. Assim, o algoritmo LRC deve escolher, para a requisição r o canal w com a menor função de reutilização $\varphi(w)$. Se mais do que um canal tiver o mesmo valor de reutilização, o intervalo com menor *void* anterior será alocado. O algoritmo 1 descreve em detalhes o algoritmo LRC.

Algoritmo 1 LRC

ENTRADA

W canais de saída de um nó i, uma requisição de reserva de recursos no intervalo $[r_i, r_f]$ para uma rajada com destino j.

SAÍDA

Comprimento de onda reservado no intervalo $[r_i, r_f]$.

LRC

1: **for all** ($w \in W$) calcule: **do**
2: A probabilidade de inversão (P_I) de acordo com as equação 3 ou 4 (a depender do algoritmo de montagem usado).
3: o tempo de vida útil dos *voids* anterior e posterior à alocação da reserva em w de acordo com as equações 1 e 2, respectivamente.
4: a função de reutilização de w de acordo com a equação 5.
5: Retorne w tal que $\varphi(w)$ seja mínimo.

5. Exemplos Numéricos

Para avaliar o desempenho dos algoritmos de escalonamento, simulações foram realizadas. A ferramenta utilizada nas simulações foi o OB2S (*Optical Burst Switching Simulator*) desenvolvido na Universidade Salvador [Maranhão et al. 2007]. Em cada simulação foram geradas 200.000 requisições de reserva de recursos para rajadas ópticas. Cada uma das simulações foi replicada 20 vezes usando diferentes sementes de geração de números aleatórios. Os resultados são reportados usando um nível de confiança de 95%.

A topologia utilizada nas simulações é apresentada na Figura 6. Cada enlace da rede é constituído por uma fibra com 16 comprimentos de onda com capacidade de transmissão de 2.5 Gbps cada.

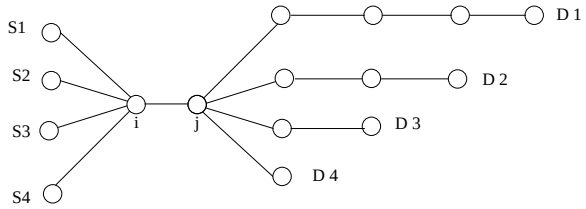
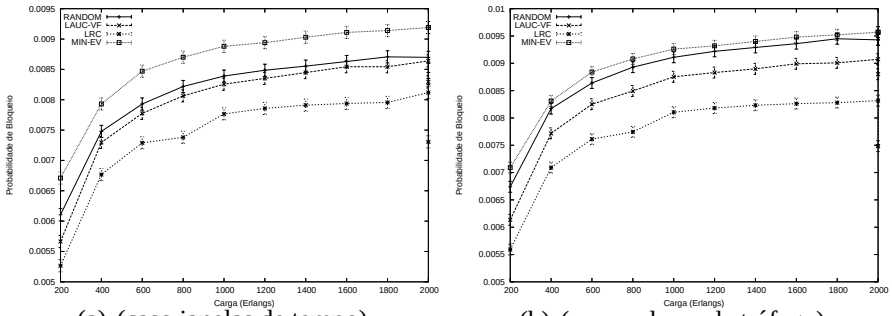


Figura 6. Topologia usada nas simulações

A rede possui quatro pares origem destino, a saber $(S_1, D_1), (S_2, D_2), (S_3, D_3), (S_4, D_4)$. O que significa dizer que todo tráfego originado em cada nó de origem S_i é destinado ao nó destino (D_i) correspondente. Todo o tráfego a que a rede é submetida é gerado nos nós de origem (S_1, S_2, S_3 e S_4) e a divisão do tráfego entre eles segue uma distribuição uniforme.

O tráfego gerado nos nós de origem segue distribuição de Poisson e o tamanho das rajadas segue uma distribuição exponencial negativa. A carga de tráfego considerada nas simulações variou de 200 a 2000 Erlangs. Dois algoritmos de montagem de rajadas foram usados pelos nós de origem. O algoritmo baseado em janelas de tempo [Ge et al. 2000] e o algoritmo baseado em volume de tráfego [Yu et al. 2002]. O tamanho médio dos pacotes IP foi de 500 bytes e os valores dos limiares de tempo e volume de bytes são ajustados de acordo com a carga em Erlangs.

Os algoritmos investigados nas simulações foram o MIN-EV, o RANDOM, o LAUC-VF e o LRC. O algoritmo MIN-SV não foi utilizado nas simulações pois é conceitualmente igual ao LACU-VF. Os algoritmos foram avaliados em uma rede sob tráfego intenso. A idéia é verificar o comportamento dos algoritmos em uma situação de alta carga e verificar a sua eficiência em tais condições.

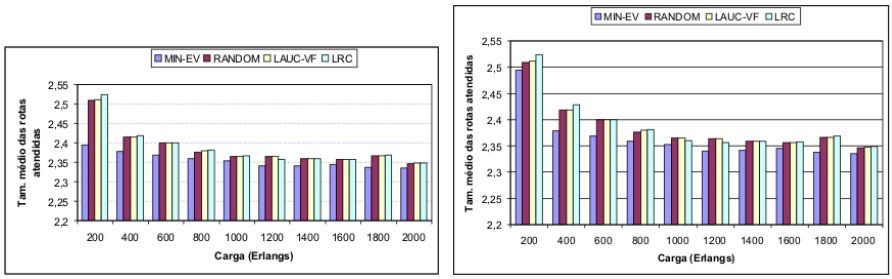


(a) (caso janelas de tempo) (b) (caso volume de tráfego)

Figura 7. Probabilidade de bloqueio de rajadas

A Figura 7 apresenta a probabilidade de bloqueio produzida pelos algoritmos. Devido à carga de tráfego na rede e à pequena quantidade de comprimentos de onda disponíveis, os algoritmos apresentam uma elevada probabilidade de bloqueio.

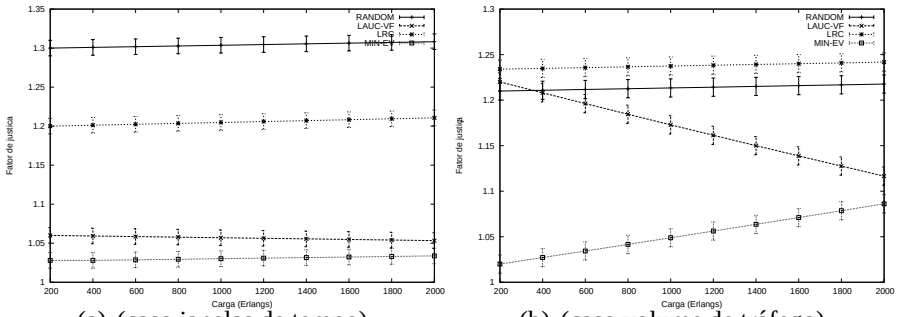
Na Figura 7 (a), percebe-se que o algoritmo MIN-EV produziu a maior probabilidade de bloqueio entre os algoritmos avaliados. Os algoritmos RANDOM e LAUC



(a) (caso janelas de tempo) (b) (caso volume de tráfego)

Figura 8. Tamanho médio das rotas atendidas

comportaram-se estatísticamente de maneira idêntica e o algoritmo LRC apresenta probabilidade de bloqueio inferior aos demais. A diferença entre as probabilidades de bloqueio produzidas pelo LRC e pelo LAUC-VF foi entre $9,1\%$ e $9,8\%$. Isso acontece dado que o LRC ao utilizar uma estratégia adaptativa, ora minimiza o *void* anterior, ora minimiza o *void* posterior, conseguindo, assim, atender a rajadas com rotas de diversos tamanhos.



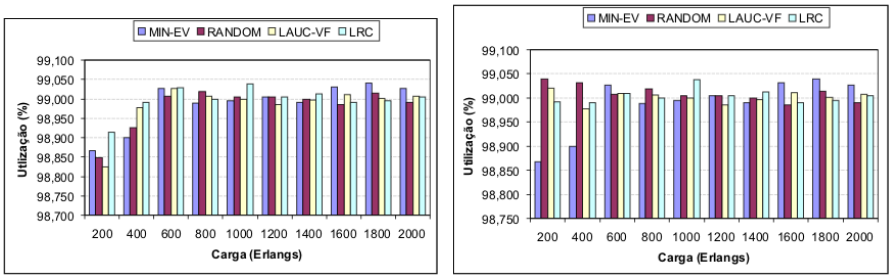
(a) (caso janelas de tempo) (b) (caso volume de tráfego)

Figura 9. Justiça

O efeito de se conseguir atender rotas com diversos tamanhos pode ser verificado nas Figuras 8 e 9. A Figura 8 apresenta o tamanho médio das rotas atendidas, mostrando uma tendência à diminuição do tamanho das rotas atendidas à medida que a carga da rede aumenta. Este é um efeito natural já que com o aumento da disputa por recursos, as rotas com tamanho maior sofrem bloqueio devido ao uso dos recursos pelas rotas de tamanho menor.

Pode-se perceber na figura que o algoritmo LRC apresenta média levemente superior. Na Figura 8 (b), percebe-se uma aproximação maior entre os resultados apresentados pelo algoritmo LRC e os demais algoritmos. Isto se deve ao fato de que o cálculo da probabilidade de acerto (equação 4) é menos preciso quando o algoritmo baseado em volume de tráfego é utilizado. Assim, o cálculo da função de reutilização pode ser menos preciso.

A análise da Figura 8 é complementada pela análise da Figura 9, que mostra o fator de justiça apresentado pelos algoritmos. O fator de justiça é definido como a razão entre a quantidade de rotas bloqueadas de tamanho máximo e o número de rotas bloqueadas de tamanho mínimo. Idealmente, o fator de justiça deve possuir valor um, indicando que o número de rotas de tamanho mínimo e máximo é igual. Quando a quantidade de



(a) (caso janelas de tempo) (b) (caso volume de tráfego)

Figura 10. Utilização da rede

rotas bloqueadas de tamanho máximo é maior do que a quantidade de rotas bloqueadas de tamanho mínimo, o fator de justiça será maior do que um. Análogamente, se a quantidade de rotas de tamanho mínimo é menor, o valor do fator de justiça será menor do que um. O fator de justiça serve como um indicativo do impacto do algoritmo em rotas de diferentes tamanhos, isto é, ao adotar determinada estratégia (minimizar *void* anterior ou posterior) o algoritmo pode penalizar rotas com diferentes tamanhos.

A Figura 9 mostra que os algoritmos LAUC-VF e LRC apresentam fator de justiça mais próximos a um, indicando assim que conseguem operar melhor em redes com rotas de tamanhos variados. Percebe-se, também, uma tendência de decrescimo no fator de justiça do LAUC. Isto acontece pois ao minimizar os *voids* anteriores, o algoritmo pode penalizar rotas menores que utilizam um tempo de ajuste pequeno. Rotas de diferentes tamanhos são beneficiadas pelo algoritmo LRC pois devido à sua estratégia de alocação, quando um nó recebe uma rajada próxima do seu destino, o *void* anterior é usado, ao passo que se a rajada for destinada a um nó mais distante, os *voids* posteriores tendem a ser minimizados.

A Figura 10 apresenta a utilização média da rede. Todos os algoritmos possuem níveis de utilização bem próximos. Porém, percebe-se que o algoritmo LRC apresenta na maior parte da simulação, utilização mais elevada (200, 400, 800, 1000, 1200 e 1400 Erlangs).

6. Conclusões e Trabalhos futuros

Um dos grandes desafios em redes OBS é o projeto de algoritmo de escalonamento rápidos que façam uso eficiente dos intervalos sem reserva ao alocá-los às novas rajadas. Um algoritmo de escalonamento ideal deve ser capaz de processar o pacote de controle rápido o suficiente de forma que um canal seja escolhido antes que a rajada correspondente chegue ao nó. Além disso, deve manter informações sobre os intervalos sem reserva fazendo alocação das rajadas nesses intervalos, garantindo, assim, o aumento da utilização da rede. Desta forma, é papel do algoritmo de escalonamento manter a baixa probabilidade de bloqueio da rede, alocando eficientemente os recursos para as rajadas, sem penalizar a alocação de recursos para as rajadas futuras.

Neste artigo, um novo algoritmo de escalonamento para redes OBS foi introduzido. O algoritmo usa informações relativas a topologia da rede e as rotas por onde as rajadas trafegarão para determinar o canal que será alocado. Resultados derivados via

simulação mostram que o algoritmo possui baixa probabilidade de bloqueio quando comparado a outros algoritmos apresentados na literatura.

Nos trabalhos futuros, serão incorporados funções de probabilidade de acerto sob tráfego com dependências de longa duração.

Referências

Chen, Y., Qiao, C., and Xiang, Y. (2004). Optical burst switching (obs): A new area in optical networking research. *IEEE Network*, 18:16–23.

Ge, A., Callegati, F., and Tamil, L. S. (2000). On optical burst switching and self-similar traffic. *IEEE Communications Letters*, 4(3):98–100.

Maranhão, J., Soares, A., and Giozza, W. F. (2007). Estudo das arquiteturas de conversão de comprimento de onda em redes wdm com comutaçãoo de rajadas Ópticas. In *XXV SBRC*, pages 133–146.

Murty, C. and Gurusamy, M. (2002). *WDM Optical Networks: Concepts, Design and Algorithms*. Prentice Hall.

Papoulis, A. (2002). *Probability, Random Variables and Sthocastic Process*, chapter 3. McGraw-Hill.

Qiao, C. and Yoo, M. (2000). Choices, Features and Issues in Optical Burst Swirching (OBS). *Optical Network Magazine*, 1:36–44.

Turner, J. (1999). Terabit burst switching. *Journal of High Speed Networking*, pages 3–16.

Xiong, Y., Vandenhoute, M., and Cankaya, C. (1999). Design and analysis of optical burst-switched networks. In *SPIE'99 Conf. All Optical Networking: Architecture, Control and Management Issues*, volume 3843, pages 112–119.

Xiong, Y., Vandenhoute, M., and Cankaya, C. (2000). Control architecture in optical burst-switched wdm networks. *IEEE Journal of Selected Areas on Communications*, pages 1838–1851.

Xu, J., Qiao, C., Li, J., and Xu, G. (2003). Efficient channel scheduling algorithms in optical burst switched networks. In *IEEE INFOCOM*.

Xu, J., Qiao, C., li, J., and Xu, G. (2004). Efficient burst scheduling algorithms in optical burst-switched networks using geometric techniques. *IEEE Journal on Selected Areas in Communications*, 22:1796–1811.

Yu, X., Chen, Y., and Qiao, C. (2002). Study of traffic statistics of assembled burst traffic in optical burst switched networks. In *Opticomm*, pages 149–159.

Yu, X., Li, J., Cao, X., Chen, Y., and Qiao, C. (2004). Traffic statistics and performance evaluation in optical burst switched networks. *Journal of Lightwave Technology*, 22(12):2722–2738.