Um algoritmo ótimo para escalonamento de canais em lote em redes OBS

Gustavo B. Figueiredo[1]*, **Eduardo C. Xavier**[1]†, **Nelson L. S. da Fonseca**[1]

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

{gustavo,ecx,nfonseca}@ic.unicamp.br

Abstract. *This paper presents an optimal batch scheduling algorithm called BATCHOPT. The algorithm considers both the requests being processed in the current batch and the requests previously scheduled to find an optimal solution. It is also proposed a batch creation strategy. Results obtained via simulations show that the BATCHOPT algorithm produces good performance when compared to other proposed existing algorithms.*

Resumo. *Este artigo apresenta um algoritmo ótimo para escalonamento em lote de canais para redes OBS, denominado BATCHOPT. O algoritmo considera tanto as requisições sendo processadas quanto as requisições já alocadas para produzir uma solução ótima para o problema. Além do algoritmo BATCHOPT é apresentado também um esquema de formação de lote que atua como extensão do protocolo JET. 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. Assim, 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.

O principal protocolo de reserva de recursos utilizado em redes OBS é o JET (*Just-Enough-Time*) [Qiao and Yoo 2000]. No JET, as rajadas são enviadas ao núcleo da rede T unidades de tempo após a transmissão do pacote de controle correspondente. Esse período, denominado tempo de ajuste, é usado para que a rajada de dados não chegue a um nó antes do término do processamento do pacote de controle. O instante de

*Este trabalho teve apoio financeiro da FAPESB
†Este trabalho teve apoio financeiro da Fapesp, Faepex and CNPq

início da reserva é o instante esperado da chegada da rajada no nó e a duração da reserva corresponde à duração da rajada.

Ao receber um pacote de controle, os nós da rede OBS devem reservar um canal de dados para a rajada. Para tanto, um algoritmo de escalonamento de canal é utilizado. Dado uma lista de reserva de recursos e uma lista de canais disponíveis, o algoritmo de escalonamento de canal é responsável por determinar os intervalos de tempo nos quais os canais de dados devem ser reservados. Diferentes critérios podem ser usados pelos algoritmos de escalonamento de canais usados nas redes OBS sendo que a maioria destes objetiva maximizar as chances de acomodar rajadas futuras.

Devido ao uso de diferentes tempos de ajustes para rajadas pertencentes à diferentes pares origem-destino, é possível que 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*. Na modelagem do problema, a ocupação de cada um dos W canais de um enlace pode ser vista 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 $b = (t', t'')$, onde t' e t'' são os instantes de início e término da rajada, se, e somente se, $s_j \leq t'$ e $e_j \geq t''$. Assim, um dos principais problemas em redes OBS é o projeto de algoritmos de escalonamento rápidos, para que não sejam criadas filas de pacotes de controle nos nós OBS, o que pode impactar negativamente no atraso fim-a-fim experimentado na rede. Além disso, tais algoritmos devem realizar boas alocações nos intervalos sem reserva para as novas rajadas a fim de que se possa minimizar a probabilidade de bloqueio e, consequentemente, aumentar a utilização da rede.

No que tange aos algoritmos de escalonamento, a minimização da probabilidade de bloqueio em uma rede OBS pode ser alcançada através da maximização das chances de acomodação de rajadas futuras. 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. Os algoritmos de escalonamento propostos em [Turner 1999, Xiong et al. 1999, Xiong et al. 2000, Xu et al. 2003b, Xu et al. 2003a] adotam uma estratégia gulosa ao fazerem a reserva dos recursos para garantir boas chances de acomodação da rajada subsequente. A estratégia é dita gulosa pois cada requisição é processada individualmente, usando somente as informações disponíveis no momento do processamento. Não existe, consequentemente, nenhum tipo de garantia de que a próxima rajada será alocada com sucesso.

A incerteza associada à alocação dos recursos nas redes OBS advém do fato de que não se sabe *a priori* os instantes de início e término e nem a prioridade das requisições que chegarão no futuro. A perda desnecessária de rajadas é possível de ocorrer por melhor que seja a estratégia adotada pelos algoritmos de escalonamento. Nesse sentido, uma estratégia ideal consiste em armazenar todas as requisições de reserva e processá-las de uma vez. Dessa forma, a melhor escolha pode ser tomada.

Tal estratégia é infactível, dado que o tempo que se pode armazenar requisições de reserva é limitado pela chegada das rajadas correspondentes, no entanto pode ser aproximada. Com este objetivo, uma nova classe de algoritmos foi proposta em [Kaheel and Al-

nuweiri 2005] e em [Charcranoon et al. 2003]: a classe dos algoritmos de escalonamento em lote. Nesta classe de algoritmos, o máximo possível de requisições são agrupadas em lotes e processados de uma só vez, garantindo-se que um número maior de requisições seja acomodado. O problema das abordagens presentes em [Kaheel and Alnuweiri 2005] e em [Charcranoon et al. 2003] é que os algoritmos propostos para escalonamento de um lote são heurísticas, e, portanto, nem sempre produzem os melhores resultados em termos de minimização da probabilidade de bloqueio.

Além disso, esses algoritmos consideram apenas o conjunto de requisições no lote e os períodos de ocupação e disponibilidade dos canais de dados, impondo assim uma restrição adicional que não permite que todos os canais possam acomodar uma requisição específica. Isso aumenta a complexidade do problema dado que nem todas as requisições podem ocupar os canais durante certos *voids*. É necessário, portanto, saber para cada *void* quais as requisições que podem ser alocadas e dentre elas quais maximizam a soma dos pesos representando as prioridades das rajadas de dados.

O presente artigo apresenta um algoritmo ótimo para o escalonamento em lote de canais em redes OBS, denominado BATCHOPT. O algoritmo considera além das requisições em processamento, o conjunto de requisições previamente alocadas. Isso permite uma modelagem única do problema de escalonamento em lote de canais dado que todos os canais podem ser usados por todas as requisições diminuindo, assim, as restrições do problema e permitindo-se usar um algoritmo ótimo e rápido, cuja complexidade depende somente do número de requisições sendo processadas. Outra característica importante do BATCHOPT é que ele é o único algoritmo de escalonamento em lote para redes OBS capaz de lidar com pesos que podem ser associados a prioridade do tráfego gerado por usuários que requerem qualidade de serviço.

Um outro fator de grande importância para os algoritmos de escalonamento em lote em redes OBS é a estratégia de formação de lote que visa determinar quais requisições serão processadas em um lote e qual a periodicidade de processamento dos lotes na rede. Os algoritmos apresentados em [Kaheel and Alnuweiri 2005] processam os lotes em intervalos de tempo fixo, o que pode ser problemático em redes OBS, já que pode ocasionar perdas desnecessárias na rede devido ao não processamento de requisições dentro do tempo exigido. Neste trabalho, uma nova estratégia de formação de lote é proposta através de adaptações no protocolo JET. Com estas adaptações, a coexistência dos algoritmos de escalonamento em lote com os algoritmos de escalonamento gulosos é viável, o que torna a rede OBS mais robusta e flexível.

O resto do artigo está organizado como segue: a seção 2 apresenta as definições e conceitos envolvidos no problema de escalonamento em lote. A seção 3 revisa os algoritmos de escalonamento de canais em lote existentes na literatura. A seção 4 apresenta o algoritmo BATCHOPT. A seção 5 apresenta uma estratégia de formação de lote. A seção 6 mostra exemplos numéricos e, por fim, a seção 7 apresenta as conclusões.

2. Algumas Definições

Denota-se por $G = (V, E)$ um grafo, onde $V(G)$ é o conjunto de vértices de G e $E(G)$ seu conjunto de arestas. Seja $u \in V(G)$ um vértice de G, a ***adjacência*** de u é definida como: $Adj(u) = \{v \in V(G); (u, v) \in E(G)\}$. Um ***subgrafo*** H de G é um grafo com $V(H) \subset V(G)$ e $E(H) \subset E(G)$. O ***grau de*** u no subgrafo H, denotado por $d(u|H)$

corresponde ao número de vértices adjacentes a u no grafo H.

Uma *clique* é um conjunto $C \subset V(G)$ tal que $\forall u, v \in C; (u, v) \in E(G)$. Uma clique C é dita *maximal* se não existir outra clique em G que tenha C como subconjunto.

G é um grafo de intervalos se existir uma correspondência biunívoca entre um conjunto de intervalos $\{I_v\}$ na reta real e o conjunto de vértices, tal que para dois vértices distintos u e v, tem-se que $(u, v) \in E(G) \Leftrightarrow I_v \cap I_u \neq \emptyset$.

Os grafos de intervalos apresentam particularidades que os tornam atraentes para a solução de diversos problemas em combinatória. Dentre tais particularidades estão o seu reconhecimento e coloração em tempo linear, o que faz com que os algoritmos baseados em grafos com tais estruturas sejam extremamente rápidos.

Um exemplo típico da aplicação dos grafos de intervalos é a sua utilização na solução do problema de *job scheduling*, descrito a seguir: Seja $I = \{J_1 = (s_1, e_1, w_1), \ldots, J_n = (s_n, e_n, w_n)\}$ uma lista de n tarefas, onde (s_i, e_i) é o tempo de início e fim da tarefa J_i, e w_i o seu peso. Tem-se k máquinas com mesma capacidade de processamento. Todas as máquinas estão inicialmente livres desde o tempo 0 até mais infinito. O objetivo do problema é selecionar uma sub-lista $I' \subseteq I$ de tarefas de peso máximo, e alocar I' nas k máquinas. O escalonamento gerado deve satisfazer o fato de que em cada máquina não pode existir sobreposição de tempo entre as tarefas.

Caso não seja imposta nenhuma restrição sobre quais tarefas possam ser processadas em cada máquina, tem-se um problema de *job scheduling com máquinas idênticas*. Caso haja restrições de que algumas tarefas não possam ser processados em um determinado conjunto de máquinas, tem-se um problema de *job scheduling com máquinas não idênticas*.

O problema de escalonamento de canais em redes OBS pode ser reduzido ao problema de *job scheduling*, onde os canais e as requisições das redes OBS correspondem, respectivamente, às máquinas e tarefas no problema de *job scheduling*.

No problema de escalonamento de lotes em redes OBS tem-se, formalmente, k canais, e uma lista $I = \{J_1 = (s_1, e_1), \ldots, J_n = (s_n, e_n)\}$ de n rajadas (correspondente a um lote), onde (s_i, e_i) é o tempo de início e fim da rajada J_i. Tem-se também uma lista S de rajadas já previamente alocadas nos canais. A lista S corresponde a rajadas que foram processadas em lotes anteriores. Tem-se, neste problema, que alocar o maior número possível de rajadas (ou o conjunto de rajadas cuja soma dos pesos é máxima) nos k canais. Duas rajadas em um mesmo canal não podem se sobrepor, ou seja, seus tempos de execução não podem ter uma intersecção.

Na seção 3, apresenta-se uma modelagem do problema de escalonamento em redes OBS através da formulação de um problema *job scheduling com máquinas não idênticas*, que é NP-Difícil, e que foi proposto em [Kaheel and Alnuweiri 2005]. Na seção 4 mostra-se como resolver o problema de escalonamento em redes OBS utilizando um algoritmo polinomial para o problema *job scheduling com máquinas idênticas*.

3. Trabalhos relacionados

Em [Kaheel and Alnuweiri 2005], o problema de escalonamento de canais é resolvido por uma formulação do problema de *job scheduling* com máquinas não idênticas. Neste caso,

o conjunto S de rajadas previamente alocadas descrevem intervalos de tempo nos quais algumas rajadas de I não podem ser alocadas. Como algumas rajadas de I não podem ser alocadas em um subconjunto de canais (quando há intersecção com rajadas de S), Kaheel e Alnuweiri assumem uma modelagem direta do problema de escalonamento de canais para o problema de *job scheduling* em máquinas não idênticas.

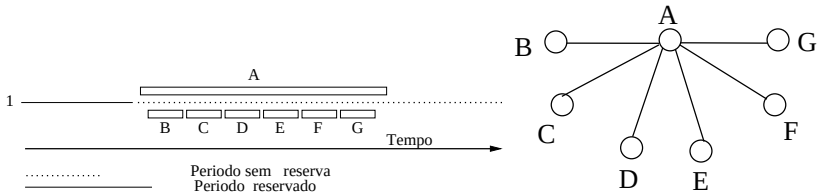
Em [Arkin and Silverberg 1987], apresenta-se um algoritmo ótimo para *job scheduling* com máquinas não idênticas cuja complexidade computacional é da ordem de $O(n^{k+1})$, o que é proibitivamente alto em redes OBS [Kaheel and Alnuweiri 2005], dada a exponencialidade em k. Dessa forma, em [Kaheel and Alnuweiri 2005] são propostas 4 heurísticas para o problema de escalonamento em lote em redes OBS. A complexidade computacional dos algoritmos é de $O(nk\log(N))$, onde n é o número de requisições sendo processadas, k o número de canais e N o número de reservas já realizadas.

A seguir serão brevemente descritas as heurísticas propostas por [Kaheel and Alnuweiri 2005]. Em todas elas, as requisições são vistas como um grafo de intervalos G.

Os vértices $v_1, ..., v_n$ de G são ditos ordenados segundo o critério *smallest-last* se v_i tem o menor grau no subgrafo induzido pelos vértices $v_1, ..., v_i$ (sendo v_n o vértice de G com menor grau). No algoritmo **Smallest Vertex Ordering (SLV)**, os intervalos são alocados na ordem *smallest last*, ou seja, a requisição correspondente a v_1 é alocada ao primeiro comprimento de onda disponível ou descartada, depois v_2 e assim por diante até o processamento de v_n.

A idéia por trás do SLV é que tendo o grafo alguns poucos nós de grau elevado, o processamento prévio desses evitará a necessidade de se usar um número grande de comprimentos de onda.

Contudo, ao contrário do que afirmam os autores de [Kaheel and Alnuweiri 2005], o processamento prévio das requisições com alto grau de intersecção pode ocasionar um número elevado de perdas.



(a) *Padrão de chegadas de requisições* (b) Grafo de intervalos correspondente

Figura 1. Problema ocasionado no escalonamento em lote

A Figura 1 ilustra o problema mencionado: suponha que as requisições se dispõem na forma apresentada na Figura 1(a). O grafo de intervalos correspondente é apresentado na Figura 1(b). Pode-se notar que o vértice "A" é o vértice com maior grau na ordenação *smallest last* (e portanto o primeiro a ser processado). Entretanto, ao se alocar o canal 1 à requisição A, todas as outras requisições serão descartadas.

Caso haja no grafo uma clique com tamanho M e no máximo k canais para alocação das requisições (com $M > k$), necessariamente $M - k$ requisições são descartadas. O

algoritmo **Maximal Cliques First (MCF)** determina, além da ordem de processamento das requisições, quais requisições deverão ser descartadas caso necessário. Para isso, o algoritmo determina todas as cliques maximais de G e as ordena, de forma crescente em relação ao tempo. Seja $\{C_1, C_2, ..., C_m, \}$ o conjunto de cliques maximais de G ordenado tal que $C_i \prec C_j$ para $i < j$ (ou seja existe uma requisição em C_i que possui tempo de início menor ou igual a cada uma das requisições em C_j). O algoritmo processa primeiro as requisições pertencentes à clique C_1 depois à C_2 e assim por diante. Se o tamanho de C_j exceder o número de canais, requisições com o menor tempo de término são descartadas.

Assim como o algoritmo SLV, a estratégia adotada pelo algoritmo MCF pode não produzir os melhores resultados. Considere novamente a Figura 1, a primeira clique a ser processada é a clique formada pelos vértices "A" e "B". Como só existe um canal disponível, a requisição "B" é descartada e a requisição "A" é alocada no canal, o que faz com que todas as outras requisições sejam descartadas.

No algoritmo **Smallest Start-time First Ordering (SSF)**, as requisições são ordenadas de acordo com seu tempo de início. Assim, as requisições com menor tempo de início são processadas primeiro. A mesma situação de perdas apresentada na Figura 1 pode ocorrer no algoritmo SSF. A primeira requisição processada seria a requisição "A", o que resultaria nas perdas descritas.

No algoritmo **Largest Interval First Ordering (LIF)**, as requisições são ordenadas de acordo com o tamanho da rajada que consiste da diferença entre o instante de término e de início da requisição. As requisições que possuem maior tamanho são processadas primeiro. Assim como os algoritmos SLV, MCF e SSF, a situação descrita na Figura 1 pode ocasionar perdas se a requisição "A" possuir maior tamanho que as demais. É importante observar que mesmo que a soma dos tamanhos das demais requisições seja superior ao tamanho da requisição "A", elas não serão consideradas.

O problema com os algoritmos descritos é que eles são baseados em heurísticas que nem sempre produzem os melhores resultados. Assim, o desempenho dos algoritmos depende da estrutura do grafo de intervalos associado ao lote de requisições, fazendo com que em alguns grafos os resultados sejam satisfatórios e em outros não.

4. Escalonamento ótimo de canais em redes OBS

Em [Kaheel and Alnuweiri 2005], assume-se que as requisições a serem processadas que possuem intersecção com requisições já alocadas não podem ser alocadas nos canais destas últimas. Desta forma, assume-se uma redução para o problema de máquinas não idênticas, pois algumas tarefas não podem ser processadas em um determinado subconjunto de máquinas. Entretanto, na estratégia proposta neste trabalho, basta garantir que o conjunto S, de requisições previamente alocadas, seja considerado no processamento do lote atual de tal forma que necessariamente as requisições em S sejam alocadas. Desta maneira, todas as requisições podem ser alocadas em qualquer um dos canais, e portanto tem-se um problema de escalonamento em máquinas idênticas, porém deve-se garantir que as requisições em S sejam alocadas. Tem-se, portanto, um algoritmo ótimo para o escalonamento de lote em redes OBS, cuja complexidade é $O(n^2 \log n)$, onde n corresponde ao número de tarefas consideradas.

Nesta seção, descreve-se um algoritmo que produz uma solução ótima para o seguinte problema: seja $I = \{J_1 = (s_1, e_1, w_1), \ldots, J_n = (s_n, e_n, w_n)\}$ uma lista de tarefas,

onde (s_i, e_i) é o tempo de início e fim da tarefa J_i, e w_i o seu peso. O algoritmo recebe um conjunto de k máquinas idênticas e também uma lista S de tarefas já alocadas nessas k máquinas. O objetivo do problema é selecionar uma sub-lista $I' \subseteq I$ de tarefas de peso máximo, e alocar I' nas k máquinas. O escalonamento gerado deve satisfazer o fato de que em cada máquina não pode existir sobreposição de tempo entre as tarefas.

Em [Arkin and Silverberg 1987] apresenta-se um algoritmo para o problema de *job scheduling* em máquinas idênticas. A seguir, expõe-se como adaptar este algoritmo para considerar a lista S de tarefas já alocados nas máquinas. O algoritmo de Arkin e Silverberg é, inicialmente, apresentado e a adaptação é, posteriormente, descrita.

Seja $I = \{J_1, \ldots, J_n\}$ um conjunto de tarefas, cada J_i com peso w_i e intervalo (s_i, e_i). O algoritmo considera o grafo de intervalos dado pelos intervalos correspondentes as tarefas. Depois, o algoritmo ordena todas as cliques maximais do grafo de intervalo de acordo com o tempo. Na Figura 2, um exemplo desta primeira construção é apresentado.

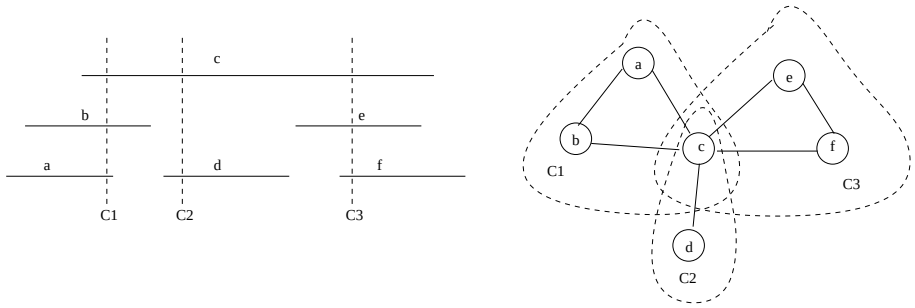


Figura 2. Requisições são representadas na esquerda. Na direita o grafo de intervalos correspondente e as suas cliques subjacentes.

Assume-se que as cliques $C_1, \ldots, C_r$ são ordenadas em forma crescente na linha do tempo. Dado este grafo de intervalos, uma rede de fluxos G' é construída como descrito a seguir: crie um nó v_0 e para cada clique C_j ($j = 1, \ldots, r$) crie um nó v_j . Crie arcos (v_j, v_{j-1}) para cada clique C_j, com custo 0 e capacidade infinita. Seja M o tamanho da clique máxima dentre as cliques $C_1, \ldots, C_r$. Para cada clique C_j, adiciona-se um arco (v_{j-1}, v_j) de custo 0 e capacidade $M - |C_j|$ que representa um *dummy job*. Para cada tarefa J_i pertencente às cliques $C_j, \ldots, C_{j+l}$, adiciona-se um arco (v_{j-1}, v_{j+l}) com capacidade 1 e custo w_i que representa todas as cliques às quais a tarefa J_i aparece (de C_j até C_{j+l}). O objetivo é encontrar um fluxo de $M-k$ unidades e de custo mínimo de v_0 até v_r no grafo construído. Os arcos que forem usados para acomodar fluxo na computação do algoritmo de fluxo máximo e de custo mínimo, correspondem as tarefas que não são alocadas.

Um exemplo da construção de G' descrita para o grafo de intervalo da Figura 2 é dado na Figura 3. Os arcos partindo de v_0 em direção a v_r representam as requisições.

Deseja-se agora adaptar o algoritmo para que o conjunto S de requisições previamente alocadas seja considerado. Para resolver este problema, faz-se a mesma construção descrita do algoritmo de Arkin e Silverberg, considerando, entretanto, o conjunto inteiro de requisições $(S \cup I)$. Para cada requisição $J_i \in S$, assume-se que seu peso é infinito. Desta forma, pode-se garantir que as requisições em S permanecerão alocadas.

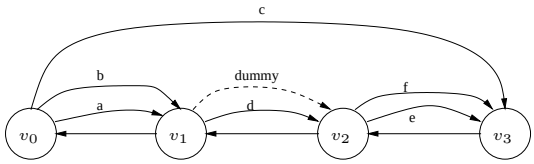


Figura 3. Exemplo de Rede de fluxo.

O algoritmo BATCHOPT possui complexidade de $O(n^2 \log n)$, onde $n = |S \cup I|$. Nota-se que diferentemente dos demais algoritmos, a complexidade do BATCHOPT depende somente do número de requisições sendo processadas. O algoritmo BATCHOPT é formalmente apresentado no Algoritmo 1.

Algoritmo 1 BATCHOPT

ENTRADA

k canais de saída de um nó i, um conjunto I de requisições a serem processadas em um lote e um conjunto S de requisições já alocadas cujo período se intersecta com o período das requisições em I.

SAÍDA

Um conjunto I' de peso máximo.

BATCHOPT

1: Atribua peso infinito para cada requisição em S.
2: Construa o grafo de intervalos G representando $I \cup S$.
3: Ordene as cliques maximais de G de acordo com o tempo.
4: Construa a rede de fluxo G' de acordo com a seção 4.
5: Calcule o fluxo de $M - k$ que possua custo mínimo.
6: Aloque todas as requisições cujos arcos representados em G' não receberam fluxo.

Teorema 1 *O algoritmo BATCHOPT resolve otimamente o problema de escalonamento em lote*

Prova. Ver apendice □

5. Estratégia de formação de lote e adaptação do protocolo de reserva

Como mencionado, em [Kaheel and Alnuweiri 2005] são propostos algoritmos para escalonamento de canais em lote. Contudo, tais algoritmos não são beneficiados por nenhuma estratégia de formação de lote, o que pode também contribuir para o aumento da probabilidade de bloqueio obtida.

A estratégia de formação de lote visa determinar quais requisições devem compor um lote e quando um lote deve ser processado. Todos os algoritmos discutidos na seção 3 processam todas as requisições que chegaram dentro de período de tempo fixo Δ, denominado janela de aceitação de requisições. O escalonador verifica periodicamente a presença de requisições para serem processadas.

O problema com essa abordagem é que em redes OBS o tempo de ajuste é variável e depende do tamanho da rota entre o nó onde o pacote de controle está sendo processado e o destino final da rajada. Assim, em uma topologia de rede real com muitos pares origem-destino, um nó pode receber requisições oriundas desses pares com diferentes tempos de

ajuste. Se uma requisição possui tempo de ajuste menor que a janela de aceitação, a rajada correspondente chegará ao nó antes do processamento do pacote de controle. É, consequentemente, difícil ajustar a janela de aceitação para garantir que todas as requisições sejam processadas antes da respectiva rajada.

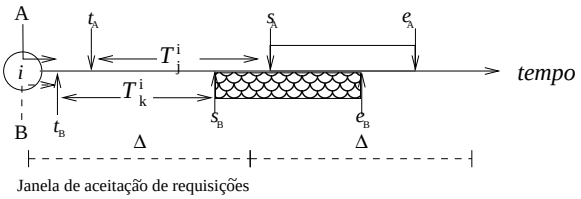


Figura 4. Problema do uso da janela de aceitação de rajadas fixa

O problema descrito é ilustrado na Figura 4. Seja t_r o instante de chegada da requisição r, Δ a janela da aceitação de requisições, e T_j^i o tempo de ajuste da requisição r ao chegar no nó i com destino j, como definido pelo protocolo JET. O pacote de controle associado à rajada A chega no tempo t_A e o pacote de controle associado à rajada B no tempo t_B. Percebe-se que o tempo de chegada da rajada B, s_B, é anterior ao término da janela de aceitação de requisições. Assim, descarta-se a rajada B. O cenário descrito decorre da inexistência de integração entre o protocolo de reservas utilizado (nesse caso o JET) com a estratégia de formação de lote.

No presente trabalho, propõe-se uma estratégia de formação de lote que pode ser considerada uma extensão do protocolo JET para o uso com algoritmos de escalonamento de canais em lote. Nesta estratégia, denominada JET-Δ, a janela de aceitação de requisições é adicionada ao tempo de ajuste da requisição, como ilustrado na Figura 5. Além disso, o instante de tempo em que se encerra o período de acomodação de requisições e se inicia o processamento das requisições (denominado limiar de processamento) é definido de forma que este nunca seja posterior ao instante de chegada da rajada mais próxima. Dessa forma, a estratégia garante que todas as requisições serão processadas antes da chegada da rajada mais próxima.

Sejam t_r, Δ e T_j^i como definidos anteriormente. O limiar de processamento (L) é definido como $L = (t_R + \Delta) - \delta$, onde $R = min_r\{t_r + T_j^i + \Delta\}$ e δ é o tempo de processamento do lote. Em redes OBS, o tempo de processamento de requisições é da ordem de alguns microssegundos, enquanto o tempo de ajuste e a janela de aceitação de requisições são da ordem de milissegundos [Rodrigues et al. 2005]. Assim, o limiar de processamento é dominado pela janela de aceitação. É importante ainda notar que à medida que novas requisições chegam ao nó i, o limiar de processamento deve ser atualizado se a nova requisição R' possui $\{t_{R'} + T_j^i + \Delta\}$ inferior ao da requisição R.

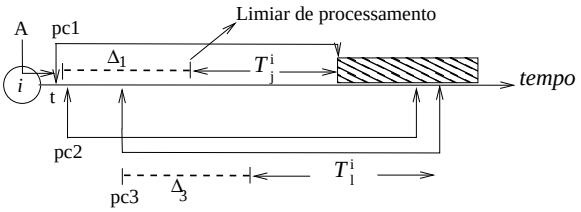


Figura 5. Estratégia de alocação de lote JET-Δ

Assim, com a determinação de L o protocolo garante que todas as requisições serão processadas antes da chegada da rajada mais próxima e assim nenhuma rajada será desnecessariamente descartada, dado que o limiar é determinado de forma que este nunca seja maior que o instante de chegada da rajada mais próxima.

Quando o limiar de processamento é alcançado, todas as requisições são processadas pelo algoritmo de escalonamento em lote utilizado. Findo o processamento, assim como em [Elhaddad et al. 2003], as requisições alocadas são agrupadas de acordo com o destino e encaminhadas em um único pacote de controle para evitar sobrecarga nos canais de controle.

A adição da janela de aceitação ao tempo de ajuste pode aumentar o atraso fim-a-fim experimentado pelos pacotes que compõem das rajadas. Todavia, esse problema pode ser minimizado levando-se em consideração o atraso máximo fim-a-fim tolerado. Nesse caso, se D é o atraso máximo fim-a-fim tolerado, T_j^s o tempo de ajuste nó nó de origem em relação ao destino j e H o número de nós na rota entre a origem e o destino, a janela de aceitação de requisições em cada nó é definida como:

$$\Delta = \frac{D - T_j^s}{H} \tag{1}$$

Uma forma de evitar o aumento do atraso fim-a-fim, é usar esquemas de montagem com predição do tamanho da rajada, como em [Morato et al. 2001] e em [Liu et al. 2003]. Assim, o atraso Δ é compensado com o envio antecipado do pacote de controle.

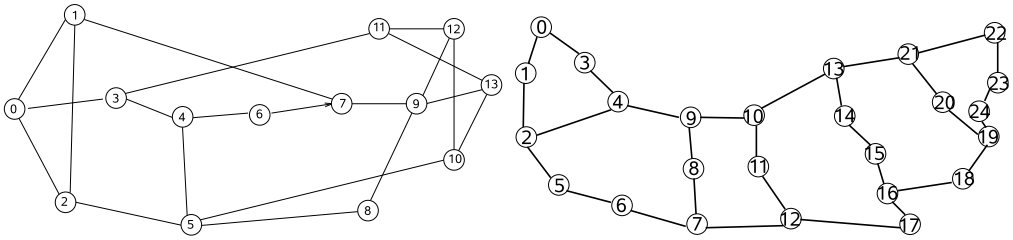
Por fim, note que, como mencionado, o protocolo JET-Δ funciona como uma extensão do protocolo JET. Assim, o JET-Δ pode ser usado tanto pelos algoritmos de escalonamento em lote apresentados neste artigo quanto pelos algoritmos de escalonamento gulosos apresentados em [Turner 1999, Xiong et al. 1999, Xiong et al. 2000, Xu et al. 2003b, Xu et al. 2003a], bastando para isso que $\Delta = 0$. O uso do JET-Δ permite, assim, que diferentes classes de algoritmos de escalonamento coexistam dentro da rede OBS, aumentando a sua robustez e flexibilidade.

6. Exemplos Numéricos

Simulações foram conduzidas para avaliar o desempenho dos algoritmos. 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 10.000 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%.

As simulações foram realizadas com as topologias NSFNet e Abilene, apresentadas 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. O tempo de processamento do pacote de controle e configuração da malha de comutação dos comutadores é de $50\mu s$.

Todos os nós da rede são potencialmente uma origem ou um destino do tráfego, o que significa dizer que a cada requisição gerada, uma origem e um destino são sorteados de acordo com uma distribuição uniforme e uma rota é então criada entre o par. O tráfego



(a) *Backbone* NSFNet (b) *Backbone* Abilene

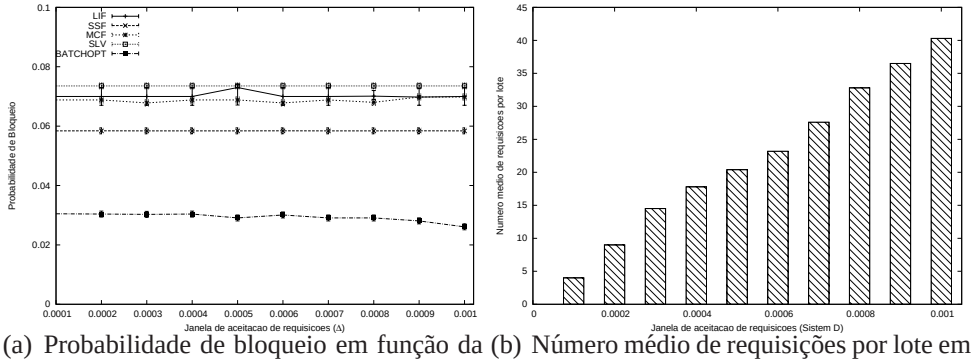
Figura 6. Topologias usadas nas simulações

é sempre gerado nos nós de origem e segue a distribuição de Poisson. O tamanho das rajadas varia de acordo com a distribuição exponencial negativa. A carga de tráfego considerada nas simulações variou de 200 a 2000 Erlangs com incremento de 200 Erlangs.

Como mencionado na seção 5, os algoritmos de escalonamento apresentados em [Kaheel and Alnuweiri 2005] utilizam uma estratégia de formação de lote ineficiente. Assim, para avaliar o desempenho dos algoritmos, todos os algoritmos avaliados usam a estratégia de formação de lote JET-Δ proposta neste trabalho.

Foi avaliado, primeiramente, o impacto da janela de aceitação de requisições (Δ) na estratégia de formação de lote JET-Δ. Para tal, a QoS foi desconsiderada e todas as requisições que adentraram a rede tinham peso unitário. Em tais simulação, a carga de tráfego na rede foi de 1000 Erlangs.

A Figura 7 mostra a probabilidade de bloqueio obtida pelos algoritmos em função de Δ (Figura 7(a)) e o número médio de requisições por lote em função de Δ (Figura 7(b)). Percebe-se que os algoritmos LIF, MCF, SSF e SLV não são sensíveis à alterações na janela de aceitações. Isso acontece pois seu critério para estabelecimento da ordem em que as requisições são alocadas é fixo. Com isso, o desempenho de tais algoritmos depende mais da estrutura do grafo de intervalos associado do que do número de requisições sendo processadas, ou seja, em alguns grafos de intervalos (lotes) tais algoritmos conseguem minimizar as perdas e em outros não.



(a) Probabilidade de bloqueio em função da janela da aceitação de requisições (b) Número médio de requisições por lote em função da janela de aceitação de requisições

Figura 7. Impacto de Δ nos algoritmos de escalonamento

O algoritmo BATCHOPT, por outro lado, consegue beneficiar-se do aumento do número médio de requisições em cada lote. Isso por que, à medida em que mais

requisições são processadas de uma vez, maior o horizonte de tempo que ele tem conhecimento sobre as requisições e mais informações ele possui acerca das intersecções entre tais requisições. Com isso, a construção proposta na seção 4 reflete com mais precisão as dependências entre as requisições e o algoritmo consegue descartar as requisições cuja aceitação seria danosa para o desempenho da rede.

Como quanto maior a janela de aceitação de requisições, melhor o desempenho do algoritmo BATCHOPT, na prática tal parâmetro pode ser ajustado dependendo dos requisitos temporais do tráfego transportado e determinado pela Equação 5. No restante das simulações, Δ foi usado com valor de $1ms$.

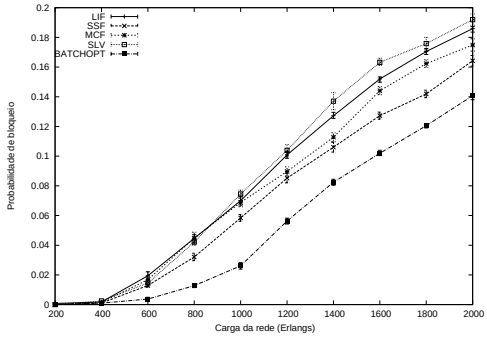


Figura 8. Probabilidade de bloqueio em função da carga de tráfego na rede

A Figura 8 mostra a probabilidade de bloqueio em função do aumento da carga na rede em uma rede sem QoS (todas as requisições possuem o mesmo peso). Com o aumento da carga de tráfego na rede, o tamanho das cliques maximais do grafo de intervalos associado às requisições é aumentado, o que explica o crescimento das perdas. Entretanto, o crescimento das perdas é menor quando o algoritmo BATCHOPT é utilizado. Isso porque, como todas as requisições têm peso unitário, o conjunto de requisições selecionado pelo algoritmo é sempre o conjunto com a maior cardinalidade e assim sempre o menor número possível de requisições é descartada.

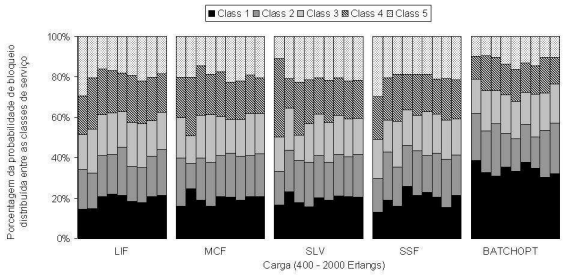


Figura 9. Porcentagem da probabilidade de bloqueio distribuída entre as classes de serviço

Foram realizadas, também, simulações levando-se em consideração requisitos de QoS. Para tal, foram usadas 5 classes de serviço de forma que a a classe i é mais prioritária que a classe j se $i > j$. Cada requisição gerada, foi associados a uma classe de serviço de maneira uniformemente distribuída. Pesos foram atribuídos a cada classe de serviço da seguinte forma: $w_1 = 1, w_2 = 2, w_3 = 4, w_4 = 8, w_5 = 16$, onde w_i é o peso das requisições de classe i.

A Figura 9 apresenta o percentual das perdas apresentadas divididas por classe de serviço. É possível perceber que o algoritmo BATCHOPT é capaz de beneficiar de forma mais eficaz as classes de mais alta prioridade. A soma do percentual de perdas das classes 4 e 5 para o algoritmo BATCHOPT foi de 33% (20% da classe 4 e 13% da classe 5). Os algoritmo LIF apresentou total de 45% (23% da classe 4 e 22% da classe 5), o algoritmo MCF apresentou total de 46% (26% da classe 4 e 20% da classe 5), o algoritmo SLV apresentou 45% (27% da classe 4 e 18% da classe 5) e o algoritmo SSF apresentou média de 53% (28% da classe 4 e 25% da classe 5). Com isso, é possível perceber que o algoritmo BATCHOPT consegue oferecer tratamento mais diferenciado às classes de serviço que os demais algoritmos avaliados.

7. Conclusões

Um dos grandes desafios em redes OBS é o projeto de algoritmos de escalonamento eficientes, capazes de reduzir a probabilidade de bloqueio, de forma que a rede seja capaz de oferecer garantias mínimas de QoS. Além disso, tais algoritmos devem manter informações sobre os intervalos sem reserva fazendo alocação das rajadas nesses intervalos, garantindo, assim, o aumento da utilização da rede.

Neste artigo, um algoritmo ótimo de escalonamento em lote para redes OBS foi proposto. O algoritmo usa uma construção especial extraída do grafo de intervalos associado às requisições para realizar a alocação das requisições. Além disso, foi proposta uma extensão do protocolo JET para atuar como estratégia de formação de lotes. Resultados de simulação mostraram que o algoritmo BATCHOPT possui desempenho superior aos algoritmos semelhantes existentes na literatura. Como trabalhos futuros, serão realizadas simulações utilizando tráfego real obtido através de traces de tráfego.

Referências

Arkin, E. M. and Silverberg, E. B. (1987). Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18:1–8.

Charcranoon, S., El-Bawab, T. S., Cankaya, H. C., and Shin, J. D. (2003). Group scheduling for optical burst switched (obs) networks. In *Globecom*, pages 2745–2749.

Elhaddad, M., Melhem, R., Znati, T., and Basak, D. (2003). Traffic shaping and scheduling for obs-based ip/wdm backbones. In *IEEE Opticom*, volume 5285, pages 336–345.

Kaheel, A. and Alnuweiri, H. (2005). Batch scheduling algorithms for optical burst switching networks. *Lecture notes in Computer Science*, 3462/2005:90–101.

Liu, J., Ansari, N., and Ott, T. J. (2003). Frr for latency reduction and qos provisioning in obs networks. *Journal on Selected Areas in Communications*, 21:1210–1219.

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.

Morato, D., Aracil, J., and Diez, L. A. (2001). On linear prediction of internet traffic for packet and burst switchin networks. In *International Conference in Computer Communications Networks*, pages 138–143.

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

Rodrigues, J. J. P. C., Freire, M. M., and Pascal, L. (2005). Impact of setup message processing and optical switch configuration times on the performance of ip over optical burst switching networks. *Lecture notes in computer science*, 3733(20):264–273.

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. (2003a). Efficient channel scheduling algorithms in optical burst switched networks. In *IEEE INFOCOM*, volume 3, pages 2268– 2278.

Xu, L., Perros, H. G., and Rouskas, G. N. (2003b). A Simulation Study of Access Protocols for Optical Burst-Switched Ring Networks. *Computer Networks*, 41:143–160.

8. Apendices

Teorema 1 *O algoritmo BATCHOPT resolve otimamente o problema de escalonamento em lote*

Prova. Primeiro, será provada a otimalidade do algoritmo. Como para cada requisição pertencente a S atribuí-se um peso infinito, o algoritmo de fluxo de custo mínimo não usará as arestas correspondentes as requisições em S; logo, necessariamente, as requisições em S permanecerão alocadas. Além disso, sabe-se que dentre as requisições em I é alocado um subconjunto de peso máximo.

Agora, mostra-se que as mudanças de canais associados a requisições cujas rajadas já estão em transito não são necessárias. Suponha que o algoritmo produz uma solução cujo início se dá em um instante de tempo t, com um conjunto S de requisições já alocadas. Seja $S' \subseteq S$ o conjunto de requisições com tempo de início menor que t. Isto significa que cada requisição em S' já está sendo transmitida e seu canal não pode ser mudado. Entretanto, note que para as requisições em $S \setminus S'$ os canais podem ser mudados sem problemas. O algoritmo BATCHOPT selecionou um conjunto I' de novas requisições tal que para qualquer tempo $t \leq t'$ existem, no máximo, k requisições de $I' \cup S$ que se intersectam com t'. Pode-se então ordenar as requisições em $I' \cup S$ por seus tempos de início e alocar as requisições nesta ordem aos canais disponíveis. Isto gera um escalonamento factível desde que em qualquer tempo t' existem no máximo k requisições intersectando com ele, e assim existe um canal disponível em cada vez que uma requisição é alocada. Note que as requisições em S foram previamente alocadas em um escalonamento factível. Assim, desde que as requisições em S' têm o menor tempo de início dentre as requisições em $S \cup I'$, elas serão processadas primeiramente e assim não haverá necessidade de mudança de canal. □