Mecanismos de controle em redes de comutação de rajadas ópticas

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Gustavo Bittencourt Figueiredo e aprovada pela Banca Examinadora.

Campinas, 6 de Dezembro de 2009.

Prof. Dr. Nelson Luís Saldanha da Fonseca
(IC/UNICAMP) (Orientador)

Dr. Marcos Rogério Salvador (Co-Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Figueiredo, Gustavo Bittencourt
F469m Mecanismos de controle em redes de comutação de rajadas ópticas / Gustavo Bittencourt Figueiredo -- Campinas, [S.P. : s.n.], 2009.

Orientadores : Nelson Luis Saldanha da Fonseca ; Marcos Rogério Salvador
Tese (Doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Desempenho - Avaliação. 2. Redes de computadores. 3. Séries de redes ópticas. 4. Comutação de rajadas ópticas (Transmissão de dados). 5. Algoritmos. I. Fonseca, Nelson Luis Saldanha da. II. Salvador, Marcos Rogério. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: Control mechanisms for optical burst switched networks

Palavras-chave em inglês (Keywords): 1. Performance - evaluation. 2. Computer networks. 3. Optical networks series. 4. Optical burst switching (Data transmission). 5. Algorithms.

Área de concentração: Redes de computadores

Titulação: Doutor em Ciência da Computação

Banca examinadora: Prof. Dr. Nelson Luis Saldanha da Fonseca (IC/UNICAMP)
Prof. Dr. César Augusto Viana Melo (DCC/ UFAM)
Prof. Dr. Edmundo Roberto Mauro Madeira (IC/UNICAMP)
Prof. Dr. Flávio Keidi Miyazawa (IC/UNICAMP)
Prof. Dr. Hélio Waldman (UFABC)
Prof. Dr. Otto Carlos Muniz Bandeira Duarte (COPPE/UFRJ)

Data da defesa: 16/10/2009

Programa de Pós-Graduação: Doutorado em Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 16 de outubro de 2009, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. César Augusto Viana Melo
DCC / Universidade do Estado do Amazonas

Prof. Dr. Edmundo Roberto Mauro Madeira
IC / UNICAMP

Prof. Dr. Hélio Waldman
FEEC / UNICAMP

Prof. Dr. Otto Carlos Muniz Bandeira Duarte
COPPE / Universidade Federal do Rio de Janeiro

Prof. Dr. Flávio Keidi Miyazawa
IC / UNICAMP

Prof. Dr. Nelson Luis Saldanha da Fonseca
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Mecanismos de controle em redes de comutação de rajadas ópticas

Gustavo Bittencourt Figueiredo*

Janeiro de 2010

Banca Examinadora:

- Prof. Dr. Nelson Luís Saldanha da Fonseca (IC/UNICAMP) (Orientador)
- Prof. Dr. César A. V. Melo (DCC/UFAM)
- Prof. Dr. Edmundo R. M. Madeira (IC/UNICAMP)
- Prof. Dr. Hélio Waldman (UFABC)
- Prof. Dr. Otto C. M. B. Duarte (COPPE/UFRJ)
- Prof. Dr. Flávio K. Miyazawa (IC/UNICAMP)

*Financiado por CAPES, FAPESB.

Agradecimentos

Escrever uma seção de agradecimentos de uma tese tem um significado bastante especial: significa que uma etapa importante da vida foi cumprida (ou quase, pelo menos até a defesa!!) e que, o alívio sentido deve-se a inúmeras contribuições de pessoas que, de uma forma ou de outra, foram importantes nessa trajetória.

Em primeiro lugar, agradeço Deus, que é o detentor de todas as honras. Agradeço pela saúde, pela família, enfim, por TUDO.

Gostaria de agradecer à minha pequena e minha grande família. Cris, Gu e Gui, vocês têm sido de imensa importância em minha vida. Aprendi a ser um pai, segundo Gutinho "bom, engraçado, pirracento, e às vezes, muito estressado", como Gui ainda não fala, acho que é só elogios!! Cris, obrigado pelo amor, paciência e dedicação.

Mãe, obrigado pelo amor incondicional e irrestrito! Sei que tem me dado muito mais, mas só isso já seria muito. Pai, obrigado pelo amor e zelo. Aos meus irmãos, Messias, Mercia, Sandra, Nando, Cal, Tiago e Rafa, saibam que TODOS são amados e muito admirados. Agradeço a vocês pelo amor e ótima convivência. Aos "co-irmãos" (cunhados é passado) Paty, Dudu, que deveriam mesmo ser chamados de irmãos. Um especial ao meu grande sobrinho (o verdadeiro!) Nandi!

Extendo esse agradecimento também à tia Bebé, Tio Juca e aos primos, Rogério, Nem, Jisa, Nadir, Neuri, Katy, e a seus filhos...Acho que todos juntos formaram uma grandissíssima família cuja base moral (e imoral hehe) criou um circulo perfeito de convivência.

Gostaria de agradecer a meu orientador e amigo. Nelson, quero aqui reconhecer que

você se preocupou e ajudou muito além do que o dever lhe exigia. Obrigado também pela franqueza e contribuições. Acho que o aprendizado foi além do campo profissional!

Agradeço aos incontáveis amigos que fiz durante esse (não tão) pequeno período de tempo. Do IC: DanLinux, Cleo, Cesar, Triste, Guido, Zeh, Glauber, Bartho, Neumar, Jorge, André Drummond (o único com sobrenome), Borin (só o sobrenome), Ju Borin (ah...estragou a piada com o Andred!!), Pará, Magrão, "Juzinha", Carlão, Bit... o povo do IC..LRC..(se esqueci alguem aq..eh pq vcs sumiram..apareçam seus sumidos!!! hhehe).De casa (como sei que todo mundo vai querer ver seu nome): S. Jaime, D. Pedrilza, Daniel, Jaque, Leco, Claudia (ops..não era pra ficar assim tão juntos? hehe), Hugo, Jam, Val, Isis, Cothoco, Clari, Leila, Gal, Gabi, Ani, Vivi, DInho, Marcia, Keila..afe..

Por fim, agradeço à CAPES a FAPESB pelo apoio financeiro.

Resumo

A demanda cada vez maior por largura de banda tem levado a implantação de uma Internet de nova geração, com *backbones* com alta capacidade de transmissão baseados nas tecnologia de multiplexação *Wavelength Division Multiplexing* (WDM). Para o eficiente transporte do tráfego da Internet, é necessário o uso de um paradigma de transmissão flexível e capaz de se adequar às flutuações do tráfego da rede. Devido a imaturidade da tecnologia de comutação de pacotes ópticos (do inglês *Optical Packet Switching* - OPS) e das desvantagens da comutação de circuitos ópticos (do inglês *Optical Circuit Switching* - OCS), a comutação de rajadas ópticas (do inglês *Optical Burst Switching* - OBS) é uma opção atrativa, dada a sua flexibilidade, maturidade tecnológica e eficiência. Nas redes OBS, os pacotes IP são agrupados em unidades de transmissão maiores, denominadas rajadas, cuja transmissão é precedida por um pacote de controle que sinaliza, entre outras coisas, o momento em que os recursos devem ser reservados. O processo de reserva de recursos é feito em uma via, o que indica que o transmissor não necessita aguardar confirmação por parte do receptor antes de enviar as rajadas. Se não houver recursos disponíveis no momento da transmissão, a rajada é sumariamente descartada.

O fato do transmissor não esperar confirmação por parte do receptor para a transmissão das rajadas, implica na necessidade de um dimensionamento adequado da rede, sob pena de alta probabilidade de bloqueio. Nesta tese, são propostos diferentes mecanismos de controle para redes OBS que podem ser usados conjuntamente, a fim de melhorar o desempenho da rede.

Esta tese apresenta, inicialmente, um estudo sobre a ocorrência de transformações

nas propriedades estatísticas do tráfego submetido à redes OBS, devido ao processo de montagem de rajadas, que ocorre na borda das redes OBS. Verificou-se que a transformação do tráfego está relacionada à escala limitante do tráfego multifractal. Além disso, verificou-se que o tráfego transformado em monofractal demanda menos recursos da rede. Assim, foi proposto um método para identificação automática da escala limitante de fluxos multifractais, além de um conjunto de algoritmos de montagem capazes de induzir as transformações nas propriedades estatísticas do tráfego.

Foi proposto, também, na tese, um algoritmo adaptativo para escalonamento de canais em redes OBS que aloca comprimentos de onda com menor chance de reutilização por rajadas futuras. Analisou-se também, nesta tese, o problema de escalonamento em lote de canais em redes OBS. Foram propostos dois algoritmos ótimos: um para o caso quando as requisições que transitam pela rede não possuem diferenciação, e outro para quando a rede exige tratamento diferenciado das requisições. Além disso, foi proposta uma estratégia para a formação dos lotes que pode ser considerada uma extensão do protocolo *Just Enough Time* - JET.

Os mecanismos propostos foram avaliados em comparação com outros existentes na literatura. Os resultados obtidos evidenciam ganhos e a adequabilidade para implementação em redes OBS, a fim de melhorar o desempenho destas redes.

Abstract

The growth of the number of Internet users has led to the increase of the bandwidth demand which, consequently, led to the need of adoption of high capacity links in the Internet backbone. Moreover, a flexible switching paradigm is necessary to provide efficient transport of Internet traffic .

Due to the limitations of both optical packet switching (OPS) and optical circuit switching (OCS), optical burst switching (OBS) emerged as an attractive switching choice. In OBS networks, IP packets are aggregated into larger transmission units, called bursts. The transmission of a burst follows the transmission of its associated control packet, which carries among other information, the time that bandwidth should be reserved for that burst. In OBS networks, the process of bandwidth reservation is done in one way, which means that the burst is discarded if there is not enough bandwidth for the transmission of the burst at a node when the burst arrives. Therefore, scheduling and burst assembling mechanisms should be conceived to avoid burst loss as well as to support the quality of service of applications running over an OBS network.

Initially, a study on traffic transformation at the edge of the network was conducted for the derivation of efficient mechanisms. The study aimed at verifying the changes of traffic descriptors due to the assembly of packets into bursts. It was found that the cutoff time scale of multifractal traffic impacts the traffic transformation. Moreover, it was found that these transformations can lead to smaller bandwidth demands. Based on findings, an automatic method was proposed for the identification of the cutoff time scale of multifractal flows, and a set of burst assembly algorithms for inducing such transformations

were introduced.

Furthermore, an adaptive algorithm for channel scheduling that allocates wavelengths with small chances of being reused by future requests was proposed. Two optimal algorithms were introduced for the provisioning of differentiated services. In addition, a batch assembly strategy, which can be considered an extension of JET protocol, was created.

The proposed mechanisms were evaluated in comparison with other mechanisms in the literature via simulation. Results evince that the mechanisms introduced in this thesis are effective for the improvement of OBS networks performance.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A demanda por largura de banda na Internet tem crescido de forma substancial nos últimos anos. Este crescimento tem sido, de certa forma, impulsionado por dois fatores: o surgimento de novas aplicações como HDTV, transmissão de áudio digital e aplicações de telefonia IP, e a necessidade crescente dos provedores de serviço de oferecer serviços cada vez mais rápidos e confiáveis.

Este aumento da demanda por largura de banda tem sugerido a implantação de uma Internet puramente óptica, baseada em WDM (*Wavelength Division Multiplexing*), capaz de operar a elevadas taxas de transmissão. Entretanto, para que estas altas taxas de transmissão no domínio óptico sejam efetivamente exploradas, é necessário minimizar as conversões de sinal entre os domínios óptico e elétrico, possibilitando transmissões fim-a-fim inteiramente no domínio óptico.

Outro fator decisivo sobre o desempenho dessa Internet de nova geração é o paradigma de comutação empregado na transmissão dos dados. Tal paradigma deve ser flexível o suficiente a ponto de se adaptar de forma eficiente às flutuações do tráfego Internet, bem como oferecer infra-estrutura dotada de mecanismos que garantam de forma eficiente a implantação de novos serviços.

Em redes WDM, três paradigmas de comutação têm sido intensamente investigados: comutação de circuitos ópticos (OCS - *Optical Circuit Switching*), comutação de pacotes

ópticos (OPS - *Optical Packet Switching*) e comutação de rajadas ópticas (OBS - *Optical Burst Switching*).

Na comutação de circuitos ópticos, um canal dedicado, denominado circuito, é estabelecido da fonte de dados ao destino através da alocação de comprimentos de onda* da origem ao destino. O processo de alocação de recursos é feito em três etapas distintas: o estabelecimento do circuito, a transmissão dos dados e a liberação do circuito. O estabelecimento dos circuitos pode ser feito de forma "manual" através de um sistema de gerenciamento ou por meio de um processo automático em duas vias, ou seja, a fonte de dados (ou comutador de ingresso) envia um pedido de reserva de recursos (estabelecimento do circuito) e recebe de volta uma resposta sobre o pedido de reserva. Em caso de resposta positiva, os dados podem ser transmitidos sem a necessidade de armazenamento temporário, pelos nós de comutação intermediários ao longo do caminho.

Apesar da relativamente baixa complexidade envolvida na comutação de circuitos ópticos, ela tem sido considerada ineficiente para suporte a tráfego com alta variabilidade devido à sua falta de flexibilidade para lidar com eventuais flutuações no tráfego da rede e mudanças no estado da mesma. Um exemplo é o *overhead* relacionado ao estabelecimento e liberação dos circuitos quando comparado à transmissão efetiva dos dados.

A comutação de pacotes ópticos é o equivalente óptico da comutação eletrônica de pacotes. Na OPS, as informações de controle são transmitidas juntamente com os dados para que as decisões de comutação possam ser tomadas. Apesar de considerado o paradigma ideal de transmissão em redes WDM, as tecnologias para armazenamento temporário de pacotes ópticos são demasiadamente imaturas para que sejam utilizadas em curto e médio prazo.

Diante das razões apresentadas, a comutação de rajadas ópticas pode ser considerada como a principal candidata a paradigma de comutação de redes WDM, a ser usado em uma Internet puramente óptica. Além disso, as redes OBS possuem algumas características que fazem com que o transporte de dados na Internet seja mais eficiente.

*Os termos comprimento de onda e canal serão usados indistintamente durante o restante do texto.

Nas redes OBS, as unidades de transmissão, denominadas rajadas, possuem uma granularidade intermediária entre o circuito e o pacote, pois uma rajada corresponde a um conjunto de pacotes agrupados. O processo de reserva de recursos é feito em uma via, ou seja, um pacote de controle é enviado da origem ao destino, informando sobre a intenção do nó origem em transmitir a rajada. O pacote de controle contém, entre outras informações, o tamanho da rajada e o tempo esperado de sua chegada. Após um tempo de ajuste, a rajada é enviada sem que nenhuma confirmação seja recebida do destino. Se no momento da transmissão das rajadas os recursos não puderem ser reservados, a rajada será sumariamente descartada. Obviamente, este processo de reserva em uma via diminui, substancialmente, o *overhead* relacionado com a reserva dos recursos das redes OBS quando comparado às redes OCS.

Outra característica importante das redes OBS é que o pacote de controle e as rajadas são transmitidos em canais separados denominados, respectivamente, canais de controle e canais de dados. Esta separação habilita a rede OBS a usar as altas taxas de transmissão disponibilizadas pela tecnologia WDM, eliminando, na transmissão dos dados, as conversões entre os domínios óptico e elétrico. Além disso, quando comparada à comutação de circuitos, a comutação de rajadas tem ganhos de multiplexação estatística; e quando comparada a comutação de pacotes, a comutação OBS não requer sincronização de FDLs (*Fiber Delay Lines*). Ainda, a rede OBS suporta transparência de dados e possui custo relativamente baixo.

Apesar das redes OBS oferecerem maior largura de banda e flexibilidade à implantação de uma Internet sobre redes ópticas de nova geração, ainda existem grandes desafios para que se alcance uma considerável melhoria nos serviços oferecidos pela rede. De um modo geral, a alta probabilidade de bloqueio [38] em redes OBS deve-se ao processo de reserva em uma via, pois não há garantias de entrega dos dados transmitidos. O sucesso das reservas remetidas à rede depende da eficiência dos mecanismos de controle, em especial dos algoritmos de escalonamento de canais utilizados.

Assim, para melhorar o desempenho da rede OBS e oferecer garantias de serviços

confiáveis, é necessário dotar a rede com mecanismos mais eficientes, capazes de diminuir a probabilidade de bloqueio da rede e torná-la mais adaptável ao suporte de tráfego Internet.

Nesta tese, são propostos diferentes mecanismos de controle que podem ser usados em conjunto para melhorar o desempenho da rede. O primeiro passo na construção de tais mecanismos foi investigar a relação entre as redes OBS e o tráfego transportado na Internet de nova geração, ou seja, o tráfego IP.

Investigou-se primeiramente o impacto dos mecanismos da rede OBS nas propriedades estatísticas do tráfego IP. É sabido que o tráfego IP pode apresentar diferentes características a depender da escala de tempo em que é observado [4]. Em grandes escalas de tempo, o tráfego IP apresenta comportamento auto-similar monofractal, já em pequenas escalas de tempo o tráfego apresenta comportamento multifractal. Trabalhos anteriores [28,31,33] investigaram a interação entre o tráfego monofractal e os mecanismos da rede OBS. Nesta tese investigou-se qual o impacto dos mecanismos de montagem de rajada no tráfego multifractal, ou seja, se a agregação de pacotes nos nós de borda da rede OBS pode causar mudanças nas propriedades estatísticas do mesmo.

Após análise exaustiva em diversos traços de tráfego coletados na Internet, constatou-se que, de fato, o processo de montagem de rajadas provoca transformações nas estatísticas do tráfego multifractal. Além disso, verificou-se a existência de uma relação entre os parâmetros usados na montagem das rajadas e a escala de tempo limitante, que é a escala a partir da qual o tráfego pode ser modelado como monofractal. Verificou-se também que o tráfego modificado demanda menos recursos da rede do que o tráfego sem transformações. Assim, foram apresentados e discutidos diversos algoritmos de montagem de rajadas capazes de induzir as transformações no tráfego da rede, garantindo uma demanda menor de recursos da rede.

Além disso, para induzir as transformações no tráfego, é necessária a determinação da escala de tempo limitante. Contudo, até então, a detecção de tal escala se dava através de inspeção visual de diagramas log-log do processo original agregado em diferentes escalas

de tempo. Para permitir a detecção da escala limitante e montagem das rajada de forma automática em nós de medição, foi proposto um método para detecção não visual da escala de tempo limitante. Tal método é baseado na análise do coeficiente de determinação de uma regressão linear realizada sobre os pontos do plano log-log do processo incremento agregado em função do intervalo de agregação. O método foi exaustivamente testado, mostrando-se eficaz na detecção da escala de tempo limitante.

Introduz-se também, nesta tese, um algoritmo adaptativo para escalonamento de canais em redes OBS. Os algoritmos de escalonamento reativos, propostos na literatura, usam um critério fixo na escolha do canal a ser utilizado. Entretanto, quando uma estratégia fixa é utilizada, podem ocorrer perdas desnecessárias de rajadas devido a falta de canais para acomodá-las. Dessa forma, o algoritmo proposto, denominado *Least Reusable Channel (LRC)*, faz uso de uma nova abordagem na determinação dos canais. O algoritmo usa uma estratégia adaptativa baseada no conceito de reutilização dos canais. A reutilização de um canal é determinada por uma função que leva em consideração o tamanho dos novos *voids*[†] que seriam gerados caso o canal fosse escolhido para acomodar a rajada cuja requisição está em processamento, além da probabilidade de cada um desses novos *voids* serem utilizados por rajadas futuras. Para calcular essa probabilidade, foi definida a probabilidade de inversão que é a probabilidade de que a ordem de chegada de duas rajadas seja inversa à ordem de chegada de seus pacotes de controle. Para tal, foram definidas expressões baseadas no tipo de montador e características estatísticas do tráfego sendo transportado. Resultados obtidos via simulação, mostram que o algoritmo LRC produz resultados melhores que dos outros algoritmos avaliados.

Além dos algoritmos gulosos (aqueles que processam uma requisição por vez) de escalonamento de canais, uma nova classe de algoritmos têm sido investigada na literatura, que é a classe dos algoritmos de escalonamento em lote de canais. Os algoritmos de escalonamento em lote agrupam as requisições em lote e fazem o processamento de todas as requisições agrupadas de uma só vez. Assim, é possível tomar decisões melhores para

[†]Período em que os canais de dados não possuem reservas.

alocação dos recursos às rajadas. De um modo geral, os algoritmos de escalonamento em lote possuem melhor desempenho do que os algoritmos gulosos em termos de probabilidade de bloqueio. Entretanto, os algoritmos propostos na literatura até então baseavam-se em heurísticas que nem sempre produziam os melhores resultados. Nesta tese, foi proposta uma nova forma para modelar o problema de escalonamento em lote de canais de forma que algoritmos ótimos pudessem ser usados com complexidade de tempo relativamente baixa.

Foram propostos dois novos algoritmos ótimos para escalonamento em lote de canais. O primeiro algoritmo, proposto inicialmente em [8] para resolver o problema de *job scheduling*, denominado aqui GreedyOPT, possui complexidade de tempo linear e seu uso é indicado para o caso em que as requisições trafegando na rede não possuem nenhum tipo de diferenciação. O segundo algoritmo possui complexidade de tempo polinomial e é indicado quando existe diferenciação da prioridade da requisições que compõem o lote. Seu emprego é útil para o caso em que a rede deseja oferecer diferenciação de serviços aos fluxos de tráfego. Além dos algoritmos propostos, foi proposta uma estratégia de formação de lote, que determina quais as requisições devem compor o lote. A estratégia de formação de lote pode ser vista como uma extensão do protocolo JET e permite a coexistência dos algoritmos de escalonamento em lote e os algoritmos gulosos no mesmo ambiente de rede.

1.1 O conteúdo desta Tese

O foco desta tese é o desenvolvimento de algoritmos de montagem e escalonamento eficientes, que reduzam a probabilidade de bloqueio em rede OBS.

No Capítulo 2 apresenta-se a arquitetura básica da rede OBS. A Seção 2.1 apresenta, em termos gerais, os conceitos de comutação de rajadas. A Seção 2.2 discute os principais paradigmas de comutação de rajadas propostos na literatura. A Seção 2.3 enfatiza o protocolo de reserva JET, principal protocolo de reservas empregado nas redes OBS.

Discute-se suas características e conceitos como reserva atrasada e o cálculo do tempo de ajuste. O processo de montagem de rajadas em redes OBS é discutido na Seção 2.4. Apresenta-se o conceito de montagem de rajadas além de ser dada uma ênfase maior aos algoritmos de montagem básicos. Na Seção 2.5, apresenta-se o conceito de escalonamento de canais.

No Capítulo 3 são discutidos conceitos de modelagem de tráfego. O capítulo limita-se a apresentar e discutir somente os conceitos necessários ao entendimento do restante da tese. Não se faz, propositadamente, uma análise aprofundada sobre modelagem de tráfego. Leitores interessados em mais detalhes podem referir-se a [4, 10, 22, 47, 58, 59, 64, 68]. Na Seção 3.1 são definidos os conceitos de processos monofractais e processos multifractais. Na Seção 3.2 discutem-se as evidências da presença da escala limitante, enfatizando-se as diferenças de comportamento geradas por processos monofractais e processos multifractais, quando a análise do diagrama log-log do processo incremento agregado de tais processos é feita.

No Capítulo 4 são descritas ferramentas e técnicas utilizadas na análise de multifractalidade de traços de tráfego IP. Além disso, são selecionados alguns traços de domínio público com tráfego real de fluxos IP e é feita uma análise de multifractalidade destes traços.

No Capítulo 5 apresenta-se uma análise da transformação do tráfego resultante da atuação dos mecanismos de montagem de rajadas das redes OBS. Na Seção 5.1 introduz-se o ambiente e os parâmetros usados no experimento. Na Seção 5.2, discute-se os resultados obtidos.

Um método para detecção não visual da escala de tempo limitante é introduzido no Capítulo 6. Primeiramente, discute-se como os pontos do processo incremento agregado em função do intervalo de agregação são dispostos no plano. Discute-se, em seguida, como o coeficiente de determinação de uma regressão linear pode ser usado para determinar a transição da região em que os pontos são dispostos de forma linear para outra região de não linearidade ou com outra inclinação. Avalia-se o algoritmo utilizando traços de

tráfego real. Discute-se, também, a complexidade computacional do método.

O Capítulo 7 apresenta quatro algoritmos de montagem de rajadas capazes da produzir transformações nas propriedades estatísticas do tráfego. Alguns trabalhos relacionados são apresentados na Seção 7.1. Na Seção 7.2 discute-se como a montagem das rajadas pode ser orientada à transformação do tráfego. Na Seção 7.3 são apresentados e discutidos os algoritmos de montagem. Na Seção 7.4 avalia-se o desempenho dos algoritmos.

No Capítulo 8, o algoritmo de escalonamento de canais LRC é introduzido. Na Seção 8.1 são apresentados alguns algoritmos de escalonamento de canais já propostos na literatura. Na Seção 8.2 discute-se como o uso de informações acerca da distância relativa entre um nó intermediário e seu destino pode ser usada no escalonamento dos canais. São introduzidos os conceitos de vida útil de um intervalo *void* e de inversão entre pacotes de controle e rajadas. Na Seção 8.2.2 calcula-se a probabilidade de inversão sob diversas premissas acerca das propriedades estatísticas do tráfego e o tipo de montador de rajadas utilizado. O algoritmo LRC é apresentado na Seção 8.4 e sua complexidade computacional analisada na Seção 8.5. Os exemplos numéricos são apresentados na Seção 8.6.

No Capítulo 9, são discutidos os algoritmos de escalonamento de canais em lote para redes OBS. Na Seção 9.1, são dados os conceitos de escalonamento em lote. Na Seção 9.2 são apresentados os conceitos de teoria dos grafos necessários ao entendimento dos algoritmos apresentados na Seção 9.3. Na Seção 9.4, discute-se a obtenção de soluções ótimas para o problema de escalonamento de canais, apresentando-se inicialmente uma formulação em programação linear, depois mostra-se como as requisições já alocadas podem ser reconsideradas na execução do algoritmo, mudando, assim, a modelagem do problema. Depois disso, são apresentados os algoritmos ótimos para a solução do problema e sua complexidade computacional é discutida. Na Seção 9.5, introduz-se a estratégia de formação de lote denominada JET-Δ. Mostra-se finalmente a avaliação de desempenho dos algoritmo discutidos na Seção 9.6.

Por fim, as conclusões são apresentadas no Capítulo 10.

1.2 Contribuições da Tese

A presente tese avança o conhecimento sobre redes OBS com as seguintes contribuições:

- detecção e análise das transformações do tráfego ocorridas devido ao processo de montagem em redes OBS;
- introdução de um método para identificação não visual da escala de tempo limitante;
- derivação de algoritmos de montagem de rajadas capazes de induzir transformações nas propriedades estatísticas do tráfego da rede;
- introdução de algoritmo adaptativo de escalonamento de canais;
- definição e cálculo da probabilidade de inversão de rajadas;
- nova modelagem do problema de escalonamento em lote de canais em redes OBS;
- formulação de algoritmo ótimo para escalonamento em lote com tempo de execução linear para requisições com peso unitário;
- derivação de algoritmo ótimo para escalonamento em lote com tempo de execução polinomial para requisições com peso não unitário;
- definição de estratégia de formação de lote para escolha das requisições que comporão o lote.

1.3 Publicações realizadas pelo autor

1.3.1 Publicações relacionadas a esta tese

As seguintes publicações têm como base os resultados apresentados nesta tese:

1. FIGUEIREDO, G. B.; XAVIER, E. C.; FONSECA, N. L. S.. An optimal batch scheduling algorithm for OBS networks. In: IEEE GLOBECOM, 2009, Hawai. IEEE Global Communications Conference, 2009.

2. FIGUEIREDO, G. B.; XAVIER, E. C.; FONSECA, N. L. S.. Um algoritmo ótimo para escalonamento de canais em lote em redes OBS. In: Simpósio Brasileiro de Redes de Computadores, 2009, Recife. XXVII SBRC, 2009. p.45-58.

3. FIGUEIREDO, G. B.; FONSECA, N. L. S.; MELO, C. A. V.. Montagem de Rajadas com Suavização de Tráfego em Redes de Comutação de Rajadas Ópticas. In: Simpósio Brasileiro de Redes de Computadores, 2008, Rio de Janeiro. XXVI SBRC, 2008. p. 637-650.

4. FIGUEIREDO, G. B.; FONSECA, N. L. S.. Escalonamento de Canais em Redes OBS usando Informações Topológicas. In: Simpósio Brasileiro de Redes de Computadores, 2008, Rio de Janeiro. XXVI SBRC, 2008. p. 623-636.

5. FIGUEIREDO, G. B. ; MELO, C. A. V.; FONSECA, N. L. S.. Identification of the Cut-off Scale of OBS Ingress Traffic. In: IEEE GLOBECOM, 2008, New Orleans. IEEE Global Communications Conference. v. 1. p. 1-6

6. FIGUEIREDO, G. B.; FONSECA, N. L. S.. The Least Reusable Channel Burst Scheduling Discipline. In: IEEE GLOBECOM, 2008, New Orleans. IEEE Global Communications Conference. v. 1. p. 1-6.

7. FIGUEIREDO, G. B.; FONSECA, N. L. S.; MELO, C. A. V.; SALVADOR, M. R.. Transformação de Tráfego em Redes Ópticas em Rajadas. In: Simpósio Brasileiro de Redes de Computadores - SBRC, 2007, Belém. Anais do XXV Simpósio Brasileiro de Redes de Computadores, 2007. p. 119-132.

8. FIGUEIREDO, G. B.; FONSECA, N. L. S.; MELO, C. A. V.; SALVADOR, M. R.. On the Transformation of Multifractal Traffic at Ingress Optical Burst Switches. In: IEEE ICC 2006, 2006, Istanbul. IEEE Interantional Conference on Communications. p. 1040-1045.

9. FIGUEIREDO, G. B.; FONSECA, N. L. S.; MELO, C. A. V.; SALVADOR, M. R..

O Impacto do Algoritmo de Montagem de Rajadas Baseado em Janelas de Tempo no Tráfego Multifractal. In: SBrT, 2005, Campinas. Anais do Simpósio Brasileiro de Telecomunicações. p. 47-52.

Capítulo 2

Redes de Comutação de Rajadas Ópticas

Neste capítulo, apresenta-se a arquitetura básica da rede de comutação de rajadas ópticas. Os mecanismos presentes na arquitetura OBS são apresentados em uma discussão geral a respeito do seu funcionamento e interação com os outros mecanismos da rede.

2.1 Comutação de rajadas ópticas

A comutação de rajadas ópticas (*Optical Burst Switching - OBS*) [56] foi proposta como um novo paradigma de comutação para redes puramente ópticas. Ela é baseada no padrão do ITU-T para comutação de rajadas em redes ATM [55], no qual a unidade de transmissão, denominada rajada, possui granularidade intermediária entre o circuito utilizado na comutação de circuitos e o pacote utilizado na comutação de pacotes.

Os pacotes IP são agregados nos nós de ingresso da rede em unidades de transmissão denominadas rajadas e transmitidos através da rede como ilustra a Figura 2.1.

Quando uma rajada alcança um nó de egresso de um domínio OBS, ela é desmontada e seus pacotes são repassados às camadas superiores como ilustra a Figura 2.2.

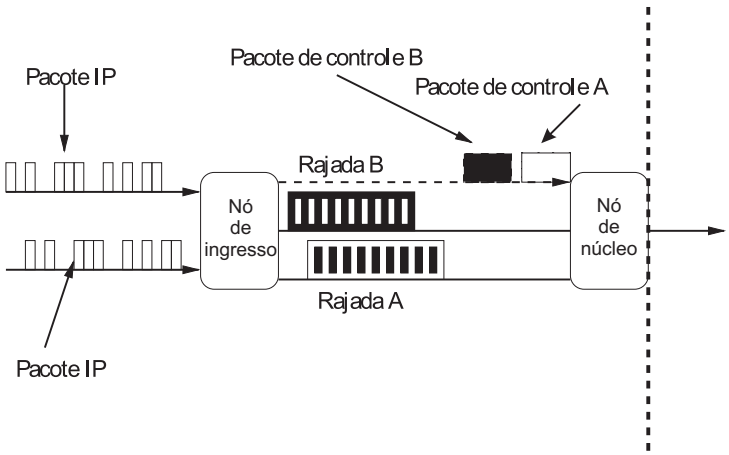


Figura 2.1: Montagem das rajadas em redes OBS.

Cada rajada é precedida por um pacote de controle*. Este pacote de controle é responsável por fazer indicação aos nós pertencentes ao caminho entre a origem dos dados e o destino dos dados sobre a chegada de uma rajada. Cada nó é, então responsável por fazer a reserva dos recursos de acordo com a sua disponibilidade. Tal processo de reserva de recursos pode variar dependendo da arquitetura OBS em questão, como será discutido na seção 2.2.

A transmissão do pacote de controle e das rajadas ocorre em canais separados, denominados, respectivamente, canais de controle e de canais de dados. Dado que o pacote de controle é significativamente menor do que uma rajada, um canal de controle é considerado suficiente para a transmissão dos pacotes de controle associados a vários canais de dados.

Cada pacote de controle sofre conversões de sinal entre os domínio óptico e elétrico e é processado eletronicamente em cada nó OBS intermediário, como ilustra a Figura 2.3. A rajada é, geralmente, enviada após um período de ajuste para compensar o atraso de processamento do pacote de controle e configuração dos nós intermediários. Se o tempo de ajuste for suficientemente grande, a rajada é transmitida totalmente no domínio óptico

*Por vezes, o termo requisição será empregado para referir-se à requisição de reserva de recursos trazida pelo pacote de controle.

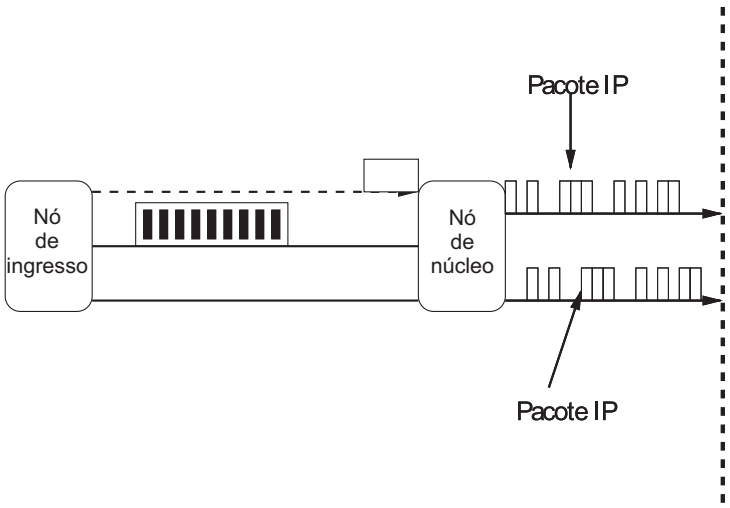


Figura 2.2: Desmontagem das rajadas em redes OBS.

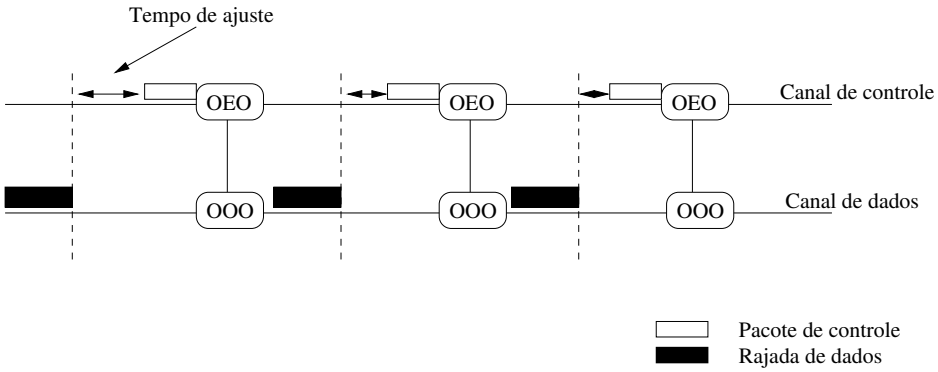


Figura 2.3: Canais separados de dados e controle.

sem sofrer atraso, em nenhum nó intermediário. Dessa forma, nenhum tipo de memória ou FDL é necessário.

Funcionalmente, os nós na rede OBS são divididos entre nós de borda e nós de núcleo, como ilustra a Figura 2.4. Os nós de borda são responsáveis por:

1. realizar a montagem da rajada;

2. determinar qual a rajada que será transmitida (em caso de haver mais de uma pronta para transmissão);

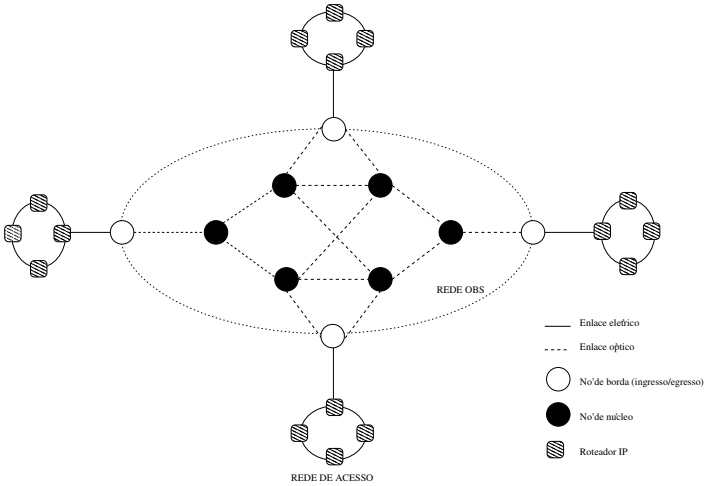


Figura 2.4: Divisão funcional da rede

3. criação do pacote de controle associado à rajada a ser transmitida;

4. determinar o tempo de ajuste entre o pacote de controle e a rajada correspondente;

5. encaminhar as rajadas ao núcleo da rede.

Afim de realizar estas funções, os nós de borda possuem interfaces eletrônicas para acomodar os pacotes oriundos das redes de acesso, e interfaces ópticas, para enviar as rajadas através da rede OBS.

Os nós de núcleo, por sua vez, são responsáveis pela reserva dos recursos que é realizada através do processamento do pacote de controle e escalonamento dos canais de dados. Além disso, os nós de núcleo participam do roteamento e da comutação da rajadas de dados.

2.2 Os paradigmas de comutação em rajadas

Várias abordagens para redes OBS foram propostas na literatura [56]. Apesar de possuirem diferentes características, todas as abordagens têm alguns pontos comuns, que são:

- o processo de montagem/desmontagem das rajadas ocorre nos comutadores de borda da rede OBS;
- os sinais de controle e de dados são transmitidos e manipulados em canais separados, fazendo com que a conversão O/E/O seja efetuada apenas num reduzido número de canais.

As principais variações de comutação em rajada são: TAG (*Tell and Go*), IBT (*In Band Terminator*) e RFD (*Reserve a Fixed Duration*) [3]. Das três, as duas primeiras já foram estudadas em redes eletrônicas. A terceira foi proposta especificamente para redes ópticas WDM.

No esquema TAG, a fonte de transmissão envia primeiramente, em um canal de controle separado, um pacote de controle para reservar recursos ao longo do caminho. A rajada é, então, imediatamente transmitida usando um dos canais de dados disponíveis, sem espera de nenhum tipo de confirmação. Após a transmissão da rajada, outro sinal de controle é enviado liberando os recursos.

No esquema IBT, cada rajada tem um cabeçalho assim como um delimitador especial (denominado terminador [30]) para indicar o final da rajada. Tal característica faz com que o esquema IBT seja bastante semelhante à comutação de pacotes, em especial no que diz respeito ao disparo da alocação e liberação de recursos. Contudo, o esquema IBT, como uma variação do paradigma OBS, utiliza o envio imediato (*cut-through*) ao invés do armazenamento e encaminhamento (*store-and-forward*) adotado na comutação de pacotes. No esquema *cut-through*, os comutadores não precisam esperar a recepção de todos os dados. Ao contrário, antes de receber o final da rajada, os comutadores podem encaminhar os pacotes que já chegaram. Com isso, a rajada sofre menos atraso e, consequentemente, menos espaço de armazenamento temporário é necessário em cada nó.

No esquema RFD, assim como no esquema TAG, um pacote de controle é enviado antes da rajada por um canal separado para reservar os recursos. A rajada de dados é enviada após um período de ajuste T. O que distingue o modelo RFD do modelo TAG é

que no primeiro, a largura de banda é reservada por uma duração especificada pelo pacote de controle que contém informação sobre o tamanho (e consequentemente a duração) da rajada. Isto implica que cada rajada terá um tamanho máximo limitado [57].

O esquema RFD tem sido considerado mais atrativo pois pode tirar vantagens do uso adequado do tempo de ajuste, ao mesmo tempo em que minimiza suas potenciais desvantagens, de forma mais efetiva que as outras duas abordagens.

Uma das vantagens de usar um tempo de ajuste é que, a partir do momento em que a rajada é armazenada na origem, não é necessário esperar pelo processamento do pacote de controle nos nós intermediários. No esquema TAG, por exemplo, as rajadas são enviadas imediatamente após o pacote de controle. Assim, é possível que a rajada chegue aos nós intermediários antes do pacote de controle ter sido processado, o que torna necessário o uso de *buffers* (FDLs) para evitar o descarte das rajadas em tais situações. No esquema RFD, por outro lado, se há *buffers* (FDLs) disponíveis nos nós intermediários, estes podem ser utilizados somente para resolução de contenção. As desvantagens de se usar tempos de ajuste na reserva dos recursos são um possível aumento do atraso fim-a-fim e o desperdício de largura de banda.

Tais desvantagens podem ser também observadas na comutação de circuitos. Contudo, na comutação de rajadas, especialmente com o esquema RFD, a relação *Custo x Benefício* pode ser otimizada. Por exemplo, na comutação de circuitos, T será, no mínimo, $2P+T_T^{(p)}$, onde P é o tempo de propagação apenas de ida e $T_T^{(p)}$ é o tempo de processamento total encontrado pela requisição de estabelecimento do circuito ao longo do caminho. Isso elimina a necessidade de armazenamento temporário nos nós intermediários ao longo do caminho. Na comutação de rajadas, os dados podem ser enviados antes dos recursos estarem reservados nos últimos segmentos do caminho. Assim, mesmo se $T \leq P+T_T^{(p)}$ não é preciso armazenamento temporário. Dessa forma, é possível para um nó intermediário qualquer tomar decisões baseadas no conhecimento que este tem sobre a duração da reserva. Caso o valor escolhido para T seja muito pequeno, ($T = 0$ por exemplo), pode não haver recursos disponíveis para que a rajada seja transmitida, fazendo com que a

mesma seja descartada.

2.3 O protocolo de reserva de recursos *Just Enough Time* (JET)

O protocolo JET foi proposto por Qiao e Yoo em [56]. Ele é uma variação do modelo RFD tradicional, no qual a rajada é precedida por um pacote de controle enviado através de um canal de sinalização. Além disso, o pacote de controle é seguido por uma rajada de dados depois de um tempo de ajuste T, com $T \geq T_T^{(p)}$, onde $T_T^{(p)}$ representa o tempo total de processamento do pacote de controle.

Enquanto a rajada não é transmitida, ela é armazenada na origem. Isto elimina a necessidade de FDLs nos nós intermediários para retardar as rajadas enquanto o pacote de controle é processado. Assim, o tempo de ajuste deve ser longo o suficiente para contabilizar o tempo de processamento do pacote de controle nos nós intermediários e no destino além do tempo de configuração do comutador de destino. Caso o tempo de ajuste seja inferior ao requerido, então existirá a possibilidade de que a rajada possa chegar em um nó antes do mesmo estar pronto para comutar a rajada. Este processo é ilustrado na Figura 2.5. Na figura, a quantidade de nós é igual 3 e o tempo de processamento em cada nó i é igual a $T_i^{(p)}$, o que resulta em um tempo total de processamento de $T_T^{(p)} = 3T_i^{(p)}$.

Assim como no esquema RFD, os recursos em um nó i são reservados até que a rajada seja transmitida. Para isso, o pacote de controle deverá incluir não só o tamanho da rajada (como no modelo RFD tradicional) mas também o valor do tempo de ajuste T_i (no nó de origem, $T_i = T$).

Considere $T_i^{(p)}$ como o tempo de processamento do pacote de controle em um nó intermediário i, $T_d^{(p)}$ o tempo de processamento do pacote de controle no nó destino e $T_d^{(s)}$ o tempo necessário para ajustar (configurar) o comutador destino. Então, o valor inicial de T será:

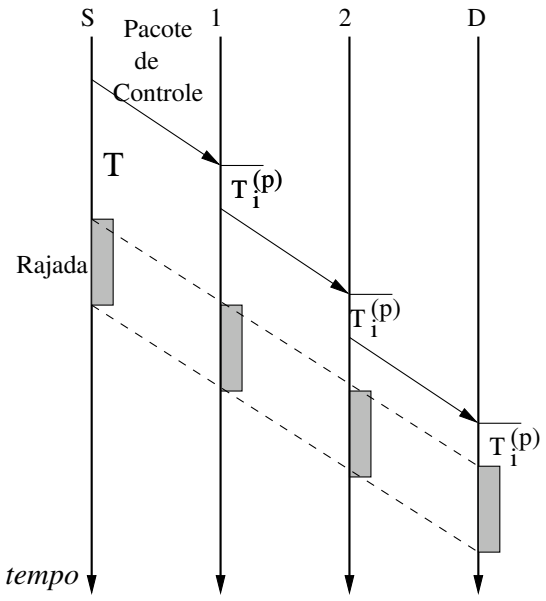


Figura 2.5: Protocolo JET.

$$T = \left(\sum_i T_i^{(p)} \right) + T_d^{(p)} + T_d^{(s)} \tag{2.1}$$

Para lidar com o tempo de processamento variável encontrado pelo pacote de controle, bem como o atraso de envio e recebimento em cada nó, o pacote de controle pode ser "marcado" com o seu tempo de chegada e agendado para transmissão tão logo o processamento termine.

Uma das diferenças mais marcantes entre o JET e os protocolos RFD, e que também pode ser considerada a principal característica do protocolo JET, é o uso de reserva atrasada (*Delayed Reservation*). Quando reserva atrasada é utilizada, a largura de banda em um nó i é reservada, somente no momento esperado da chegada da rajada e não ao final do processamento do pacote de controle.

Seja t_{in} o instante de chegada de um pacote de controle em um nó i, t' o instante do término do processamento do pacote de controle e t_{out} o instante da transmissão do pacote de controle (onde $t_{in} < t' < t_{out}$). Os recursos serão reservados em $t = T_i + t_{in}$, o

que implica que na Figura 2.6(a), $T(i) = T_i - (t' - t_{in})$ e o pacote de controle conterá o valor atualizado de T_i que é $T_i - (t_{out} - t_{in})$.

Se os recursos requisitados não estão disponíveis, a rajada é dita bloqueada e é descartada. O uso de reserva atrasada pode reduzir os bloqueios das rajadas como ilustrado na Figura 2.6(b). Suponha que um dado nó i recebe dois pacotes de controle. O término do processamento do pacote de controle 1 acontece antes do término do processamento do pacote de controle 2, ou seja, $t'_1 < t'_2$. Quando o segundo pacote de controle chega, é possível saber se $t_2 > t_1 + l_1$ (a segunda rajada chegará após a transmissão da primeira - caso 1), ou se $t_2 + l_2 < t_1$ (caso 2 - a segunda rajada chega e é transmitida antes da chegada da rajada 1). Nos dois casos, os recursos podem ser reservados para as duas rajadas. Por outro lado, se o mecanismo de reserva atrasada não for utilizado, não existe forma de saber se a segunda rajada é pequena o suficiente para ser transmitida antes da primeira rajada ou se ela chegará depois da primeira rajada. Desse modo, a segunda rajada pode ser descartada.

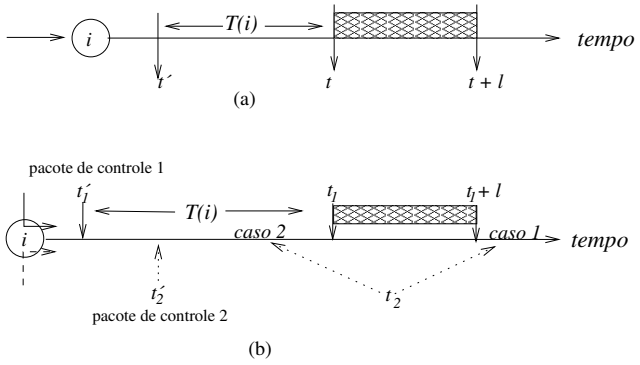


Figura 2.6: Reserva atrasada.

2.4 Processo de Montagem das Rajadas nas Redes OBS

A montagem de rajadas é o processo de agregar vários pacotes, às vezes provindos de várias fontes distintas, em rajadas. Este processo ocorre na unidade de montagem do nó de ingresso da rede OBS e sua arquitetura básica é ilustrada na Figura 2.7.

Cada nó de ingresso mantém filas, nas quais os pacotes IP provindos das portas de entrada são armazenados. A quantidade de filas mantidas em cada nó depende dos critérios adotados para a montagem das rajadas. Um nó pode, por exemplo, usar uma fila para cada classe de serviço ou uma fila para cada destino da rede. A Figura 2.7 ilustra a arquitetura básica de uma rede OBS, enfatizando um nó de borda com duas filas. A fila de montagem à qual os pacotes IP são encaminhados depende do critério de montagem adotado. Os critérios são os mais variados e podem ser pacotes que pertencem à mesma classe de serviço, pacotes com um mesmo destino, ou mesma origem.

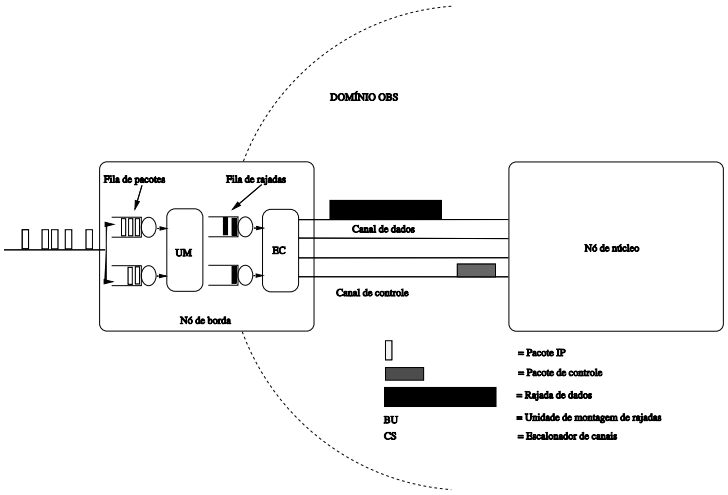


Figura 2.7: Processo de montagem de rajadas

A montagem das rajadas pode tambem adotar diversos critérios. Em [43], o critério

de montagem das rajadas depende da classe de serviço dos pacotes, do tamanho máximo permitido e do tempo máximo de montagem de cada rajada. No mecanismo proposto em [69], uma rajada é criada contendo pacotes de várias classes de serviço. Os pacotes são posicionados na rajada em ordem decrescente de prioridade. Dessa forma, na iminência de uma colisão, a parte de menor prioridade pode ser descartada sem que haja perda completa da rajada. Nesse caso, o critério de montagem das rajadas depende do número de classes em questão, da quantidade máxima de pacotes de cada classe que pode ser colocada em uma rajada e do tempo de montagem das rajadas.

Apesar da possibilidade de uso de diferentes critérios na montagem das rajadas, dois critérios podem ser considerados como fundamentais para a maioria dos algoritmos de montagem de rajadas: o tempo máximo de montagem das rajadas [28] e o tamanho mínimo exigido de cada rajada [78].

No primeiro algoritmo, denominado aqui de algoritmo baseado em janelas de tempo, pacotes IP provindos das portas de entrada são colocados em diferentes filas à espera da montagem da rajada. Quando o primeiro pacote de uma dada fila é armazenado, um temporizador é iniciado. Quando o temporizador expira após um tempo t_i, uma rajada é criada e enviada.

Em redes operacionais, o tempo de montagem de rajadas depende dos requisitos temporais de QoS das classes de tráfego envolvidas. Obviamente, quanto mais restrito o requisito temporal, menor será o tempo das montagem de rajadas, evitando assim que pacotes sofram grandes atrasos no processo de montagem. Os valores típicos costumam variar de 1ms [29] a 600ms [43]. Os parâmetros da unidade de montagem de rajadas sugeridos em [43] são apresentados na Tabela 2.1.

No segundo algoritmo, um contador de pacotes ou de bytes é mantido para cada fila. Quando o número de pacotes, ou de bytes, alcança um limite pré-estabelecido, a rajada é criada e enviada. A utilização exclusiva do primeiro critério implica na possibiilidade de existência de rajadas de tamanho variável, enquanto que a utilização exclusiva do segundo critério, implica que somente rajadas de tamanho fixo são permitidas.

Tabela 2.1: Parâmetros sugeridos por [43] para a unidade de montagem de rajadas.

Classe de Serviço	Tam. min	Tam. Max	t_i
EF	5KB	5KB	4.8ms
AF	30KB	50KB	55ms
BE	125KB	125KB	600ms

2.4.1 Predição do tamanho das rajadas

O atraso sofrido por uma rajada consiste de três componentes: o atraso da montagem das rajadas nos comutadores de borda, o atraso da reserva dos recursos e o atraso de propagação. O atraso causado pela montagem das rajadas deve-se ao fato de que, normalmente, o pacote de controle só é enviado após o término da montagem da rajada correspondente. Das três componentes citadas, o atraso causado pela montagem das rajadas provoca o maior impacto no atraso fim-a-fim das rajadas.

Alguns esquemas de predição de tráfego têm sido propostos para reduzir o atraso causado pela montagem das rajadas. A idéia por trás da predição de tráfego na montagem das rajadas, é conseguir em um dado instante de tempo t, prever a quantidade de dados que chegará até um outro instante de tempo $t + \tau$. Com isso, a unidade de montagem de rajadas consegue fazer o disparo do pacote de controle antes mesmo de todos os pacotes da rajada terem chegado. Dessa forma, a rajada pode ser enviada assim que todos os pacotes (previstos) tiverem chegado.

A escolha do método de predição usado é um compromisso entre o custo computacional, o intervalo de predição e o erro na predição do tráfego. O método deve ser simples o suficiente para não sobrecarregar os comutadores, mas deve possuir boa precisão para que os danos causados por eventuais falhas na predição sejam mínimos.

Em [49], propõe-se um esquema baseado em predição de tráfego que visa minimizar o atraso causado pela montagem das rajadas. A idéia é parecida com a apresentada nas Figuras 2.8 e 2.9.

Em [42], um novo esquema de reserva de recursos baseado na predição do tamanho

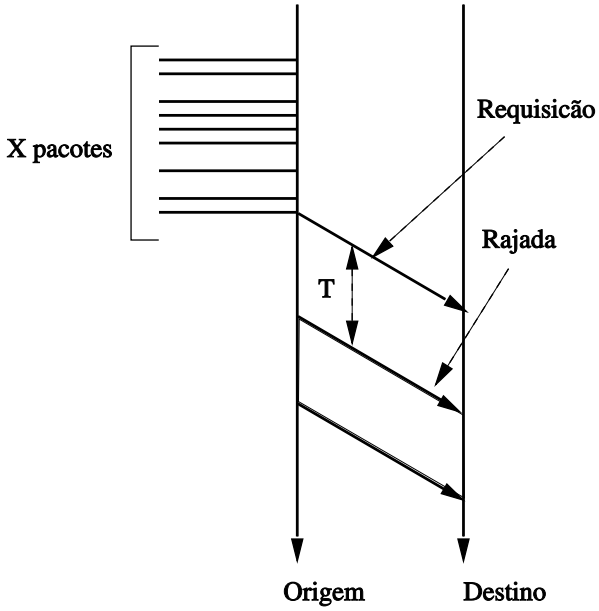


Figura 2.8: Montagem de rajadas sem predição de tráfego.

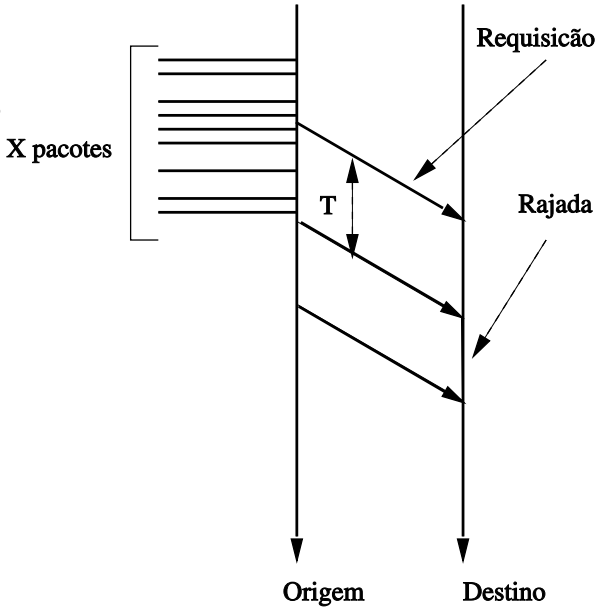


Figura 2.9: Montagem de rajadas com predição de tráfego.

das rajadas é apresentado. O esquema funciona em três fases: fase de predição, fase de pré-transmissão e fase de exame.

Na fase de predição, tão logo a montagem de uma nova rajada inicie, a unidade de montagem prediz o tamanho que a mesma deve ter, baseada no método de predição linear. Na fase de pré-transmissão, ao invés de esperar o término da montagem da nova rajada, um pacote de controle é enviado logo após o término da predição. O pacote de controle contém as informações necessárias à reserva dos recursos, inclusive o tamanho da nova rajada.

Na fase de exame, quando a rajada está efetivamente montada, o seu tamanho é comparado com o que havia sido previsto. Se o tamanho previsto for maior ou igual ao da rajada, a mesma é enviada. Caso a previsão tenha sido menor que o tamanho real da rajada, outro pacote de controle é enviado informando o tamanho correto da rajada. Para diminuir a margem de erro da previsão, um fator de correção de erro é adicionado ao pedido de reserva. Assim, na fase de pré-transmissão, ao invés de fazer reserva para

uma quantidade X de bytes, o nó de borda envia um pacote de controle pedindo uma reserva para uma quantidade $X + \delta$ de bytes, onde δ é o fator de correção.

2.5 Escalonamento de canal em redes OBS

Ao receber um pacote de controle, os nós da rede OBS devem reservar um canal de dados para transmitir a rajada ao nó subsequente. Um algoritmo de escalonamento (ou seleção) de canal é utilizado com esta finalidade. Dadas uma lista de reservas 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 e quais as rajadas ocuparão os períodos reservados. Esse processo é ilustrado na Figura 2.10

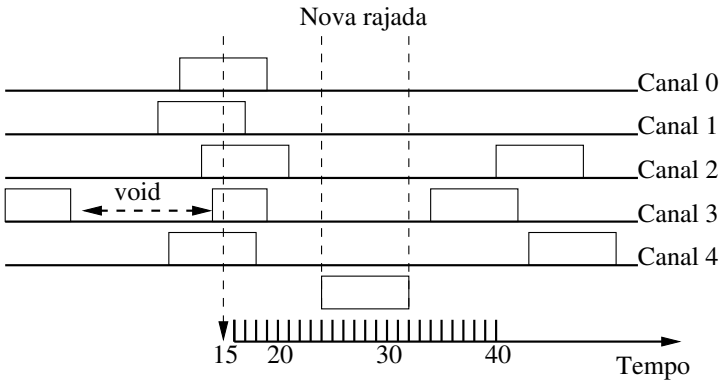


Figura 2.10: Escalonamento dos canais.

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 a 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*. Cada um dos W canais de um enlace pode ser visto inicialmente 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 outro 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).

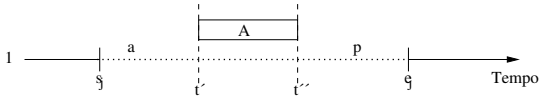


Figura 2.11: Criação de novos intervalos *void*.

A Figura 2.11 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 para acomodar as reservas. O desempenho de um algoritmo de escalonamento de canais é, usualmente, medido em função da probabilidade de bloqueio por ele apresentada. Assim, quanto menor a probabilidade de bloqueio das rajadas, melhor o algoritmo de escalonamento.

Vários algoritmos de escalonamento de canal têm sido propostos na literatura [50, 65, 73, 79]. Xiang Yu et al. [79] classificam os algoritmos em dois grupos: algoritmos

de escalonamento reativos e algoritmos de escalonamento pró-ativos. O primeiro grupo é caracterizado por não considerar o impacto causado nas rajadas futuras gerado pela seleção de um canal para uma dada rajada. O segundo grupo é formado por algoritmos que evitam a contenção de rajadas futuras. Tais algoritmos serão discutidos com mais detalhes no Capítulo 8

2.6 Qualidade de serviço em redes OBS

Provisão de QoS em redes OBS é um campo fértil de pesquisa. Diversos mecanismos têm sido propostos usando diferentes abordagens para provisão de QoS em redes OBS [13, 23, 37, 41, 48, 69, 77, 80]. Esta seção apresenta apenas um subconjnto de tais mecanismos. Leitores interessados podem encontrar mais informações a respeito em [15, 34].

Prioritized Just Enough Time (PJET)

O protocolo PJET [77] utiliza o tempo de ajuste como forma de prover QoS a diferentes classes de serviço. No protocolo PJET um tempo de ajuste adicional, denotado t_{qos}, é atribuido a classes de alta prioridade. O valor de t_{qos} é constante e consideravelmente mais alto do que o utilizado no protocolo JET. Além disso, ele deve ser maior que o tamanho máximo das rajadas das classes inferiores. Assim, com o uso do tempo de ajuste e de reserva atrasada, é possível fazer reserva de recursos *a priori*, garantindo a classes de alta prioridade o acesso aos recursos mesmo quando as requisições das classes de mais baixa prioridade chegam com antecedência. A transmissão só tem sucesso se a chegada prevista da rajada de alta prioridade se dá após o término da transmissão da rajada de baixa prioridade. A idéia é que a utilização de um tempo de ajuste razoavelmente grande, torne a probabilidade de perda das rajadas da classe de alta prioridade independente da carga oferecida pelas classes de baixa prioridade e somente como função da carga oferecida pela classe de alta prioridade. O principal problema com esta abordagem, entretanto, é o aumento significativo do atraso fim-a-fim no tráfego de alta prioridade.

Preemptive Prioritized Just Enough Time (PPJET)

O protocolo PPJET [37] provê QoS a classes de alta prioridade através da interrupção das requisições das classes de baixa prioridade. A idéia é que as reservas das classes de alta prioridade não possam ser bloqueadas por reservas futuras de classes de mesma prioridade ou por reservas de classes de baixa prioridade que tenham tempo de início no futuro. Assim, uma reserva de classe de alta prioridade só pode ser bloqueada por uma reserva já existente de uma classe de mesma prioridade ou por reservas de classes de mais baixa prioridade que já estejam sendo atendidas.

Diferenciação Proporcional em redes OBS

Embora consigam prover QoS a classes de alta prioridade em detrimento das classes de baixa prioridade, o uso de um tempo de ajuste adicional tem sido amplamente apontado na literatura como um fator negativo desse tipo de mecanismo [13, 41, 80]. Os principais problemas apontados são: i) tais esquemas requerem mecanismos de reserva homogêneos em que o tamanho da rajada e o tempo de ajuste de cada classe de serviço deve ser ajustado e mantido em todos os comutadores do domínio OBS, sob pena de degradação para um serviço sem prioridades como nas redes OBS originais. ii) tais esquemas não limitam a quantidade de recursos que uma classe de alta prioridade deve consumir, dessa forma, as classes de baixa prioridade podem nem mesmo ter acesso aos recursos. iii) um possível aumento do atraso fim-a-fim, já que um novo tempo de ajuste é adicionado.

Para solucionar este tipo de problema, foi proposto em [13] um esquema de QoS proporcional. Neste modelo, a única garantia é de que rajadas pertencentes a uma classe de mais alta prioridade não receberão menos recursos do que rajadas pertencentes a classes de mais baixa prioridade. Entretanto, pode-se ajustar os parâmetros de diferenciação de serviços de tal forma que a QoS exigida pelas aplicações possa ser atendida.

O modelo é baseado na seguinte relação para todas as classes de serviço:

$$\frac{q_i}{q_j} = \frac{s_i}{s_j}(i, j = 0...N) \tag{2.2}$$

onde q_i e s_i são uma métrica de QoS e um fator de diferenciação, respectivamente. Tal modelo é válido tanto em grandes escalas de tempo quanto em pequenas escalas de tempo. Dessa forma, a natureza em rajada do tráfego Internet pode ser levada em consideração. Assim, tem-se:

$$\frac{\overline{q}_i(t, t+\tau)}{\overline{q}_j(t, t+\tau)} = \frac{s_i}{s_j} \tag{2.3}$$

onde τ é denominado intervalo de monitoração e $\overline{q}_i(t, t+\tau)$ é a média da métrica de QoS neste intervalo.

Para fazer a efetiva implementação do modelo de diferenciação proporcional, um esquema de *descarte intencional de rajadas* foi proposto. Neste esquema, quando a equação 2.3 é violada, as rajadas de baixa prioridade são descartadas. A idéia é deixar os canais de dados com menos tráfego para dar oportunidade às classes de mais alta prioridade.

Preemptive Wavelength Reservation Protocol (PWRP)

O descarte intencional empregado em [13] provoca um número excessivo de descartes, o que faz com que a rede trabalhe sempre com baixa utilização e alta probabilidade de bloqueio.

Em [41] propõe-se um novo mecanismo que visa prover diferenciação de serviços em redes OBS. O mecanismo considera K classes de serviço, $c_1, c_2, ..., c_k$ em ordem ascendente de prioridade. Cada classe "i" possui um limite de uso p_i, tal que $0 \leq p_1 < p_2 < ... < p_i < ... < p_k \leq 1$ e $\sum_{i=1}^{k} p_i = 1$. Cada comutador monitora o limite de uso de cada classe para determinar se a mesma está de acordo com um perfil pré-determinado. Uma classe i é dita dentro do perfil se a sua utilização (ρ_i) é menor do que o seu limite de uso (p_i), ou seja, se $\rho_i \leq p_i$, onde $\rho_i = \sum_{j=1}^{n_i} l_j / \sum_{i=1}^{k} \sum_{j=1}^{n_i} l_j$ onde k é o número de classes, n_j é

o número de requisições escalonadas da classe c_j no tempo $(t, t+\tau)$ e l_j é o tamanho da rajada da requisição R_j da classe c_j.

Considerando a natureza em rajada do tráfego, o monitoramento da utilização da classe i deve ser feito em uma pequena escala de tempo τ denominada escala de tempo de monitoração [41].

Ao receber uma requisição r, o comutador verifica se existe um canal que possa acomodar a rajada. Caso possa, a reserva é realizada. Caso não possa, o comutador verifica se a classe em questão está dentro do perfil. Caso esteja, o comutador procura um canal utilizado por uma classe fora do perfil para acomodar a requisição. Contudo, uma classe fora do perfil só pode ser bloqueada se já fez pelo menos uma reserva no mesmo canal que, ao ser bloqueada, possa acomodar a rajada da classe dentro do perfil.

Early Dropping e Wavelength Grouping

Os modelos de QoS proporcional são, também, chamados de modelos de QoS relativos. Neste modelo, o desempenho de cada classe não é definido quantitativamente em termos absolutos. Ao contrário, a QoS de uma classe é definida relativamente em comparação à de outras classes. Por exemplo, a uma classe de alta prioridade é garantido um menor retardo e menor taxa de perdas do que é garantido às classes de baixa prioridade. Entretanto, essas métricas de QoS das classes de alta prioridade ainda dependem das condições do tráfego de baixa prioridade. O modelo de QoS absoluto contrapõe-se a este modelo. O modelo de QoS absoluto garante um limite definido e absoluto em relação a tais métricas.

Em [80] propõe-se um esquema de provisão de QoS baseado no modelo absoluto usando como métrica a taxa de perdas de rajadas. Tal esquema é baseado na combinação de dois mecanismos: *early dropping* e *wavelength grouping*. O mecanismo de *early dropping* descarta probabilisticamente rajadas das classes da baixa prioridade para garantir a taxa máxima de perdas suportada pelas classes de alta prioridade. O mecanismo *wavelength grouping* reserva canais para as classes de alta prioridade. A reserva dos canais pode ser estática na qual um conjunto fixo de canais é estritamente reservado para cada classe

de serviço, ou dinâmica, na qual uma quantidade de canais é reservada mas não necessariamente um conjunto de canais. Para implementar tal esquema em toda a rede, um mecanismo denominado *path clustering* é utilizado. Este último mecanismo prioriza rajadas baseado na quantidade de nós do seu caminho. Dessa forma, a taxa de perda do tráfego sem garantias é também reduzida, enquanto a QoS do tráfego com garantias é provida.

Load-Level-Based Admission Control

Em [48] propõe-se um mecanismo qua faz o controle de admissão das rajadas baseado num parâmetro denominado nível de carga (*load level*). Este parâmetro indica o número máximo de comprimentos de onda que uma rajada de uma determinada classe i pode ocupar. Dessa forma, uma rajada pertencente à classe i só é admitida se no instante de tempo t, o número total de comprimentos de onda utilizados é menor do que o nível de carga l_i. Como verifica o número total de comprimentos de onda e não o número de comprimentos de onda de cada classe, tal mecanismo necessita de um número menor de parâmetros do que o mecanismo proposto em [80]. Em última instância, isso indica que um número menor de estados em cada comutador deve ser armazenado, o que faz com que menos esforço computacional seja despendido para tomar a decisão de admitir ou não uma rajada.

Generalized Burst Assembly

Em [69] é proposto um esquema de diferenciação de serviços realizado no processo de montagem de rajadas. Para tal, são associadas prioridades às rajadas e um esquema de segmentação, no qual parte da rajada pode ser descartada, é implementado. Com isso, os pacotes com mais alta prioridade são colocados no início da rajada. Caso haja contenção, a parte da rajada contendo pacotes de baixa prioridade é descartada. Além disso, os parâmetros usados na construção das rajadas são variáveis e ajustados de forma a reduzir o atraso e a probabilidade de bloqueio experimentados pelas rajadas.

QoS absoluta em redes OBS baseadas em GMPLS

Em [23] é proposto um esquema de provisão de QoS absoluta em redes OBS operando com plano de controle GMPLS, através do uso conjunto de um mecanismo para controle de admissão, como proposto em [80], e de técnicas de roteamento explícito, que permitam que as rajadas com mais alta prioridade sejam encaminhadas por rotas com menor intensidade de tráfego.

2.7 Resumo Conclusivo

Este capítulo apresentou os principais mecanismos propostos para a reserva de recursos, montagem das rajadas, escalonamento de canais e provisão de qualidade de serviço em redes OBS. Dos protocolos propostos para reserva de recursos, o JET destaca-se por fazer uso de reserva atrasada e do tempo de ajuste, permitindo que a rede opere sem a necessidade de FDLs. Além disso, nenhum pacote adicional é necessário para o término da reserva e liberação dos recursos. Ele é um protocolo eficiente e amplamente aceito na literatura.

Dentre os algoritmos de montagem de rajada, os que apresentam melhor desempenho são os algoritmos adaptativos. Tais algoritmos são flexíveis o suficiente para se adaptarem às condições da rede otimizando, assim, a montagem das rajadas e em algumas abordagens atendendo requisitos de QoS das camadas superiores.

Capítulo 3

Modelagem de tráfego com propriedades sensíveis à escala de tempo

Uma característica importante do tráfego gerado pelo protocolo IP (*Internet Protocol*) é a existência de padrões *scaling*. Tais padrões, impactam de forma significativa o desempenho dos mecanismos de controle de tráfego, e por isto, tem sido foco de intensa pesquisa [54]. Evidências da existência da natureza fractal em longas escalas de tempo do tráfego IP, bem como a presença de padrões divergentes em pequenas escalas de tempo foram identificadas há quase uma década [22] [25] [24] [18] [39].

Em escalas de tempo da ordem de centenas de milissegundos e maiores, o comportamento do tráfego pode ser modelado adequadamente por modelos auto-similares. Ao se variar a escala de tempo de observação de processos auto-similares, o comportamento observado é único e invariante no tempo, ou seja, tais processos apresentam um comportamento monofractal e normalmente dependem apenas de um único parâmetro, o parâmetro de *Hurst*.

Em pequenas escalas de tempo, na ordem de centenas de milissegundos e menores, o tráfego IP é altamente variável e a variabilidade difere das existentes em longas escalas de

tempo. Em tais escalas, o tráfego pode ser bem representado por modelos multifractais, já que os processos multifractais possibilitam comportamentos variantes em diferentes escalas de tempo, proporcionando maior flexibilidade em descrever fenômenos irregulares que são localizados no tempo.

Neste capítulo, apresenta-se o fenômeno *scaling*, bem como os demais conceitos sobre modelagem de tráfego utilizados no restante da tese.

3.1 Fenômeno *scaling*

Esta seção descreve brevemente diversos fenômenos *scaling* e a terminologia relacionada relevante a esta tese. O termo *"scaling"* provém da análise do tráfego em diferentes escalas de tempo.

Para descrição das propriedades *scaling*, considera-se o sinal $X(t)$, $t \in \mathbb{R}$ e o seu processo incremental X_Δ definido como

$$X_\Delta(k) := X((k+1)\Delta) - X(k\Delta), \quad k \in \mathbb{Z}, \tag{3.1}$$

onde Δ é uma constante, e sua derivada $Z(t)$ definida como

$$X(t) = \int_{\tau=0}^{t} dZ(\tau). \tag{3.2}$$

No contexto de tráfego em redes, $X(t)$ representa o total de bytes que chega no intervalo de tempo $[0, t]$, $X_\Delta(k)$ é o processo bytes-por-tempo obtido através da agregação do tráfego em intervalos de tempo de tamanho Δ e $Z(t)$ é a taxa de tráfego instantânea no tempo t.

3.1.1 Auto-similaridade

O processo $X(t)$, $t \in \mathbb{R}$, é auto-similar (monofractal) com parâmetro de *Hurst* $H > 0$, se

$$X(at) \stackrel{fd}{=} a^H X(t),\ t \in \mathbb{R},\ \forall a > 0, \tag{3.3}$$

onde $\stackrel{fd}{=}$ denota igualdade em distribuições de dimensões finitas. Essencialmente, nas diversas escalas de tempo, processos auto-similares mantêm suas propriedades estatísticas exceto por um fator constante escalar. Um exemplo importante de processo auto-similar é o movimento Browniano fracional (fBm - fractional Brownian motion).

Em séries temporais que representam o tráfego medido em pacotes, bytes ou bits, a auto-similaridade é uma propriedade referente à invariância da distribuição para qualquer incremento do processo, ou quando a série temporal apresenta a mesma estrutura de autocorrelação para diferentes níveis de agregação. A auto-similaridade do tráfego manifesta-se visualmente como uma invariância em relação à escala de tempo. Se forem construídos gráficos "Pacotes/Unidade de Tempo" x "Unidade de Tempo", para diversas escalas de tempo, pode-se verificar que o tráfego parece o "mesmo" nas diferentes escalas, ou seja, a agregação não resulta em suavidade.

Plotando-se o número de pacotes que chegam a cada 0,1 milisegundos, de um dos traços de nosso estudo, observou-se um processo altamente variável (primeiro gráfico da Figura 3.1). Por outro lado, plotando-se o número de pacotes que chegam em níveis de agregação maiores, a cada 1ms, 10ms, e 100ms (segundo, terceiro e quarto gráfico da Figura 3.1 respectivamente), ao invés de se obter um processo mais nivelado, o processo permaneceu tão variável quanto os de escalas de tempo menores. Este comportamento revela a natureza auto-similar do tráfego.

A importância do comportamento auto-similar está no fato de ser difícil determinar uma escala de tempo para o dimensionamento, pois o tráfego não se torna suavizado em torno de um valor médio para escalas maiores. Esta característica tem influência direta na distribuição da ocupação e do tempo de espera dos pacotes em uma fila.

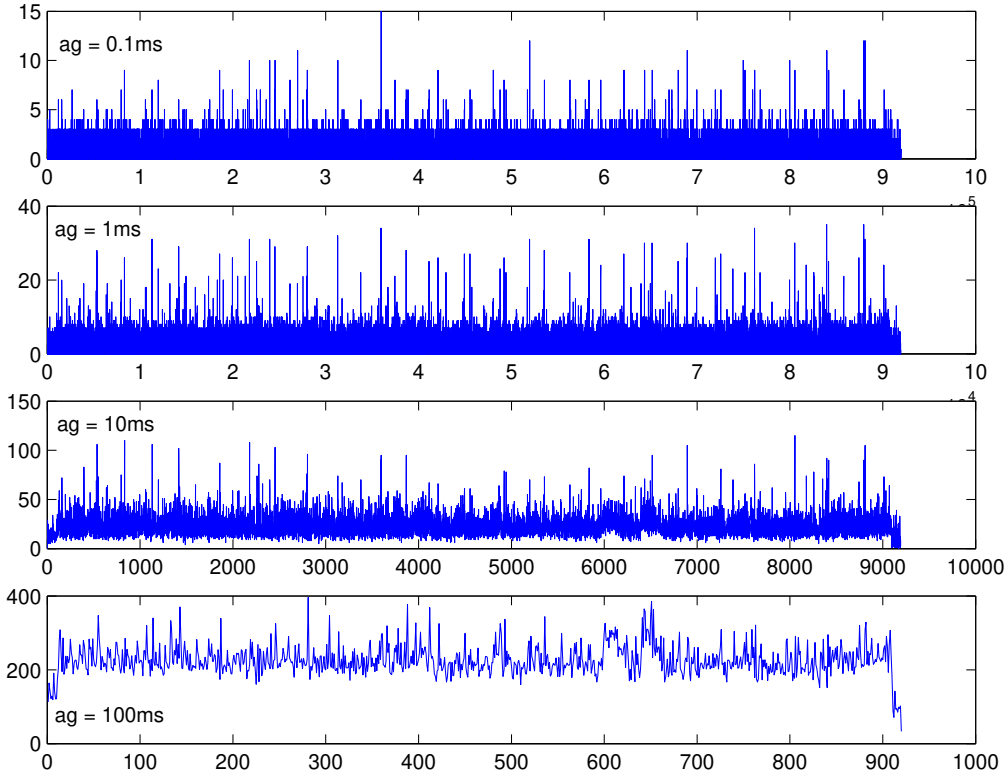


Figura 3.1: Quantidade de pacotes que chegam em escalas de tempo de 0,1ms, 1ms, 10ms e 100ms respectivamente (traço TXS-1111527905) [2].

3.1.2 LRD e processos $1/f$

Um processo estacionário $Y(k)$, $k \in \mathbb{Z}$, com função de autocovariância $r_Y(k)$ e função densidade espectral (SDF) $\Gamma_Y(f)$ apresenta dependência de longa duração (LRD - Long-Range Dependence) [17], se

$$\sum_k r_Y(k) = \infty \quad ou \quad \Gamma_Y(0) = \infty. \tag{3.4}$$

As correlações não desprezíveis quando ocorre grandes atrasos de tempo ou, equivalentemente, os componentes de freqüências baixas intensas, fazem a estimativa estatística dos processos LRD não trivial [7].

Exemplos clássicos de processos LRD são processos $1/f$- *noise* cujas funções densidade espectral se comportam como

$$\Gamma(f) \sim c|f|^{1-2h}, \quad f \to 0, \tag{3.5}$$

onde c é uma constante não nula e o expoente *scaling* $h \in (0,5\ ,\ 1)$. Aqui, $f(x) \sim l(x)$ denota $\lim_{x\to 0} f(x)/l(x) = 1$ quando $x \to 0$. O valor de h igual a $0,5$ corresponde a ruído branco (*white noise*) ou ausência de correlação, enquanto que h próximo de 1 indica um processo com intensa correlação.

3.1.3 Multifractalidade

Em contraste com a monofractalidade, a multifractalidade é uma propriedade que aparece em pequenas escalas de tempo e é caracterizada por momentos de ordem maior.

Pode-se definir valor para o expoente de *Hölder* do processo $X(t)$ (ou singularidade) $\beta(t)$ no tempo t, se

$$|X(t+\epsilon) - X(t)|^{\epsilon\to 0} \approx \epsilon^{\beta(t)}. \tag{3.6}$$

Um processo monofractal possui o mesmo $\beta(t)$ em todos os instantes de tempo t, enquanto que um processo multifractal possui $\beta(t)$ variando em t. O fBm é um exemplo de processo monofractal com $\beta(t) = H$, para todo t. No capítulo 4, será descrita a técnica que usa a transformada *wavelet* da derivada $Z(t)$ de $X(t)$ utilizada para determinar se $X(t)$ é multifractal.

Enquanto a monofractalidade é definida para escalas de tempo assintoticamente grandes, a multifractalidade é definida para escalas de tempo pequenas. Na prática, deve-se analisar os dados dentro de um intervalo finito de escalas de tempo.

3.2 Escala limitante

O tráfego IP pode apresentar comportamento diferente a depender da magnitude da escala de tempo em que é observado. Em escalas de tempo da ordem de centenas de milisegundos e acima, o tráfego IP apresenta comportamento auto-similar que é modelado com precisão por processos monofractais [4]. Em escalas de tempo menores, o tráfego IP pode apresentar comportamento *multi-scaling* não capturado completamente por processos monofractais, sendo melhor caracterizado por processos multifractais [4]. Tal comportamento pode ser constatado através da análise visual do gráfico log-log do processo incremento agregado $X_\Delta(i)$ em função do intervalo de agregação Δ, como descrito a seguir.

3.2.1 *Scaling* multifractal

Associado ao processo $X(t)$ de chegadas do tráfego no intervalo $[0, t]$ tem-se o processo incremento agregado $X_\Delta(i)$ definido por:

$$X_\Delta(i) = X(i\Delta) - X((i-1)\Delta) \tag{3.7}$$

Os momentos estatísticos do processo incremento comportam-se como [59]:

$$\sum_i X_\Delta(i)^q \sim c(q)\Delta^{-\tau(q)} \qquad \Delta \to 0 \tag{3.8}$$

quando $\tau(q)$ é linear em q, o comportamento *scaling* do tráfego é monofractal. Caso contrário é multifractal.

Aplicando-se o logaritmo na Equação 3.8, tem-se:

$$\log(\sum_i X_\Delta(i)^q) \approx -\tau(q)\log(\Delta) + \log(c(q)) \tag{3.9}$$

O que mostra que o $\log(\sum_i X_\Delta(i)^q)$ depende linearmente do $\log(\Delta)$ e que $\tau(q)$ representa a inclinação da reta obtida.

Para avaliar a função $\tau(q)$ recorre-se à função soma partição. Considere o processo de chegada $X(i), 1 \leq i \leq N$, no intervalo $[0, T]$, em uma escala $\delta = T/N$. Define-se soma partição como:

$$\sum_i X_\Delta(i)^q = \sum_{k=1}^{N/\Delta} (X_k^\Delta)^q \tag{3.10}$$

onde

$$X_k^\Delta = \sum_{l=1}^{\Delta} X[(k-1)\Delta + l] \tag{3.11}$$

é o processo original observado em uma escala de agregação $\delta = T\Delta/N$.

Agregando-se o processo $X(i)$ usando a função soma partição para um dado valor de q_i e variando-se Δ, obtém-se um conjunto de pontos no plano $\log(\Delta)\text{x}\log(\sum_i X_\Delta(i)^q)$. Assim, a função $\tau(q_i)$ pode ser obtida fazendo-se regressão linear nos pontos do plano $\log(\Delta)\text{x}\log(\sum_i X_\Delta(i)^q)$. Constrói-se finalmente a função $\tau(q)$ através da interpolação dos valores de $\tau(q_i)$ em diferentes momentos estatísticos.

3.2.2 *Scaling* monofractal

Considere um processo $Y(t)$ e seu processo associado de incrementos estacionários $Y_\Delta(i) = Y(i\Delta) - Y((i-1)\Delta)$. Assuma que $Y(t)$ é monofractal como definido na Equação 3.3 i.e.

$$Y(Ci) \stackrel{d}{=} C^H Y(i).$$

Seja

$$Y_k^\Delta = \frac{1}{\Delta} \sum_{l=1}^{\Delta} Y[(k-1)\Delta + l] = \frac{1}{\Delta} \overline{Y}_k^\Delta.$$

Então [59]:

$$Y_\Delta(i) \stackrel{d}{=} \Delta^{1-H} Y_k^\Delta. \tag{3.12}$$

Como proposto em [64] um teste da característica auto-similar do tráfego pode ser

realizado através da análise do comportamento dos momentos do processo. Seja:

$$\mathbb{E}|Y^{\Delta}|^q = \frac{1}{N/\Delta}\sum_{k=1}^{N/\Delta}|Y_k^{\Delta}|^q$$

Em analogia a análise de multifractalidade apresentada na seção 3.2.1 tem-se:

$$\sum_i Y_{\Delta}(i)^q = \sum_{k=1}^{N/\Delta}|\overline{Y}_k^{\Delta}|^q = \Delta^{q-1}N\cdot\mathbb{E}|Y^{\Delta}|^q$$

Se $Y_{\Delta}(i)$ é auto-similar como definido pela Equação 3.12 tem-se:

$$\log(\sum_i Y_{\Delta}(i)^q) \approx \gamma(q)\log(\Delta) + const \tag{3.13}$$

Além disso, $\gamma(q)$ é linear em q tal que:

$$\gamma(q) = qH - 1. \tag{3.14}$$

Dessa forma, o teste de auto-similaridade proposto por Taqqu et al. [64] determina se existe γ tal que a Equação 3.13 é valida. Se γ depende linearmente de q então o tráfego é auto-similar.

3.2.3 Identificação da característica scaling do tráfego

Como discutido nas seções 3.2.1 e 3.2.2, o tráfego de rede pode apresentar comportamentos distintos em diferentes escalas de tempo. Tal comportamento pode ser visualmente observado no gráfico log-log plot do processo agregado de incremento $X_{\Delta}(i)$ como uma função do intervalo de agregação Δ.

Em grandes escalas de tempo, o tráfego é modelado por processos monofractais (Equação 3.12), fazendo com que o gráfico log-log apresente comportamento linear, de acordo com a Equação 3.13. A inclinação das retas é, então, dada pela Equação 3.14.

Em pequenas escalas de tempo o tráfego é modelado por processos multifractais, apresentando comportamento de acordo com a Equação 3.8. O gráfico em tais escalas de tempo também pode apresentar comportamento linear [59], entretanto, é possível observar no gráfico log-log uma mudança de comportamento das curvas regida pela presença de diferentes coeficientes angulares ($\tau(q)$ e $\gamma(q)$), orientando a inclinação das retas formadas nos diferentes regimes. A escala Δ^* identificada como escala de transição entre os dois regimes é denomidada escala limitante, ou escala de tempo limite (do inglês cutoff time scale) [22].

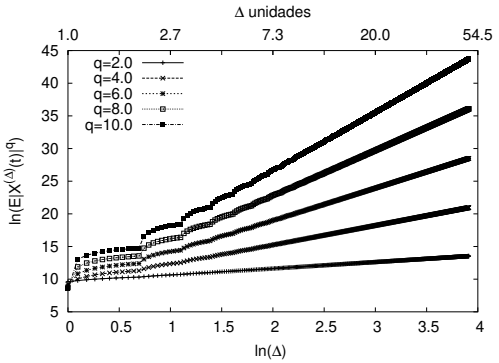


Figura 3.2: Escala de tempo limite

Na Figura 3.2, mostram-se os resultados obtidos quando se avalia a relação entre os momentos estatísticos e as escalas de tempo de uma medida multifractal. Esta medida é uma cascata multiplicativa binomial com os multiplicadores distribuídos conforme uma função Beta simétrica de parâmetro $p = 1.6$. Na avaliação realizada, as escalas de tempo Δ indicam o número de realizações do processo original $X(t)$ que é usado para gerar uma realização do processo agregado $\sum_i X_\Delta(i)^q$. As curvas da Figura 3.2 apresentam dois comportamentos: o primeiro é o comportamento não-linear das curvas, principalmente quando a escala de tempo (Δ) é inferior a 3 (três) unidades. O segundo comportamento verificado é a não convergência das curvas para um valor específico e uma mudança brusca no comportamento das curvas a partir da escala de tempo $\Delta = 3$. A não linearidade das curvas é destacada pela ocorrência da escala limitante, conforme definição em [22], com

uma mudança brusca no comportamento das curvas. A escala limitante pode então ser definida a grosso modo como a escala de agregação além da qual o processo é considerado monofractal.

3.3 Resumo conclusivo

No tráfego IP, há evidências da existência da natureza fractal em longas escalas de tempo, bem como a presença de padrões divergentes em pequenas escalas de tempo. Neste capítulo foi apresentada uma breve descrição de diversos fenômenos *scaling* existentes no tráfego das redes. Ficou claro que em escalas de tempo de centenas de milissegundos e maiores, o tráfego apresenta comportamento monofractal, enquanto que em escalas menores o tráfego apresenta comportamento multifractal. Foi também evidenciada a presença da escala limitante, que é a escala de agregação do tráfego a partir da qual o tráfego deixa de ser modelado como multifractal e passa a ser modelado como monofractal.

Capítulo 4

Análise do comportamento do tráfego IP em diferentes escalas de tempo

Ferramentas estatísticas baseadas em transformadas *wavelet* de séries temporais são utilizadas para o estudo das propriedades do comportamento *scaling* local e global [5]. Estas ferramentas são apropriadas para a detecção e identificação de comportamentos *scaling* e para a estimativa estatística subsequente de parâmetros relevantes de tais séries. Neste capítulo, tais ferramentas são apresentadas. Além disso, apresenta-se resultados de estudo realizado para verificar a ocorrência do fenômeno *scaling* em traços de tráfego IP, utilizados no restante desta tese, quando estes são agregados em escalas de tempo pequenas.

4.1 Detecção da natureza *scaling* do tráfego

Nesta seção, são apresentados alguns conceitos sobre a natureza *scaling* de tráfego, discutidos sob o ponto de vista das transformadas *wavelet*. Um processo auto-similar (mono-

fractal) $X(t)$ de ordem q apresenta momentos estatísticos definidos por [4]:

$$E|X(t)|^q = E|X(1)|^q.|t|^{qH}, \tag{4.1}$$

onde H é o parâmetro de *Hurst*. Esta definição dos momentos estatísticos de um processo monofractal $X(t)$, impõe a restrição de uniformidade nas variações (explosividade) de $X(t)$ em diferentes instantes de tempo, ou seja, assume-se que o processo possui explosividade uniforme, medida por H, em diferentes instantes de tempo. Um processo multifractal não apresenta a restrição de uniformidade nas suas variações e tem seus momentos estatísticos definidos por [4]:

$$E|X(t)|^q = E|X(1)|^q.|t|^{\zeta(q)}, \tag{4.2}$$

onde $\zeta(q)$ é a função cascateamento. Esta função apresenta um comportamento não-linear para os diferentes momentos q, o que caracteriza a ocorrência de multifractalidade.

No domínio wavelet, os momentos são expressos da seguinte forma:

$$E|d_X(j,k)|^q \approx 2^{j\zeta(q)} \qquad j \to -\infty, \tag{4.3}$$

onde $d_X(j,k)$ é a série de incrementos (detalhes) obtidos pela decomposição do processo $X(t)$ usando a transformada wavelet discreta. Os coeficientes da transformada *wavelet* discreta são definidos como:

$$d_X(j,k) = \int_{-\infty}^{\infty} X(t)\psi_{j,k}(t)dt, \tag{4.4}$$

onde $X(t)$ é um processo estacionário de tempo contínuo e o membro $\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k)$ é gerado pela wavelet mãe $\psi(t)$ através da dilatação de uma escala $x = 2^j$ e translação de $2^j k$. Em um octavo fixo j, a sequência $d_X(j,\cdot)$ corresponde à análise de $X(t)$ na escala 2^j. Octavo j é o logaritmo da escala a na base 2, ou seja, a escala $a = 2^j$.

A função cascateamento $\zeta(q)$ é então definida por:

$$\zeta(q) = \alpha_q - \frac{q}{2}, \tag{4.5}$$

onde α_q é chamado de expoente de cascateamento (do inglês *scaling exponent*). Este expoente tem o seu valor relacionado à explosividade do tráfego, que no caso de processos multifractais varia para os diferentes momentos estatísticos q.

4.1.1 Diagrama logescala

A estimativa de α_q é realizada através do método diagrama logescala (do inglês *Logscale Diagram*-LD) do q-ésimo momento. Neste método, α_q é definido pela inclinação de uma reta que se aproxima da curva gerada pela relação entre $S_q(j)$ e 2^j em uma escala logarítmica. A determinação de $S_q(j)$ é discutida a seguir.

Inicialmente, são estudadas as propriedades globais da série, que são as estatísticas das séries temporais vistas em cada nível de resolução (ou escala), definidas como uma função escalar. A energia média contida em cada escala da série é examinada e a mudança destes valores na medida em que se vai de escalas mais grossas (escalas maiores) para escalas mais finas (escalas menores) é, então, medida. A energia média na escala j é a média da soma dos coeficientes *wavelet* elevado ao quadrado,

$$S_2(j) = \frac{1}{n_j} \sum_k |d_X(j,k)|^2, \tag{4.6}$$

onde n_j é o número de coeficientes *wavelet* $d_X(j,k)$ disponíveis no octavo j. O gráfico do logaritmo destas estimativas versus j é o chamado Diagrama logescala (LD):

$$\text{LD: } log_2 S_2(j) \;\; vs \;\; log_2 a = j \;,$$

Para determinar a propriedade global da série, determina-se qualitativamente sob qual intervalo existe uma relação linear entre $S_2(j)$ e 2^j em escala logarítimica, ou seja, sob qual intervalo de escalas de tempo existe *scaling*.

Em [66] encontra-se disponível um conjunto de rotinas desenvolvido por Abry e Veitch implementado no Software Matlab. Estas rotinas generalizam o Diagrama logescala, que faz a análise do *scaling* de estatísticas de segunda-ordem, nos Diagramas logescala de q-ésimas-ordens, para $q \in \mathbb{R}$, usando as estimativas da função partição:

$$S_q(j) = \frac{1}{n_j} \sum_{k=1}^{n_j} |d_X(j,k)|^q \approx E|d_X(j,k)|^q, \tag{4.7}$$

onde n_j é número de detalhes $d_X(j,.)$, na escala de tempo j, gerados pela decomposição de $X(t)$, usando-se uma transformada wavelet discreta.

O gráfico de $S_q(j)$ versus 2^j em escala logarítimica representa o Diagrama logescala de q-ésima-ordem (q-LD) e permite a análise do *scaling* local (Figura 4.1).

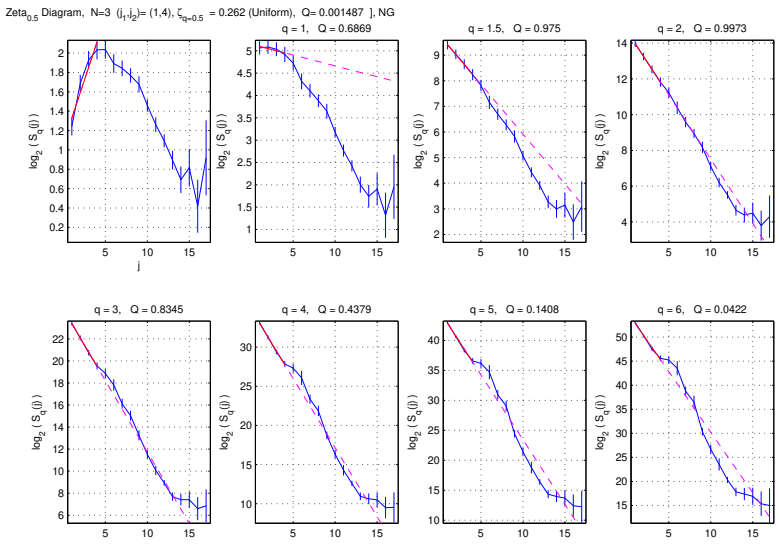


Figura 4.1: Diagramas logescala de q-ésima ordem do fluxo IP do traço MEM-1111247410.

Esta generalização é fundamental para o estudo da multifractalidade, pois um processo multifractal não pode ser descrito por um único expoente como o parâmetro de *Hurst* H, e momentos de maiores ordens também precisam ser analisados, fazendo-se necessário obter

um conjunto de expoentes α_q.

Detecção de *scaling*

O objetivo da análise do *scaling* local é coletar informações características sobre os picos e a localização dos vários expoentes α_q. Para um q fixo, um padrão é visto como uma linha reta no Diagrama logescala de q-ésima ordem, com pelo menos quatro octavos de comprimento, e a medida de sua inclinação (*slope*) é a estimativa do expoente *scaling* α_q correspondente (Figura 4.1). Se para cada q, uma linha reta é encontrada no q-LD sob uma mesma faixa de escalas, então há presença de *scaling*. O intervalo de valores usado foi $q \in [0, 6]$. Este intervalo foi escolhido através de uma escolha conservadora baseada em estudos relatados em [67].

A princípio não se sabe em quais escalas, se houver alguma, existe *scaling*. Na prática, esta decisão é feita com os valores octavos de corte menor e maior, j_1 e j_2 respectivamente, de cada intervalo suspeito de haver *scaling*. A heurística usada consiste na existência de uma linha de regressão que corte cada intervalo de confiança do intervalo $[j_1, j_2]$ escolhido, isto é, o intervalo $[j_1, j_2]$ deve apresentar uma linearidade nos q-LD. Para a regressão ser bem definida ao menos quatro escalas de tempo são necessárias e, sendo Q o valor retornado no teste *Chi-square goodness of fit*, Q deve ser maior que 0,05. Quanto maior for o valor de Q, maior será a linearidade do intervalo.

4.1.2 Diagrama multiescala

Para testar se um processo possui multifractalidade em um intervalo $[j_1, j_2]$ que possui *scaling*, dois parâmetros foram introduzidos: $\zeta_q = \alpha_q - q/2$ e $h_q = \zeta_q/q$. Em [4], descreve-se um método, chamado de diagrama multiescala (do inglês *Multiscale Diagram*-MD), para se determinar a ocorrência de multifractalidade em um processo. Este método consiste em verificar o comportamento da função $\zeta(q)$ em relação aos momentos estatísticos q. Caso a função $\zeta(q)$ apresente um comportamento não-linear verifica-se a ocorrência de multifractalidade no processo avaliado.

Além do Diagrama Multiescala, o Diagrama Multiescala Linear (do inglês *Linear Multiscale Diagram*) também pode ser usado para detectar multifractalidade. Ele traça $h_q = \zeta_q/q$ *versus* q. Neste diagrama, a monofractalidade é revelada por uma curva horizontal, ao passo que a multifractalidade pode ser detectada por uma curva não horizontal.

O MD e o LMD do fluxo IP (Figura 4.2) do traço MEM-1111247410 apresentaram respectivamente não-linearidade e não-horizontalidade, indicando evidências de multifractalidade.

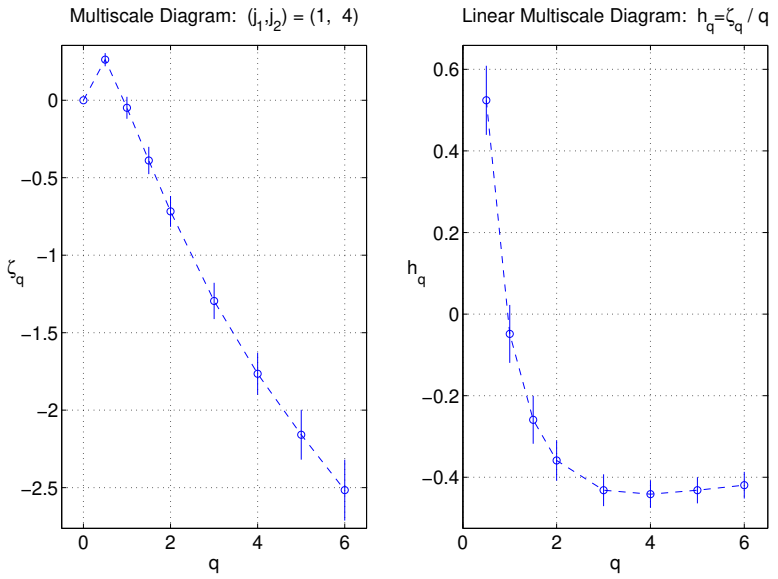


Figura 4.2: Diagrama Multiscale e Diagrama Multiscale Linear do fluxo IP do traço MEM-1111247410.

Interpretação do *scaling*

A existência de linearidade no Diagrama multiescala e horizontalidade no Diagrama multiescala Linear descrevem processos monofractais, enquanto que a não-linearidade no Diagrama multiescala e a não-horizontalidade no Diagrama multiescala Linear descrevem processos multifractais. Um ponto essencial no uso da ferramenta é que o conceito de

alinhamento é relativo aos intervalos de confiança, e não ao alinhamento próximo dos expoentes ζ_q em si. Os intervalos de confiança dos expoentes aumentam monotonicamente com o aumento do valor de j para escalas cada vez maiores.

As conclusões sobre a natureza *scaling* dos traços selecionados para este estudo são descritas e detalhadas a seguir.

4.2 Descrição dos traços analisados

Foram analisados traços de pacotes coletados da Internet com diferentes perfis obtidos do projeto de pesquisa PMA (The NLANR Passive Measurement and Analysis Project), que disponibiliza em seu WEB *site* dezenas de traços por dia, provenientes de pontos de coleta de diversas partes do mundo [2].

Os traços de tráfego real utilizados tiveram a ocorrência de multifractalidade verificada através dos métodos diagrama multiescala e diagrama multiescala linear. Os traços usados nessa avaliação são de domínio público e contêm o registro do tráfego de redes operacionais no período de 2003 a 2005. Esses traços registraram o tráfego em pontos de agregação das redes vBNS e Internet2 ABILENE e foram obtidos no sítio NLANR (www.nlanr.net). A Tabela 4.1 resume as informações dos traços.

Para garantir a generalidade do estudo, os traços oriundos de pontos de coleta diversos foram escolhidos aleatoriamente. Os traços estudados são provenientes de enlaces OC3, OC48 e OC192. Os traços brutos foram processados com a Ferramenta CAIDA Coralreef [62] e com programas em C++ desenvolvidos pelos autores, permitindo a extração do horário de chegada (com precisão em microsegundos), protocolo IP e tamanho (em bytes) de cada pacote do traço. Neste estudo somente as informações relativas aos fluxos IP são utilizadas.

A Figura 4.3 apresenta os gráficos log-log dos momentos estatísticos *q's* do processo incremento agregado do tráfego IP dos traços ANL-1111548257, MEM-1111247410, MEM-1111679715, MEM-1112013766 e TXS-1113503155 e a Figura 4.4 os gráficos log-log dos

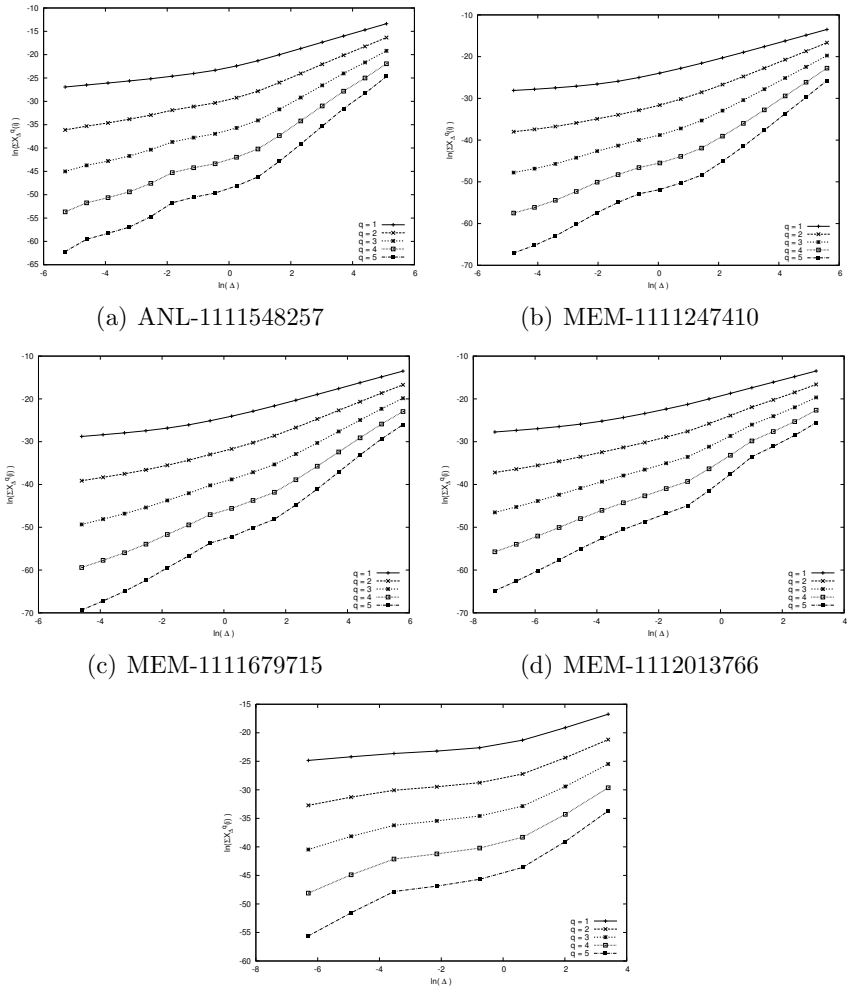
Tabela 4.1: Descrição dos traços de tráfego real usados na tese.

Traço	Data	Tamanho	Numero de pacotes IP	Link	Taxa (Mbps)	Ponto de agregação
ANL-1111548257	03/22/2005 20:11	561 KB	42420	OC3	2,3	Argonne National Lab.
MEM-1111247410	03/19/2005 07:56	1,1 MB	75122	OC3	2,6	University of Memphis
MEM-1111679715	22/03/2005 14:10	1.5MB	103372	OC3	4,4	University of Memphis
MEM-1112013766	03/28/2005 04:49	980 Kb	70259	OC3	1,8	University of Memphis
MEM-1053844177	05/24/2003 23:54	1,2MB	78122	OC3	2,65	University of Memphis
TXS-1113503155	04/14/2005 11:31	226 KB	16895	OC3	6,8	Texas universities
IPLS-CLEV-20020814-090000-0	14/08/2002 09:00	782MB	42998801	OC48	412,131	Abilene (Indianapolis)
IPLS-CLEV-20020814-090000-1	14/08/2002 09:00	730MB	40153607	OC48	457,852	Abilene (Indianapolis)
IPLS-CLEV-20020814-091000-0	14/08/2002 09:10	767MB	41230376	OC48	363,754	Abilene (Indianapolis)
20040601-193121-0	06/01/2004 19:00	1,2GB	56624949	OC192	770,00	Abilene (Indianapolis)
20040601-193121-1	06/01/2004 19:00	1,3GB	67363371	OC192	1.648	Abilene (Indianapolis)
20040601-194000-1	06/01/2004 20:00	1,4GB	71710474	OC192	828,616	Abilene (Indianapolis)

Tabela 4.2: Característica *Scaling* dos traços de tráfego usados na tese.

Trace	Δ^*	Valor médio do expente de Holder	Var	I. C.
ANL-1111548257	2.7ms	0.726	0.03	0.007
MEM-1111247410	3.3ms	0.695	0.0085	0.01
MEM-1111679715	5.4ms	0.758	0.008	0.031
MEM-1112013766	3.0ms	0.72	0.009	0.005
MEM-1053844177	6.7ms	0.687	0.007	0.003
TXS-1113503155	1.3ms	0.89	0.0408	0.03
IPLS-CLEV-20020814-090000-0	3ms	0.83	0.0400	0.03
IPLS-CLEV-20020814-090000-1	2,8ms	0.792	0.007	0.001
IPLS-CLEV-20020814-091000-0	3,2ms	0.675	0.001	0.003
20040601-193121-0	1.6ms	0.683	0.0014	0.009
20040601-193121-1	1.3ms	0.689	0.0021	0.0013
20040601-194000-1	1.3ms	0.622	0.0027	0.001

momentos estatísticos *q's* do processo incremento agregado do tráfego IP dos traços IPLS-CLEV-20020814-090000-0, IPLS-CLEV-20020814-090000-1, IPLS-CLEV-20020814-091000-0, 20040601-193121-0, 20040601-193121-1 e 20040601-194000-1 . Cada curva de cada gráfico representa um momento estatístico $q = [1, 2, 3, 4, 5, 6]$. O valor de Δ^* é definido pela escala de tempo na qual ocorre a mudança de comportamento das curvas dos momentos. Como observado nas figuras, há uma mudança no comportamento das curvas dos fluxos no ponto correspondente à escala de tempo limite. Por exemplo, na Figura 4.3(a), há uma mudança no comportamento das curvas do traço ANL-1111548257 quando $ln(\Delta)$ é aproximadamente 1,1. Como $ln(\Delta^*) \sim 1,1$ então, $\Delta^* \sim 3ms$. Todas as escalas abaixo de Δ^* estão tipicamente no regime de pequenas escalas e fazem parte das escalas de interesse deste estudo.



(a) ANL-1111548257

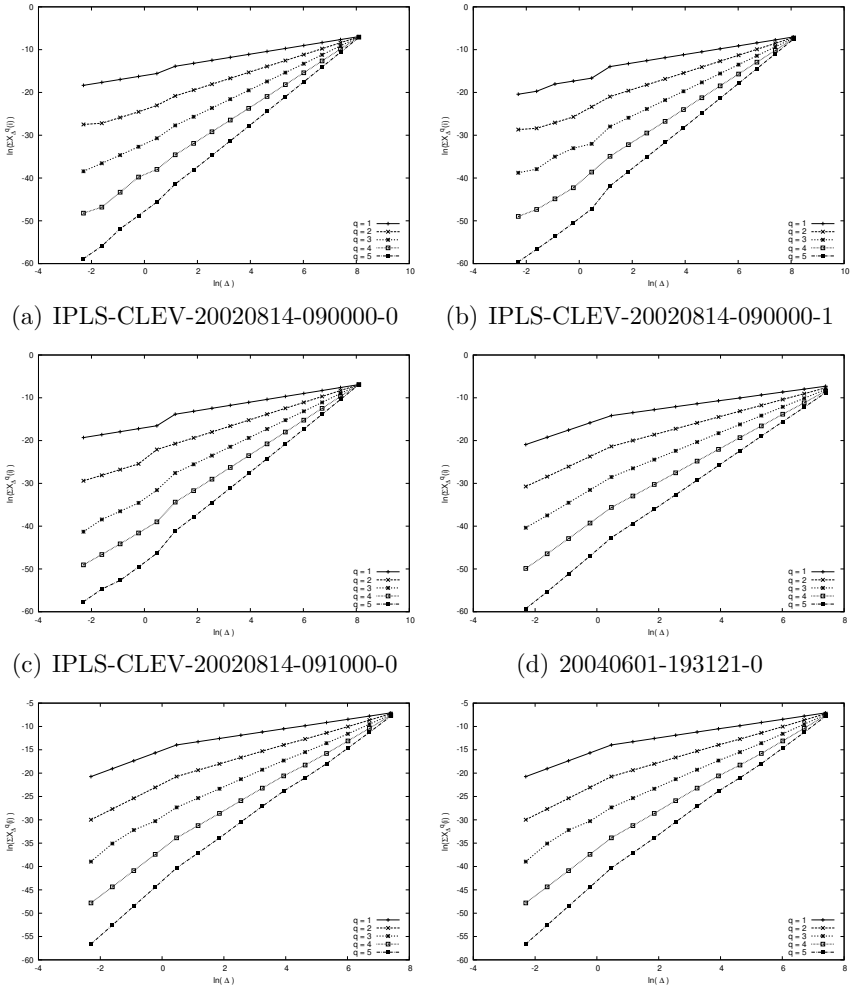
(b) MEM-1111247410

(c) MEM-1111679715

(d) MEM-1112013766

(e) TXS-1113503155

Figura 4.3: Log-log plot dos momentos estatísticos do processo incremento agregado dos fluxos IP dos traços ANL-1111548257, MEM-1111247410, MEM-1111679715, MEM-1112013766 e TXS-1113503155 .



(a) IPLS-CLEV-20020814-090000-0 (b) IPLS-CLEV-20020814-090000-1

(c) IPLS-CLEV-20020814-091000-0 (d) 20040601-193121-0

(e) 20040601-193121-1 (f) 20040601-194000-1

Figura 4.4: Log-log plot dos momentos estatísticos do processo incremento agregado dos fluxos IP dos traços IPLS-CLEV-20020814-090000-0, IPLS-CLEV-20020814-090000-1, IPLS-CLEV-20020814-091000-0, 20040601-193121-0, 20040601-193121-1 e 20040601-194000-1.

Para identificar as propriedades *scaling* do tráfego, foram utilizadas as ferramentas descritas anteriormente. As análises estatísticas de segunda ordem e de ordens superiores foram feitas usando os Diagramas logescala de q-ésima ordem, o Diagrama multiescala e o Diagrama multiescala Linear.

Através do Diagrama logescala de q-ésima ordem, para todos os traços estudados foi encontrado um intervalo contendo *scaling*, cujas escalas de tempo são menores que o respectivo Δ^*.

Os Diagramas logescala de q-ésima ordem resultantes da análise dos fluxos IP do traço ANL-1111548257 são mostrados na Figura 4.5. O valor de Q, exibido no canto superior direito de cada diagrama, corresponde ao valor de linearidade retornado no teste de regressão para a reta contida no intervalo $[j_1, j_2]$. Os altos valores de Q encontrados indicam linearidade, e portanto, revelam a presença de *scaling* no intervalo testado $[j_1, j_2] = [1, 4]$.

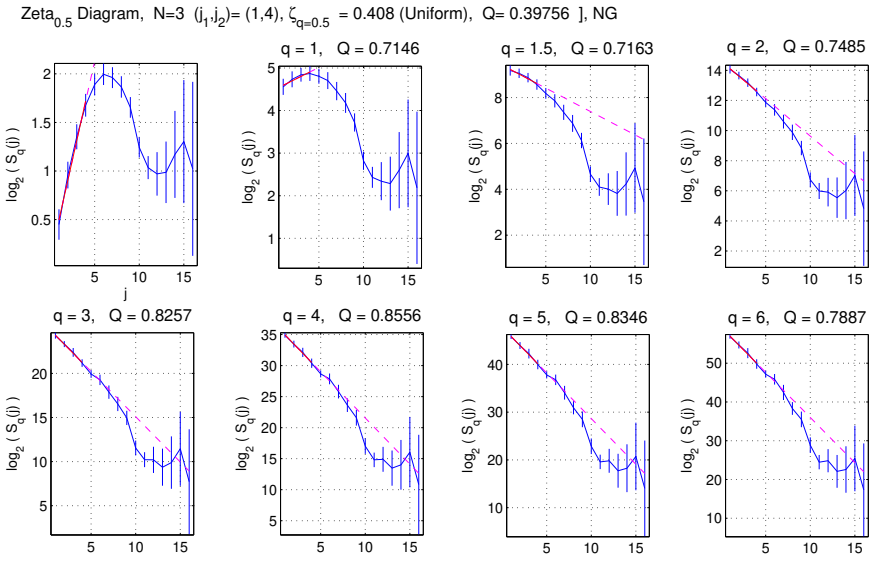


Figura 4.5: Diagramas logescala de q-ésima ordem do fluxo IP do traço ANL-1111548257.

Para testar se um processo é multifractal, estimou-se para cada Diagrama logescala de q-ésima ordem a inclinação da reta encontrada nas pequenas escalas de tempo.

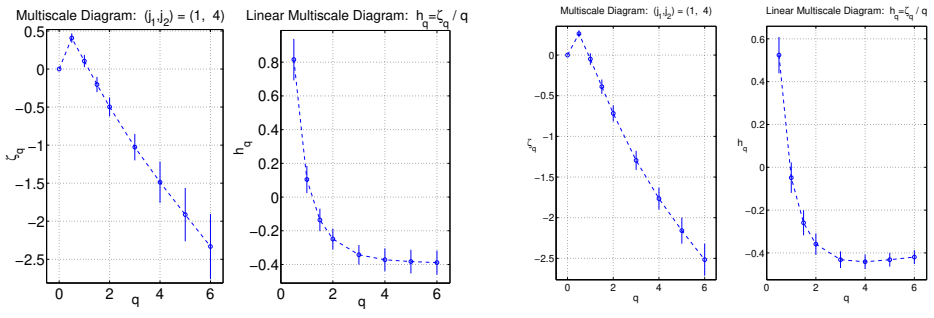


Figura 4.6: Diagrama Multiscale e Diagrama Multiscale Linear do fluxo IP do traço ANL-1111548257 (a) e MEM-1111247410 (b).

A partir das inclinações estimadas, finalmente, obteve-se os Diagramas multiescala e os Diagramas multiescala Lineares para os fluxos IP de cada traço.

Exemplificando, os MD e LMD dos fluxos IP dos traços ANL-1111548257(Figura 4.6(a)) e MEM-1111247410 (Figura 4.6(b)) apresentaram, respectivamente, não-linearidade e não-horizontalidade, indicando evidências de multifractalidade.

4.3 Resumo conclusivo

Este capítulo apresentou conceitos sobre o comportamento *scaling* do tráfego expressos no domínio *wavelet*. Apresentou também as ferramentas utilizadas para avaliação do comportamento *scaling* observado nos traços de tráfego utilizados o restante desta tese. Finalmente, apresentou de forma resumida os resultados da avaliação do traços de tráfego.

Capítulo 5

Mudanças de escala no tráfego em redes OBS

Um dos aspectos fundamentais para o dimensionamento da rede é a caracterização das estatísticas do tráfego a ser transportado e, em especial, caracterização da explosividade do tráfego em diferentes escalas de tempo. No capítulo 3 discutiu-se a presença da escala de tempo limitante em fluxos multifratctais. Viu-se que o tráfego quando agregado em escalas de tempo acima da escala de tempo limitante pode ser modelado como monofractal, enquanto que o tráfego agregado em escalas de tempo menores que a escala de tempo limitante é modelado como multifractal. Dado que as redes OBS operam agrupando pacotes em rajadas, é possível que esse processo cause mudanças nas propriedades do tráfego. Neste capítulo, avalia-se se o processo de montagem de rajadas usado nas redes OBS pode realizar transformações nas propriedades estatísticas do tráfego que adentra a rede OBS.

5.1 Análise de Mudança de Escala

Para avaliar as mudanças das características estatísticas do tráfego, experimentos de simulação foram realizados usando o simulador NS-2 equipado com o módulo OBSns [1].

Os experimentos foram conduzidos de forma a avaliar as mudanças das propriedades estatísticas do tráfego decorrente do processo de montagem de rajadas. Nos experimentos, um nó OBS de borda é alimentado com tráfego IP multifractal. O tráfego de saída do montador é então coletado e analisado como ilustrado na Figura 5.1.

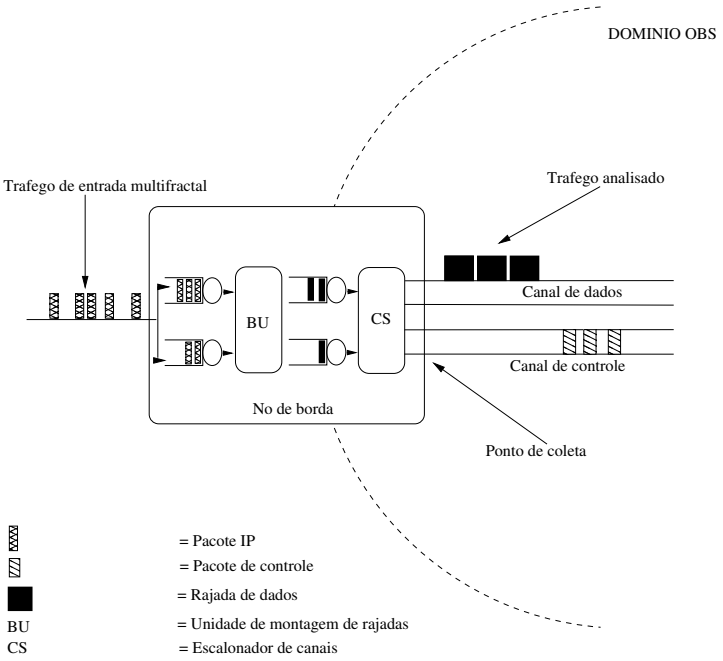


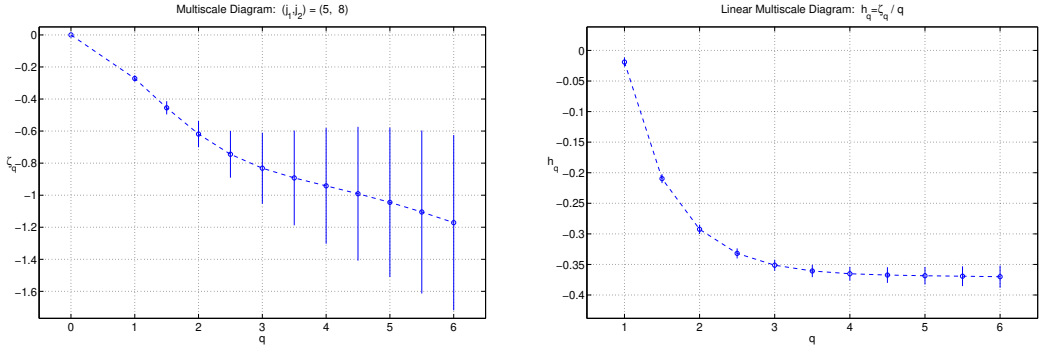
Figura 5.1: Cenário usado nas simulações.

O expoente de Holder para os traços foram calculados de acordo com o procedimento apresentado em [10]. A Tabela 5.1 apresenta a escala limitante (Δ^*) e o valor máximo do temporizador usado na montagem das rajadas (t_i), a média e a variância do expoente de Holder, bem como o intervalo de confiança usado para o cálculo da média. O valor escolhido para o temporizador abaixo da escala limitante ($t_i < \Delta*$) corresponde a dez vezes o valor usado para realizar a agregação do respectivo traço de tráfego.

As Figuras 5.2 e 5.3 mostram o Diagrama Multiescala (MD) e o Diagrama Linear Multiescala (LMD) do tráfego que alimenta o nó de borda da rede OBS. É importante notar que o fenômeno da multifractalidade pode ser observado tanto no diagrama MD quanto no LMD.

Tabela 5.1: Expoente de Holder e escala de tempo limite do tráfego multifractal de entrada.

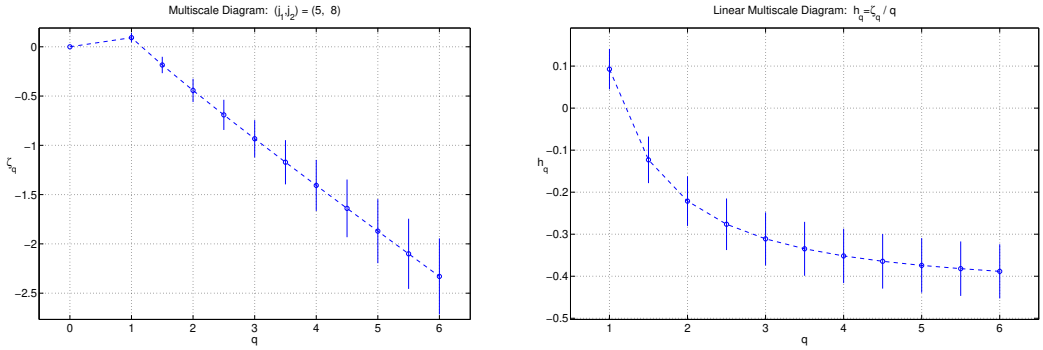
Traço	Δ^*	$t_i > \Delta^*$	$t_i < \Delta^*$	Valor médio do expente de Holder	Var	I. C.
ANL-1111548257	2.7ms	3ms	1ms	0.726	0.03	0.007
MEM-1111247410	3.3ms	4ms	1ms	0.695	0.0085	0.01
MEM-1111679715	5.4ms	6ms	3ms	0.758	0.008	0.031
MEM-1112013766	3.0ms	4ms	1ms	0.72	0.009	0.005
MEM-1053844177	6.7ms	7ms	6ms	0.687	0.007	0.003
TXS-1113503155	1.3ms	2ms	1ms	0.89	0.0408	0.03
IPLS-CLEV-20020814-090000-0	3ms	4ms	1ms	0.83	0.0400	0.03
IPLS-CLEV-20020814-090000-1	2,8ms	4ms	1ms	0.792	0.007	0.001
IPLS-CLEV-20020814-091000-0	3,2ms	4ms	1ms	0.675	0.001	0.003
20040601-193121-0	1.6ms	2ms	1ms	0.683	0.0014	0.009
20040601-193121-1	1.3ms	2ms	1ms	0.689	0.0021	0.0013
20040601-194000-1	1.3ms	2ms	1ms	0.622	0.0027	0.001



(a) Diagrama Multiescala do Traço MEM-1111679715.

(b) Diagrama Multiescala Linear do Traço MEM-1111679715.

Figura 5.2: Diagramas Multiescala e Multiescala Linear do Traço MEM-1111679715.



(a) Diagrama Multiescala do traço TXS-1113503155.

(b) Diagrama Multiescala Linear do Traço TXS-1113503155.

Figura 5.3: Diagramas Multiescala e Multiescala Linear do Traço TXS-1113503155.

A próxima seção apresenta os resultados obtidos, avaliando a influência das políticas de montagem baseadas em janelas de tempo e em volume de tráfego nas propriedades do tráfego resultante do processo de montagem de rajadas. Devido a limitações de espaço, resultados são apresentados apenas para os traços MEM-1111679715 and TXS-1113503155.

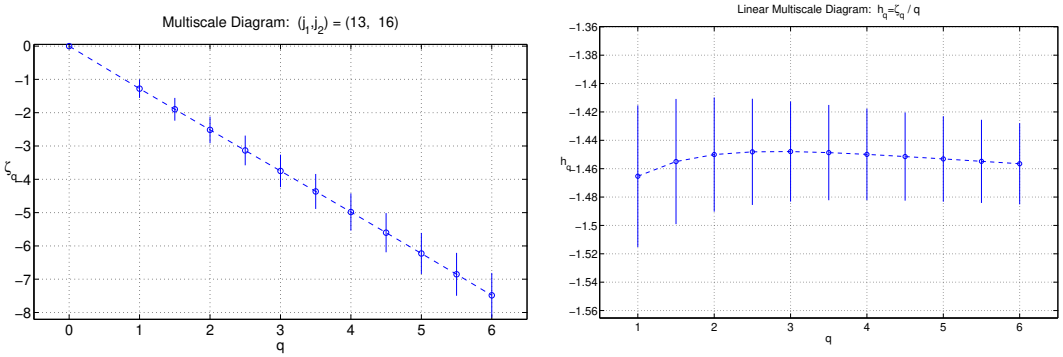
5.2 Resultados numéricos

5.2.1 O impacto dos limiares de tempo na escala do tráfego de saída

No primeiro cenário, t_i é maior que o valor da escala limitante (Δ^*) do tráfego de entrada. No segundo cenário, t_i é menor que Δ^*. Com esses cenários, pretende-se verificar o relacionamento entre o valor do limiar usado na montagem das rajadas e as propriedades estatísticas do tráfego resultante.

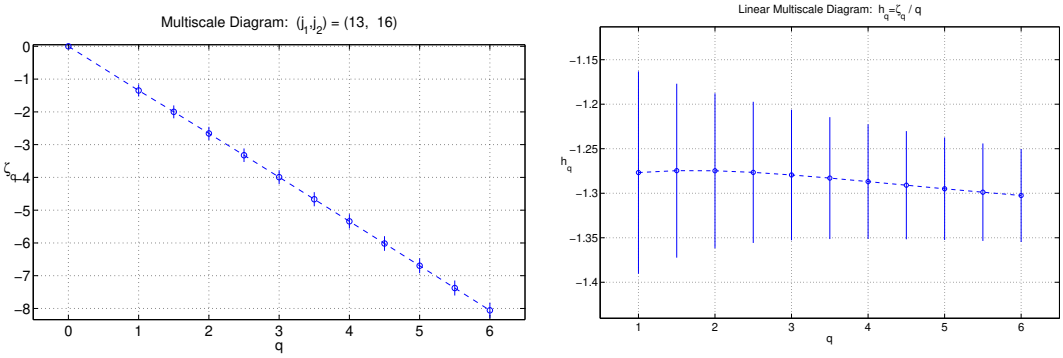
Cenário 1: $t_i > \Delta^*$

As Figuras 5.4 e 5.5 mostram os diagramas MD e LMD do tráfego de saída do montador de rajadas. Este tráfego é resultante das transformações do tráfego de entrada dos traços MEM-1111679715 and TXS-1113503155, respectivamente. O comportamento linear das curvas no diagrama MD indica a monofractalidade do tráfego de saída, dado que a função de cascateamento $\zeta(q)$ apresenta comportamento linear nos vários momentos estatísticos q, o que pode ser confirmado pelo diagrama LMD que mostra alinhamento horizontal para os dois traços. Pode-se concluir que o processo de montagem de rajadas em escalas de tempo maiores que a escala limitante de um tráfego multifractal transforma as características do tráfego de entrada de multifractal para monofractal. A segunda coluna da Tabela 5.3 mostra o parâmetro de Hurst (H_t) do tráfego monofractal resultante do processo de montagem usando janelas de tempo.



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagrama Linear Multiescala Linear do Tráfego de Saída.

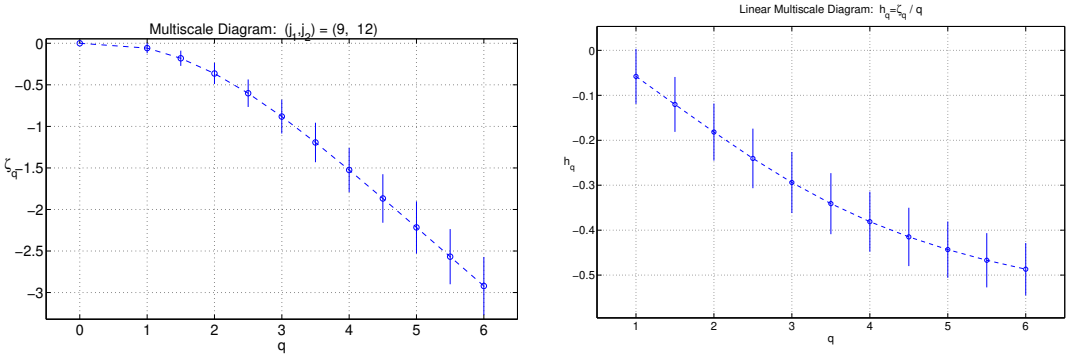
Figura 5.4: Análise do Traço MEM-1111679715 para $t_i > \Delta^*$.



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagrama Multiescala Linear do Tráfego de Saída.

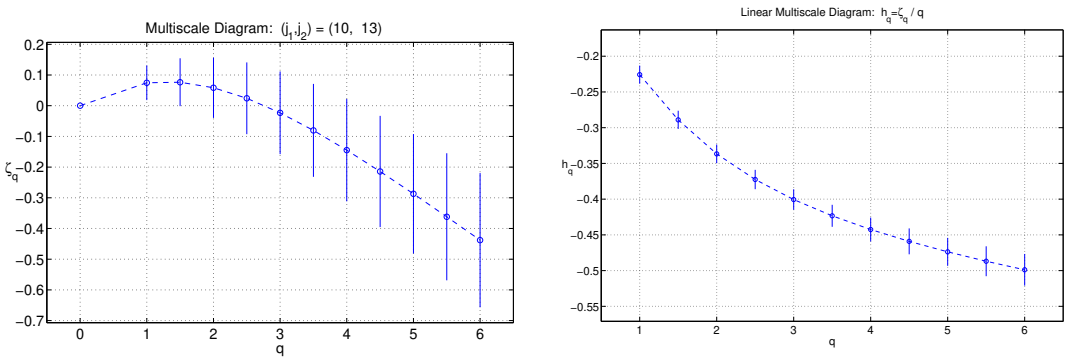
Figura 5.5: Análise do Traço TXS-1113503155 para $t_i > \Delta^*$.

Cenário 2: $t_i < \Delta^*$



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagram Multiescala Linear do Tráfego de Saída.

Figura 5.6: Análise do traço MEM-1111679715 para $t_i < \Delta^*$.



(a) Diagrama Multiescala do Tráfego d Saída. (b) Diagrama Multiescala do Tráfego de Saída.

Figura 5.7: Análise do Traço TXS-1113503155 para $t_i < \Delta^*$.

As Figuras 5.6 e 5.7 mostram os resultados para $t_i < \Delta^*$. As curvas do diagrama MD tem um comportamento não linear que indica a multifractalidade. O padrão multiescala pode ser percebido pela ausência de alinhamento horizontal nas curvas do diagrama LMD. Pode-se concluir, portanto, que o processo de montagem de rajadas em escalas de tempo menores que a escala limitante do tráfego de entrada mantém as propriedades multifractais do tráfego.

5.2.2 O impacto do limiar do volume de tráfego na escala do tráfego de saída

Como ilustrado na seção anterior, a natureza das propriedades estatísticas do tráfego resultante do processo de montagem de rajadas, depende do relacionamento entre o valor limite do temporizador (t_i) usado na política de montagem e o valor da escala limitante do tráfego de entrada.

Para investigar o relacionamento entre a escala de tempo do processo de montagem e a escala limitante do tráfego de entrada, o volume de tráfego de entrada (b_i) foi dividido pela taxa média de chegadas (λ). Essa escala de tempo é então comparada à escala limitante do tráfego de entrada.

Os experimentos foram conduzidos usando limiares para o contador de bytes que implicassem em escalas de montagem abaixo da escala limitante do tráfego de entrada ou em escalas de montagem acima da escala limitante do tráfego de entrada. Os valores escolhidos foram, respectivamente, 1KB [29] e 125KB [43].

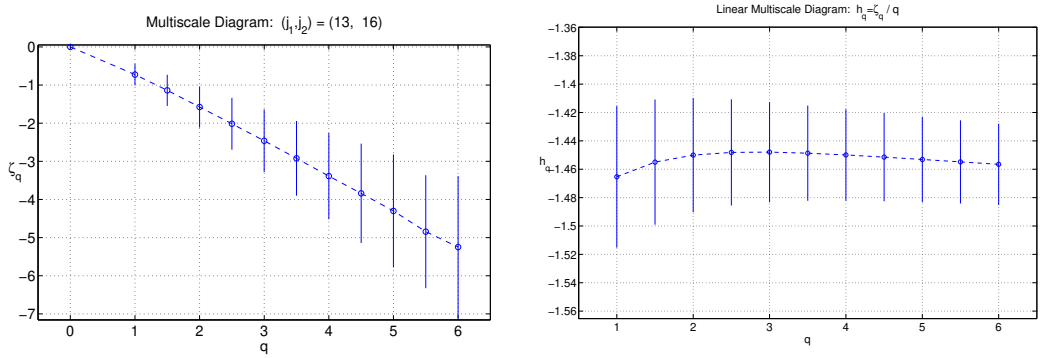
Os traços MEM-1111679715 e TXS-1113503155 têm taxa média de chegadas (λ) de 4.4 e 6.8 Mbps, respectivamente. O contador de bytes de 1KB produz uma escala de tempo de montagem (t_i) de 1,8 e 1,2 ms para os traços MEM-1111679715 e TXS-1113503155. Esses valores são menores que a escala limitante limite (Δ^*) do tráfego de entrada. Por outro lado, o contador de bytes de 125KB corresponde a escala do tempo de 0,22 e 0,15 segundos para os traços MEM-1111679715 e TXS-1113503155. Essas escalas são maiores que a escala limitante do tráfego de entrada.

Cenário 1: $b_i/\lambda > \Delta^*$

As Figuras 5.8 e 5.9 mostram os resultados da análise do tráfego resultante do processo de montagem de rajadas com $b_i/\lambda > \Delta^*$. Os diagramas MD mostram comportamentos similares. A função de cascateamento $\zeta(q)$ apresenta comportamento linear. O comportamento monofractal é confirmado pelo alinhamento horizontal apresentado no diagrama

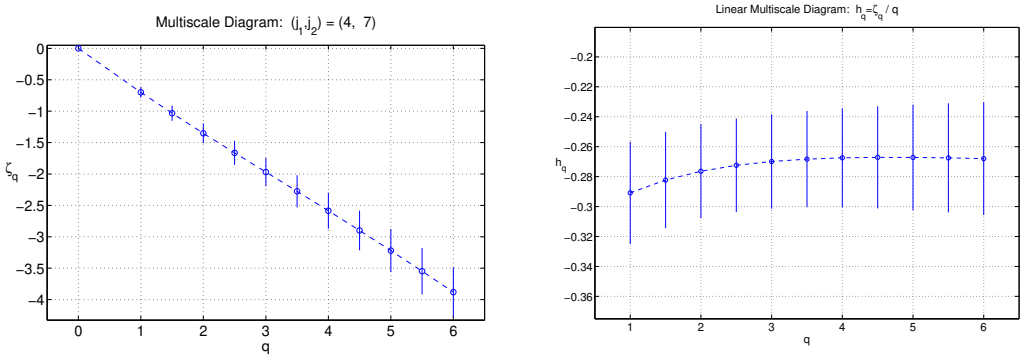
LMD. Tal comportamento da função de cascateamento caracteriza a ocorrência de monofractalidade.

Assim, o processo de montagem de rajadas usando limiares maiores que o produto entre a taxa média de chegadas e o valor da sua escala limitante transforma um tráfego com características multifractais em um tráfego com características monofractais. A terceira coluna da Tabela 5.3 mostra o parâmetro de Hurst (H_v) do tráfego resultante do processo de montagem, quando a política de montagem de rajadas baseada em volume de tráfego é utilizada.



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagrama Multiescala do Tráfego de Saída.

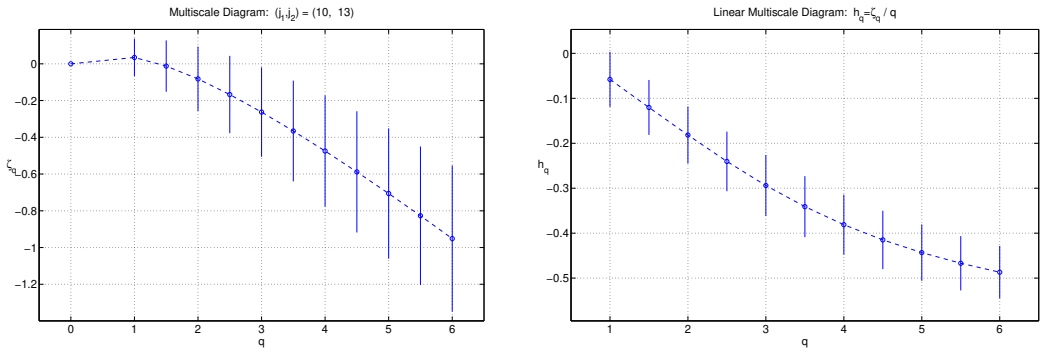
Figura 5.8: Análise do Traço MEM-1111679715 para $b_i/\lambda > \Delta^*$.



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagrama Multiescala Linear do Tráfego de Saída.

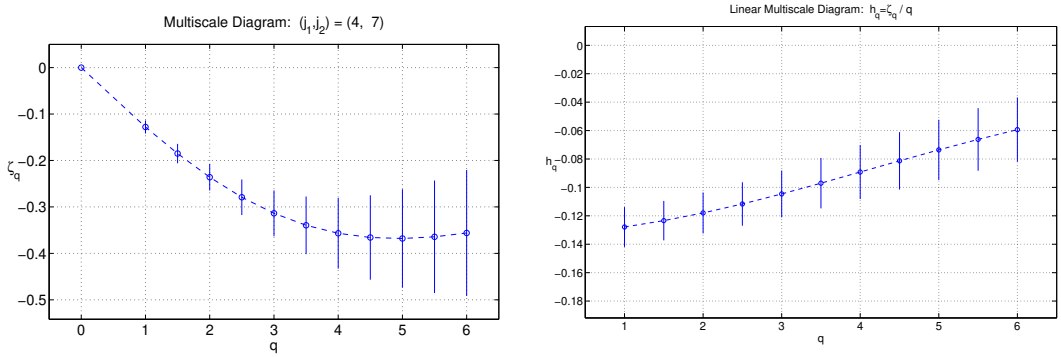
Figura 5.9: Análise do Traço TXS-1113503155 para $b_i/\lambda > \Delta^*$.

Cenário 2: $b_i/\lambda < \Delta^*$



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagrama Multiescala do Tráfego de Saída.

Figura 5.10: Análise do Traço MEM-1111679715 para $b_i/\lambda < \Delta^*$.



(a) Diagrama Multiescala do Tráfego de Saída. (b) Diagrama Multiescala Linear do Tráfego de Saída.

Figura 5.11: Análise do Traço Trace TXS-1113503155 para $b_i/\lambda < \Delta^*$.

As Figuras 5.10 a 5.11 mostram a análise do tráfego resultante do processo de montagem de rajadas com $b_i/\lambda < \Delta^*$. No diagrama MD, as curvas apresentam comportamento não linear o que indica a presença de multifractalidade no tráfego. Além disso, o diagrama LMD mostra alinhamento não horizontal.

Tabela 5.2: Expoente de Holder do Tráfego Multifractal de Saída.

Traço	$Media_t$	Var_t	$I.C._t$	$Media_v$	Var_v	$I.C._v$
MEM-1111247410	0.601	0.006	0.009	0.653	0.002	0.001
MEM-1111679715	0.598	0.008	0.011	0.632	0.004	0.002
MEM-1112013766	0.601	0.009	0.003	0.622	0.004	0.002
ANL-1111548257	0.546	0.008	0.002	0.603	0.016	0.020
TXS-1113503155	0.655	0.010	0.013	0.732	0.008	0.015

Tabela 5.3: Parâmetro de Hurst do Tráfego Monofractal de Saída.

Traço	H_t	H_v
MEM-1111247410	0.669	0.682
MEM-1111679715	0.595	0.687
MEM-1112013766	0.629	0.675
ANL-1111548257	0.598	0.639
TXS-1113503155	0.672	0.797

5.2.3 O Efeito Suavizador das Políticas de Montagem de Rajadas

A fim de avaliar a suavização do tráfego causada por diferentes processos de montagem de rajadas, as propriedades estatísticas do tráfego produzido por essas políticas foram comparadas.

A Tabela 5.2 apresenta a média e a variância dos valores assumidos pelo expoente de Holder do tráfego multifractal resultante do processo de montagem, bem como o intervalo de confiança do valor médio. Os sub-índices t e v denotam, respectivamente, as políticas baseadas em janelas de tempo e em volume de tráfego. A Figura 5.12 apresenta os resultados para comparação visual.

Quando comparados à média e à variância do tráfego de entrada, fica claro que o processo de montagem suaviza o tráfego de entrada. Isto é evidenciado pela redução da média e da variância do expoente de *Holder*. A redução da média do expoente de Holder

Figura 5.12: Média do expoente de Holder dos tráfegos de entrada e saída do montador de rajadas.

indica que a explosividade do tráfego em escalas de tempo menores foi reduzida. Além disso, a redução da variância indica que a variabilidade da explosividade do tráfego nessas escalas de tempo também foi reduzida. Comparando as duas políticas, pode-se notar que a média do expoente de Holder produzida pela política de montagem de rajadas baseada em janelas de tempo é menor que o valor produzido pela política baseada em volume de tráfego, o que indica uma suavização do tráfego em pequenas escalas de tempo.

A Tabela 5.3 apresenta o parâmetro de *Hurst* do tráfego resultante do processo de montagem, quando as políticas baseadas em janelas de tempo e em volume de tráfego foram utilizadas. O parâmetro de Hurst foi calculado usando o estimador A-V disponível em [66]. Pode-se notar que as políticas baseadas em volume de tráfego produzem tráfego monofractal com parâmetro de Hurst mais alto que o parâmetro de Hurst produzido pela política baseada em janelas de tempo. Uma hipótese para a explicação desse fenômeno é que a política baseada em volume de tráfego produz rajadas maiores que a política baseada em janelas de tempo (Tabela 5.4). Na política baseada em janelas de tempo, alguns pacotes pertencente ao mesmo fluxo são transmitidos em rajadas diferentes enquanto que na política baseada em volume de tráfego esses mesmos pacotes são transmitidos em uma única rajada. Períodos maiores de atividade e de silêncio são, consequentemente, produzidos nas políticas baseadas em volume de tráfego o que pode, hipoteticamente, levar a maiores dependências de longa duração.

5.2.4 O impacto das mudanças do tráfego no dimensionamento da rede

Para avaliar o impacto causado pelas transformações das propriedades estatísticas do tráfego no dimensionamento da rede OBS, o cenário descrito na Figura 5.13 foi utilizado nas simulações. A figura apresenta uma rede composta por 3 nós de borda (2 de ingresso e 1 de egresso) e um nó de núcleo. Os nós de borda de ingresso são alimentados com tráfego multifractal gerados a partir dos traços de tráfego citados na Tabela 4.1. Cada enlace possui uma única fibra com 16 canais de dados e 2 canais de controle.

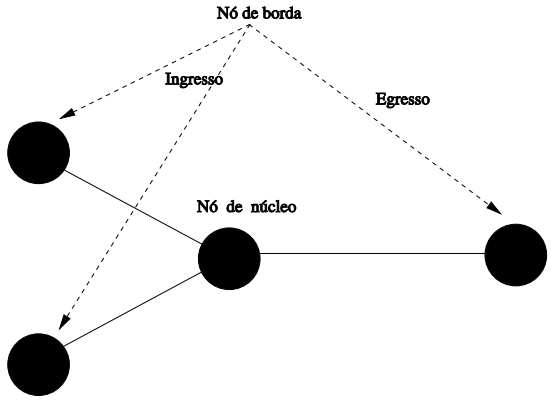


Figura 5.13: Cenário utilizado na avaliação do impacto do tráfego no dimensionamento da rede.

A idéia é avaliar se as transformações ocorridas no tráfego têm impacto significativo no dimensionamento da rede. Para tal, foi medida a probabilidade de bloqueio (PB), a quantidade de canais adicionais (em relação ao dimensionamento usado para medir a probabilidade de bloqueio) para que a rede experimente probabilidade de bloqueio igual a zero (C) e o número médio de pacotes em cada rajada (NP).

A Tabela 5.4 apresenta os resultados da simulação. Pode-se perceber que tanto na política de montagem baseada em volume de tráfego, quanto na política baseada em janelas de tempo, a rede OBS experimenta maior probabilidade de bloqueio, quando o tráfego resultante do processo de montagem de rajadas possui características multifractais. Isto acontece devido à alta frequência

Tabela 5.4: Probabilidade de Bloqueio e Demanda de Canais Adicionais.

Política	Volume de tráfego						Janelas de tempo					
Tráfego	Multifractal			Monofractal			Multifractal			Monofractal		
Trace	PB	C	NP	PB	C	NP	PB	C	NP	PB	C	NP
ANL57	4%	1	3,1	0%	0	255	2%	1	3,0	0%	0	258
MEM10	2%	1	4,7	0%	0	351,8	1%	1	2,4	0%	0	351,5
MEM15	1%	1	5,6	0%	0	289,1	2%	1	1,8	0%	0	267,2
MEM66	3%	1	2,8	0%	0	711,4	1,3%	1	2,7	0%	0	451,2
TXS55	0%	0	7,4	0%	0	451,2	0.1%	1	1,4	0%	0	306,5
Média	2%	1	4,7	0%	0	411,7	1,3%	1	2,0	0%	0	326,9

de rajadas e pacotes de controle gerados. Como pode ser visto na Tabela 5.4, a quantidade de pacotes por rajadas (NP) é de 4,7, quando a política de motagem baseada em volume de tráfego é utilizada e de 2,0 quando a política baseada em janelas de tempo é utilizada. Isto gera um número elevado de pacotes de controle na rede e, consequentemente, uma quantidade maior de canais é necessária para realizar o escalonamento de todas as rajadas e pacotes de controle. A ausência temporária desses recursos faz, obviamente, com que a probabilidade de bloqueio seja maior.

Comparando as políticas de montagem, percebe-se que a política baseada em volume de tráfego produz probabilidades de bloqueios mais altas do que a política baseada em janelas de tempo, dado que a política baseada em volume de tráfego produz tráfego com maior explosividade e a sua demanda de recursos é mais elevadas do que a política baseada em janelas de tempo.

5.3 Resumo conclusivo

Investigou-se, neste capítulo, o efeito de políticas de montagem de rajada nas características *scaling* do tráfego IP que adentra um domínio OBS. Resultados obtidos via simulação revelaram que os mecanismos de montagem de rajadas causam o efeito de suavização da explosividade do tráfego de entrada da rede. A média e a variância do expoente de Holder do tráfego multifractal resultante do processo de montagem de rajadas são menores do que aquelas encontradas no

tráfego que adentra o domínio OBS. Além disso, resultados mostram, também, que a escolha dos valores usados no temporizador (em caso de políticas baseadas em janelas de tempo) ou no contador de bytes (no caso das políticas baseadas em volume de tráfego) têm impacto significativo nas propriedades estatísticas do tráfego.

Mostrou-se que se o limiar do temporizador (no caso da política de montagem baseada em janelas de tempo) ou do produto entre a taxa média do tráfego de entrada e o valor da escala limitante (no caso específico de políticas baseadas em volume de tráfego) for maior que o valor da escala limitante do tráfego original, a montagem de rajadas produz tráfego com características monofractais, enquanto que o uso de valores menores que o valor da escala de tempo limitante produz tráfego com características multifractais.

As políticas baseadas em volume de tráfego produzem tráfego monofractais com parâmetro de Hurst mais alto que as políticas baseadas em janelas de tempo. Além disso, políticas baseadas no volume de tráfego produzem tráfego multifractal cujo expoente de Holder possui valores mais altos que os produzidos por políticas baseadas em janelas de tempo. Indica-se, portanto, o uso de políticas de montagem de rajadas baseadas em janelas de tempo.

Além disso, para diminuir a demanda de recursos da rede, recomenda-se que os valores dos limiares usados nas políticas de montagem de rajadas sejam escolhidos, de forma a produzir tráfego com características monofractais. Recomenda-se, portanto, que limiares para políticas de montagem sejam adotados em função da escala limitante do tráfego que adentra um domínio OBS.

Capítulo 6

Algoritmo para determinação automática da escala limitante

Como discutido no Capítulo 5, o processo de montagem das redes OBS podem provocar suavização no tráfego da rede e uma diminuição na demanda por recursos da mesma. Além disso, viu-se que tais transformações dependem da escolha dos parâmetros de montagem de rajadas em função da escala limitante dos fluxos multifractais. Entretanto, como mencionado, a detecção da escala limitante de um fluxo de tráfego multifractal é feita através de inspeção visual do diagrama log-log do processo incremento agregado em função do intervalo de agregação. Esta análise por vezes resulta em erro já que pequenas variações na estrutura dos gráficos são comuns e quase imperceptíveis de serem visualmente notadas.

Neste capítulo, apresenta-se um método baseado no modelo de regressão linear dos mínimos quadrados para obtenção da escala limitante de um fluxo multifractal.

6.1 Detecção da escala limitante de fluxos multifractais

Como exposto anteriormente, a escala limitante determina a escala de agregação a partir da qual o fluxo apresenta estatísticas monofractais. Isto implica que, a partir da escala limitante,

o conjunto de pontos (x_i, y_i); $x_i = \log(\Delta)$ e $y_i = \log(\sum_i X_\Delta(i)^q)$, dispõem-se linearmente no plano $\log(\Delta)\text{x}\log(\sum_i X_\Delta(i)^q)$.

Dado um conjunto de n pontos $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, a qualidade do modelo de uma regressão linear simples é definida pelo coeficiente de determinação, R^2. Ele é uma medida da proporção da variabilidade em uma variável, que é explicada pela variabilidade da outra. Assim em uma correlação perfeita, $R^2 = 1$. O coeficiente de determinação é dado pela seguinte expressão:

$$R^2 = \frac{\left(\left(\sum_{i=1}^{n} y_i^2\right) - n\overline{y}^2\right) - \sum_{i=1}^{n} y_i^2 - b_0 \sum_{i=1}^{n} y_i - b_1 \sum_{i=1}^{n} x_i y_i}{\left(\sum_{i=1}^{n} y_i^2\right) - n\overline{y}^2} \tag{6.1}$$

com $b_0 = \overline{y} - b_1\overline{x}$ e $b1 = \frac{\sum xy - n\overline{xy}}{\sum x^2 - n(\overline{x})^2}$.

Considere o conjunto de pares ordenados $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$. Sabe-se que existe um alinhamento entre os pontos $\{(x_{\Delta^*}, y_{\Delta^*}), ..., (x_n, y_n)\}$, onde $(x_{\Delta^*}, y_{\Delta^*})$ representa o ponto correspondente à escala limitante. Além disso, sabe-se que a função formada pelos pontos $\{(x_1, y_1), ..., (x_{\Delta^*-1}, y_{\Delta^*-1})\}$, ou apresenta comportamento não linear, ou apresenta comportamento linear com inclinação diferente da apresentada pelos pontos $\{(x_{\Delta^*}, y_{\Delta^*}), ..., (x_n, y_n)\}$. Assim, o método proposto para detecção da escala limitante consiste na observação do comportamento da função formada pelos diferentes valores assumidos pelo coeficiente de determinação à medida que novos pontos (x_i, y_i) são adicionados à regressão.

Partindo do ponto (x_n, y_n) o método incrementalmente re-calcula o valor do coeficiente de determinação ao adicionar mais pontos ao conjunto. Como os pontos de $\{(x_{\Delta^*}, y_{\Delta^*}), ..., (x_n, y_n)\}$ possuem alinhamento, é esperado que o valor de R^2 seja próximo de um, quando a qualidade da regressão linear é boa. Por outro lado, à medida que os pontos pertencentes à região de não linearidade são adicionados ao conjunto de pontos, o valor de R^2 se deteriora. Desta forma, é possível saber o ponto de transição entre os regimes monofractal e multifractal. O método é apresentado no Algoritmo 1.

O algoritmo recebe, para cada momento estatístico q (q variando entre q_{max}, que corresponde ao maior valor que q pode assumir e q_{min}, o menor valor que q pode assumir), um conjunto S, de n pontos (correspondente ao número de escalas de agregações a que o

Algoritmo 1 calculaΔ^*

ENTRADA

Conjunto de n pares ordenados, o maior momento estatístico a ser analisado (q_{max}), o menor momento estatístico a ser analisado (q_{min}).

SAÍDA

Escala limitante Δ^*.

Calcula Δ^*

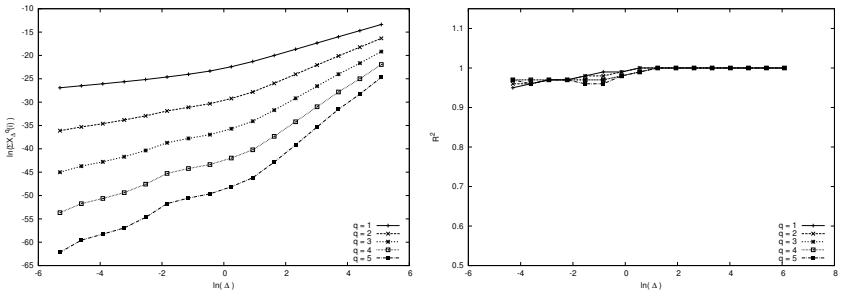
```
1: M ← 0
2: for all ( q_min ≤ q ≤ q_max): do
3:    S ← {}
4:    i ← n
5:    S ← ∪{(x_i, y_i)}
6:    Calcule R^2 sobre os pontos de S
7:    while ((i ≥ 1) ∧ (R^2 ≥ δ)) do
8:       i ← (i − 1)
9:       S ← ∪{(x_i, y_i)}
10:      Calcule R^2 sobre os pontos de S
11:   if (R^2 < δ) then
12:      if (i ≥ M) then
13:         M ← i
14: Δ* ← M
15: Retorne Δ*
```

processo $X(t)$ foi submetido) e retorna o valor aproximado da escala limitante Δ.

O valor de R^2 é calculado sobre um conjunto de pontos contidos em S. Aumenta-se S incrementalmente, adicionando-se novos pontos. R^2 é re-calculado até que esteja abaixo de um limiar pré-determinado, δ, que pode ser usado para ajustar o nível de precisão do método. Se o valor de δ for escolhido próximo a um, isto significa que os pontos de S devem estar alinhados precisamente.

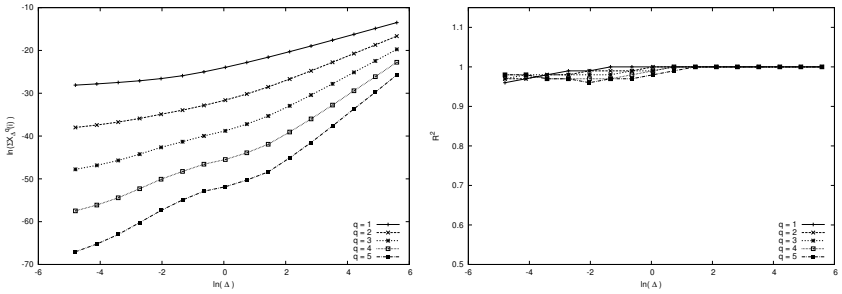
O algoritmo ainda mantém a variável M, que armazena o valor máximo, entre todos os momentos estatísticos q, da escala de tempo em que a regressão linear deixa de ser aceitável, correspondendo assim à escala limitante.

As Figuras 6.1 a 6.11 apresentam os resultados da avaliação do método proposto quando aplicado aos traços apresentados no Capítulo 4. Por exemplo, a Figura 6.1(a) apresenta a escala limitante de um fluxo multifractal encontrado no traço de tráfego ANL-1111548257. Percebe-se que a escala limitante é encontrada quando $\ln(\Delta) \approx 1.048$,



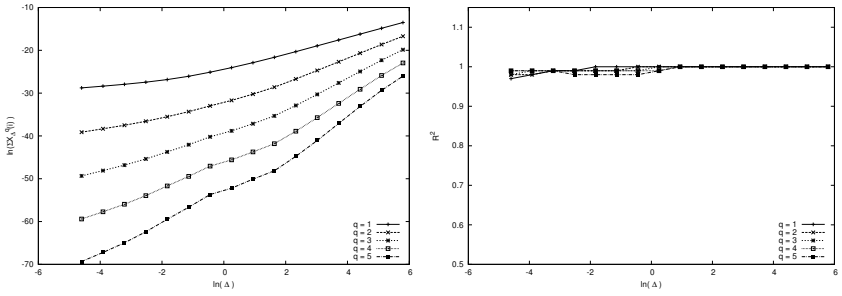
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.1: Detecção da escala limitante do traço ANL-1111548257.



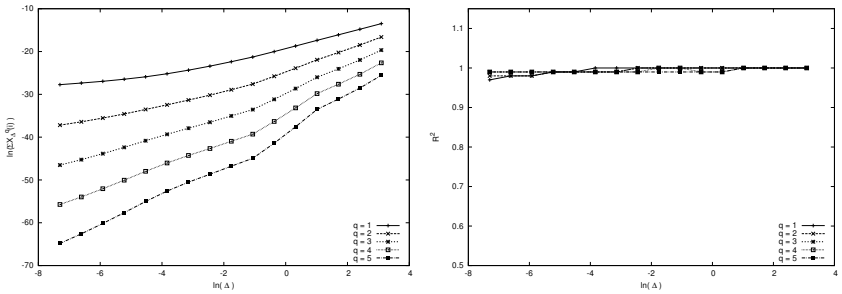
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.2: Detecção da escala limitante do traço MEM-1111247410.



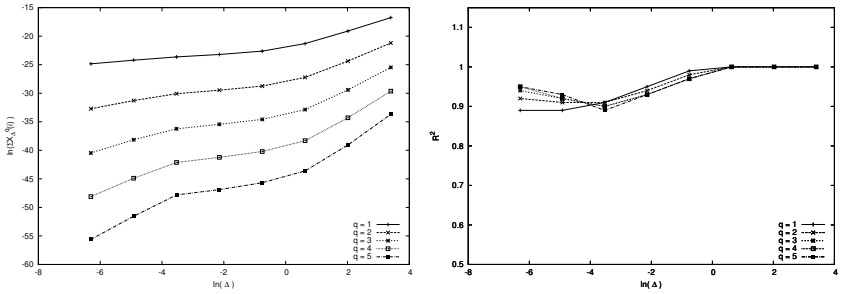
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.3: Detecção da escala limitante do traço MEM-1111679715.



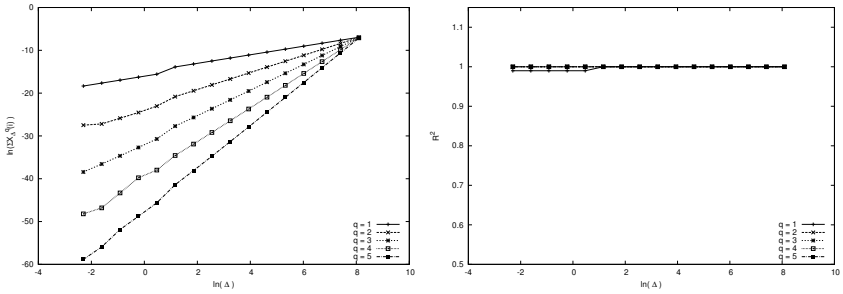
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.4: Detecção da escala limitante do traço MEM-1112013766.



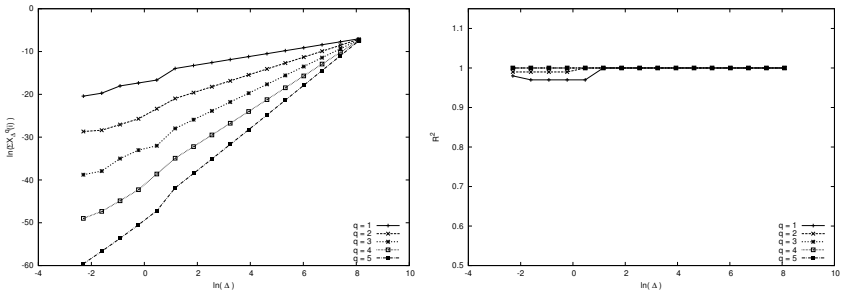
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.5: Detecção da escala limitante do traço TXS-1113503155.



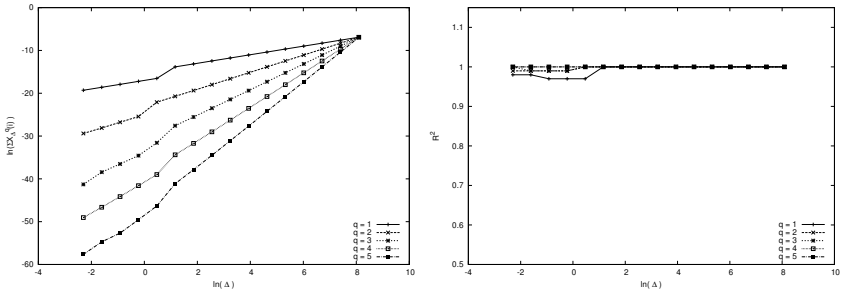
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.6: Detecção da escala limitante do traço IPLS-CLEV-20020814-090000-0.



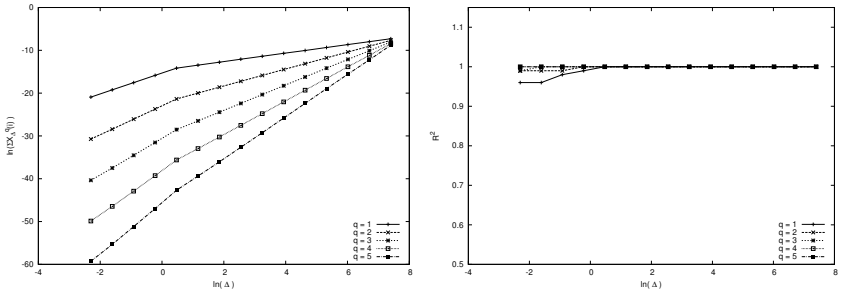
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.7: Detecção da escala limitante do traço IPLS-CLEV-20020814-090000-1.



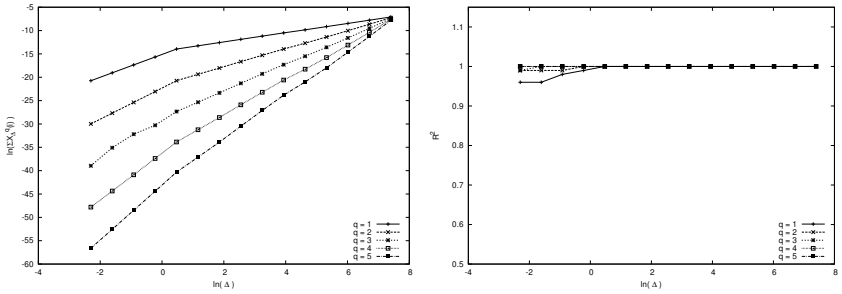
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.8: Detecção da escala limitante do traço IPLS-CLEV-20020814-091000-0.



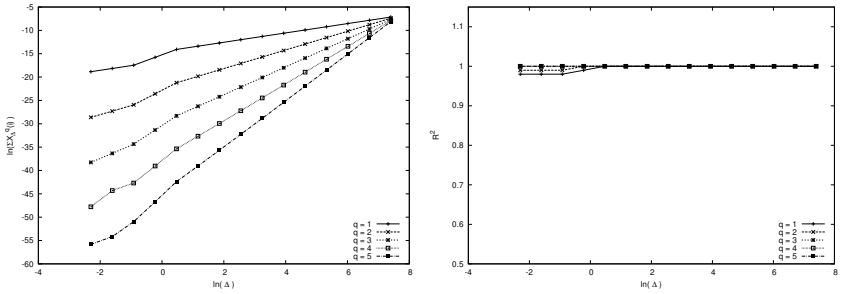
(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.9: Detecção da escala limitante do traço 20040601-193121-0.



(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.10: Detecção da escala limitante do traço 20040601-193121-1.



(a) Escala limitante do traço. (b) Coeficiente de determinação.

Figura 6.11: Detecção da escala limitante do traço 20040601-194000-1.

já que a função soma partição apresenta comportamento linear em todos os momentos estatísticos. A Figura 6.1(b) mostra o coeficiente de determinação da regressão linear em função das escalas em que o processo original foi agregado. É possível observar no ponto correspondente à escala limitante $R^2 \approx 0.98$, tendo uma queda a partir desse ponto.

O algoritmo calculaΔ^* pode ser utilizado para fornecer aos algoritmos de montagem de uma rede OBS informações sobre a escala limitante de um fluxo multifractal que alimenta a rede OBS. É importante salientar que, se no exemplo proposto, o valor escolhido para δ for maior que 0.98, o método fornecerá ao algoritmo de montagem uma escala de agregação superior à escala limitante, entretanto, ao montar as rajadas em escalas acima da escala limitante, o algoritmo continuará produzindo tráfego monofractal [26].

6.1.1 Complexidade computacional do método

A detecção da escala limitante implica antes de mais nada na agregação do tráfego em diversas escalas de tempo. Dessa forma, a complexidade da detecção da escala limitante depende também da complexidade de agregação do tráfego.

Seja $X(t)$ o processo de chegadas de pacotes no intervalo $[0, t]$. A agregação do tráfego em escalas de tempo corresponde a calcular a função soma partição apresentada pela Equação 3.2.1 para cada momento estatístico $q_{min} \leq q \leq q_{max}$, variando-se a escala de agregação Δ, no intervalo $[\Delta_{min}, \Delta_{max}]$. Seja $T(n)$ o tempo de execução do algoritmo de agregação, o Teorema 1 vale:

Teorema 1. *O tempo de execução do algoritmo de agregação é linear em n, onde n é o número de pontos do processo $X(t)$.*

Prova. Para valores fixos de Δ e q, a função soma partição é calculada. Para calcular a Equação 3.2.1 são necessárias N/Δ iterações, e para cada iteração da Equação 3.2.1 são realizadas Δ iterações no processo original $X(t)$, o que corresponde a um total de $\frac{N\Delta}{\Delta}$ iterações. Seja $|M|$ o número de momentos estatísticos e $|E|$ o número de escalas de

agregação. A agregação é realizada

$$(|M|) \cdot (|E|) \cdot n = O(n)$$

vezes, fazendo com que, assintoticamente, o tempo de execução do método de agregação seja linear. □

□

Após a agregação do processo $X(t)$, o algoritmo calculaΔ^* é executado. Seja $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ o conjunto de pontos resultantes do processo de agregação do tráfego, onde $x_i = \log(\Delta)$ e $y_i = \log(\sum_i X_\Delta(i)^q)$. O Teorema 2 vale:

Teorema 2. *O tempo de execução do algoritmo calculaΔ^* é linear em n, onde n é o número de pontos do conjunto S.*

Prova. O algoritmo inicia sua execução sobre um conjunto com um único ponto e progressivamente vai adicionando novos pontos ao conjunto até que o mesmo contenha n pontos ou o valor do coeficiente de determinação seja menor do que o limiar pré-determinado, δ. No pior caso, todos os n pontos são adicionados ao conjunto. Mantendo armazenados os valores das somas parciais de y_i e x_i para um conjunto S' com j pontos, é possível calcular a Equação 6.1 para um conjunto S'' com $j + 1$ pontos usando os valores das somas parciais. Assim, para cada novo ponto adicionado ao conjunto S apenas a atualização das somas parciais é realizada, resultando num total de n atualizações para um conjunto com n pontos. O algoritmo calculaΔ^* é executado para cada momento estatístico q. Assim, tem-se um tempo de execução total de

$$(|M|) \cdot n = O(n)$$

onde $|M|$ é o número de momentos estatísticos. □

□

6.2 Resumo conclusivo

Este capítulo apresentou um método para identificação automática da escala de tempo limitante. O método baseia-se na verificação do coeficiente de determinação da regressão linear realizada nos pontos do plano formado pelo processo incremento agregado em função da escala de agregação. O método foi aplicado em um conjunto de traços de tráfego real e sua aplicabilidade foi confirmada.

Capítulo 7

Algoritmos de montagem de rajadas com moldagem de tráfego

Neste capítulo, são apresentados algoritmos de montagem de rajadas capazes de induzir transformações nas propriedades estatísticas do tráfego da rede, transformando-o de multifractal para monofractal. Determina-se primeiramente o tamanho mínimo que cada rajada deve possuir para que haja transformação. O tamanho mínimo da rajada é determinado de forma que o processo que modela a chegada do tráfego ao nó OBS de ingresso seja agregado em uma escala de tempo em que o tráfego apresenta características monofractais. Resultados obtidos, através de simulação, mostram a capacidade dos algoritmos em realizar tais transformações.

7.1 Trabalhos relacionados

Vários algoritmos de montagem de rajadas têm sido propostos na literatura [9, 20, 28, 51, 63, 73, 78]. A principal tarefa de um algoritmo de montagem de rajadas é escolher uma quantidade de pacotes de uma ou mais filas da unidade de montagem e enviá-los ao núcleo da rede como uma rajada.

Segundo [79], a maioria dos algoritmos de montagem de rajadas usam ou tempo de

montagem [28, 78] ou o tamanho das rajadas [9], como critério principal na criação das rajadas. O uso de um limiar de tempo limita o atraso sofrido pelos pacotes nas filas de montagem, quando o tráfego na rede não é intenso. Por outro lado, o uso de um limitante para o tamanho das rajadas reduz o atraso do processo de montagem de rajadas, quando a intensidade do tráfego é alta. Os principais parâmetros envolvidos no processo de montagem de rajadas são um limitante de tempo t, um tamanho de rajada b e um tamanho mínimo de rajada b_{min}.

Os algoritmos de montagem de rajadas podem ser classificados em quatro categorias diferentes: algoritmos de montagem baseados em janelas de tempo, algoritmos de montagem baseados no tamanho das rajadas, algoritmos de montagem baseados em tempo e tamanho e algoritmos de montagem adaptativos [79].

A categoria dos algoritmos de montagem de rajadas baseados em janelas de tempo utiliza como principal critério para montagem de rajadas um limiar de tempo t [28, 78], isto é, a cada intervalo de tempo t, uma rajada é montada e enviada ao núcleo da rede. A segunda categoria contém algoritmos de montagem de rajadas que utilizam como principal critério o tamanho da rajada. Nesta categoria, os algoritmos montam as rajadas assim que o tamanho das mesmas atinge um valor pré-definido b [9].

Estas duas categorias possuem algoritmos simples e de fácil implementação. Entretanto, eles podem ser considerados problemáticos em algumas circunstâncias. Por exemplo, desde que os algoritmos pertencentes à segunda classe não possuem restrição sobre a duração do tempo de montagem, é provável que estando a rede sob baixa carga, o atraso fim-a-fim experimentado pelos pacotes não possua um limitante superior.

De forma inversa, os algoritmos de montagem pertencentes à primeira classe apresentam problemas quando o tráfego é intenso. Como não existe o limiar para o tamanho das rajadas, as mesmas são enviadas ao núcleo da rede com um tamanho excessivamente grande. A Figura 7.1, que ilustra o problema, P1 indica o ponto de montagem das rajadas quando os algoritmos da primeira categoria são usados. O ponto P2 ilustra analogamente o ponto de montagem das rajadas quando os algoritmos da segunda categoria são usados.

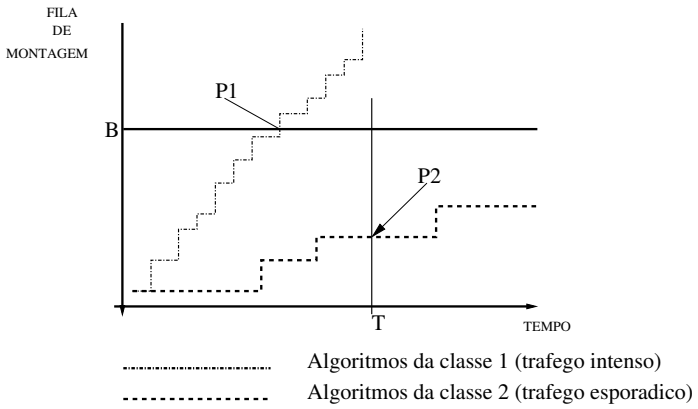


Figura 7.1: Processo de montagem das rajadas.

Através da análise ilustrada na Figura 7.1, é possível idealizar um algoritmo que se comporte como os algoritmos da primeira classe, quando a rede está sob tráfego esporádico, e que se comporte como os algoritmos da segunda classe quando a rede está sob tráfego intenso. Este é o principal objetivo dos algoritmos pertencentes à terceira classe de algoritmos de montagem de rajadas.

Como exemplos de algoritmos desta terceira classe, pode-se destacar o algoritmo proposto por Xiong et al [73] e o algoritmo proposto por Yu et al. [78]. O algoritmo de Xiong et al. utiliza como critérios de montagem, o tempo de montagem e o tamanho máximo das rajadas. Se um dos dois limiares for violado, a rajada é montada e enviada. O algoritmo de Yu et. al é bastante similar ao algoritmo de Xiong, com a diferença que o primeiro usa o tamanho mínimo das rajadas em adição aos parâmetros utilizados por Xiong. Assim, é adicionada a restrição de que a rajada, para ser montada e enviada, deve possuir um tamanho mínimo b_{min}. Se o temporizador expirar e a rajada não possuir o tamanho de b_{min}, são usados bytes de enchimento para completar o tamanho mínimo.

Outra sutil diferença entre os algoritmos citados é o resíduo deixado nas filas de montagem, após o envio da rajada. Enquanto no algoritmo de Xiong todos os dados são removidos da fila após o envio de uma rajada, no algoritmo de Yu os pacotes residuais (aqueles que não foram usados na transmissão) esperam o próximo período de montagem.

A quarta categoria de algoritmos de montagem de rajadas é composta pelos algoritmos

adaptativos. Tais algoritmos usam informações sobre o tráfego ou o estado corrente da rede para fazer a montagem das rajadas, adaptando-se mais facilmente às condições do tráfego e alcançando melhor desempenho na provisão de QoS do que os algoritmos apresentados até então.

Em [9], Cao et al. propuseram um algoritmo baseado em janela de tempo adaptativa. Neste algoritmo, o valor do tempo de montagem depende do tamanho médio das rajadas, da largura de banda dos enlaces e da quantidade de canais. A idéia é que para evitar que um determinado fluxo TCP venha a retransmitir pacotes desnecessariamente, o tempo de montagem de rajadas não deve ser maior que o tempo do *timeout* de retransmissão (*Retransmit Timeout Value - RTO*) menos o RTT (*Round Trip Time*) associado com o fluxo.

7.2 Montagem de rajadas com moldagem de tráfego

No Capítulo 5 mostrou-se que as propriedades estatísticas do tráfego podem ser alteradas dependendo dos parâmetros utilizados no processo de montagem de rajadas em uma rede OBS.

Verificou-se que o tráfego é:

$$monofractal \quad , \text{se } t \geq \Delta^* \; ; \tag{7.1a}$$

$$multifractal \quad , \text{se } t \leq \Delta^* \tag{7.1b}$$

onde t é o tempo de montagem das rajadas e $\Delta*$ é a escala limitante do tráfego multifractal.

Seja b o limiar do tamanho das rajadas usado pelo montador baseado em volume de tráfego e λ a taxa média de chegadas de pacotes (expressa em bytes por segundo), sabe-se

também que:

$$b = t \cdot \lambda \tag{7.2}$$

De 7.1a e 7.2, conclui-se que se:

$$\frac{b}{\lambda} > \Delta^* \tag{7.3}$$

o tráfego tem suas características multifractais alteradas para monofractais. Os resultados da Equação 7.3 são usados quando o montador de rajadas baseado em volume de tráfego é empregado.

Observa-se ainda, que além de mudanças nas características *scaling* do tráfego, o processo de montagem provoca suavização do tráfego cujo impacto é a redução de até 4% na probabilidade de bloqueio experimentada pela rede, quando o tráfego é transformado para monofractal.

Dessa forma, a montagem da rajada pode ser realizada, ajustando-se os parâmetros t ou b de acordo com os valores calculados de Δ^* e λ, garantindo-se, assim, a transformação do tráfego para monofractal. Para o cálculo de Δ^*, o método calculaΔ^* descrito no Capítulo 6 pode ser utilizado. A taxa de transmissão das fontes pode ser obtida através de acordos de nível de serviço (do inglês SLA - *Service Level Agreement*).

7.3 Montagem de rajadas usando técnicas *composite*

O problema de montagem de rajadas é mais desafiador, quando restrições temporais são introduzidas no processo de montagem, ou seja, se obrigatoriamente o tempo de montagem das rajadas tiver que ser menor do que a escala limitante. Dessa forma, o simples ajuste de t ou b para acima da escala limitante, pode causar um efeito negativo nas aplicações cujo tráfego está sendo transportado. Este cenário pode ser comum quando a rede transporta tráfego com requisitos de QoS.

Considere um conjunto $\mathcal{C}$ de classes de serviço. Cada uma das classes possui um requisito temporal D_i que é o atraso máximo fim-a-fim permitido para que os pacotes

pertencentes à classe i sejam adequadamente entregues ao destino.

Em uma rede OBS, o atraso fim-a-fim experimentado pelos pacotes pode ser expresso de forma que:

$$D_i \geq \alpha_i + T_i + d_i(b), \tag{7.4}$$

onde T_i é o tempo de ajuste usado na classe i, d_i é o fator que considera o tempo de propagação e transmissão de uma rajada de tamanho b da classe i e α_i é o tempo máximo gasto no processo de montagem. Seja t_i o tempo de montagem, b_i o limiar de bytes e λ_i a taxa média de chegadas de pacotes (expressa em bytes por segundo) da classe i. Tem-se a seguinte relação $\alpha_i = t_i$ ou $\alpha_i = \frac{b_i}{\lambda_i}$ dependendo do montador utilizado.

Novamente, se, para todas as classes, o tempo máximo de montagem é maior que a escala limitante, o montador pode ajustar t_i ou b_i para a produção das rajadas com tráfego monofractal. Entretanto, se existe alguma classe de serviço cujo tempo de montagem é inferior à escala limitante então a transformação do tráfego só é possível através do envio de rajadas mistas (contendo pacotes de diferentes classes).

Em redes OBS cujas rajadas possuem um tamanho mínimo s e um tamanho máximo S conhecidos, é possível, determinar através da Equação 7.3, para valores fixos de λ_i e $\Delta*$, o tamanho β ($s \leq \beta \leq S$) que as rajadas devem possuir de forma que haja transformação no tráfego.

Seja $\mathcal{C}$ o conjunto das classes de serviço, $\mathcal{C}'$ um subconjunto de $\mathcal{C}$ contendo classes que possuem pacotes nas filas de transmissão e seja p_i a prioridade da classe i de forma que $p_0 > p_1 > \cdots > p_n$. Seja, também, Q_i a fila de pacotes da classe i e $|R|$ o tamanho atual da rajada sendo montada. Assume-se que a escala limitante do fluxo multifractal (Δ^*) e a taxa média de chegadas de pacotes da classe i, λ_i, são conhecidos. A partir daí, β é determinado.

7.3.1 Algoritmos de montagem de rajadas com suavização de tráfego

Uma forma de criar rajadas mistas é através de uma política de escalonamento *round-robin* para preenchimento do tamanho da rajada. Assim, quando um limiar (t_i ou b_i) da classe i é alcançado, cria-se uma rajada contendo todos os pacotes da classe i. Caso o tamanho seja inferior a β os pacotes disponíveis da classe $i+1$ são adicionados, e assim sucessivamente até que a rajada tenha tamanho pelo menos igual a β. Detalha-se o processo de montagem no Algoritmo 2, denominado *Round Fit*.

Algoritmo 2 Round Fit (RF)

ENTRADA
Escala limitante Δ^*, t_i (ou b_i) e λ_i
SAÍDA
Rajada de dados com tamanho β
Round Fit
1: Calcule β, a partir da expressão 7.3
2: **while** $Q_i \neq 0$ **do**
3: $R \leftarrow R \cup head(Q_i)$
4: **if** $|R| < \beta$ **then**
5: $j \leftarrow (i+1) mod |\mathcal{C}'|$
6: **while** $|R| < \beta$ **do**
7: **while** Q_j **do**
8: $R \leftarrow R \cup head(Q_j)$
9: $j \leftarrow (i+1) mod |\mathcal{C}'|$

O algoritmo RF funciona da seguinte forma: na linha 1 o tamanho mínimo da rajada é determinado através da equação 7.3. Nas linhas 2 - 3 os pacotes da *i-ésima* classe (classe cujo temporizador expirou) são adicionados à rajada. A linha 4 testa se o tamanho da rajada após a inclusão dos pacotes da *i-ésima* classe é inferior a β. Caso afirmativo, os pacotes das outras classes são adicionados à rajada numa ordem *round-robin* até que o tamanho da rajada seja igual a β (linhas 6 - 9). É importante destacar na linha 5 que a próxima classe a ser escolhida é a classe seguinte à *i-ésima* classe.

O escalonamento *round-robin* pode não priorizar o atendimento de classes com mais alta prioridade. Dessa forma, ao colocar os pacotes da classe $i+1$ após os pacotes da classe

i, a classe de mais alta prioridade pode ser prejudicada. Para contornar essa situação, as classes de mais alta prioridade devem ser servidas prioritariamente. Para tal, introduz-se um algoritmo, doravante denominado *High Priority Fit (HPF)*, que opera da seguinte forma: quando um limiar (t_i ou b_i) da *i-ésima* classe é alcançado, a rajada é criada contendo todos os pacotes desta classe. Caso o tamanho da rajada seja inferior a β, a mesma é completada com os pacotes disponíveis da classe de mais alta prioridade, e assim sucessivamente até que a rajada tenha tamanho pelo menos igual a β. O procedimento $max\_unused(\mathcal{C}')$ devolve a classe de mais alta prioridade cujos pacotes ainda não foram adicionados à rajada. A operação do algoritmo HPF é detalhada no Algoritmo 3.

Algoritmo 3 High Priority Fit (HPF)

ENTRADA
Escala limitante Δ^*, t_i (ou b_i) e λ_i
SAÍDA
Rajada de dados com tamanho β
High Priority Fit (HPF)

```
1: Calcule β, a partir da expressão 7.3
2: while Q_i ≠ 0 do
3:    R ← R ∪ head(Q_i)
4: if |R| < β then
5:    j ← max_unused(C')
6:    while |R| < β do
7:       while Q_j do
8:          R ← R ∪ head(Q_j)
9:       j ← max_unused(C')
```

O algoritmo HPF funciona da seguinte forma: na linha 1, o valor de β é determinado de acordo com a equação 7.3. Nas linhas 2 - 3 os pacotes da *i-ésima* classe (classe cujo temporizador expirou) são adicionados à rajada. A linha 4 testa se o tamanho da rajada após a adição dos pacotes da *i-ésima* classe é inferior a β. Caso afirmativo, uma nova classe é escolhida para ter seus pacotes adicionados à rajada. Esta sentença é realizada na linha 5. É importante destacar que a classe a ser escolhida é determinada pela função $max\_unused(\mathcal{C}')$, que retorna sempre a classe de mais alta prioridade com pacotes nas suas filas de montagem. Este ponto marca a diferença entre o algoritmo HPF e o algoritmo

RF, já que neste último a nova classe escolhida é a classe $i + 1 \bmod |\mathcal{C}'|$. No restante do algoritmo, os pacotes das novas classes escolhidas pela função $max\_unused(\mathcal{C}')$ são colocados na rajada até que a mesma possua tamanho igual a β (linhas 6 - 9).

Se o tráfego oriundo das classes de mais alta prioridade for sempre intenso, as classes de mais baixa prioridade podem simplesmente não ser atendidas pelo HPF. No terceiro algoritmo, denominado *Proportional Fit(PF)*, esse problema é resolvido da seguinte forma: quando um limiar (t_i ou b_i) da classe i é alcançado, a rajada é criada contendo todos os pacotes da classe i. Caso o tamanho da rajada seja inferior a β a rajada é completada com pacotes de todas as classes que possuem pacotes em suas filas de montagem até que a rajada alcance tamanho pelo menos igual a β. O número de pacotes em cada classe é proporcional a ocupação da classe na fila. A operação do algoritmo PF é detalhada no Algoritmo 4.

Algoritmo 4 Proportional Fit (PF)

ENTRADA
Escala limitante Δ^*, t_i (ou b_i) e λ_i
SAÍDA
Rajada de dados com tamanho β
Proportional Fit

```
1: Calcule β, a partir da expressão 7.3
2: while Q_i ≠ 0 do
3:    R ← R ∪ head(Q_i)
4: if |R| < β then
5:    X ← (β − |R|)
6:    F ← ⌈X/(C' − 1)⌉
7:    j ← max_unused(C')
8:    l ← 0
9:    while |R| < β do
10:      while (Q_j) ∧ (l < F) do
11:         R ← R ∪ head(Q_j)
12:         l ← l + 1
13:      l ← 0
14:      j ← max_unused(C')
```

O algoritmo PF funciona da seguinte forma: na linha 1, o valor de β é determinado de acordo com a equação 7.3. Nas linhas 2 - 3 os pacotes da *i-ésima* classe (classe cujo

temporizador expirou) são adicionados à rajada. A linha 4 testa se o tamanho da rajada após a adição dos pacotes da *i-ésima* classe é inferior a β. Caso afirmativo, a rajada é completada proporcionalmente com pacotes pertencentes a todas as classes de serviço. Na linha 5 calcula-se a quantidade de pacotes que ainda faltam para completar o tamanho mínimo β. Após isso, na linha 6, são calculados, dentre as classes de serviço que possuem pacotes em suas filas de montagem, quantos pacotes serão usados de cada classe de serviço. A variável j é mantida para indicar a próxima classe de serviço de onde serão tirados os pacotes e a variável l é utilizada para indexar a quantidade de pacotes que já foram retirados de cada classe de serviço. Da linha 10 à linha 14, uma nova classe de serviço é escolhida e os pacotes desta classe são colocados na rajada.

Introduz-se, também, um algoritmo denominado *Backlog Fit (BF)* que opera da seguinte forma: quando um limiar (t_i ou b_i) da classe i é alcançado, a rajada de tamanho maior do que β é criada contendo, pacotes de todas as classes que possuem pacotes em suas filas de montagem em proporção à sua ocupação na fila. A diferença do algoritmo *Backlog Fit* em relação ao algoritmo *Proportional Fit*, é que no primeiro são colocados proporcionalmente pacotes de todas as classes que cujas filas de montagem não estão zeradas, enquanto que no segundo, somente o complemento da rajada (pacotes que faltam para que o tamanho β seja alcançado) é dividido proporcionalmente entre as outras classes de serviço, além da classe i. A operação do algoritmo BF é detalhada no Algoritmo 5.

O algoritmo BF funciona da seguinte forma: na linha 1, o valor de β é determinado de acordo com a equação 7.3. Na linha 2, calcula-se a quantidade de pacotes de cada classe de serviço que a rajada conterá. Assim como no algoritmo PF, as variáveis j e l são mantidas para indexar, respectivamente, a classe de serviço de onde os pacotes são retirados e a quantidade de pacotes de tal classe. Nas linhas 3 - 10, os pacotes de cada classe de serviço são retirados e colocados na rajada.

A Figura 7.2 ilustra o funcionamento dos algoritmos. A figura mostra um nó de borda com $|\mathcal{C}|$ filas de montagem. Na figura, a classe 1 é a classe de mais alta prioridade e a classe $\mathcal{C}$ a de mais baixa prioridade. A unidade de montagem (BU) é responsável por

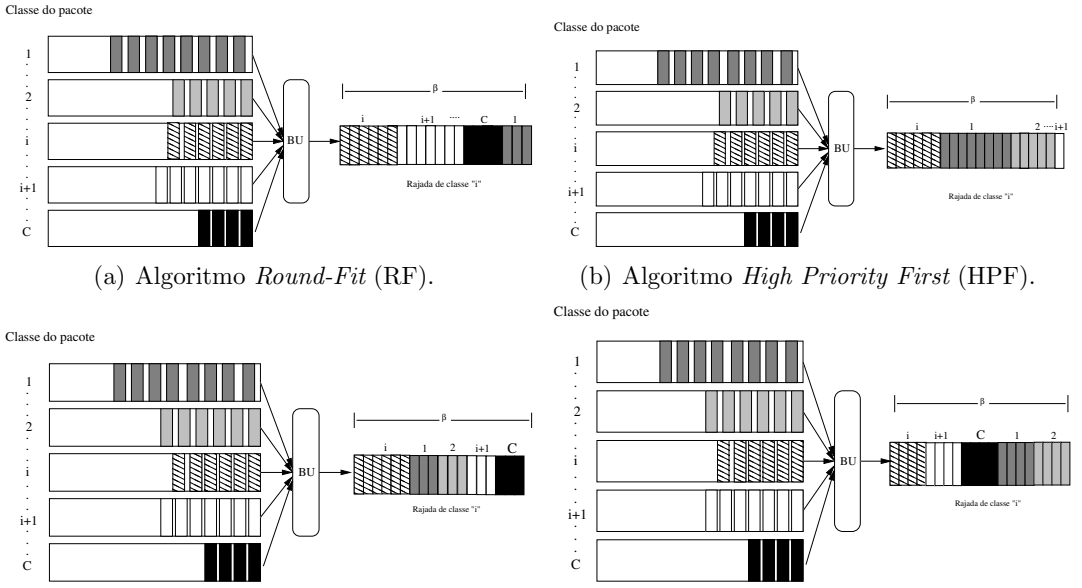
Algoritmo 5 Backlog Fit (BF)

ENTRADA
Escala limitante Δ^*, t_i (ou b_i) e λ_i
SAÍDA
Rajada de dados com tamanho β
Backlog Fit

```
1: Calcule β, a partir da expressão 7.3
2: F ← ⌈β/(C')⌉
3: j ← max_unused(C')
4: l ← 0
5: while |R| < β do
6:    while (Q_j) ∧ (l < F) do
7:       R ← R ∪ head(Q_j)
8:       l ← l + 1
9:    l ← 0
10:   j ← max_unused(C')
```

escolher pacotes de uma ou mais filas e realizar a montagem de uma rajada de tamanho (pelo menos) β. Na Figura 7.2(a), o algoritmo RF é ilustrado. É possível perceber que após o término do temporizador da *i-ésima* classe, todos os pacotes desta classe (pacotes hachurados) são adicionados à rajada. A rajada é, posteriormente, completada com pacotes da classe subsequente (pacotes brancos), e assim por diante, até que o tamanho igual a β tenha sido alcançado.

Na Figura 7.2(b), após colocar os pacotes da *i-ésima* classe, todos os pacotes da classe de mais alta prioridade (cinza escuros) são colocados na rajada. Os pacotes da classe com segunda maior prioridade são, então, colocados na rajada (cinza claros) e assim por diante. Na Figura 7.2(c), $\beta = 18$. Assim, após serem colocados na rajada os pacotes da *i-ésima* classe (6 pacotes), o restante da rajada (12 pacotes) é dividida entre as classes que possuem pacotes em suas filas de montagem (4 classes), o que dá 3 pacotes de cada classe de serviço. Por fim, na Figura 7.2(d) assume-se que $\beta = 20$. Assim, a quantidade de pacotes de cada classe de serviço (5 classes) (dentre aquelas classes que contém pacotes em suas filas de montagem) é igual a 4.



(a) Algoritmo *Round-Fit* (RF).

(b) Algoritmo *High Priority First* (HPF).

(c) Algoritmo *Proportional Fit* (PF).

(d) Algoritmo *Backlog Fit* (BF).

Figura 7.2: Algoritmos de montagem de rajadas.

7.4 Avaliação de desempenho dos algoritmos de montagem

Para avaliar o desempenho dos algoritmos de montagem apresentados, simulações usando a ferramenta NS-2 com módulo OBSns [1] foram realizadas. As simulações foram divididas em dois cenários distintos: cenário com tráfego baseado em traços de tráfego e cenário com tráfego sintético.

7.4.1 Simulações com tráfego baseado em traços de tráfego real

No primeiro cenário, o objetivo foi verificar a ocorrência de transformações nas características *scaling* do tráfego. Para tal, um nó de borda servindo 5 classes de serviço e equipado com 32MB de buffer de entrada foi alimentado com tráfego multifractal obtido a partir de traços de tráfego real. O tráfego de cada classe de serviço foi gerado a partir

de uma cópia do traço e o tamanho mínimo da rajada (β) necessário para a mudança de escala de forma que $\beta/\lambda > \Delta^*$ (Tabela 7.1).

Para garantir que o tráfego de entrada tivesse intensidade suficiente para que rajadas de tamanho mínimo igual a β fossem geradas (o que garantiria a mudança nas características scaling do tráfego) somente os traços com maior taxa de transmissão dentre aqueles apresentados na Tabela 4.1 foram utilizados. Na Tabela 7.1, todos os traços usados na simulação são listados.

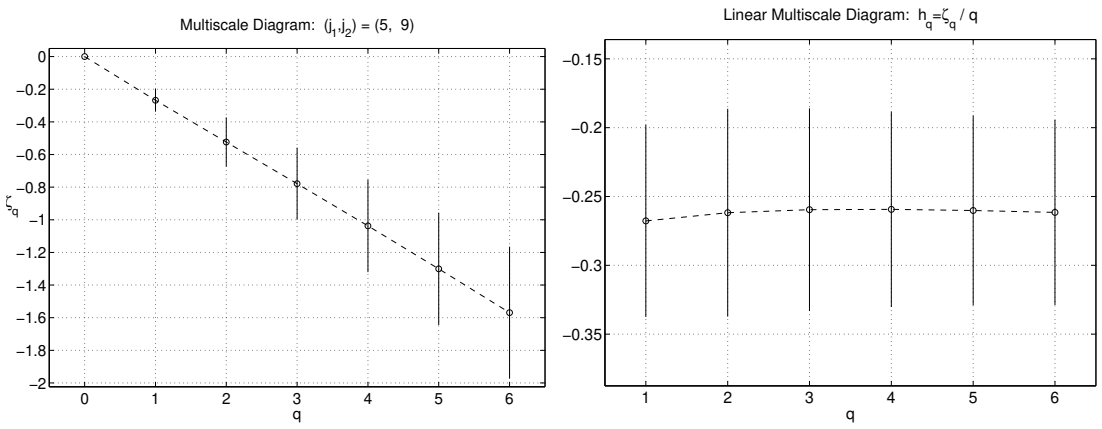
O tempo de montagem de rajada de cada classe i (t_i), foi ajustado para 1ms (abaixo da escala limitante de todos os traços) e para garantir que o critério de montagem de rajadas fosse sempre determinado pelo estouro do temporizador t_i, o que justificaria o uso dos algoritmos propostos, o contador de bytes de cada classe (b_i) foi ajustado para 32MB

Tabela 7.1: Características dos traços de tráfego real usados na avaliação de desempenho das políticas propostas

Traço	λ (Mbps)	Δ^*	β	Holder	Var	I.C.
IPLS-CLEV-090000-0	412.131	3ms	205KB	0.83	0.0400	(0. 80, 0.86)
IPLS-CLEV-090000-1	457.852	2,8ms	230KB	0.792	0.007	(0.791, 0.793)
IPLS-CLEV-091000-0	363.754	3,2ms	180KB	0.675	0.001	(0.672, 0.678)
20040601-193121-0	770.00	1.6ms	200KB	0.683	0.0014	(0.674, 0.692)
20040601-193121-1	1,648	1.3ms	410KB	0.689	0.0021	(0.676, 0.702)
20040601-194000-1	828.616	1.3ms	210KB	0.622	0.0027	(0.621, 0.623)

A Figura 7.3 mostra o diagrama multiescala e o diagrama multiescala linear correspondente ao algoritmo BF quando alimentado com o traço IPLS-CLEV-090000-0. Pode ser facilmente percebido que o tráfego injetado no núcleo da rede é monofractal, a despeito da política usada. Os diagramas relativos às outras políticas foram omitidos devido a limitações de espaço.

A Tabela 7.2 mostra o parâmetro de Hurst resultante, o atraso experimentado pelos pacotes IP na montagem das rajadas e a vazão obtida pelos algoritmos de montagem para o traço de tráfego IPLS-CLEV-090000-0. É possível observar pelo valor do parâmetro de



(a) Diagrama Multiescala do tráfego gerado pelo algoritmo BF.

(b) Diagrama Multiescala Linear do tráfego gerado pelo algoritmo BF.

Figura 7.3: Diagramas Multiescala Linear e Multiescala do tráfego de saída gerado pelo algoritmo BF.

Tabela 7.2: Parametro de Hurst, Retardo de montagem e vazão dos traços IPLS-CLEV-090000-0

Algoritmo	H	Retardo de montagem (sec)	Vazão (Mbps)
RF	0.745	0.001s	410.41
HPF	0.728	0.001s	410.41
PF	0.751	0.0098s	412.02
BF	0.781	0.00993s	411.99

Hurst que houve uma diminuição no nível de explosividade do tráfego quando comparado ao valor médio do expoente de Holder apresentado na Tabela 7.1. Além disso, a tabela mostra o tempo médio em que os pacotes IP permanecem nas filas de montagem do nó de borda. Como esperado, percebe-se que o tempo médio é limitado superiormente em $1ms$, quando encerra o tempo para a montagem das rajadas (t_i). É possível observar que as políticas PF e BF produzem um menor atraso. Isso acontece por que todas as rajadas enviadas possuem pacotes de todas as classes de serviço, o que faz com que o tempo de espera dos pacotes seja menor. Por outro lado, nas políticas RF e HPF, quando o limite do temporizador é alcançado, primeiramente, são colocados pacotes da classe cujo temporizador "estourou", só então, são colocados pacotes de outras classes. Isso faz com que as rajadas possuam majoritariamente pacotes IP da classe sendo montada, o que implica que grande parte dos pacotes da fila de montagem tiveram que esperar todo o tempo de montagem para serem transmitidos.

É possível perceber que os algoritmos PF e BF apresentam maior vazão do que os algoritmos RF e HPF. Isso acontece pois a cada envio de rajada, os algoritmos PF e BF conseguem o envio de dados de todas as classes de serviço sem que os temporizadores associados à montagem de rajadas dessas classes sejam zerados. Este fato faz com que um número maior de dados seja enviado em um mesmo intervalo de tempo se os algoritmos RF e HPF forem usados.

7.4.2 Simulações com tráfego sintético

No segundo cenário, as simulações foram realizadas em um ambiente no qual a taxa de chegadas dos pacotes IP pudesse ser controlada e o comportamento dos algoritmos analisado. Para tal, as simulações foram realizadas com tráfego monofractal gerado a partir da multiplexação de dezenas de fontes ON-OFF Pareto implementadas no módulo OBS-ns [1]. Os períodos ON e OFF tiveram a mesma duração média de 100ms e quando em estado ON as fontes transmitiam com uma taxa de 50Mbps.

No primeiro conjunto de experimentos, o atraso sofrido na montagem dos pacotes IP

e a vazão média são medidos em função do número de classes de serviço em uso na rede. Nestes experimentos, a taxa média de chegadas de cada classe de serviço foi de 25Mbps. As rajadas de cada classe de serviço possuiram tamanho mínimo para montagem mista (β) de 30KB e tamanho mínimo para montagem individual (b_i) de 100KB.

A Figura 7.4 mostra a vazão média agregada, em função do aumento do número de classes de serviço.

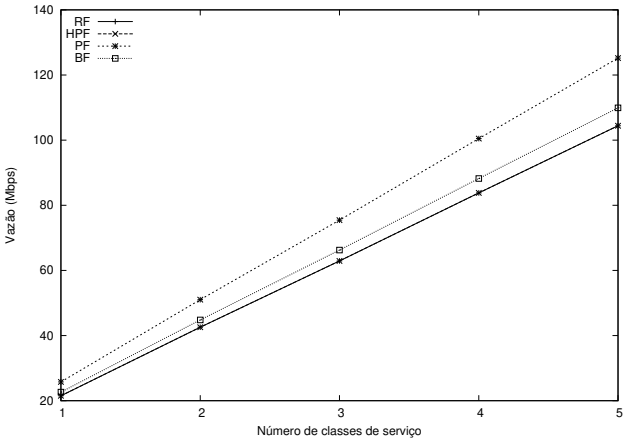


Figura 7.4: Vazão em função do número de classes de serviço.

É possível observar que há um incremento na vazão de todos os algoritmos. Isso acontece por que à medida em que mais classes de serviço são usadas, o algoritmo tem mais opções de classes para preencher a rajada. Assim, se uma classe não possui dados para enviar, pacotes de outra classe podem ser usados.

Dentre os algoritmos avaliados, os algoritmos RF e HPF possuem praticamente o mesmo comportamento. Como as fontes associadas às classes de serviço possuem a mesma taxa média, o fato de utilizar os dados da classe mais prioritária ou da próxima classe para completar a rajada é indiferente, quando a vazão agregada é considerada. O algoritmo que obtém o segundo melhor ganho à medida que o número de classes de serviço é aumentado é o algoritmo BF. Por fim, o algoritmo com melhor desempenho foi o algoritmo PF. Este desempenho é devido a uma diferença sutil no funcionamento dos algoritmos PF e BF em relação ao RF e HPF.

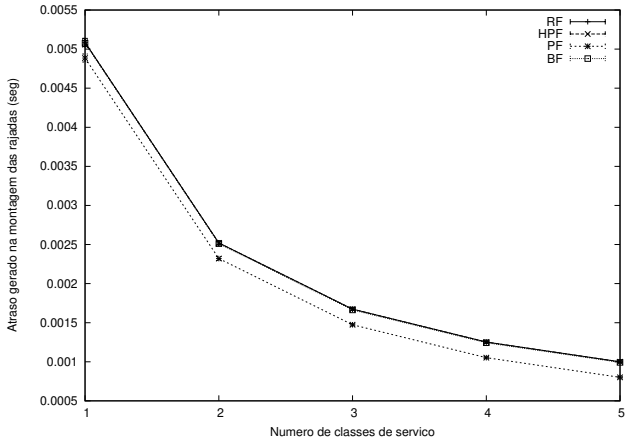


Figura 7.5: Atraso de montagem em função do número de classes de serviço.

Nos algoritmos RF e HPF todos os pacotes da *i-ésima* classe são colocados na rajada (o temporizador é zerado) bem como todos os pacotes da classe subsequente. Assim, essa nova classe também tem seu temporizador zerado. Dessa forma, a rajada de *i-ésima* classe e da classe subsequente só enviarão dados de "carona"nas rajadas de outras classes ou quando seu próprio temporizador expirar novamente. No caso dos algoritmos PF, quando estoura o temporizador da *i-ésima* classe todos os pacotes desta classe são colocados na rajada (temporizador é zerado) mas só uma fração dos pacotes das outras classes é colocada, o temporizador dessas classes não é zerado, o que faz com que em pouco tempo uma nova rajada de tais classes seja enviada. Como a rajada de *j-ésima* classe enviou dados na rajada de *i-ésima* classe é possível que ela possua menos dados para enviar na próxima rajada assim, uma fração maior de pacotes de outras classes é tomada emprestada. Isso faz com que as classes de serviço estejam enviando dados continuamente. Um processo parecido acontece com o algoritmo BF. Contudo, como o algoritmo BF divide proporcionalmente a composição das rajadas entre todas as classes de serviço, a ocupação das filas de montagem de todas as classes é menor. Dessa forma, as classes tem sempre menos pacotes para enviar, o que pode reduzir a vazão.

Um raciocínio similar pode ser construído para explicar o menor atraso de montagem dos algoritmos BF e PF apresentado na Figura 7.5. Como os temporizadores das classes

de "carona"não são zerados, o período de montagem é menor quando comparado aos algoritmos HPF e RF.

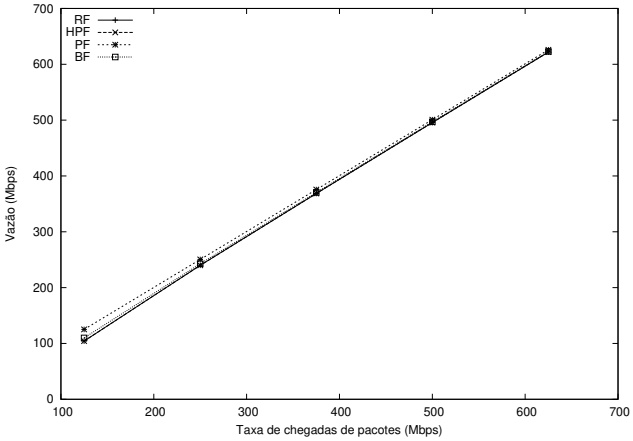


Figura 7.6: Vazão como função da taxa média de chegada de pacotes IP.

Num segundo conjunto de experimentos, o atraso de montagem e a vazão foram medidos em função do aumento da taxa de chegadas de pacotes IP. Neste cenário, o nó OBS de ingresso serve 5 classes de serviço e assim como nos experimentos anteriores, as rajadas de cada classe de serviço possuem tamanho mínimo para montagem mista (β) de 30KB e tamanho mínimo para montagem individual (b_i) de 100KB.

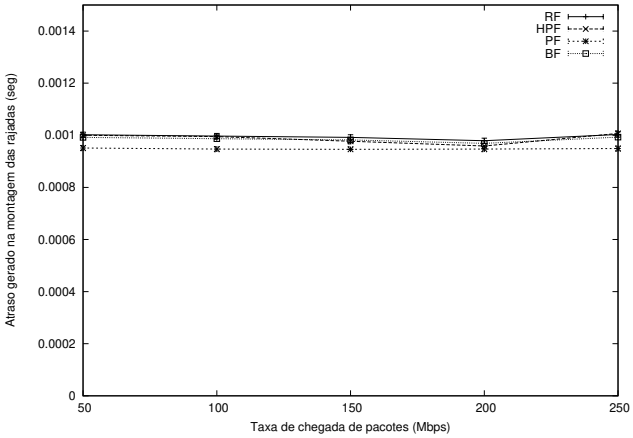


Figura 7.7: Atraso de montagem como função da taxa média de chegada de pacotes IP.

A Figura 7.6 mostra a vazão média agregada em função do aumento da taxa de chegada

de pacotes ao nó OBS de ingresso. Percebe-se um aumento linear da vazão em todos os algoritmos apresentados. À medida em que se aumenta a taxa de chegadas de pacotes, as rajadas passam a ser montadas seguindo o critério de tamanho mínimo para montagem com pacotes de uma única classe (b_i). Com isso, as rajadas tornam-se maiores à medida em que a taxa aumenta fazendo com que um número maior de bytes seja transmitido. Outro fator importante, é que, com o aumento da taxa, o tempo entre chegada de pacotes diminui. Isso implica no fato de que quando uma rajada da *i-ésima* classe é enviada um tempo menor é necessário para que se inicie o processo de montagem da próxima rajada desta classe, o que faz com que um número maior de rajadas seja enviada ao final da simulação.

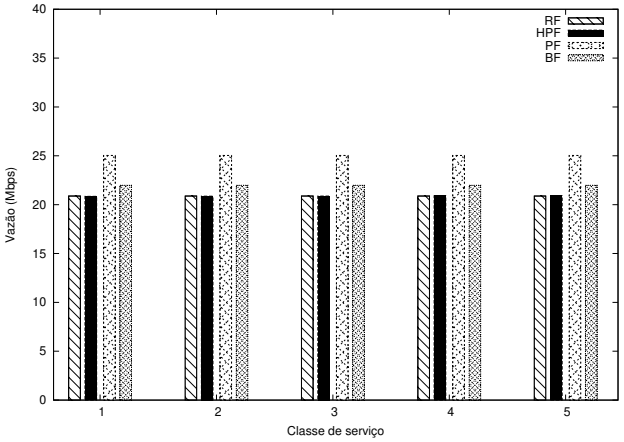


Figura 7.8: Vazão distribuída entre as classes de serviço.

Nota-se também, que à medida em que a taxa de chegadas de pacotes IP aumenta, a diferença entre os algoritmos diminui. Pois as rajadas montadas passam a conter apenas pacotes de uma única classe. Assim, todos os algoritmos comportam-se da mesma forma.

A Figura 7.7 apresenta o atraso sofrido na montagem em função do aumento da taxa de chegadas de pacotes. É possível notar que o atraso introduzido pelos algoritmos não sofre impacto com o aumento da taxa de chegada de pacotes; os algoritmos BF e PF produzem de novo os menores atrasos de montagem.

As Figuras 7.8 e 7.9 apresentam a vazão e o atraso de montagem experimentados

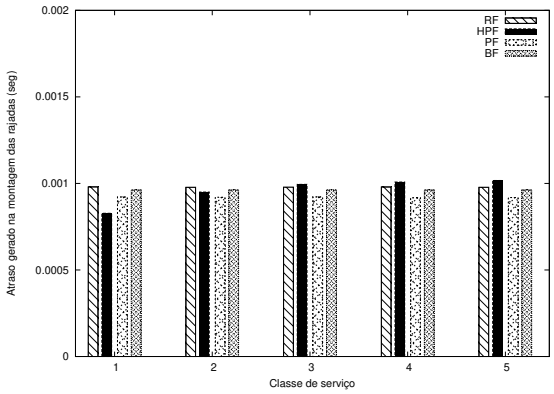


Figura 7.9: Atraso de montagem distribuído entre as classes de serviço.

individualmente por cada classe de serviço. Com exceção do algoritmo HPF, todos os algoritmos dividem tanto a vazão quanto o atraso uniformemente entre as classes. O atraso menor e a vazão maior produzidos pelo algoritmo HPF é devido ao fato deste utilizar sempre os pacotes das classes mais prioritárias na complementação das rajadas.

7.5 Resumo conclusivo

Neste capítulo, quatro políticas de montagem de rajadas foram apresentadas. As políticas são usadas para criar rajadas mistas de forma que possuam um tamanho mínimo (neste contexto determinado para provocar transformações nas propriedades estatísticas do tráfego da rede).

Simulações foram realizadas para avaliar o desempenho dos algoritmos. Dentre os algoritmos apresentados, o algoritmo *Proportional fit* apresentou melhor desempenho no que se refere a atraso de montagem de rajadas e vazão. Além disso, todos os algoritmos apresentados foram capazes de realizar as transformações nas propriedades estatísticas do tráfego da rede.

Capítulo 8

O algoritmo de escalonamento *Least Reusable Channel*

Este capítulo introduz 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 a reutilização de canais, e utiliza 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 determinar o 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.

8.1 Trabalhos relacionados

Vários algoritmos de escalonamento de canal foram propostos na literatura [50, 65, 73, 79]. Xiang Yu et al. [79] classificam os mesmos em dois grupos: algoritmos reativos [14, 65, 73] e algoritmos proativos [40, 71]. O primeiro grupo é caracterizado por não considerar o impacto causado nas rajadas futuras gerado pela seleção de um canal para uma dada rajada. O segundo grupo é formado por algoritmos que evitam a contenção de rajadas futuras.

8.1.1 Algoritmos de escalonamento proativos

Escalonamento proativo foi proposto por Wang, Morikawa e Aoyama em [71], na definição do algoritmo *Priority-based Wavelength Assignment (PWA)*. Neste algoritmo, cada nó de ingresso mantém uma base de dados contendo informações sobre a prioridade de cada comprimento de onda para cada nó de destino. Quando chega uma rajada , o nó de ingresso realiza uma busca na sua base de dados. Caso o comprimento de onda com a maior prioridade esteja disponível, a rajada é enviada neste comprimento de onda. Caso contrário, o algoritmo verifica o comprimento de onda com a segunda maior prioridade.

As prioridades dos comprimentos de onda são continuamente atualizadas, de acordo com a porcentagem de rajadas da origem ao destino que foram descartadas naquele comprimento de onda. Estudos realizados através de simulação, mostram que o PWA consegue uma redução considerável na taxa de perda de rajadas.

Inspirados no PWA, Li e Qiao [40] propuseram diversos algoritmos de escalonamento. Os algoritmos propostos em [40] são coletivamente, chamados de BORA (*Burst Overlapping Reduction Algorithm*). Estes baseiam-se no fato de que se o número total de rajadas que chegam simultaneamente a uma porta de saída de um comutador for maior do que a quantidade de comprimentos de onda disponíveis, as perdas serão inevitáveis (assumindo a ausência de FDLs). Dessa forma, a idéia do BORA é diminuir o número total de rajadas simultâneas chegando em cada porta.

O algoritmo BORA é baseado numa métrica denominada grau de sobreposição (*do inglês overlapping degree*) que é definida como o número de rajadas que simultaneamente chegam a um enlace. Existe uma relação entre o grau de sobreposição e a taxa de perdas de rajadas tal que, quanto maior o grau de sobreposição maiores as chances de rajadas serem descartadas.

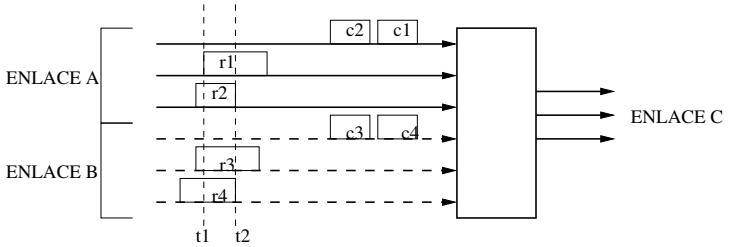


Figura 8.1: Contenção de rajadas simultâneas.

A Figura 8.1 mostra um exemplo no qual o LAUC-VF (e outros algoritmos de escalonamento) falha no escalonamento de todas as rajadas. No exemplo, um nó OBS intermediário tem dois enlaces de entrada e um enlace de saída. Cada enlace tem dois canais de dados e um canal de controle. Se quatro rajadas (r1,r2,r3 e r4) chegam ao nó OBS e todas devem usar o enlace de saída (ENLACE C), desde que elas se sobrepõem entre si com grau 4 no intervalo de tempo $[t1, t2]$, duas das quatro rajadas serão descartadas.

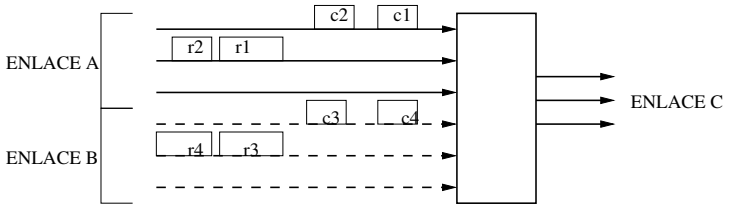


Figura 8.2: Evitando a contenção de rajadas simultâneas.

Se o grau de sobreposição puder ser reduzido através do atraso das rajadas r2 e r4 em um nó de ingresso tal que r2 chegue ao enlace "A" após a chegada da rajada r1, e r4 chegue ao enlace "A" após a rajada r3 (Figura 8.2), o nó OBS será capaz de escalonar as rajadas sem nenhuma perda.

Para uma rede OBS, é razoável pensar que em cada enlace existem mais do que um caminho conectando os múltiplos pares de origem e destino da rede. Assim, para reduzir o grau de sobreposição em um determinado enlace, é necessário que cada caminho reduza seu próprio grau de sobreposição. Além disso, não se pode reduzir a sobreposição nos comutadores de núcleo já que os mesmos não têm como retardar as rajadas para diminuir a sobreposição. Dessa forma, a redução deve ser realizada nos comutadores de ingresso da rede.

No exemplo da Figura 8.1, se as rajadas r1 e r2 são geradas pelo mesmo comutador de ingresso, uma forma direta de reduzir o grau de sobreposição é retardar a transmissão da rajada r2 até a saída da rajada r1. Do mesmo modo, quando uma nova rajada estiver montada e pronta para transmissão, ela deverá ser enviada no mesmo canal que as rajadas r1 e r2. O problema com essa abordagem é que o retardo introduzido pode ser muito alto. O objetivo do BORA então, é limitar esse atraso.

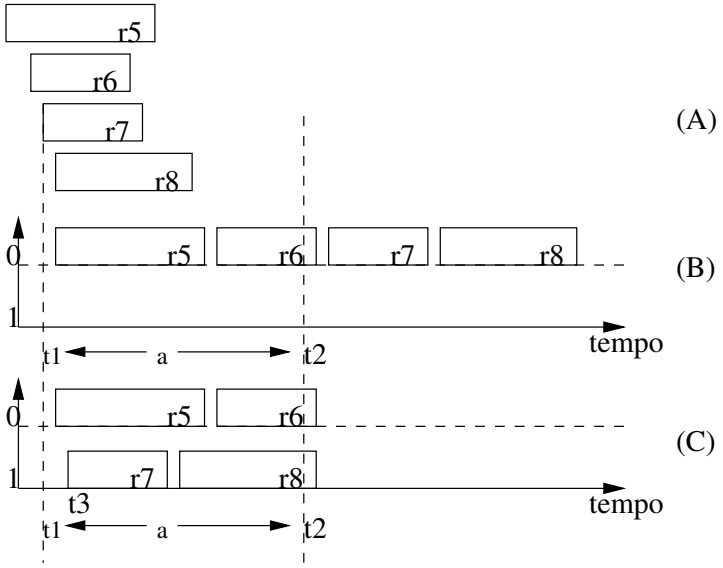


Figura 8.3: Serialização das rajadas no nó OBS de ingresso.

A Figura 8.3 ilustra um exemplo de como um nó OBS de ingresso escalona um conjunto de rajadas geradas localmente. A Figura 8.3(A) mostra quando quatro rajadas r5, r6, r7 e r8 chegam sequencialmente ao escalonador de rajadas. A Figura 8.3(B) ilustra o

resultado do escalonamento quando o escalonador ignora o limite de atraso das rajadas. Pode-se notar que o atraso sofrido pelas rajadas r7 e r8 pode exceder um limite superior para o atraso, representado por "a" na figura.

A Figura 8.3(C) mostra quando o escalonador leva o limite máximo de atraso em consideração. Depois que as rajadas r5 e r6 são escalonadas no comprimento de onda "0", o escalonador tenta usar o mesmo comprimento de onda, entretanto, como o limite para o atraso é excedido, o escalonador utiliza o comprimento de onda "1" para transmitir a rajada r7. Da mesma forma, quando o nó deseja transmitir a rajada r8, ele primeiramente verifica o comprimento de onda 0. Por exceder o atraso, o comprimento de onda "1" será utilizado pela rajada r8.

8.1.2 Algoritmos de escalonamento reativos

Turner [65] propôs um algoritmo de escalonamento de canal denominado *Horizon* (também conhecido como *Latest Available Unused Channel* - LAUC, [72]). Neste algoritmo, o escalonador armazena, para cada canal, apenas as informações sobre o tempo depois do qual não existem reservas (horizonte). O escalonador aloca a rajada sendo processada considerando o horizonte mais próximo do início da reserva. Na Figura 8.4, por exemplo, o canal escolhido seria o canal 0. O LAUC minimiza 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 $O(\log |W|)$, para um enlace com W canais, entretanto, ele produz baixa utilização da rede e altas taxas de perda dado que todos os intervalos *void* são desconsiderados.

Em [72], foram propostas variações do LAUC. Tais variações possuem as mesmas informações 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. [73]

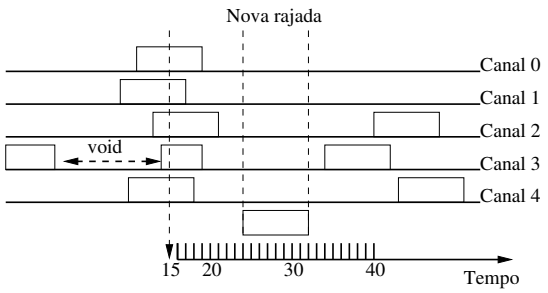


Figura 8.4: Escalonamento dos canais.

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 início 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 $O(|W|\log(|S|))$, onde S representa o conjunto de reservas em todos os canais [14]. Uma adaptação simples do LAUC-VF é o algoritmo *RANDOM*. Neste algoritmo, os *voids* viáveis são selecionados. Depois, um deles é aleatoriamente escolhido. No cenário descrito na Figura 8.4, o canal selecionado pelo LAUC-VF é o canal 2. O algoritmo *RANDOM* escolhe aleatoriamente um dentre os 5 canais.

Xu et al. [74, 75] 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 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 $O(\log(|S|))$.

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 acomodar a rajada no menor *void* viável. Experimentos de simulação mostraram que o desempenho do algoritmo MIN-SV é superior quando comparado com o MIN-EV e o *Best Fit*. Os canais da Figura 8.4 selecionados pelos algoritmos MIN-EV, MIN-SV e *Best-Fit* são os canais 3, 2 e 3, respectivamente.

Detti et al. [19] apresentaram um algoritmo de escalonamento de rajadas com resolução de contenção denominado OCBS (*Optical Composite Burst Switching*), que procura encontrar um intervalo *void* que acomode a nova rajada ou um que minimize a porção da rajada que será descartada em caso de contenção. Em [70], Vokkarane et al. propuseram um algoritmo similar, porém mais elaborado. Neste novo esquema, uma das cinco políticas pode ser usada:

1. *Drop Policy*: A rajada original (aquela para a qual já existe reserva) é preservada enquanto a nova rajada é descartada.

2. *Segment and Drop Policy*: A nova rajada é mantida e a rajada original é segmentada com a cauda (ou final) da rajada original sendo descartada.

3. *Deflect Drop Policy*: A nova rajada é redirecionada a uma porta de saída alternativa, caso exista uma. Caso não exista porta de saída alternativa, a nova rajada é descartada.

4. *Segment First and Deflect Policy*: A rajada original é segmentada e a sua cauda é redirecionada a uma porta de saída alternativa, se existir uma. Caso não exista uma porta de saída alternativa, a cauda da rajada original é descartada

5. *Deflect First Segment and Drop Policy*: A nova rajada é redirecionada a uma porta de saída alternativa, caso exista. Caso não exista porta de saída alternativa, a rajada original é segmentada e a cauda da rajada original é descartada enquanto a nova rajada é encaminhada pela porta de saída original

Em [11] propõe-se um algoritmo de escalonamento na ordem de chegada das rajadas ao invés da ordem dos pacotes de controle. Cada rajada é escalonada em um canal ou em

uma FDL.

8.2 Uso de informações topológicas no escalonamento de canais em redes OBS

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 de uma alocação dos recursos bloquear várias requisições de alocação posteriores.

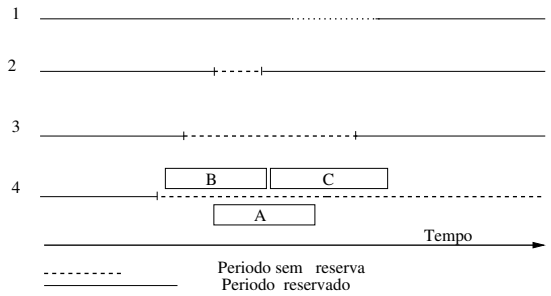


Figura 8.5: Problema no escalonamento de canais.

A Figura 8.5 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 caso o canal 4 seja usado para acomodar a reserva A, as reservas B e C serão descartadas. Entretanto, se o canal 3 for usado para acomodar a

reserva A, ambas as reservas (B e C) podem ser acomodadas com sucesso.

Para considerar a chegada de requisições futuras, o algoritmo precisa, ao fazer a alocação de um canal a uma requisição, observar dois fatores importantes. O primeiro fator é a capacidade que os voids anterior e posterior gerados pela alocação da requisição em processamento têm de acomodar novas requisições. O segundo fator é a probabilidade de que as novas requisições venham a usar os voids criados com a alocação da requisição em processamento.

8.2.1 Vida útil de um intervalo *void*

Para avaliar o capacidade que os *voids* gerados pela alocação da requisição em processamento têm de serem usados para alocar novas requisições, foi criada uma métrica denominada a vida útil de um intervalo. Tal métrica deve ser levada em consideração pelo algoritmo de escalonamento já que a capacidade que os *voids* têm de acomodar novas requsições é progressivamente diminuída à medida que o tempo passa, como ilustrado na Figura 8.6.

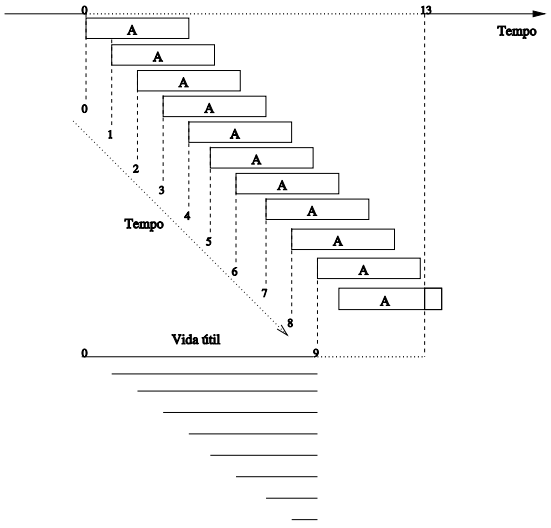
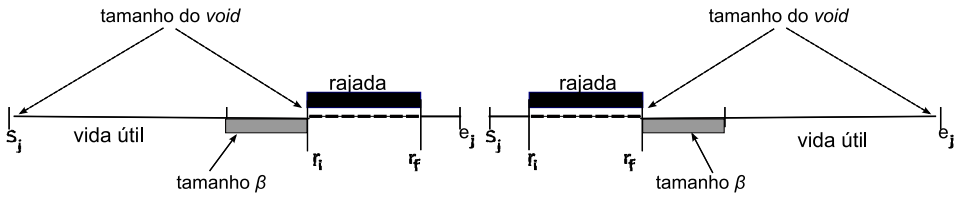


Figura 8.6: Tempo de vida útil de um intervalo.

Como dito, a vida útil de um intervalo *void* representa a sua capacidade de acomodar

reservas futuras. Ela é definida como o tamanho do novo intervalo gerado, diminuído do tamanho médio ($\overline{\beta}$), em unidades de tempo, das rajadas que transitam pela rede. Para realizar o cálculo, assume-se que o tempo é discretizado e a unidade de medida usada na discretização possui tamanho 1.



(a) Tempo de vida útil (*void* anterior). (b) Tempo de vida útil (*void* posterior).

Figura 8.7: Cálculo do tempo de vida útil.

Assim, sejam s_j , e_j, r_i e r_f o início e o final do *void* e da r-ésima requisição, respectivamente e $\overline{\beta}$ o tamanho médio das rajadas (em unidades de tempo). O *void* anterior, a_j, gerado pela alocação da *r-ésima* requisição (vide Figura 2.11), só pode acomodar uma rajada de tamanho $\overline{\beta}$ até o instante $r_i - \overline{\beta} + 1$. Assim, o tempo de vida útil do intervalo a_j pode ser calculado como:

$$\upsilon(a_j) = r_i - \overline{\beta} + 1 - s_j = r_i - (s_j + \overline{\beta}) + 1 \tag{8.1}$$

de modo análogo, o tempo de vida útil do intervalo p_j (*void* posterior gerado pela alocação da *r-ésima* requisição) pode ser calculado como:

$$\upsilon(p_j) = e_j - (r_f + \overline{\beta}) + 1 \tag{8.2}$$

A Figura 8.7 ilustra a idéia do tempo de vida útil. Na Figura 8.7(a) é possível observar o tempo de vida útil do *void* anterior gerado pela alocação da rajada em um *void* no intervalo $[s_j, e_j]$. De modo análogo, a Figura 8.7(b) mostra o tempo de vida útil do *void* posterior.

Com o cálculo do tempo de vida útil do intervalo, é então possível ter uma estimativa

sobre a capacidade que cada um dos novos intervalos têm de acomodar novas requisições.

8.2.2 Inversão na ordem de chegada das requisições

Para identificar o potencial de re-utilização de cada novo *void* gerado pela alocação de uma requisição (vide Figura 2.11), é preciso avaliar as chances de que as próximas requisições tenham tempo de início anterior ao tempo de início da requisição sendo processada. Em outras palavras, é preciso saber as chances de haver uma inversão na ordem de chegada de pacotes de controle e rajadas, de forma que um pacote de controle futuro contenha um pedido de reserva, em um período de tempo anterior ao período reservado para o pacote de controle em processamento.

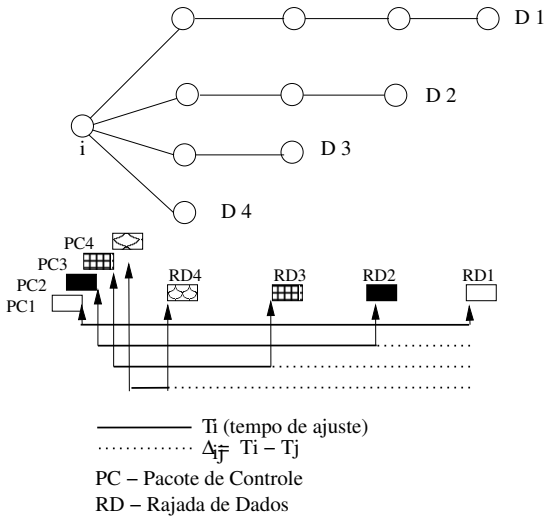


Figura 8.8: Inversão da ordem de chegada das requisições.

A Figura 8.8 ilustra uma situação potencial 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.

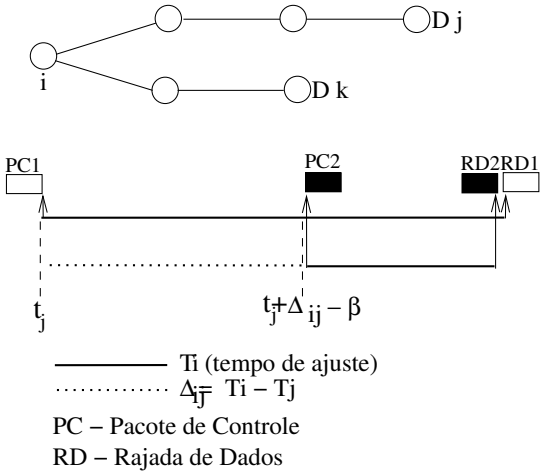


Figura 8.9: Inversão: chegada do segundo pacote de controle dentro do intervalo $[t_j; t_j + \Delta_{jk} - \beta]$.

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 j). A segunda condição é que o pacote de controle pc_2 chegue ao nó i no intervalo de tempo

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

que é o intervalo de tempo compreendido entre a chegada do primeiro pacote de controle somado à diferença entre os tempos de ajuste T_i^j e T_i^k, onde β representa o tamanho da

rajada destinada ao nó k, como ilustra a Figura 8.9.

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, como ilustra a Figura 8.10.

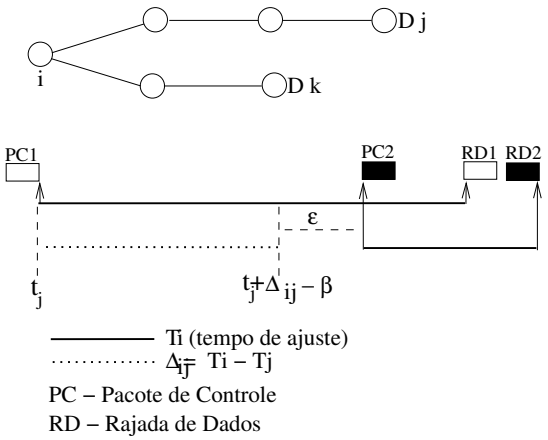


Figura 8.10: Não inversão: chegada do segundo pacote de controle após o intervalo $[t_j; t_j + \Delta_{jk} - \beta]$.

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 na origem dos dados, 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.

8.3 Probabilidade de inversão

Nas próximas subseções, a probabilidade de inversão será calculada considerando as políticas de montagem baseada em janelas de tempo e baseada em volume de tráfego que são as políticas de montagem mais comuns em redes OBS.

8.3.1 Cálculo da probabilidade de inversão com algoritmo de montagem baseado em janelas de tempo

Caso o algoritmo utilizado seja 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 8.11). Seja t_j o instante de chegada do último pacote de controle pertencente ao par (S_j, D_j) e N

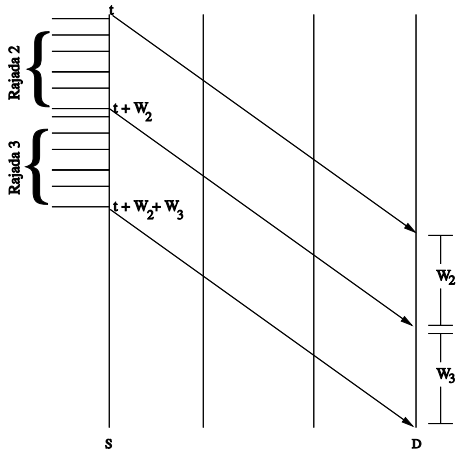


Figura 8.11: Tempo entre chegada de rajadas (algoritmo baseado em janelas de tempo).

o número de pares origem-destino. A probabilidade de inversão é calculada, tomando-se o número médio de pares origem destino (S_k, D_k) em que seus pacotes de controle podem chegar no intervalo $[t_j; t_j + \Delta_{jk} - \overline{\beta}]$, a probabilidade de inversão é dada por:

$$P_I = \frac{1}{N} \sum_{k=1}^{N} X_k \tag{8.3}$$

onde,

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

e $\overline{\beta}$ representa o tamanho médio das rajadas do par (S_k, D_k).

8.3.2 Cálculo da probabilidade de inversão com algoritmo de montagem baseado em volume de tráfego

A seguir, são derivadas expressões para o caso em que a montagem das rajadas é baseada em volume de tráfego. São contemplados os casos em que o tráfego pode ser modelado como um processo *Poisson* e quando o tráfego exibe características auto-similares.

Cálculo da probabilidade de inversão para tráfego *Poisson*

Se o algoritmo de montagem de rajadas utilizado for baseado em volume de tráfego, uma rajada só é montada (e o pacote de controle associado liberado) após a chegada de b_i bytes. Isso equivale ao fato de que aproximadamente n pacotes com tamanho médio igual a B, de forma que $n = b_i/B$, (onde b_i é o limiar para a criação da rajada), chegaram ao nó de montagem do par (S_k, D_k).

Seja (S_j, D_j) o par origem-destino para o qual o pacote de controle está em processamento no nó i. Seja t_j o instante de chegada desse pacote de controle. Como mencionado, para que haja inversão entre os pacotes de controle e rajadas futuras destinadas a outros pares origem-destino em relação ao pacote de controle do par (S_j, D_j) que chegou no instante t_j, os novos pacotes de controle devem chegar no intervalo $[t_j; t_j + \Delta_{jk} - \overline{\beta}]$.

Considere t_k o instante da chegada do último pacote de controle, por exemplo p_1, do par (S_k, D_k) ao nó i. Para que o novo pacote de controle, p_2, destinado ao nó k chegue dentro do intervalo $[t_j; t_j + \Delta_{jk} - \overline{\beta}]$ dado que p_1 chegou ao nó i no instante t_k, é necessário que o intervalo de tempo gasto na montagem da rajada associada a p_2 dure, no máximo, $[t_k; t_j + \Delta_{jk} - \overline{\beta}]$. Em outras palavras, é necessário que n pacotes cheguem ao nó de montagem dentro do referido intervalo. *Assumindo que o processo de chegadas de pacotes segue a distribuição de Poisson,* a probabilidade de que n pacotes cheguem dentro do intervalo $[t_k; t_j + \Delta_{jk} - \overline{\beta}]$, o que fará com que o pacote de controle p_2 destinado ao nó k chegue no intervalo $[t_j; t_j + \Delta_{jk} - \overline{\beta}]$ é dada pela Equação [53]:

$$P_I = \sum_{k=1}^{N} \frac{(\lambda_k([t_j - (t_k + \overline{\beta}) + \Delta_{jk}])^{n-1} e^{-\lambda_k([t_j - (t_k + \overline{\beta}) + \Delta_{jk}])}}{(n-1)!} \tag{8.4}$$

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

Cálculo da probabilidade de inversão para tráfego auto-similar usando limitantes de tráfego

Se o tráfego que adentra a rede OBS tiver características auto-similares, uma aproximação da probabilidade de inversão é possível de ser derivada através do uso de processos envelope. Considere uma rede OBS, em que o tráfego do par origem-destino (S_i, D_i) exibe características auto-similares. Seja $A(t)$ a quantidade de tráfego que chega a um nó dentro do intervalo $[0, t)$ e seja $\widehat{A}(t)$ seu processo envelope [46].

Sabe-se que o processo envelope determina a quantidade máxima de bytes que chega a um nó dentro do intervalo $[0, t)$. Com isso, para determinar a probabilidade de que uma rajada seja criada dentro de um intervalo de tamanho $[t_k; tj + \Delta_{jk} - \overline{\beta}]$, é preciso saber se o tamanho mínimo da rajada, b_i pode ser alcançado.

Sabe-se que se, $\widehat{A}(t_j + \Delta_{jk} - \overline{\beta}) - \widehat{A}(t_k) < b_i$, a probabilidade de que uma rajada seja criada no intervalo $[t_k; tj + \Delta_{jk} - \overline{\beta}]$ é nula. Assim, uma aproximação da probabilidade de inversão pode ser dada pela seguinte expressão:

$$P_I = \frac{1}{N} \sum_{k=1}^{N} X_k \tag{8.5}$$

onde,

$$X_k = \begin{cases} 1 & \text{, se } \widehat{A}(t_j + \Delta_{jk} - \overline{\beta}) - \widehat{A}(t_j) \geq b_i; \\ 0 & \text{, caso contrário} \end{cases}$$

8.4 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 8.1.2 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 requisiçõe futuras que cheguem em um intervalo de tempo curto. Para ter mais sucesso na sua estratégia de acomodação de rajadas, o *void* anterior deve ser maior, o que faz com que o algoritmo minimize o *void* posterior.

Se uma estratégia fixa é utilizada, os resultados podem ser insatisfatórios. Ao minimizar 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 deste tipo de estratégia não levam em consideração a capacidade dos outros *voids* de acomodar novas requisições.

Diferentemente dos algoritmos discutidos na seção 8.1, o algoritmo LRC não usa uma estratégia fixa. Seu critério de seleção depende da potencial capacidade de reutiilização por futuras requisições dos *voids* anterior e posterior criados após a alocação da requisição corrente. Para inferir sobre reutilização, 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 permite acomo-

dar a rajada no *void* com menor chances de ser reutilizado, deixando livres os *voids* com maiores chances de serem utilizados por reservas futuras.

O 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.

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:

$$\varphi(w) = P_I.\upsilon(a_j) + (1 - P_I).\upsilon(p_j) \tag{8.6}$$

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.

O algoritmo LRC é formalmente descrito no Algoritmo 6. Seja i o nó executando o algoritmo LRC. O algoritmo recebe como entrada o conjunto de canais de saída (W) e uma requisição de reserva de recursos no período $[r_i, r_f]$. Ao receber uma requisição de

reserva de recursos, o algoritmo LRC primeiro determina o conjunto W_v formado pelos canais capazes de acomodar a requisição, ou seja, o conjunto de canais que não possuem reservas no período solicitado (linha 1).

Para cada um dos canais $w \in W_v$, o algoritmo calcula: a probabilidade de inversão, o tempo de vida útil dos *voids* anterior e posterior e a função de reutilização.

A probabilidade de inversão (linha 2) é usada para determinar as chances de que uma nova requisição demande uma reserva para um intervalo de tempo anterior ao intervalo de tempo requerido pela reserva atual. Este cálculo não leva em consideração todos os nós da rede. Ao contrário, o cálculo da probabilidade de inversão é realizado sobre o conjunto de pares origem-destino que já enviaram, pelo menos uma rajada através do nó i. Desta forma é possível que o nó i tenha informações sobre o último instante de chegada do pacote de controle desses pares origem-destino. Além disso, o cálculo sobre uma fração dos nós diminui drasticamente a complexidade computacional do algoritmo.

Calcula-se, posteriormente o tempo de vida útil dos novos *voids* anterior (a_w) e posterior (p_w) gerados caso o canal w seja escolhido para acomodar a requisição (linha 4). De posse desse valor, é possível saber se o *void* a ser minimizado será o *void* anterior ou o posterior. Em outras palavras, caso a probabilidade de inversão seja alta é melhor que um canal que minimize o *void* posterior seja escolhido, caso contrário, um canal que minimize o *void* anterior deve ser escolhido.

O algoritmo determina, posteriormente, a função de reutilização do canal w (linha 5) de acordo com a Equação 8.6. Esta etapa corresponde a determinar as 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 $\varphi(w)$. Caso mais do que um canal tenha o mesmo valor de reutilização, o intervalo com menor *void* anterior será alocado. Por fim, o canal com a menor chance de reutilização futura é utilizado para acomodar a requisição em processamento (linha 6).

Algoritmo 6 LRC

ENTRADA

Um conjunto W de 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: Determine o conjunto de canais $W_v \subseteq W$ capazes de acomodar a requisição.
2: Determine a probabilidade de inversão (P_I) como discutido na seção 8.2.2 de acordo com o algoritmo de montagem utilizado.
3: **for all** ($w \in W_v$): **do**
4: o tempo de vida útil dos *voids* anterior e posterior à alocação da reserva em w de acordo com as equações 8.1 e 8.2, respectivamente.
5: a função de reutilização de w de acordo com a equação 8.6.
6: Retorne w tal que $\varphi(w)$ seja mínimo.

8.5 Complexidade Computacional

Esta seção, discute a complexidade computacional do algoritmo LRC, estabelecida pelo Teorema 1.

Teorema 1. *Seja N_p o conjunto de pares origem-destino da rede, W o conjunto de canais e S o conjunto de reservas já efetuadas em todos os canais. A complexidade do algoritmo LRC é da ordem de $O(|N_p||W|\log(|S|))$.*

Prova.

Para o cálculo do conjunto de canais (W_v) que podem acomodar a requisição em processamento, bem como a execução das linhas (3- 4), é necessário que o algoritmo LRC mantenha as mesmas informações que o algoritmo LAUC-VF. Assim, no caso em que $W_v = W$, tem-se uma complexidade de $O(|W|\log(|S|))$. Além disso, o cálculo da probabilidade de inversão (linha 2) em um nó i, é realizado sobre um sub-conjunto do conjunto de pares origem-destino, a saber, (N_p), o que resulta em uma complexidade computacional de $O(|N_p||W|\log(|S|))$.

□

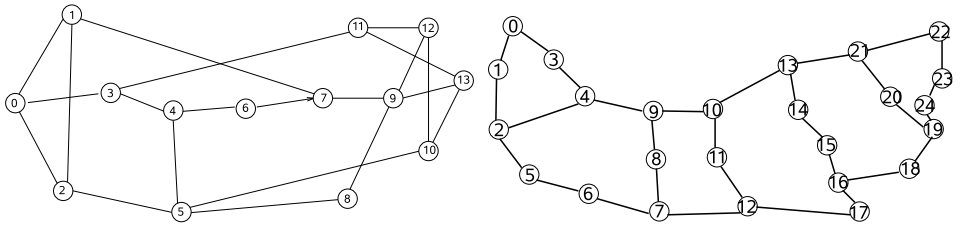
Algumas considerações são importantes a respeito da complexidade computacional do

algoritmo LRC. Na análise de pior caso, o algoritmo LRC apresenta elevada complexidade computacioal. Por exemplo, seja N_n o número total de nós da rede. Tomando-se o número de pares origem-destino em função do número de nós da rede, tem-se que o número máximo de pares origem-destino cujo tráfego passa pelo nó i. em uma topologia em que i é intermediário na rota entre todos os pares origem-destino $|N_p| = \frac{N_n(N_n-1)}{2}$, o que resulta em uma complexidade, em função do número de nós da rede, de $O(N_n^2|W|\log(|S|))$.

Contudo, em redes operacionais o grau de conectividade entre os nós permite o uso de técnicas de engenharia de tráfego como balanceamento de carga e roteamento baseado em restrições, em especial roteamento com interferência mínima [27], capazes de minimizar o compartilhamento de enlaces entre múltiplos pares origem-destino. Assim, somente um pequeno número de pares origem-destino compartilham enlaces de uma rota, o que diminui drasticamente a complexidade do algoritmo LRC.

8.6 Exemplos Numéricos

Para avaliar o desempenho dos algoritmos de escalonamento, simulações foram realizadas. A ferramenta utilizada nas simulações foi o simulador OB2S (*Optical Burst Switching Simulator*) desenvolvido na Universidade Salvador [44]. 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 intervalos de confiança possuem um nível de confiança de 95%.



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

Figura 8.12: Topologias usadas nas simulações.

Os experimentos foram realizados utilizando as topologias das redes Abilene e NSFNet apresentadas na Figura 8.12*. Cada enlace da rede é constituído por uma fibra com 32 comprimentos de onda com capacidade de transmissão de 2.5 Gbps cada. Em cada nó, é empregada a conversão total dos comprimentos de onda, o que implica dizer que todos os comprimentos de onda na saída do comutador podem ser usados pelo algoritmo de escalonamento, independentemente de qual comprimento de onda tiver sido usado na chegada do pacote de controle. 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. A rota entre cada par origem-destino é determinada no início da operação da rede pelo algoritmo de menor caminho em relação ao número de nós.

Como existem diferentes métodos para cálculo da probabilidade de inversão, dependendo do algoritmo de montagem de rajadas utilizado no nó de borda da rede OBS, dois algoritmos de montagem de rajadas foram usados nos nós de origem. O algoritmo baseado em janelas de tempo [28] e o algoritmo baseado em volume de tráfego [78].

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 LAUC-VF. As métricas utilizadas na avaliação de desempenho foram a probabilidade de bloqueio (ou taxa de perdas de rajadas), a utilização da rede, a utilização efetiva da rede, o tamanho médio das rotas atendidas e o fator de justiça.

A probabilidade de bloqueio é a razão entre o número de rajadas (pacotes de controle) descartados em relação ao número total de rajadas gerado. A utilização é calculada como a razão entre o tempo em que os recursos da rede (comprimentos de onda, malha de comutação etc) estiveram reservados por todas as rajadas em relação ao tempo total de observação. A utilização da rede é calculada como a média aritimética da utilização

*Por questões de organização, somente serão apresentados os resultados para a rede NSFNet. Os resultados das simulações na rede Abilene estão em perfeita coerência com os apresentados nesta seção

em cada um dos comutadores da rede (malha de comutação e comprimentos de onda das portas de saída), levando-se em consideração o tempo em que os recursos ficaram utilizados em todas as rajadas, tanto as descartadas quanto as bem sucedidas. Já a utilização efetiva, mede a utilização da rede somente pelas transmissões bem sucedidas.

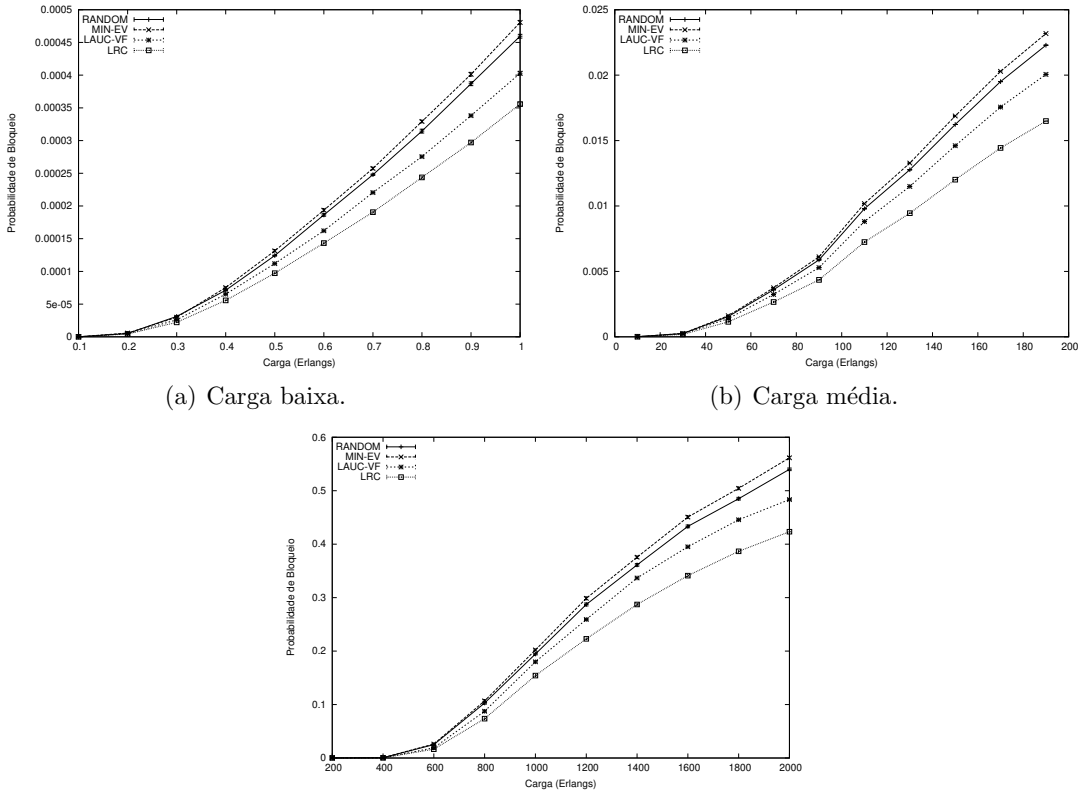
A probabilidade de inversão na ordem entre os pacotes de controle e rajadas depende, dentre outros fatores, do tamanho do tempo de ajuste presente nos pacotes de controle transmitidos na rede. O tempo de ajuste, como apresentado no Capítulo 2, por sua vez, depende do tamanho da rota à qual pertence o pacote de controle. Assim, a estratégia de minimizar, ora os *voids* anteriores, ora os *voids* posteriores, pode ser benéfica ao atendimento das rajadas pertencentes a rotas de diferentes tamanho. Para se avaliar o desempenho, o tamanho médio das rotas atendidas foi utilizado como métrica de desempenho. O tamanho médio das rotas atendidas é calculado como a média aritimética dos tamanhos das rotas em cada transmissão de rajada bem sucedida.

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. O fator de justiça deve, idealmente, possuir valor um, indicando que o número de rotas bloqueadas de tamanho mínimo e máximo é igual. Quando a quantidade de 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. Analogamente, se a quantidade de rotas de tamanho mínimo é maior, o valor do fator de justiça será menor do que um. O fator de justiça também serve como um indicativo do impacto do algoritmo em rotas de diferentes tamanhos, isto é, ao adotar uma determinada estratégia (minimizar *void* anterior ou posterior), o algoritmo pode penalizar rotas com diferentes tamanhos.

Para avaliar o impacto do cálculo da probabilidade de inversão no desempenho dos algoritmos. foram realizados experimentos usando diferentes tipos de algoritmos de montagem nos nós de borda da rede. A seguir, os resultados dos experimentos de simulação são apresentados.

8.6.1 Experimentos com algoritmo de montagem de rajadas baseado em janelas de tempo

Nestes experimentos, o nó de borda é alimentado por tráfego gerado segundo a distribuição de *Poisson* e o tempo de montagem das rajadas é de 1ms. As simulações foram realizadas com tráfego em três diferentes intensidades: carga baixa (carga variando de 0.1 a 1 Erlang), carga média (carga variando de 10 a 190 Erlangs) e carga alta (carga variando de 200 a 2000 Erlangs).



(a) Carga baixa.

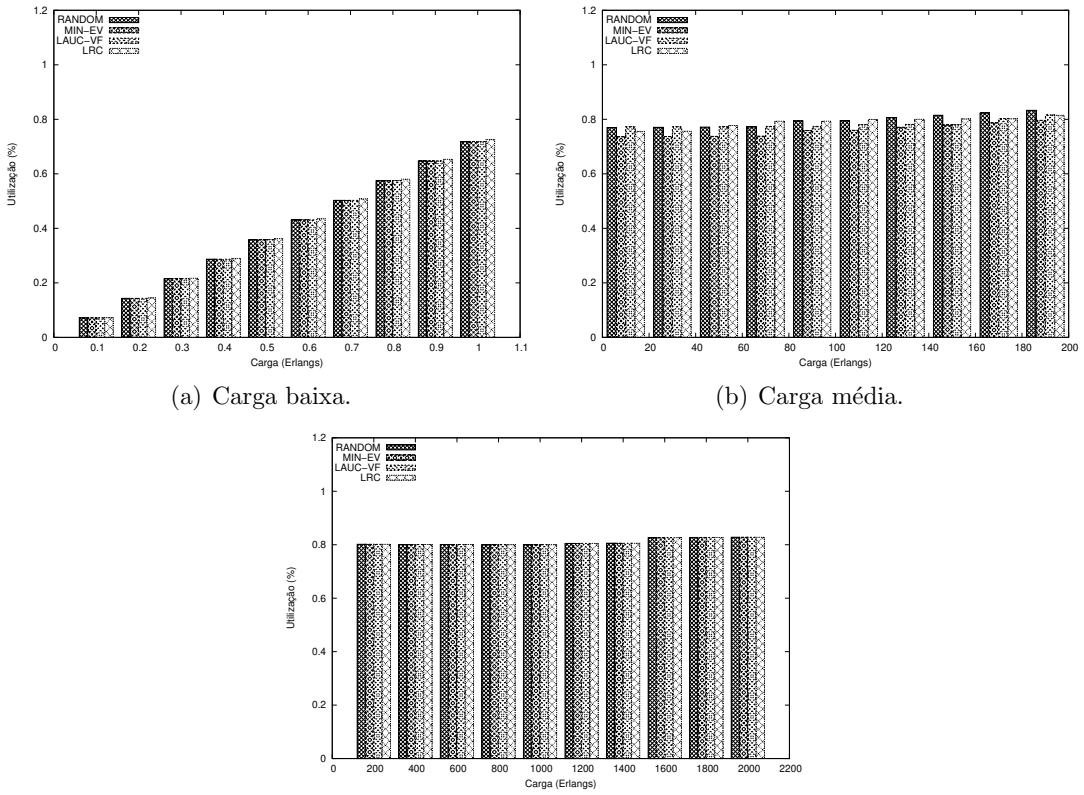
(b) Carga média.

(c) Carga alta.

Figura 8.13: Probabilidade de bloqueio (janelas de tempo).

A Figura 8.13 ilustra a probabilidade de bloqueio produzida pelos algoritmos. De um modo geral, o algoritmo de pior desempenho foi o MIN-EV. O algoritmo RANDOM apre-

sentou o segundo pior desempenho. O algoritmo LRC apresentou a menor probabilidade de bloqueio entre os algoritmos avaliados, com um ganho médio de 13% em relação ao LAUC-VF (segundo melhor algoritmo) e de $27,6\%$ em relação ao pior algoritmo.



(a) Carga baixa.

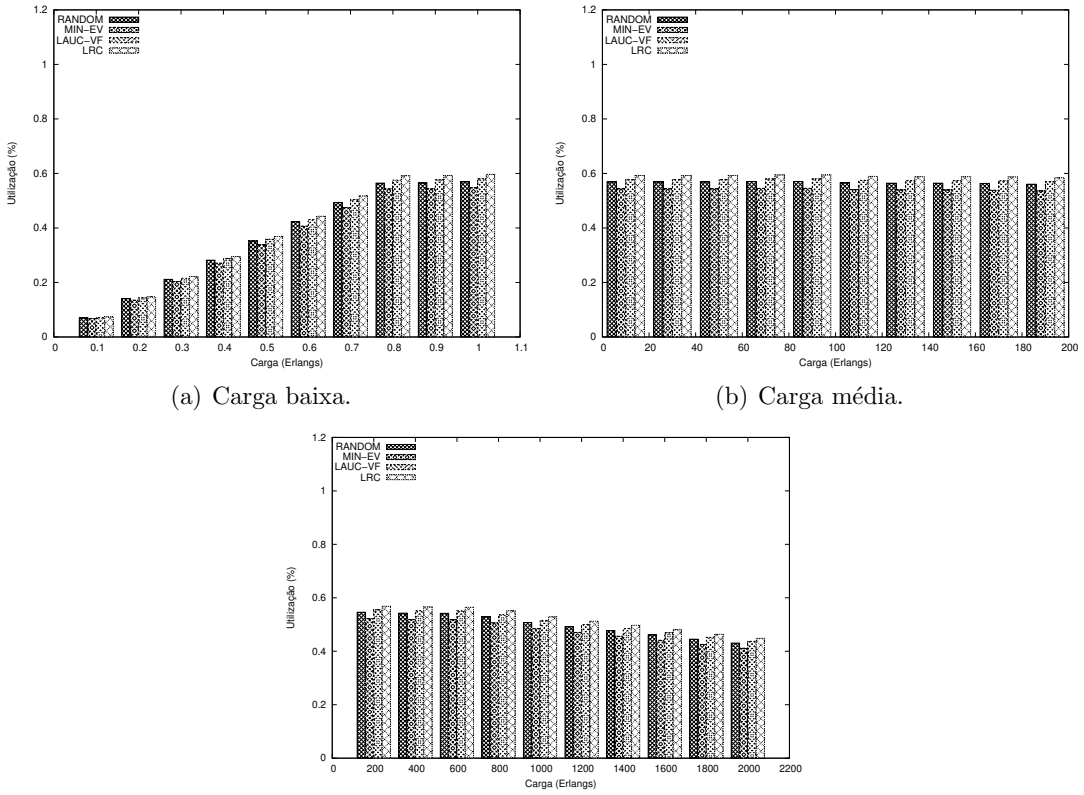
(b) Carga média.

(c) Carga alta.

Figura 8.14: Utilização média da rede (janelas de tempo).

A Figura 8.14 mostra a utilização em função do aumento da carga de tráfego. É possível notar que a utilização dos algoritmos cresce com o crescimento da carga de tráfego, pois o número de requisições transitando na rede aumenta, o que aumenta, por sua vez, o número de reservas realizadas. Esse efeito acontece de forma quase linear até uma carga de um Erlang, onde a utilização alcança 78%. A partir desse ponto, a taxa de aumento da utilização diminui sensívelmente, já que a rede vai se tornando saturada e grande parte das

novas requisições é perdida. Além disso, percebe-se que reservas não foram realizadas em cerca de 20% do tempo. Isso quer dizer que nesses casos, os *voids* não foram preenchidos com nenhuma alocação.



(a) Carga baixa.

(b) Carga média.

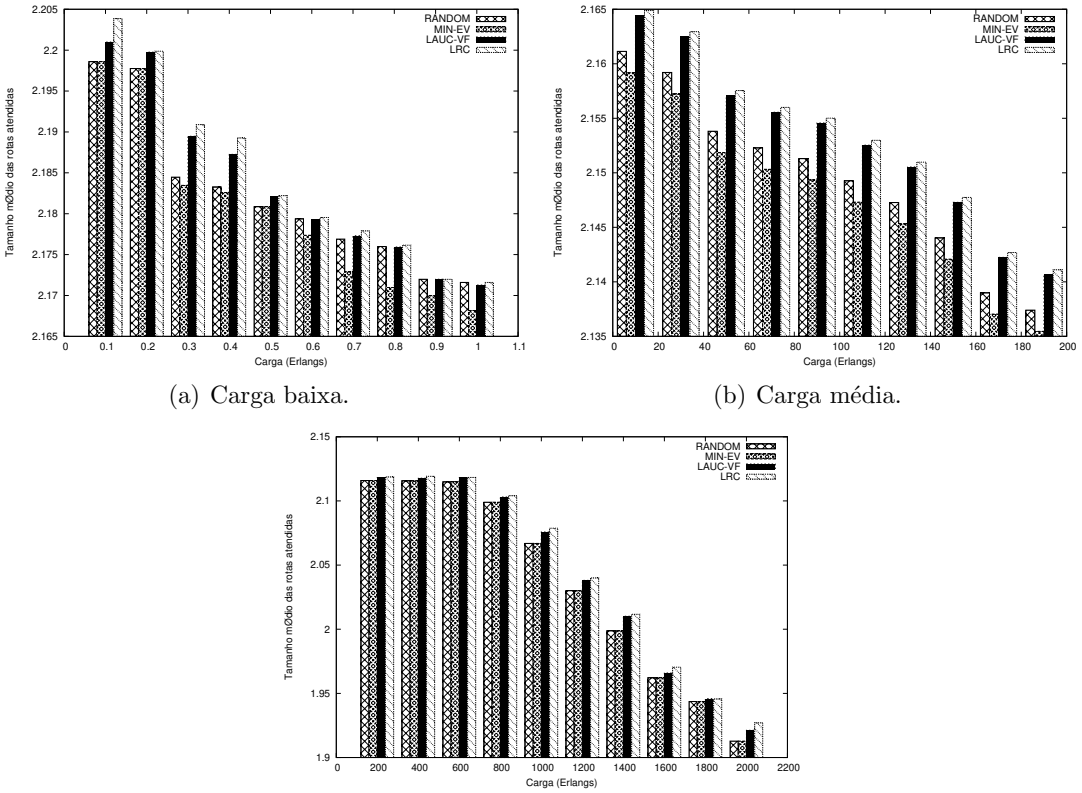
(c) Carga alta.

Figura 8.15: Utilização Efetiva média da rede (janelas de tempo).

A Figura 8.15 mostra a utilização efetiva da rede em função do aumento da carga. É possível perceber que existe um crescimento quase linear até aproximadamente 0.8 Erlangs e uma estagnação a partir deste ponto até que a carga da rede atinja aproximadamente 800 Erlangs, onde a utilização efetiva apresenta um leve decrescimento.

Como a utilização efetiva é medida em função das transmissões bem sucedidas, o algoritmo que apresentou a maior utilização dentre os algoritmos avaliados foi aquele

que apresentou a menor probabilidade de bloqueio e conseguiu atender rotas de maior tamanho, o LRC.



(a) Carga baixa.

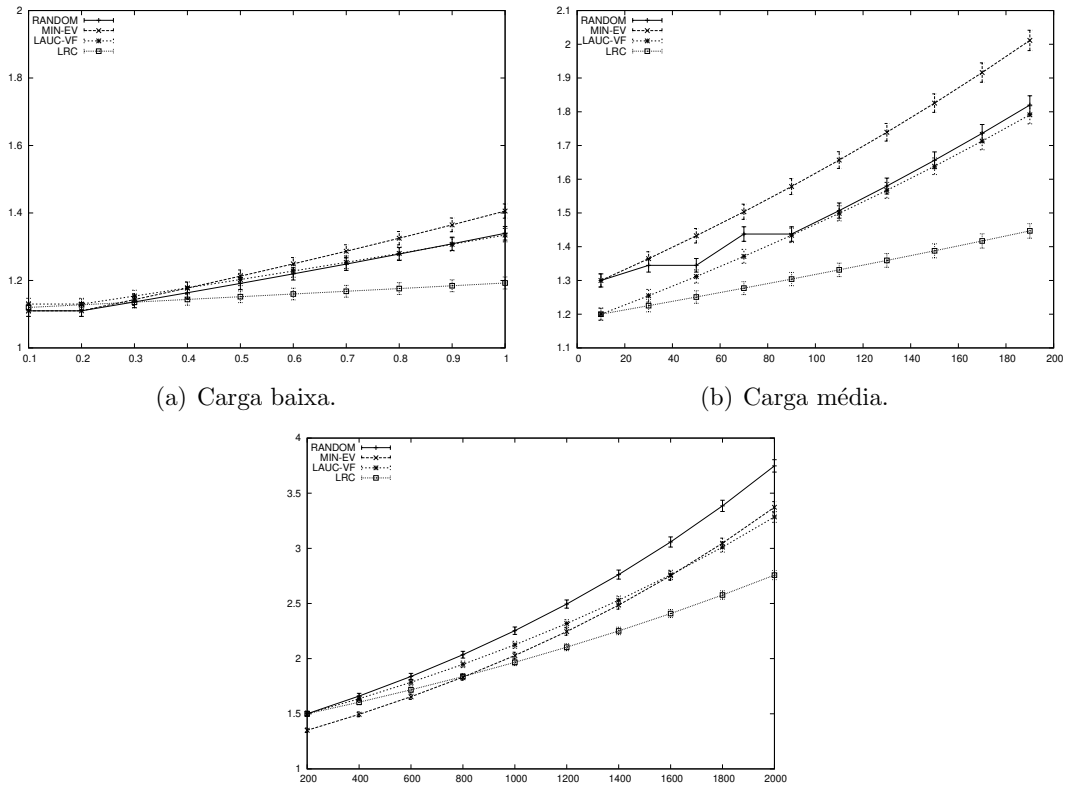
(b) Carga média.

(c) Carga alta.

Figura 8.16: Tamanho médio das rotas atendidas (janelas de tempo).

O efeito de se conseguir atender rotas com diversos tamanhos pode ser verificado nas Figuras 8.16 e 8.17. A Figura 8.16 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, dado 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. Contudo, é possível notar que o algoritmo LRC consegue minimizar esse efeito quando comparado aos demais algoritmos.

O algoritmo LRC consegue este efeito através do uso de sua estratégia dinâmica na escolha dos canais, que depende da capacidade de reutilização dos mesmos. Isso permite que o algoritmo ao realizar a alocação de uma rajada já próxima ao seu destino e cuja probabilidade de inversão é pequena, ocupe os *voids* anteriores. Por outro lado, ao realizar a reserva de requisições de rotas maiores, os pacotes de controle contém tempo de ajuste grande, e consequentemente alta probabilidade de inversão. Assim, os *voids* posteriores podem ser usados, deixando *voids* anteriores para requisições futuras que causarão a inversão.



(a) Carga baixa.

(b) Carga média.

(c) Carga alta.

Figura 8.17: Fator de justiça (janelas de tempo).

A Figura 8.17 mostra que o algoritmo LRC apresenta fator de justiça mais próximos

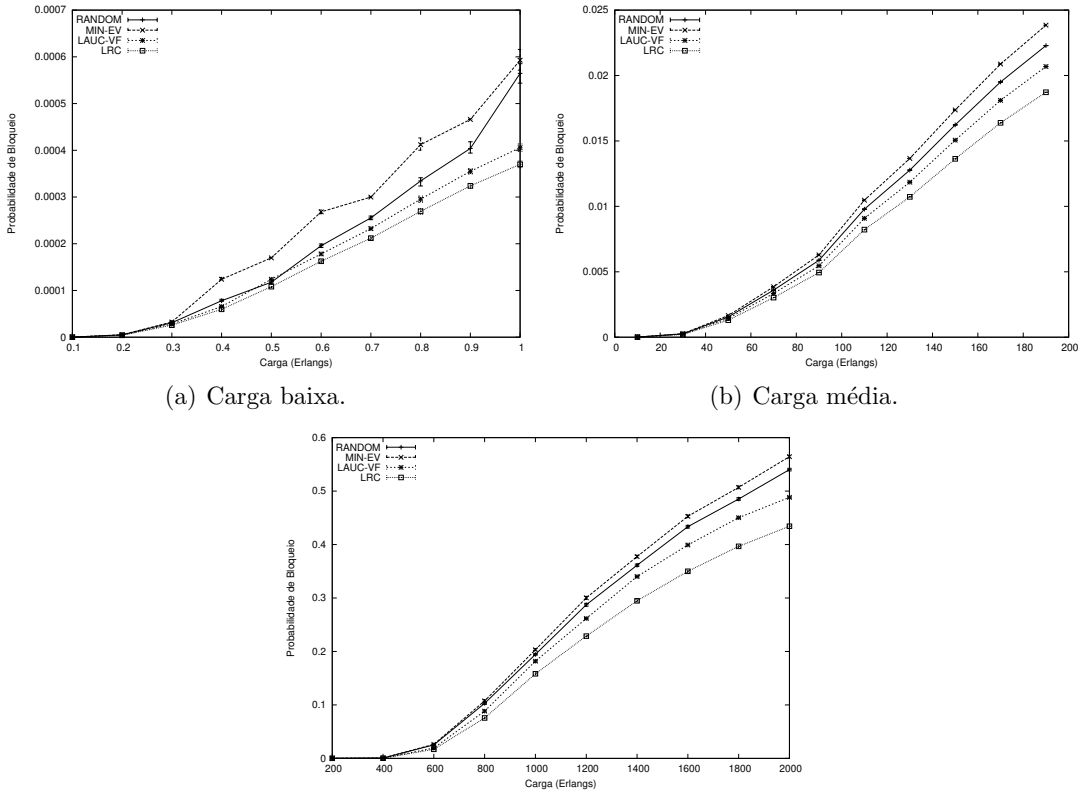
a um, indicando assim que consegue operar melhor em redes com rotas de tamanhos variados. À medida em que a carga aumenta, ocorre a diminuição do tamanho médio das rotas e as rotas maiores são mais penalizadas. Entretanto o algoritmo LRC produz um crescimento menos acentuado, o que indica que as rotas de tamanho máximo sofrem menos perdas quando o algoritmo LRC é utilizado.

8.6.2 Experimentos com algoritmo de montagem de rajadas baseado em volume de tráfego (tráfego *Poisson*)

Neste conjunto de experimentos, o nó de borda é alimentado por tráfego gerado segundo a distribuição de *Poisson*. O limiar do tamanho da rajada utilizado foi de 1280KB. As simulações foram realizadas com tráfego em três diferentes intensidades: carga baixa (carga variando de 0.1 a 1 Erlang), carga média (carga variando de 10 a 190 Erlangs) e carga alta (carga variando de 200 a 2000 Erlangs).

A Figura 8.18 ilustra a probabilidade de bloqueio gerada pelos algoritmos quando o montador utilizado nos nós de borda é baseado em volume de tráfego. Os resultados são absolutamente coerentes com aqueles reportados na seção anterior, entretanto, o ganho apresentado pelo algoritmo LRC em relação ao algoritmo de pior desempenho (MIN-EV) caiu para $23,16\%$ e o ganho em relação ao algoritmo com a segunda menor probabilidade de bloqueio (LAUC-VF) caiu para $9,1\%$. Essa queda pode ser explicada por uma queda na precisão da estimação da probabilidade de inversão, quando o algoritmo de montagem baseado em volume de tráfego é utlizado.

As Figuras 8.20 a 8.22 mostram a utilização da rede, o tamanho médio das rotas atendidas e o fator de justiça em função do aumento da carga de tráfego na rede. Os resultados confirmam que o algoritmo LRC produz uma maior utilização da rede, enquanto consegue atenuar a penalização sofrida pelas rotas com um número maior de saltos quando a carga aumenta.



(a) Carga baixa.

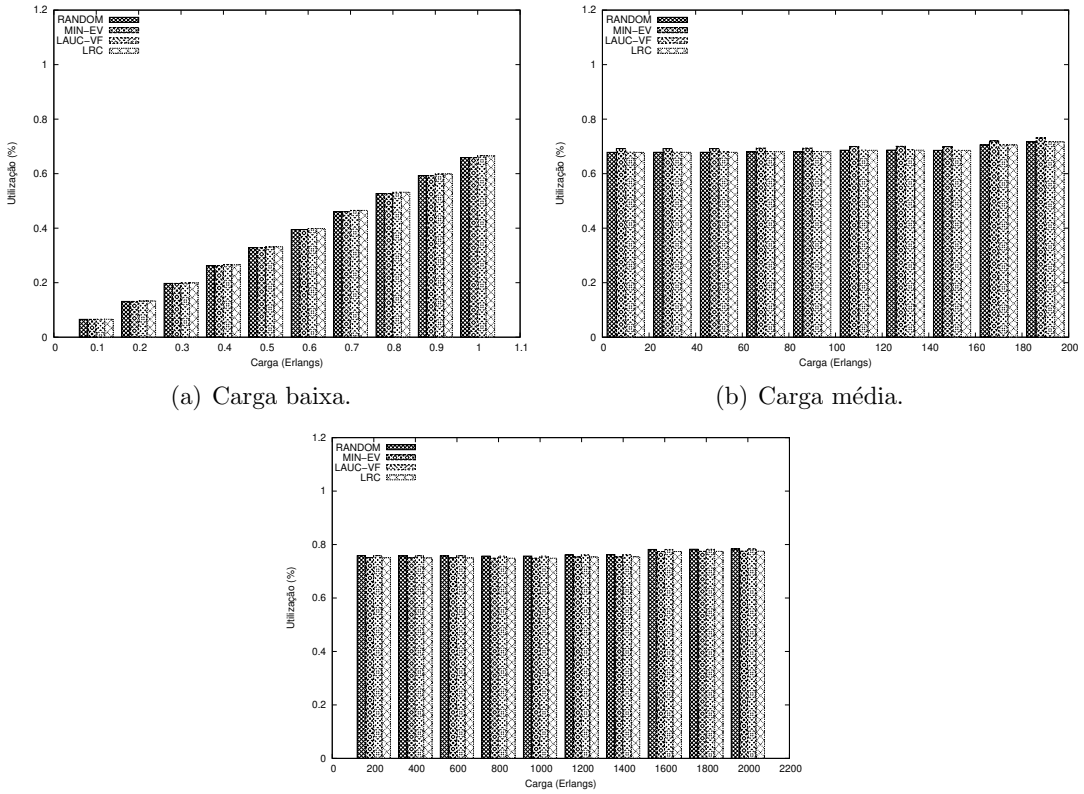
(b) Carga média.

(c) Carga alta.

Figura 8.18: Probabilidade de bloqueio (volume de tráfego).

8.6.3 Experimentos com algoritmo de montagem de rajadas baseado em volume de tráfego (tráfego auto-similar)

Neste conjunto de experimentos, os nós de borda são alimentados por tráfego gerado através dos traços de tráfego real 20040601-194000-1 e 20040601-193121-1, descritos no Capítulo 4. O limiar do tamanho da rajada é de 1280KB. O processo de montagem de rajadas é realizado de forma *offline*, ou seja, primeiramente, os nós foram alimentados com os traços 20040601-194000-1 e 20040601-193121-1 e as rajadas montadas de acordo com o algoritmo baseado em volume de tráfego apresentado em [9]. O registro de tempo e tamanho da nova rajada criada foi então gravado, criando-se assim um novo traço com



(a) Carga baixa.

(b) Carga média.

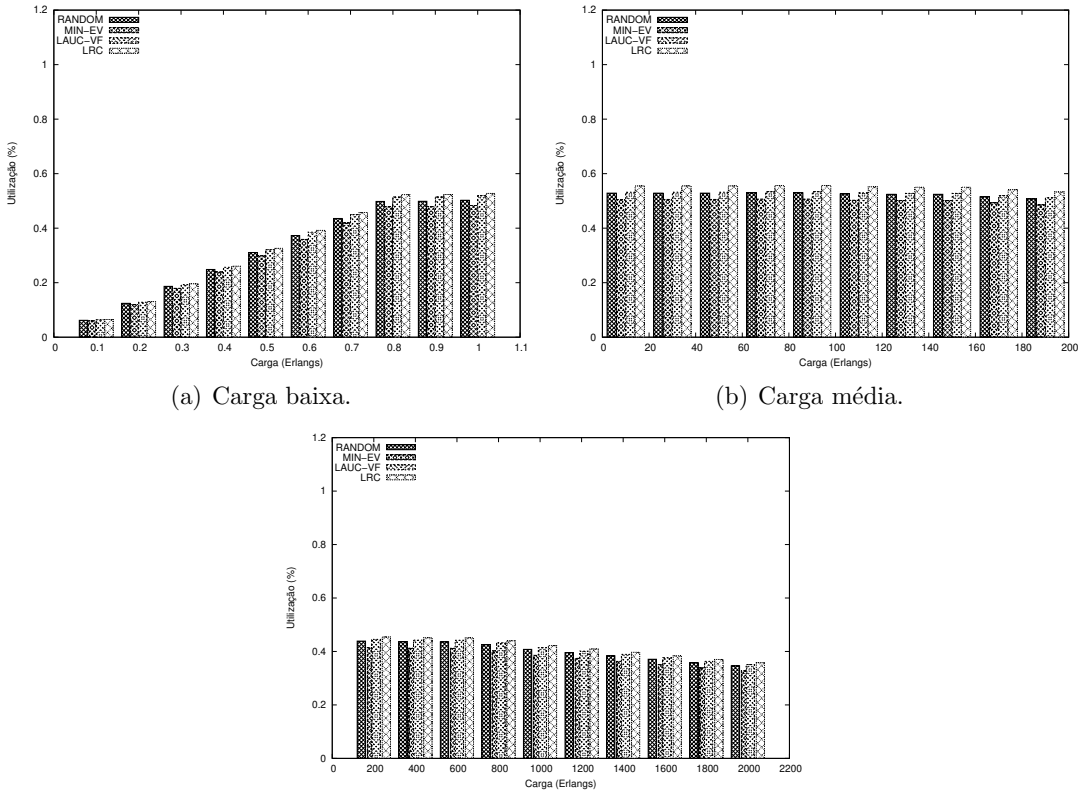
(c) Carga alta.

Figura 8.19: Utilização média da rede (volume de tráfego).

informações sobre o tempo inter-chegada de rajadas e seu tamanho correspondente.

De acordo com o que foi discutido no Capítulo 4, o tráfego contido nos novos traços gerados possui características auto-similares monofractais. O processo envelope desses traços foi calculado de acordo com procedimento em [46] e essa informação foi armazenada em todos os nós da rede. A partir daí, com os parâmetros do processo envelope já calculados a probabilidade de inversão foi então calculada como discutido na seção 8.3.2

Os resultados das simulações são apresentados nas Tabelas 8.1 e 8.2. As legendas Alg, PB, U, UE, Tam e FJ representam, respectivamente o algoritmo avaliado, a probabilidade de bloqueio, a utilização, a utilização efetiva, o tamanho das rotas e o fator de



(a) Carga baixa.

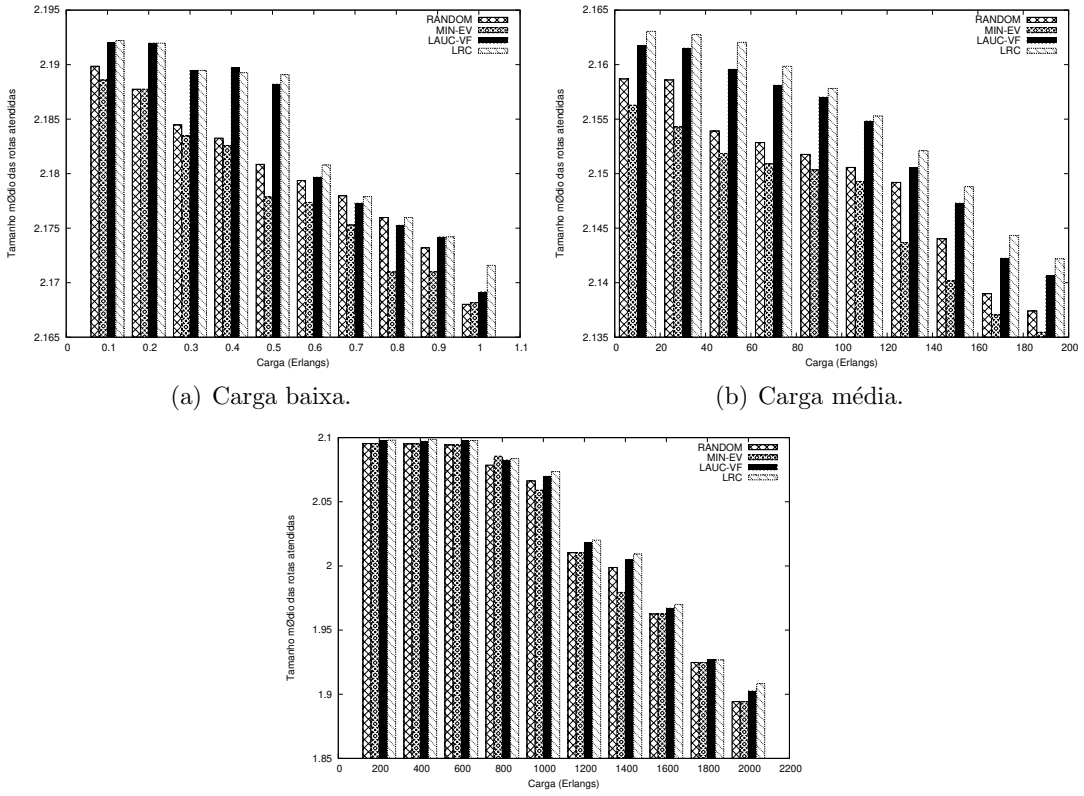
(b) Carga média.

(c) Carga alta.

Figura 8.20: Utilização Efetiva média da rede (volume de tráfego).

justiça. É possível perceber que o algoritmo LRC produz ainda a menor probabilidade de bloqueio dentre os algoritmos avaliados, entretanto, o ganho em relação ao pior (MIN-EV) e segundo melhor (LAUC-VF) algoritmos foi reduzido a $12,5\%$ e 3% para o traço 20040601-193121-1 e $16,34\%$ e $4,3\%$.

Tal redução deve-se ao fato de que o cálculo da probabilidade de inversão baseado em processos envelope apresenta deficiências em duas situações. A primeira situação é que o processo envelope é um limitante superior do tráfego. Apesar de estudos realizados em [46, 47] mostrarem que processos envelopes tanto de fluxos multifractais quanto de fluxos monofractais limitam o tráfego com alto grau de precisão além de serem justos,



(a) Carga baixa.

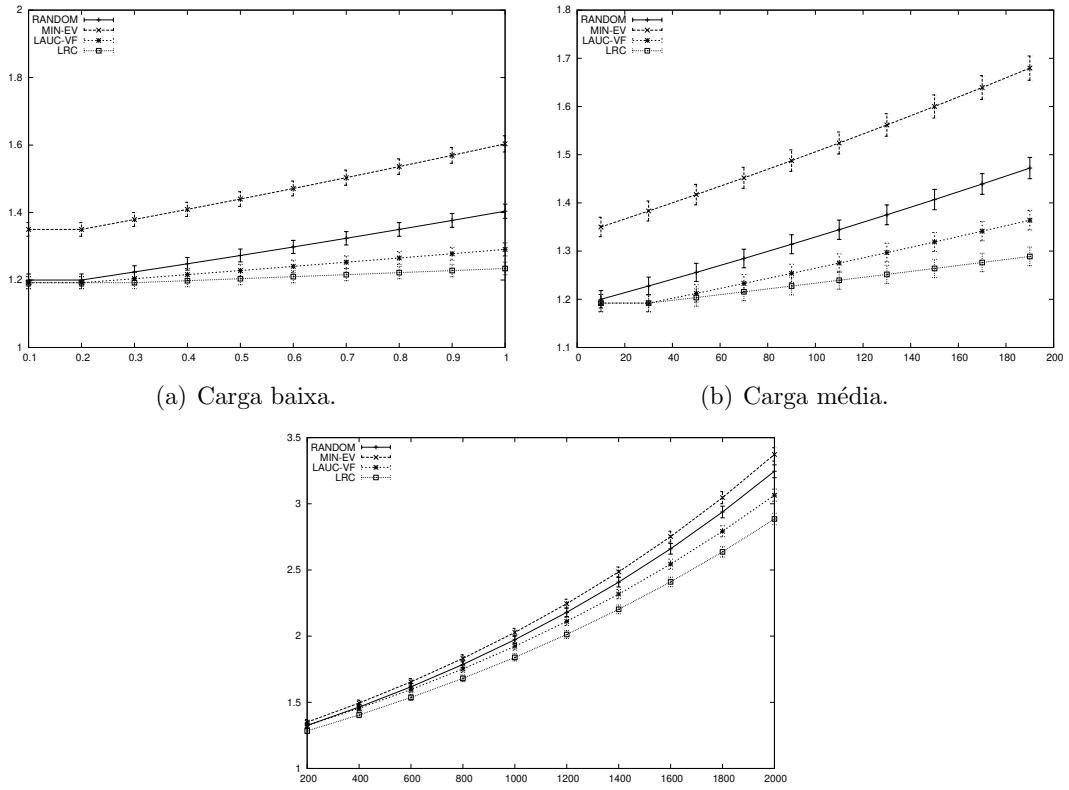
(b) Carga média.

(c) Carga alta.

Figura 8.21: Tamanho médio das rotas atendidas (volume de tráfego).

a alta variabilidade do tráfego pode fazer com que em determinados instantes, o volume de tráfego seja bem inferior ao previsto pelo processo envelope. Em outras palavras, uma situação em que não haveria inversão pode ser errôneamente identificada como uma situação de inversão.

A segunda situação é que o processo envelope pode ser violado com probabilidade κ. Uma situação identificada como de não inversão pode, devido à violação do processo envelope, levar ocasionalmente, a inversão.



(a) Carga baixa.

(b) Carga média.

(c) Carga alta.

Figura 8.22: Fator de justiça (volume de tráfego).

8.7 Resumo conclusivo

Neste capítulo, foi apresentado o algoritmo LRC. O algoritmo, diferentemente dos demais algoritmos discutidos, não usa uma estratégia fixa para alocação dos canais às reservas. Ao contrário, o algoritmo LRC utiliza o canal com menos chances de ser utilizado por reservas futuras. Com isso, consegue uma menor probabilidade de bloqueio, quando comparado aos demais algoritmos existentes.

Apesar do bom desempenho do algoritmo LRC, melhorias no cálculo da probabilidade de inversão, discutido na seção 8.2.2 podem ser realizadas, levando a uma eventual melhoria no desempenho do algoritmo.

Tabela 8.1: Resultados de simulação para o traço 20040601-193121-1.

Alg	PB	U	UE	Tam	FJ
RANDOM	$5.601x10^{-4}$	65.4%	50.1%	2,18	1.37
MIN-EV	$5.824x10^{-4}$	65.3%	49.7%	2,13	1.40
LAUC-VF	$5.208x10^{-4}$	65.2%	50.9%	2,20	1.3
LRC	$5.096x10^{-4}$	66%	50.3%	2,20	1.3

Tabela 8.2: Resultados de simulação para o traço 20040601-194000-1.

Alg.	PB	U	UE	Tam	FJ
RANDOM	$1.601x10^{-4}$	67.4%	52.4%	2,18	1.42
MIN-EV	$1.664x10^{-4}$	67%	67.4%	2,17	1.44
LAUC-VF	$1.456x10^{-4}$	66.9%	54.1%	2,19	1.36
LRC	$1.392x10^{-4}$	67.5%	55.0%	2,270	1.34

Além disso, a complexidade computacional do algoritmo pode ser elevada em alguns casos. Parte desta complexidade advém do cálculo da probabilidade de inversão. Assim, em trabalhos futuros, esforços podem ser realizados para, além de melhorar a precisão do cálculo da probabilidade de inversão, diminuir a complexidade computacional do algoritmo LRC.

Capítulo 9

Algoritmos de escalonamento de canais em lote para redes OBS

Este capítulo apresenta dois algoritmos ótimos para o escalonamento em lote de canais em redes OBS. O primeiro deles, denominado GreedyOPT, é apropriado para o problema de escalonamento em lotes quando todas as requisições da rede possuem pesos idênticos. O segundo, denominado BATCHOPT, é apropriado para o caso em que as requisições da rede possuem pesos não idênticos. Os algoritmos consideram, além das requisições em processamento, o conjunto de requisições previamente escalonadas. Isso permite uma modelagem única do problema de escalonamento em lote de canais, dado que todos os canais podem ser potencialmente usados por todas as requisições, diminuindo assim, as restrições do problema de alocação das requisições em processamento, o que permite a elaboração de algoritmos ótimos e rápidos.

Uma contribuição adicional apresentada neste capítulo é uma nova estratégia de formação de lote 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.

9.1 Escalonamento em lote de canais

A minimização da probabilidade de bloqueio em uma rede OBS pode ser alcançada através da maximização da chance 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 chance de ser acomodada.

Os algoritmos de escalonamento discutidos no Capítulo 8 adotam uma estratégia gulosa ao fazerem a reserva dos recursos para garantir boas chances de acomodação da rajada subseqüente. A estratégia é dita gulosa pois cada requisição é processada individualmente, usando somente as informações disponíveis no momento do processamento. Assim, não existe nenhum tipo de garantia de que a próxima rajada será alocada com sucesso.

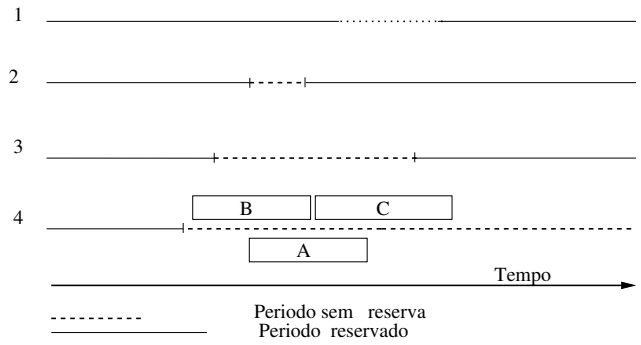


Figura 9.1: Problema no uso de estratégia gulosa no escalonamento de canais.

A Figura 9.1 ilustra uma situação que pode ocorrer com certa frequência quando tal estratégia gulosa é utilizada. 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. Percebe-se que se o canal 4 for usado para acomodar a reserva A, nenhum *void* pode ser utilizado pelas reservas B e C, entretanto, se o canal 3 for usado para acomodar a reserva A, ambas

as reservas (B e C) podem ser acomodadas com sucesso.

É fácil assim perceber a existência de um fator de incerteza associado à alocação dos recursos nas redes OBS, que advém do fato de que não se sabe *a priori* os instantes de início e término (a duração) e nem a prioridade das requisições que chegarão no futuro. Assim, por melhor que seja a estratégia gulosa adotada pelos algoritmos, é provável que algumas rajadas sejam desnecessariamente perdidas.

Para diminuir tal incerteza, uma nova classe de algoritmos de escalonamento foi proposta em [35] e em [12]: a classe dos algoritmos de escalonamento em lote. A idéia é diminuir o grau de incerteza, agrupando o máximo possível de requisições e processando-as de uma única vez, minimizando assim, as chances de perdas, dado que a melhor combinação pode ser escolhida, como ilustrado na Figura 9.2.

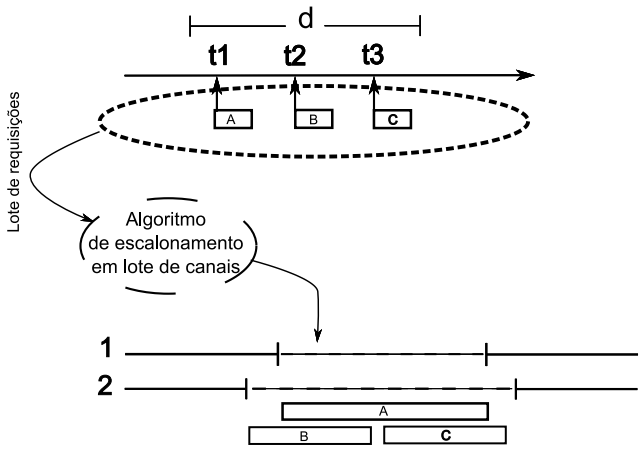


Figura 9.2: Escalonamento em lote de canais.

Uma estratégia ideal consiste em armazenar todas as requisições de reserva que transitarão pela rede e processá-las de uma vez usando um algoritmo ótimo. Dessa forma, a melhor escolha pode ser tomada. Apesar de tal estratégia ser, obviamente infactível, dado que o tempo que se pode armazenar requisições de reserva é limitado pela chegada das rajadas correspondentes, aproximações podem ser derivadas. Com o objetivo de criar aproximações, os algoritmos de escalonamento em lote foram propostos.

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. Assim, a incerteza sobre as requisições futuras presente na estratégia de escalonamento gulosa pode ser diminuída.

De um modo geral, a abordagem adotada consiste nos seguintes passos:

1. Atribuição de uma ordem linear nas requisições do lote;

2. Percorrendo as requisições nesta ordem, associação de cada requisição ao canal com menor índice que não está sendo utilizado por nenhuma requisição que precede a requisição sendo processada.

O problema das abordagens presentes em [35] e em [12], 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 tanto o conjunto de requisições do lote quanto 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*.

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 [35] e em [12] 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 capítulo, 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.

9.2 Algumas Definições

Nesta seção são apresentados conceitos necessários para o entendimento dos algoritmos de escalonamento discutidos neste capítulo.

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. Dado um conjunto $V(H) \subseteq V(G)$, o subgrafo de G induzido por $V(H)$ é o grafo $H = (V(H), E(H))$, onde $E(H) = \{(u, v) \in E(G); u, v \in V(H)\}$.

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.

Um vértice v de G é simplicial se sua vizinhança (conjunto de vértices adjacentes a v) induz uma clique. Um esquema de eliminação perfeito é uma ordenação $[v_n, ..., v_1]$ dos vértices de G tal que cada vértice v_i (com $1 \leq i \leq n-1$) é um vértice simplicial no subgrafo induzido por $V(G) \setminus v_{i+1}$

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 $I = \{J_1 = (s_1, e_1, p_1), \ldots, J_n = (s_n, e_n, p_n)\}$ uma lista de n tarefas, onde (s_i, e_i) é o tempo de início e fim da tarefa J_i, e p_i o seu peso. Tem-se um conjunto de W 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 W máquinas. O escalonamento gerado deve satisfazer o fato de que em cada máquina não poder 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, um conjunto de W 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. Objetiva-se, neste problema, alocar o maior número possível de rajadas (ou o conjunto de rajadas cuja soma dos pesos é máxima) nos W canais. Duas rajadas em um mesmo canal não podem se sobrepor, ou seja, seus tempos de execução não podem ter uma intersecção.

Na seção 9.3, 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 [35]. Na seção 9.4, mostra-se como resolver o problema de escalonamento em redes OBS utilizando algoritmos polinomiais para o problema *job scheduling com máquinas idênticas*.

9.3 Trabalhos relacionados

Em [35], o problema de escalonamento de canais é resolvido através de uma formulação do problema de *job scheduling* com máquinas não idênticas. Neste caso, o conjunto S de

rajadas previamente alocadas representam 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.

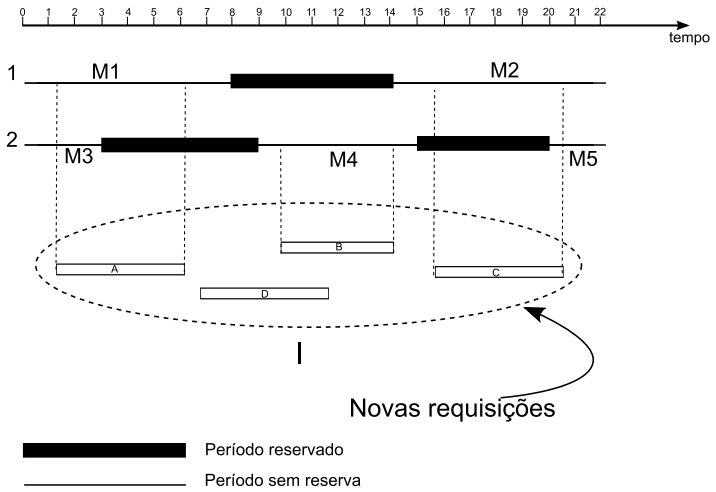


Figura 9.3: Modelagem do problema de escalonamento em lote de canais como *job scheduling* com máquinas não idênticas.

A Figura 9.3 ilustra a modelagem do problema de escalonamento em lote de canais em redes OBS como um problema de *job scheduling* com máquinas não idênticas. Na figura, existem dois canais de dados que podem ser utilizados para acomodar as requisições. É possível perceber que o canal 1 está reservado no intervalo de tempo compreendido entre os instantes 8 e 14 e os *voids* formados pelos intervalos $[0, 8)$ e $(14, \infty)$ podem ser usados para novas reservas. Dessa forma, os *voids* são modelados como máquinas distintas (M1 e M2) com períodos de funcionamento restrito. Da mesma forma, o canal 2 possui reservas nos intervalos $[3, 9]$ e $[15, 20]$. Assim, os períodos usados para acomodar novas reservas são: $[0, 3)$, $(9, 15)$ e $(20, \infty)$, o que na modelagem do problema de *job scheduling* seria equivalente a três máquinas distintas, M3, M4 e M5. Dessa forma, tem-se cinco máquinas com diferentes períodos de operação (diferentes capacidades).

Em [6], apresenta-se um algoritmo ótimo para *job scheduling* com máquinas não idênticas cuja complexidade computacional é da ordem de $O(n^{|W|+1})$, o que é proibitivamente alto em redes OBS [35], dada a exponencialidade em $|W|$, onde W é o conjunto de máquinas. Dessa forma, em [35] são propostas 4 heurísticas para o problema de escalonamento em lote em redes OBS.

A seguir, serão brevemente descritas as heurísticas propostas por [35]. Em todas elas, as requisições são vistas como um grafo de intervalos G.

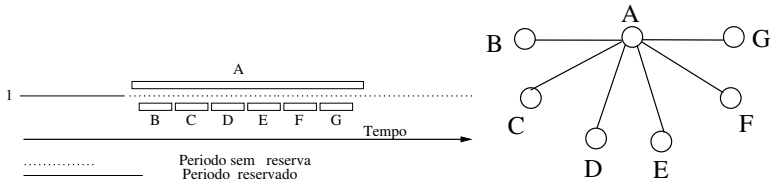
9.3.1 Algoritmo *Smallest Vertex Ordering (SLV)*

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 complexidade computacional do algoritmo é de $O(n + n^2 + n|W|log(|S|))$, dado que demanda-se $O(n)$ operações para realizar a ordenação *smallest-last* [45], $O(n^2)$ para realizar a construção do grafo de intervalos e $O(n|W|log(|S|))$ para a alocação da reserva [35].

A idéia central 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 [35], o processamento prévio das requisições com alto grau de intersecção pode ocasionar um número elevado de perdas.

A Figura 9.4 ilustra o problema mencionado: suponha que as requisições se dispõem na forma apresentada na Figura 9.4(a). O grafo de intervalos correspondente é apresentado na Figura 9.4(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.



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

Figura 9.4: Problema ocasionado no escalonamento em lote.

9.3.2 Algoritmo *Maximal Cliques First (MCF)*

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

A implementação da ordenação MCF possui complexidade computacional de $O(n^2)$ e a reserva do recurso possui complexidade de $O(n|W|log(|S|))$. Isso resulta em uma complexidade computacional da ordem de $O(n^2 + n|W|log(|S|))$ [35].

Assim como o algoritmo SLV, a estratégia adotada pelo algoritmo MCF pode não produzir os melhores resultados. Considere novamente, a Figura 9.4, 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.

9.3.3 O algoritmo *Smallest Start-time First Ordering (SSF)*

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. O algoritmo SSF possui claramente complexidade de $O(nlog(n) + n|W|log(|S|))$.

A mesma situação de perdas apresentada na Figura 9.4 pode ocorrer no algoritmo SSF. A primeira requisição processada seria a requisição "A", o que resultaria nas perdas descritas.

9.3.4 O algoritmo *Largest Interval First Ordering (LIF)*

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 o SSF, o algoritmo LIF possui complexidade de $O(nlog(n) + n|W|log(|S|))$.

Assim como os algoritmos SLV, MCF e SSF, a situação descrita na Figura 9.4 pode ocasionar perdas se a requisição "A" for maior do 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.

9.3.5 Algoritmo Max-SS

Em [12], é proposto um algoritmo denominado Max-SS (*Max Stable Set Algorithm*). A idéia do algoritmo é encontrar o conjunto independente de cardinalidade máxima, garantindo, assim, que o maior número de requisições disjuntas será alocado.

O algoritmo usa uma busca em largura lexicográfica para obter um esquema de eliminação perfeito σ. A partir daí, uma sequência de vértices $v_1, ..., v_n$, formando um conjunto independente máximo é formada como segue: $v_1 = \sigma(1)$, v_i é o primeiro vértice em σ que segue v_{i-1} e que não está em $X_{v_1} \cup ... \cup X_{v_{i-1}}$, onde $X_v = \{x \in Adj(v) | \sigma(v) \prec \sigma(x)\}$.

Ao encontrar um conjunto independente de cardinalidade máxima, o algoritmo Max-SS garante que o maior número de requisições disjuntas será alocado. Entretanto, o algoritmo Max-SS possui algumas limitações. A primeira delas é que o conjunto independente máximo deve ser encontrado separadamente para cada um dos canais, o que eleva a sua complexidade computacional. A segunda limitação, é que ele não leva em consideração a presença de *voids*, fazendo com que a utilização dos enlaces seja baixa em relação aos demais algoritmos apresentados.

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. A seguir será discutido como esse problema pode ser resolvido de forma ótima com tempo de execução polinomial.

9.4 Escalonamento ótimo de canais em redes OBS

Nesta seção, o problema de escalonamento em lote de canais é discutido visando a obtenção de soluções ótimas. Primeiramente será exibida uma formulação em programação linear para o problema, apresentada em [6]. Após isso, será discutida uma mudança na modelagem do problema que permite a solução ótima do problema em tempo polinomial. Estas abordagens serão discutidas em seguida.

9.4.1 Formulação em programação linear inteira

Em [6], apresenta-se a seguinte formulação em programação linear inteira (PI) para o problema de *job scheduling*.

$$
\begin{array}{lrcll}
\max & \sum_{R \in \mathcal{R}} x_R \, w_R & & & \\
\text{r.a.} & Ax & \leq & e^T W & (1) \\
& x & \in & \{0,1\} & (2)
\end{array}
$$

onde w é um vetor de n colunas representando o valor das requisições, x é um vetor binário em que $x_j = 1$ indica que a requisição j será reservada, A é uma matriz de incidência clique-vértice m x n (m é o número de cliques maximais), W é o número de canais/máquinas e e^T é o vetor unitário de dimensão m.

Discute-se em [6], propriedades especiais do problema que permitem uma solução ótima em tempo polinomial. Partindo de tais propriedades, as seções 9.4.3 e 9.4.4 enfatizam a solução de diferentes versões do problema usando algoritmos cuja complexidade computacional é polinomial.

9.4.2 Remodelagem do problema de escalonamento em lote em redes OBS

Na seção 9.3, discutiu-se como o problema de escalonamento em lote de canais foi modelado como um problema de *job scheduling* com máquinas não idênticas [35]. O fato que justifica tal modelagem, como ilustrado na Figura 9.3, é que os períodos de disponibilidade dos canais de dados alternam-se com períodos em que os mesmos estão reservados, o que leva a um mapeamento direto entre os *voids* (no problema de escalonamento em lote) e as máquinas (no problema de *job scheduling*). Contudo, isso aumenta o número de restrições do problema, e consequentemente a complexidade computacional necessária à obtenção de uma solução ótima.

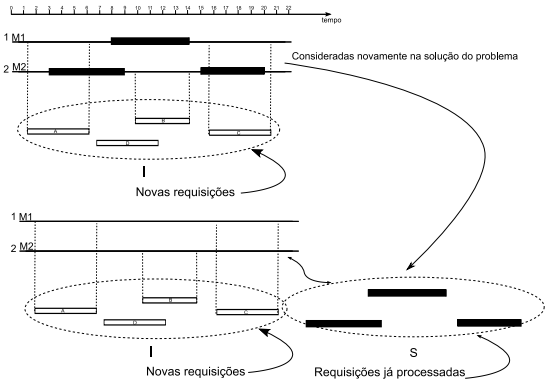


Figura 9.5: Escalonamento em lote de canais com máquinas idênticas.

Neste trabalho, uma nova forma de modelar o problema é proposta. Esta nova modelagem será usada pelos algoritmos de escalonamento em lote apresentados neste capítulo.

Nesta nova modelagem, ao invés de modelar cada *void* como uma máquina distinta, cada canal representa uma máquina. Assim, para garantir que todas as máquinas sejam idênticas, o conjunto S de reservas já efetuadas cujas rajadas ainda não chegaram, é considerado na obtenção da nova solução. Com isso, todos os canais estarão disponíveis para receber requisições em qualquer instante de tempo, fato que os torna com mesma capacidade e que, em última instância, permite a transformação do problema de escalonamento em lote de canais em redes OBS ao problema de *job scheduling* com máquinas idênticas.

Este processo é ilustrado na Figura 9.5, que mostra dois canais de dados usados para acomodar as requisições. Em um dado momento, o canal 1 está reservado no período $[8, 14]$ e o canal 2 nos períodos $[3, 9]$ e $[15, 20]$. Em seguida, chegam requisições de reservas A ($[1, 6]$), B ($[10, 14]$), C ($[16, 21]$) e D ($[7, 12]$). As requisições A, B, C e D são agrupadas em um lote, juntamente com as reservas previamente realizadas. Assim, ambos os canais podem ser usados para acomodar qualquer requisição, o que é determinado pela solução obtida pelo algoritmo de escalonamento.

Gerenciamento das reservas já realizadas

Desde que as reservas já realizadas são por vezes usadas na obtenção de novas soluções, é útil manter estruturas de dados adicionais para agilizar o processamento dos lotes. No caso dos algoritmos descritos a seguir, é preciso determinar de forma rápida quais das reservas já realizadas se sobrepõem às requisições do lote a ser processado. Para tal, para cada canal pode-se manter uma lista auxiliar de requisições ordenadas de forma decrescente em relação ao tempo de término. Assim, o tempo gasto será apenas da ordem de $O(|S|+|W|)$ para determinar as requisições desejadas. Caso a lista auxiliar seja armazenada na forma de um *heap* [16] ordenado pelo tempo de término, o tempo necessário para mantê-lo será da ordem de $O(|S| \log(|S|))$.

9.4.3 Algoritmo ótimo para requisições com pesos idênticos

Quando não existe nenhum tipo de diferenciação nos serviços oferecidos pela rede OBS, todas as requisições, e consequentemente, todas as rajadas possuem a mesma prioridade. Isso faz com que o peso associado a cada uma das requisições seja idêntico e tenham valor igual a p. Nesta seção será apresentado um algoritmo para escalonamento de canais quando todas as requisições da rede possuem pesos idênticos. Formalmente, tem-se um conjunto de W canais de dados, e uma lista $I = \{J_1 = (s_1, e_1, p), \ldots, J_n = (s_n, e_n, p)\}$ de n requisições (correspondente a um lote), onde (s_i, e_i, p) é o tempo de início, o tempo de fim da rajada e o peso da requisição 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. *Objetiva-se, neste problema, alocar o maior número possível de rajadas de* $I \cup S$ *nos* W *canais.*

Em [8] é apresentado um algoritmo para escalonamento de tarefas em um conjunto de W máquinas idênticas cuja complexidade computacional é de $O(n \max(\log(n), |W|))$. Este algoritmo, denominado aqui, GreedyOPT será utilizado para solucionar o problema de escalonamento de canais em redes OBS. A idéia fundamental do algoritmo baseia-se no Teorema 1.

Teorema 1. *Dado um conjunto* $I = \{J_1 = (s_1, e_1, 1), \ldots, J_n = (s_n, e_n, p)\}$ *com* n *requisições a serem escalonadas. Suponha que uma das requisições é encurtada, ou seja, seu tempo de início permanece inalterado, enquanto o tempo de término é antecipado. Se as demais requisições são inalteradas, então o número de requisições alocadas ou permanece inalterado ou é incrementado.*

Prova. A prova é baseada no fato de que quando uma requisição é encurtada, o número de requisições com as quais ela se intersecta, ou diminui, ou permanece inalterado. Seja $I^* \subseteq I$ um escalonamento e J_i a requisição "encurtada". Se J_i não foi inicialmente selecionada para o escalonamento, significa que existe um conjunto $C \subseteq I^*$ de requisições, com as quais J_i se intersecta. Quando J_i é encurtada, ela pode deixar de se intersectar

com as requisições de C fazendo com que possa ser adicionada a I^*, aumentando assim a quantidade de requisições escalonadas. Note que a inclusão de J_i é incapaz de diminuir a quantidade de requisições já escalonadas. Isso por que, J_i só é adicionada ao escalonamento se não mais possui intersecção com as requisições de C. Além disso, como mencionado, ao ser encurtada J_i, o número de requisições com as quais ela se intersecta não aumenta, deixando assim as demais requisições de I^* inalteradas. Se, por outro lado, quando J_i for encurtada ainda persistirem intersecções com algumas requisições de C, a requisição J_i não é adicionada ao escalonamento I^*, deixando-o inalterado.

Um raciocínio semelhante pode ser usado para o caso de quando J_i faz parte do escalonamento. Seja J_l uma requisição não pertencente ao escalonamento I^* e assuma que a única requisição pertencente a I^* com a qual J_l se intersecta é a requisição J_i. Se quando J_i for encolhida, a intersecção com J_l não mais existir, J_l poderá ser adicionada ao escalonamento, incrementando, assim, I^*. Observe que a inclusão de J_l não afeta as demais requisições de I^*, já que J_l só possui intersecção com J_i. Caso J_l possua com outras requisições em I^*, ela não poderá ser adicionada ao escalonamento, deixando-o intacto. Além disso, se após J_i ser encolhida, a intersecção entre J_i e J_l persistir, J_l não poderá ser adicionada ao escalonamento, deixando-o intacto. □

A idéia do algoritmo é processar sequencialmente as requisições, tentando acomodá-las em um dos $|W|$ canais. Caso a requisição não possa ser acomodada, o algoritmo tenta substitui-la por alguma das requisições já acomodadas cujo tempo de término seja o maior. O algoritmo GreedyOPT é apresentado no Algoritmo 7.

O algoritmo GreedyOPT funciona da seguinte forma: na linha 2 as requisições são ordenadas em ordem crescente de tempo de início. Na linha 3, a requisição em processamento é adicionada ao conjunto de requisições que pertencem à solução e na linha 4 verifica-se se a requisição pode ser acomodada em um dos canais. Caso a requisição não possa ser acomodada em um dos canais, a requsição com o maior tempo de término é removida. Na prática, o que o algoritmo faz, caso a requisição não possa ser acomodada em um dos canais, é tentar substituir a requisição com maior tempo de término pela

Algoritmo 7 GreedyOPT

ENTRADA

Um conjunto W de 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 cardinalidade máxima.

GreedyOPT

1: $N \leftarrow |I| + |S|$
2: Ordene todas as requisições de $I \cup S$ de acordo com o tempo de início de forma que $s_1 \leq s_2 \leq ... \leq s_N$.
3: Considere as requisições sequencialmente, adicionando-as ao conjunto I'
4: Caso não haja canal para acomodar a última requisição, remova de I' a requisição com o maior tempo de término.

requisição em processamento.

O algoritmo GreedyOPT resolve otimamente o problema de escalonamento em lote de canais em redes OBS quando todas as requisições sendo processadas na rede possuem peso unitário, como se pode constatar através do Teorema 2

Teorema 2. *Seja t_0 o menor tempo em que mais que $|W|$ requisições se intersectam, seja J o conjunto dessas requisições e $e_k = max_{j \in J} e_j$. Então, existe um escalonamento ótimo que não inclui k.*

Prova. *

Se no instante de tempo t_0 existem mais do que $|W|$ requisições se intersectadas, claramente, pelo menos uma das requisições não poderá ser alocada. A escolha de qual requisição será descartada depende só do futuro ($t > t_0$), desde que todas as requisições que iniciaram antes do instante de tempo t_0, ou já terminaram, ou pertencem ao conjunto J. Assim, o tempo de início de cada requisição em J pode ser redefinido como sendo t_0. Suponha que alguma requisição i é removida ao invés da requisição k. Ao trocar a requisição k pela requisição i, o que acontece é que uma requisição mais longa é substituída por uma requisição mais curta; em outras palavras, uma requsição é encurtada.

*A prova deste teorema pode ser encontrada em [8] e será reproduzida aqui por questões de completude.

Assim, pelo Teorema 1, incrementa-se o número de requisições no escalonamento, ou o escalonamento permanece intacto. □

É importante nesse ponto mencionar que as requisições pertencentes ao conjunto S não possuem nenhum tipo de garantia de que permanecerão nas novas soluções obtidas. Contudo, dado que o algoritmo é ótimo, as requisições de S só deixarão de ser alocadas para dar lugar a um novo conjunto de requisições com cardinalidade máxima. Além disso, já que as redes OBS usam esquema de reserva de recursos em uma via, o fato de alguma requisição de S ser descartada não gera nenhum tipo de mensagem adicional aos nós de origem dos dados.

Complexidade computacional

Teorema 3. *Seja $|W|$ o número de canais de saída de um nó OBS, n o número de requisições a serem processadas em um lote, S o conjunto de requisições já alocadas cujo período de tempo se sobrepõe às requisições do lote e $s = |S|$, a cardinalidade do conjunto S. A complexidade computacional do algoritmo GreedyOPT é de $O(N\log(N) + N|W|) = O(N\max(\log(N), |W|))$, onde $N = n + s$.*

Prova.

Como descrito na seção 9.4.2, demanda-se $O(s\log(s))$ para determinar quais as reservas ainda não utilizadas devem ser reprocessadas. Demanda-se também $O(N\log(N))$ para realizar a ordenação das requisições do lote, bem como $O(N|W|)$ para realizar o teste de existência de canal para acomodar a nova requisição realizado na linha 4. Assim, a complexidade do GreedyOPT é de $O(s\log(s) + N\log(N) + N|W|)$. Dado que $s\log(s) \leq N\log(N)$, fazendo $s = N$ tem-se que a complexidade do algoritmo é de $O(2N\log(N) + N|W|) = O(N\log(N) + N|W|) = O(N(\log(N) + |W|))$. Fazendo-se a complexidade em função do termo que majora a soma $(\log(N) + |W|)$, tem-se $O(N\max(\log(N), |W|))$,

□

9.4.4 Algoritmo ótimo para requisições com pesos não idênticos

Quando o tráfego da rede requer diferenciação ou existe presença de mecanismos de provisão de QoS, os pacotes de controle da rede são marcados com a prioridade com que devem ser tratados nos nós ao longo do caminho por onde trafegarão. O valores com que são marcados os pacotes de controle, denominados pesos, refletem a sua importância e a sua precedência em relação à alocação dos recursos em caso de escassez dos mesmos.

Nesta subseção, será apresentado um algoritmo ótimo para escalonamento em lote de canais para o caso em que as requisições da rede possuem pesos com valores não unitários. Na estratégia proposta, garante-se que o conjunto S, de requisições previamente alocadas, é considerado no processamento do lote atual, de tal forma que necessariamente as requisições em S sejam alocadas. Tem-se, como pretendido, um mapeamento para o problema de *job scheduling* com máquinas idênticas e garante-se que as requisições de S permanecerão na nova solução, fazendo com que o tempo necessário à reserva dos recursos e ajuste da malha de comutação para tais requisições seja economizado.

Tem-se, formalmente, o seguinte problema: seja $I = \{J_1 = (s_1, e_1, p_1), \ldots, J_n = (s_n, e_n, p_n)\}$ uma lista de tarefas, onde (s_i, e_i) é o tempo de início e fim da tarefa J_i, e p_i o seu peso. O algoritmo recebe um conjunto de W máquinas idênticas e também uma lista S de tarefas já alocadas nas máquinas de W. O objetivo do problema é selecionar uma sub-lista $I' \subseteq I$ de tarefas de peso máximo, e alocar I' nas $|W|$ máquinas. O escalonamento gerado deve satisfazer o fato de que em cada máquina não pode existir sobreposição de tempo entre as tarefas.

Em [6], 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 p_i e intervalo (s_i, e_i). O algoritmo considera o grafo de intervalos dado pelos intervalos correspondentes às tarefas. Depois, o algoritmo ordena todas as cliques maximais $C_1, \ldots, C_r$, do grafo de intervalo, de

acordo com o tempo. Na Figura 9.6 apresenta-se um exemplo desta primeira construção.

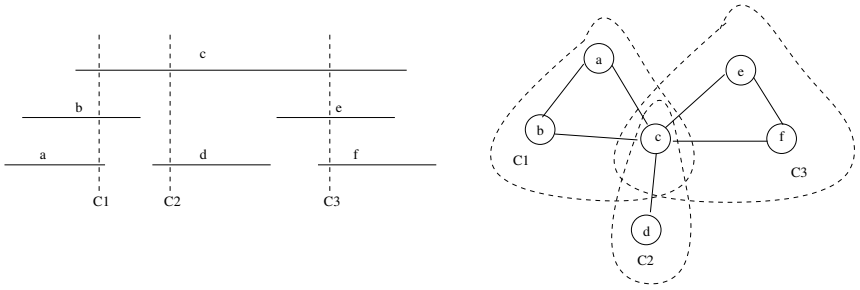


Figura 9.6: **Requisições são representadas na esquerda. Na direita o grafo de intervalos correspondente e as suas cliques subjacentes.**

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 p_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 $M - |W|$ 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 9.6 é dado na Figura 9.7. Os arcos partindo de v_0 em direção a v_r representam as requisições.

Teorema 4. *O algoritmo de Arkin e Silverberg [6] resolve otimamente o problema de job scheduling com máquinas idênticas.*

Prova. Uma prova da otimalidade do algoritmo pode ser encontrada em [6]. Entretanto, por razões de completude será apresentada aqui.

O objetivo do problema é alocar um conjunto de tarefas de peso máximo em $|W|$ máquinas. Logo, tem-se a condição de que a qualquer instante, no máximo $|W|$ tarefas

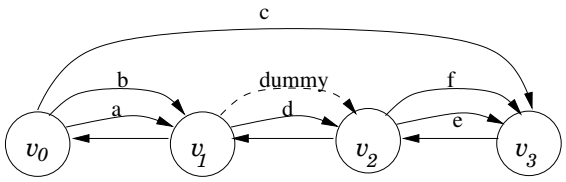


Figura 9.7: Exemplo de Rede de fluxo.

podem estar em processamento. Esta condição é a mesma que impõe que em cada uma das cliques maximais no máximo $|W|$ tarefas devem ser escolhidas, portanto, pretende-se remover as tarefas de peso mínimo que não podem ser alocadas. Assim, em cada clique maximal C_i, deve-se assegurar que $|C_i| - |W|$ tarefas são removidas.

A rede de fluxos G' é construída, de tal forma que sobre cada vértice v_i é possível passar um fluxo de M unidades (Neste caso, passar um fluxo não significa somente transitar diretamente por v_i pois uma aresta correspondente a uma tarefa que pertence a clique C_i também pode passar de um vértice anterior a v_i até um vértice posterior a v_i, como por exemplo a tarefa c e a clique C_2). É possível passar o fluxo de M unidades sobre cada vértice v_i, pois acrescenta-se uma aresta (v_{i-1}, v_i) com custo 0 e capacidade $M - |C_i|$ para cada clique maximal C_i.

O algoritmo acha um fluxo de $M - |W|$ unidades e custo mínimo de v_0 para v_r. Em cada vértice v_i de G', deve-se passar um fluxo de $M - |W|$ unidades. Como a princípio, pode-se passar um fluxo de M unidades sobre cada um dos vértices v_i, e está passando um fluxo de $M - |W|$ unidades, tem-se, então, que $|W|$ unidades de fluxo não estão sendo usadas. Estas $|W|$ unidades correspondem às tarefas (arestas de capacidade 1) que serão alocadas, e em cada clique C_i haverá, consequentemente, no máximo, $|W|$ tarefas alocadas. Como o fluxo encontrado pelo algoritmo é de custo mínimo, as arestas não usadas correspondem a um conjunto de tarefas de peso máximo.

□

Deseja-se agora adaptar o algoritmo de Arkin e Silverberg para que o conjunto S de requisições previamente alocadas seja considerado e, mais ainda, esteja necessariamente na

nova solução obtida. 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 custo é infinito. Desta forma, pode-se garantir que as requisições em S permanecerão alocadas. O algoritmo BATCHOPT é formalmente apresentado no Algoritmo 8.

Algoritmo 8 BATCHOPT

ENTRADA

Um conjunto W 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 9.4.
5: Calcule o fluxo de $M - |W|$ unidades que possua custo mínimo.
6: Aloque todas as requisições cujos arcos representados em G' não receberam fluxo.

Por ser uma extensão do algoritmo de Arkin e Silverberg, o algoritmo BATCHOPT é capaz de resolver o problema de escalonamento em lote em redes OBS de forma ótima, como é provado no seguinte teorema:

Teorema 5. *O algoritmo BATCHOPT resolve otimamente o problema de escalonamento em lote em redes OBS*

Prova.

Será provada, primeiramente, a otimalidade do algoritmo. Como para cada requisição pertencente a S atribui-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, pelo resultado do Teorema 4, 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 trânsito 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, $|W|$ requisições de $I' \cup S$ que se intersectam no tempo 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 $|W|$ 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.

□

9.4.5 Complexidade computacional

Teorema 6. *Seja n o número de reservas a serem processadas no lote, S o conjunto de reservas já processadas cujo período se intersecta com as reservas do lote, $s = |S|$, o algoritmo BATCHOPT possui complexidade de $O(N^2 log(N) + N)$, onde $N = n + s$.*

Prova.

Como descrito na seção 9.4.2, gasta-se $O(s \log(s))$ para determinar o conjunto S, caso tal conjunto seja armazenado em um *heap*. Além disso, o algoritmo de Arkin e Silverberg possui complexidade $O(N^2 \log(N)$ [6], o que dá uma complexidade de $O(s \log(s) + N^2 \log(N))$. Dado que $s \log(s) \leq N \log(N)$, fazendo-se $s = N$ tem-se uma complexidade de $O((N^2 + N) \log(N))$. Como $N^2 + N = O(N^2)$ tem-se uma complexidade de

$O(N^2 \log(N))$. Após determinar quais requisições deverão ser alocadas, é possível realizar a alocação das requisições em tempo de $O(N)$, usando algoritmo simples para coloração de grafos de intervalos [52]. Assim, a complexidade do algoritmo BATCHOPT é de $O(N^2 \log(N) + N)$. □

Diferentemente do algoritmo GreedyOPT, o algoritmo BATCHOPT garante que as requisições do conjunto S estarão na nova solução. Apesar de não garantir que o novo escalonamento seja máximo caso os pesos originais das requisições de S sejam utilizados, existem algumas vantagens ao garantir que as requisições de S permaneçam alocadas. Uma delas é que o tempo gasto com a insercção na lista de reservas já realizadas torna-se menor, dado que as requisições de S já estão inseridas. Além disso, ao garantir que reservas já realizadas permaneçam inalteradas, minimiza-se um efeito cascata em que cada nova requisição pode bloquear uma requisição existente, levando a rede a um estado de volatilidade em que nenhuma requisição pode ser efetivamente atendida.

9.4.6 Evitando o reprocessamento das requisições já alocadas

Uma limitação da estratégia proposta para a redução do problema de escalonamento em lote em redes OBS ao problema de *job scheduling* com máquinas idênticas é o fato de que as requisições já alocadas são reprocessadas, o que pode aumentar o tempo de processamento das requisições no lote. Além disso, no caso específico do algoritmo GreedyOPT, não existe nenhuma garantia de que requisições que já foram reservadas no passado permaneçam assim até a chegada da rajada correspondente.

Uma forma de contornar tais problemas, seria adiar o processamento das requisições cujo período pode se intersectar com o período das requisições de lotes futuros, fazendo, dessa forma, com que as primeiras sejam reprocessadas. O raciocínio por trás de tal estratégia, é que as requisições que necessitam ser reprocessadas quando foram reservadas pela primeira vez, o foram com uma grande antecedência em relação à chegada da rajada correspondente (em outras palavras, possuíam tempo de ajuste alto). Assim, as requisições que chegaram para compor lotes futuros e que não possuem tempo de ajuste

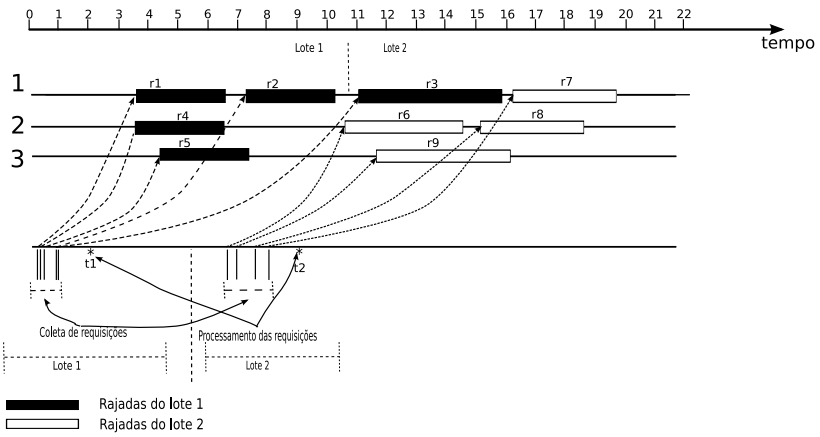


Figura 9.8: Processamento antecipado de requisições.

tão alto têm seus períodos de duração intersectando com as requisições já reservadas.

A Figura 9.8 ilustra a idéia. Nela, vêem-se dois lotes de requisições. O lote 1 é composto pelas requisições r1, r2, r3, r4 e r5, ao passo que o lote 2 é composto pelas requisições r6, r7, r8 e r9. Como pode ser visto, a requisição $r3$ é processada juntamente com as demais requisições de seu lote, no instante $t1 = 2$. Com a chegada das requisições do lote 2, vê-se que o canal 1 não pode ser usado no período de $[11, 16]$, pois o mesmo está reservado para a rajada correspondente à requisição $r3$. Dessa forma, a requisição $r3$ deve ser reprocessada. Entretanto, se no instante de processamento do lote 1, a decisão de adiar o processamento da requisição $r3$ tivesse sido tomada, ela poderia ter sido processada uma única vez juntamente com as requisições do lote 2, no instante de tempo $t2 = 9$.

Uma forma de inferir sobre quando adiar o processamento de uma requisição pode ser feita através da probabilidade de inversão discutida no Capítulo 8. Como visto, a inversão ocorre quando a ordem de chegada de duas rajadas é inversa à ordem de chegada de seus pacotes de controle e ela ocorre quando o tempo de ajuste da segunda requisição é menor do que o tempo de ajuste da primeira. Isso quer dizer que se o processamento de requisições com tempo de ajuste alto for adiado, existe uma alta probabilidade de que outras requisições cheguem no futuro para formar um lote com a requisição adiada. Com isso, para diminuir o reprocessamento de requisições, basta adiar o processamento

de todas as requisições com probabilidade de inversão próxima a 1.

Obviamente, existe a necessidade de cálculo da probabilidade de inversão para saber quando adiar o processamento de uma requisição, o que exige processamento. A complexidade do cálculo da probabilidade de inversão em um nó i, depende do número de pares origem-destino cujas rotas passam por i. Assim, existe um compromisso entre realizar o reprocessamento das requisições e realizar o cálculo da probabilidade de inversão para determinar se uma requisição deve ou não ser processada no lote atual. Se o número de pares origem-destino que usam o nó i em sua rota for pequeno, pode ser vantajoso realizar o cálculo da probabilidade de inversão. Caso contrário, pode ser mais vantajoso realizar o reprocessamento das requisições.

9.5 Estratégia de formação de lote e adaptação do protocolo de reserva

Como mencionado, em [35] 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 9.3 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.

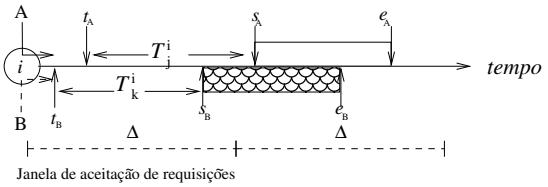


Figura 9.9: Problema do uso da janela de aceitação de rajadas fixa.

A Figura 9.9 ilustra o problema descrito. 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 ao 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. Descarta-se, portanto, 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 9.10. Além disso, o instante de tempo em que se encerra o período de acomodação de requisições e se inicia o seu processamento (denominado limiar de processamento) é definido de forma que este não 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 e seja $T_L^{(p)}$, o tempo de processamento do lote. O limiar de processamento (L) é definido como o instante de tempo em que se inicia o processamento do lote. Nesse ponto, é importante discutir o papel do nó de

ingresso da rede OBS na determinação do limiar de processamento. O tempo de ajuste é função do tempo de processamento do pacote de controle nos nós da rota por onde a rajada trafegará. Assim, se o nó de ingresso possui conhecimento sobre o tipo de algoritmo de escalonamento empregado no núcleo da rede, e se o tempo de processamento do lote é conhecido, então, o tempo de ajuste determinado no nó de ingresso será:

$$T = \left(\sum_{i} T_L^{(p)} \right) + T_d^{(p)} + T_d^{(s)} \tag{9.1}$$

fazendo com que $L = (t_R + \Delta)$, onde $R = min_r\{t_r + T_j^i + \Delta\}$. Note que a requisição que determina o instante do processamento é aquela, dentre as requisições do lote, cuja rajada chegará no instante de tempo mais próximo.

Por outro lado, caso diferentes algoritmos de escalonamento sejam implementados nos nós de núcleo, torna-se mais difícil determinar o tempo de processamento a partir do nó de ingresso. Assim, o tempo de ajuste calculado no nó de ingresso pode conter somente o tempo necessário para o processamento de um único pacote de controle ($T^{(p)}$) em cada nó, somado à janela de aceitação de requisições. Logo, é preciso que o tempo extra necessário para o processamento de todo o lote seja diminuído do tempo de coleta de requisições (Δ) da requisição que determina o limiar de processamento. Dessa maneira, o limiar de processamento do lote é definido como $L = (t_R + \Delta) - T_L^{(p)}$. A fração de tempo diminuída da janela de aceitação de requisições equivale a $T_L^{(P)} - T^{(P)}$, já que o tempo de processamento de um único pacote de controle deve ser descontado.

À medida em que cresce o número de requisições do lote, o tempo de processsamento do lote também aumenta, diminuindo, assim, o tempo destinado à coleta de novas requisições. Para evitar que o tempo de processamento do lote cresça indefinidamente ao ponto do nó não conseguir processar o lote antes da chegada das rajadas, um tempo máximo para o processamento do lote $T_{max}^{(p)}$ é determinado. Assim, uma rajada só é inserida no lote se $T_L^{(p)}(n) \leq T_{max}^{(p)}$, onde $T^{(p)}(n)$ é o tempo de processamento de um lote com n requisições e pode ser estimado, por exemplo, pela função que representa a complexidade do algoritmo

no pior caso. Na prática, porém, pode-se determinar um número máximo de requisições que o lote deve conter, de forma que $T_{max}^{(p)}$ seja limitado.

É importante, porém destacar, que 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 [60]. Assim, o tempo de processamento do lote é quase desprezível, fazendo com que o limiar de processamento seja dominado pela janela de aceitação. É importante também 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.

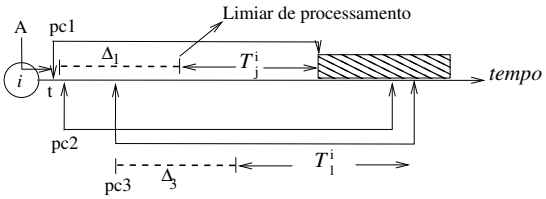


Figura 9.10: Estratégia de alocação de lote JET-Δ.

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 [21], 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 as 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 no 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:

$$\Delta = \frac{D - T_j^s}{H} \tag{9.2}$$

Uma forma de evitar o aumento do atraso fim-a-fim, é usar esquemas de montagem com predição do tamanho da rajada, como em [49] e em [42]. 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 [65, 72–74, 76], 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.

9.6 Exemplos Numéricos

Esta seção compara os algoritmos propostos com outros algoritmos da literatura. Devido ao fato de ser proposto para redes OBS com apenas um canal de dados, o algoritmo MAX-SS não foi usado nas comparações. Para avaliar o desempenho dos algoritmos, foram realizadas simulações usando a ferramenta OB2S (*Optical Burst Switching Simulator*) desenvolvido na Universidade Salvador [44]. Os algoritmos foram implementados em linguagem C, usando o conjunto de bibliotecas disponíveis em [61].

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 e os intervalos de confiança possuem um nível de confiança de 95%.

As simulações foram executadas em dois cenários distintos. No primeiro cenário, reproduz-se ambiente proposto em [35]. No segundo cenário, avalia-se as diferentes es-

tratégias de formação de lote e o desempenho dos algoritmos em topologias de redes reais com um conjunto de parâmetros mais realista. Os resultados são apresentados nas subseções a seguir.

9.6.1 Simulações em topologia com nó OBS concentrador

Como dito, neste primeiro conjunto de simulações verifica-se o comportamento dos algoritmos propostos em um cenário controlado com poucos parâmetros de ajuste, além de comparar os resultados obtidos com aqueles já publlicados em [35].

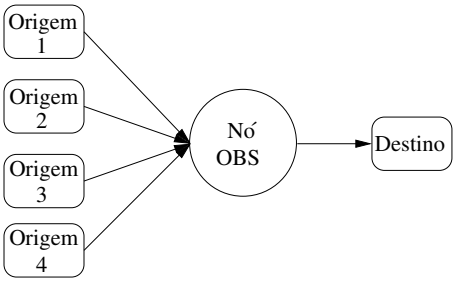


Figura 9.11: Topologia usada no primeiro cenário de simulação.

A Figura 9.11 apresenta a topologia usada no primeiro ambiente de simulação. Ela consiste de um único nó OBS de núcleo conectado a diversas fontes de tráfego e um único destino. Cada enlace de entrada possui dois comprimentos de onda separados (um para dados e um para controle). O enlace ligando o nó OBS de núcleo ao destino possui cinco comprimentos de onda (quatro para dados e um para controle) e cada um dos comprimentos de onda utilizados possui capacidade de 2.5 Gbps (OC-48).

Para estar em conformidade com os resultados discutidos em [35], foi definida a constante τ como o tempo de transmissão de 1024 bits em um dos comprimentos de onda, ou seja, $\tau = \frac{1024}{2377728000} = 4.3e-7$ segundos. As rajadas são geradas pelas fontes de tráfego de acordo com uma distribuição de *Poisson* com tamanho médio das rajadas $\overline{b} = 81920$ bits. O tempo de ajuste é gerado de acordo com uma distribuição uniforme sobre o intervalo $[130\tau, 150\tau]$. A janela de aceitação de requisições possuiu um valor arbitrário de 100τ, o que dá aproximadamente $40\mu sec$.

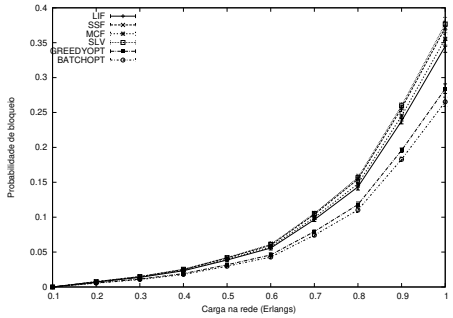


Figura 9.12: Probabilidade de bloqueio experimentada pelos algoritmos.

A Figura 9.12 mostra a probabilidade de bloqueio experimentada pelos algoritmos. É possível ver claramente que o algoritmo com menor probabilidade de bloqueio é do BATCHOPT, seguido pelo GreedyOPT. Apesar de tanto o algoritmo GreedyOPT quanto o algoritmo BATCHOPT alocarem de forma ótima as requisições que compõem cada lote, o fato do algoritmo GreedyOPT não garantir que as requisições reprocessadas continuem alocadas aumenta a sua probabilidade de bloqueio, já que tais requisições podem desperdiçar recursos que poderiam ter sido ocupados por outras requisições.

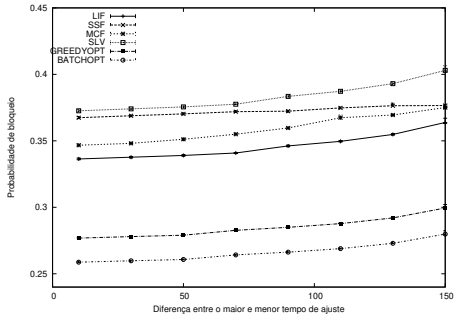


Figura 9.13: Efeito da diferença do tempo de ajuste na probabilidade de bloqueio experimentada pelos algoritmos.

Foi avaliado, também, o efeito da diferença do tempo de ajuste na probabilidade de bloqueio. A diferença do tempo de ajuste foi definida como a diferença entre o maior e o menor tempo de ajuste gerados aleatoriamente, isto é, $T_{max} - T_{min}$. O valor de T_{max} foi de 200τ e o valor de T_{min} foi gradativamente aumentado, diminuindo assim $T_{max} - T_{min}$. Neste caso, a carga de tráfego na rede foi de 99% da capacidade do enlace.

A Figura 9.13 mostra que, de um modo geral, assim como observado nos experimentos em [35], à medida em que se aumenta a diferença do tempo de ajuste, aumenta-se a probabilidade de bloqueio. É possível observar que a probabilidade de bloqueio experimentada pelo algoritmo BATCHOPT é inferior aos demais algoritmos, seguido pelo algoritmo GreedyOPT. O incremento da probabilidade de bloqueio à medida em que se aumenta a diferença entre os tempos de ajuste máximo e mínimo é um fenômeno comum em redes OBS que operam com o protocolo JET. Tal fenômeno é denominado *retro-blocking* [36] e acontece devido ao uso de reserva retardada no protocolo JET. Assim, uma reserva, r_i, pode ser bloqueada por outra reserva, r_{i+1}, cujo tempo de início é anterior ao início da reserva r_i.

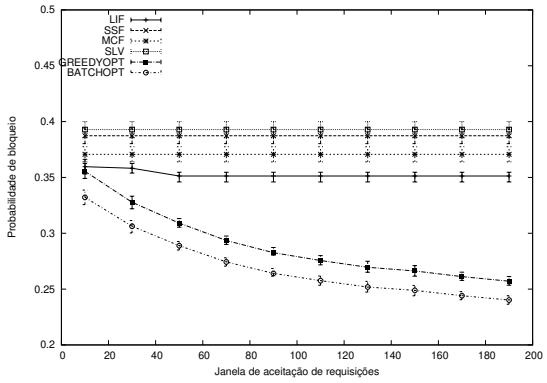


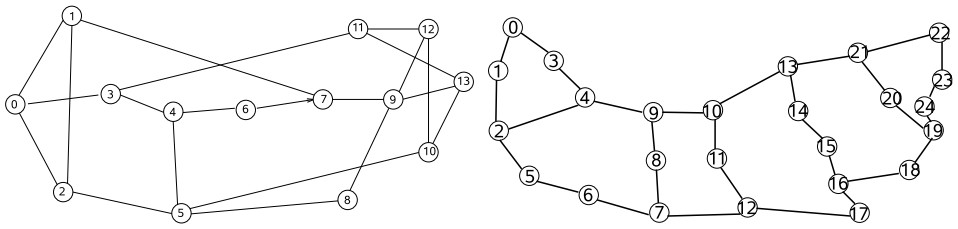
Figura 9.14: Efeito da janela de aceitação de requisições na probabilidade de bloqueio experimentada pelos algoritmos.

A Figura 9.14 mostra a probabilidade de bloqueio em função da janela de aceitação de requisições. Nesta avaliação, a diferença entre o maior e o menor tempo de ajuste foi de 50τ, a carga na rede foi de 99% da capacidade do enlace e a janela de aceitação de requisição variou entre 10τ e 190τ. É possível perceber que à excessão dos algoritmos GreedyOPT e BATCHOPT, a probabilidade de bloqueio experimentada pelos algoritmos não depende do aumento da janela de aceitação de requisições. Na realidade, no caso dos algoritmos LIF, SSF, MCF e SLV, o sucesso ao alocar as requisições do lote depende muito mais do padrão de início e término das requisições do lote do que da quantidade de requisições.

No caso dos algoritmos GreedyOPT e BATCHOPT, a probabilidade de bloqueio diminui com o aumento da janela de aceitação de requisições, pois à medida em que se aumenta a jenela de aceitação de requisições, o número de requisições compondo cada lote aumenta. Como os algoritmos são ótimos, conseguem sempre alocar o número máximo de requisições em cada lote, o que, ao final dos experimentos, resulta em uma probabilidade de bloqueio mais baixa.

9.6.2 Simulações com topologias de redes operacionais

Neste cenário, as simulações foram realizadas com as topologias das redes NSFNet e Abilene, apresentadas na Figura 9.15. Cada enlace da rede é constituído por uma fibra com 32 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 $50\mu s$. Assumiu-se que os nós de borda não possuem conhecimento sobre o tipo de algoritmo de escalonamento de canais usado no núcleo da rede. Assim, todos os pacotes de controle enviados ao núcleo da rede possuiam o tempo de ajuste baseados no tempo de processamento de um único pacote de controle em cada nó, somado à janela de aceitação de requisições. Além disso, assumiu-se também que o tempo de processamento do lote cresce linearmente à medida em que novas requisições são adicionadas ao lote.



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

Figura 9.15: Topologias usadas nas simulações.

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

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.

Avaliação de desempenho das estratégias de formação de lote

Nos experimentos realizados na seção 9.6.1, ficou claro que o algoritmo BATCHOPT produz os melhores resultados em termos de probabilidade de bloqueio, entretanto, como mencionado, em [35], a estratégia de formação de lote utilizada é de processar todas as requisições que chegam dentro de intervalos de tempo fixo. Contudo, como discutido na seção 9.5 esta estratégia pode ser ineficiente. Assim, para avaliar os benefícios da estratégia de formação de lote proposta neste trabalho foram realizadas outras simulações.

Como argumentado anteriormente, o dimensionamento da janela de aceitação torna-se mais crítico em topologias complexas com diversos pares origem-destino. Assim, o desempenho das estratégias de formação de lote foi avaliado usando tais topologias.

O algoritmo de escalonamento usado nestas avaliações foi o BATCHOPT, já que o mesmo apresentou os melhores resultados na avaliação realizada na seção 9.6.1. O BATCHOPT foi utilizado com a estratégia JET-Δ proposta nesse trabalho e também com a estatégia, denominada aqui de FIXA, proposta em [35].

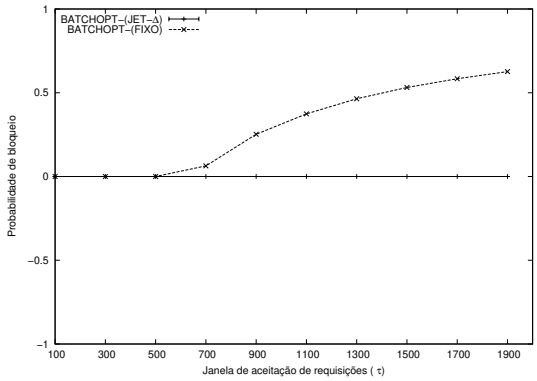


Figura 9.16: Probabilidade de bloqueio do BATCHOPT com diferentes estratégias de formação de lote.

A Figura 9.16 apresenta a probabilidade de bloqueio em função do tamanho da janela

de aceitação de requisições. Percebe-se que enquanto a janela de aceitação é menor do que o menor tempo de ajuste, a probabilidade de bloqueio do BATCHOPT é igual independente da estratégia adotada. Contudo, à medida em que a janela torna-se maior do que o menor tempo de ajuste, o desempenho da estratégia FIXA degrada-se. Isto acontece porque como a janela de aceitação é maior do que o menor tempo de ajuste presente na rede, as rajadas correspondentes às requisições do lote começam a chegar aos nós da rede antes que a respectiva requisição tenha sido processada, fazendo com que as rajadas sejam sumariamente descartadas.

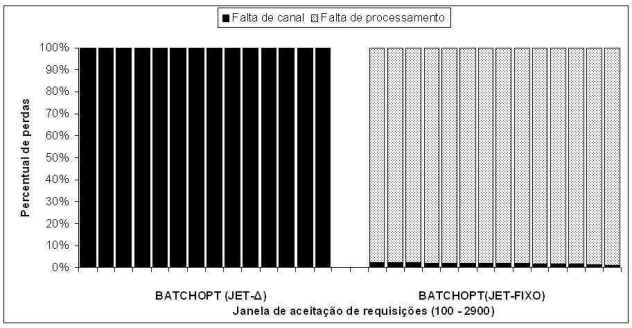
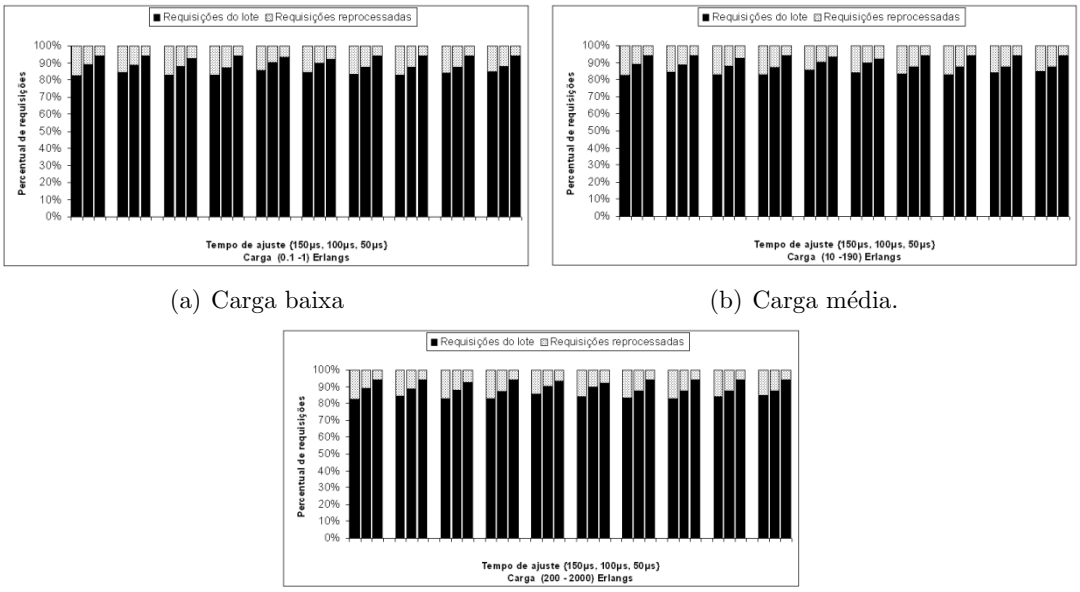


Figura 9.17: Percentual de perdas distribuído entre requisições não processadas e perdas por falta de canal de dados.

A Figura 9.17 apresenta o percentual de perdas de rajadas dividido entre as requisições que não foram processadas e aquelas descartadas por falta de canal para acomodá-las. Esta medida é feita em função da janela de aceitação de requsições. A janela de aceitação de requisições variou de 100τ a 2900τ, com incremento de 200τ. É possível observar que quando a estratégia JET-Δ é utilizada, todas as perdas devem-se somente a falta de canais para acomodá-la. Por outro lado, quando a estratégia FIXA é utilizada, cerca de 95% das perdas se dá por falta de processamento das requisições. Percebe-se também que à medida em que a janela de aceitação de requisições é aumentada, o percentual de perdas devido à falta de processamento aumenta suavemente. Isto acontece pois a janela de aceitação de requisições torna-se maior do que o tempo de ajuste. Com isso, as rajadas correspondentes às requisições do lote chegam ao nó antes mesmo que haja

o processamento do lote. À medida em que aumenta-se a janela, maior o percentual de requisições que sofre com esse fenômeno.



(a) Carga baixa

(b) Carga média.

(c) Carga alta.

Figura 9.18: Percentual das requisições reprocessadas em cada lote.

Dado que os algoritmos ótimos apresentados nesse capítulo adicionam requisições já processadas cujas rajadas ainda não foram transmitidas ao lote em processamento, um experimento para medir o percentual de requisições reprocessadas em cada lote foi realizado. A estratégia de formação de lote JET-Δ foi usada com janela de aceitação de requisições igual a 1ms. Os resultados reportados foram calculados tomando-se a média aritimética em todos os nós da rede.

A Figura 9.18 mostra os resultados do percentual de reprocessamento dividido entre as requisições pertencentes ao lote (aquelas processadas pela primeira vez) e as requisições reprocessadas. Os resultados são apresentados em função do tempo de ajuste contido nas requisições (sem levar em consideração a janela de aceitação de requisições). As barras verticais são dispostas em grupos de três. Cada barra dentro de um grupo representa o

percentual de reprocessamento de requisições contendo um tempo de ajuste dentre $50\mu s$, $100\mu s$ e $150\mu s$. Cada grupo representa uma carga da rede. As simulações foram realizadas com três cenários de carga de tráfego: carga baixa, variando de 0.1 a 1 Erlangs com incremento de 0.1 Erlangs; carga média, variando de 10 a 190 Erlangs, com incremento de 20 Erlangs; e carga alta, variando de 200 a 2000 Erlangs, com incremento de 200 Erlangs.

É possível perceber que, estatísticamente, não existe alteração no percentual de requisições reprocessadas em relação às requisições do lote (Figuras 9.18(a), 9.18(b) e 9.18(c)). Isto acontece pois à medida em que aumenta-se a carga na rede, mantendo-se inalterados os tempos de ajuste, aumenta-se proporcionalmente o número de requisições em cada lote e o número de requisições já processadas a espera da respectiva rajada. É possível também observar que o percentual de requisições reprocessadas aumenta à medida em que se aumenta o tempo de ajuste contido nas requisições. Isto acontece pois quanto maior o tempo de ajuste, mas tempo se passa entre o instante de término do processamento da requisição e a transmissão da rajada correspondente. Assim, a reserva fica mais tempo armazenada, permitindo, assim, que cheguem outras requisições e outros lotes sejam formados.

Avaliação de desempenho dos algoritmos de escalonamento em lote

Como visto na seção 9.6.2, os algoritmos de escalonamento apresentados em [35] utilizam uma estratégia de formação de lote ineficiente. Para avaliar o desempenho dos algoritmos, todos os algoritmos avaliados usaram 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 custo unitário. Em tais simulações, a carga de tráfego na rede foi de 1000 Erlangs.

A Figura 9.19 mostra a probabilidade de bloqueio obtida pelos algoritmos em função de Δ e a Figura 9.20 o número médio de requisições por lote em função de Δ. Percebe-se que os algoritmos LIF, MCF, SSF e SLV não são sensíveis a alterações na janela de aceitações.

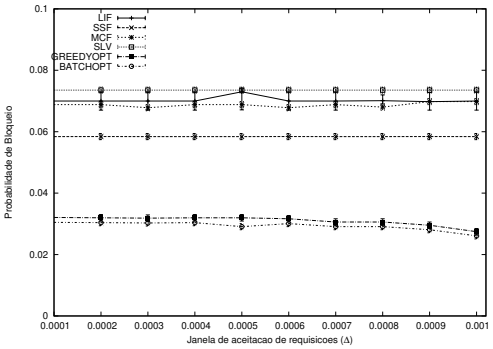


Figura 9.19: Probabilidade de bloqueio em função da janela da aceitação de requisições.

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.

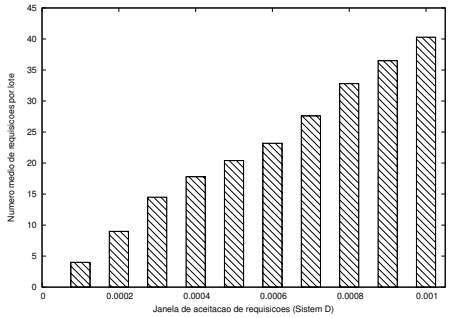


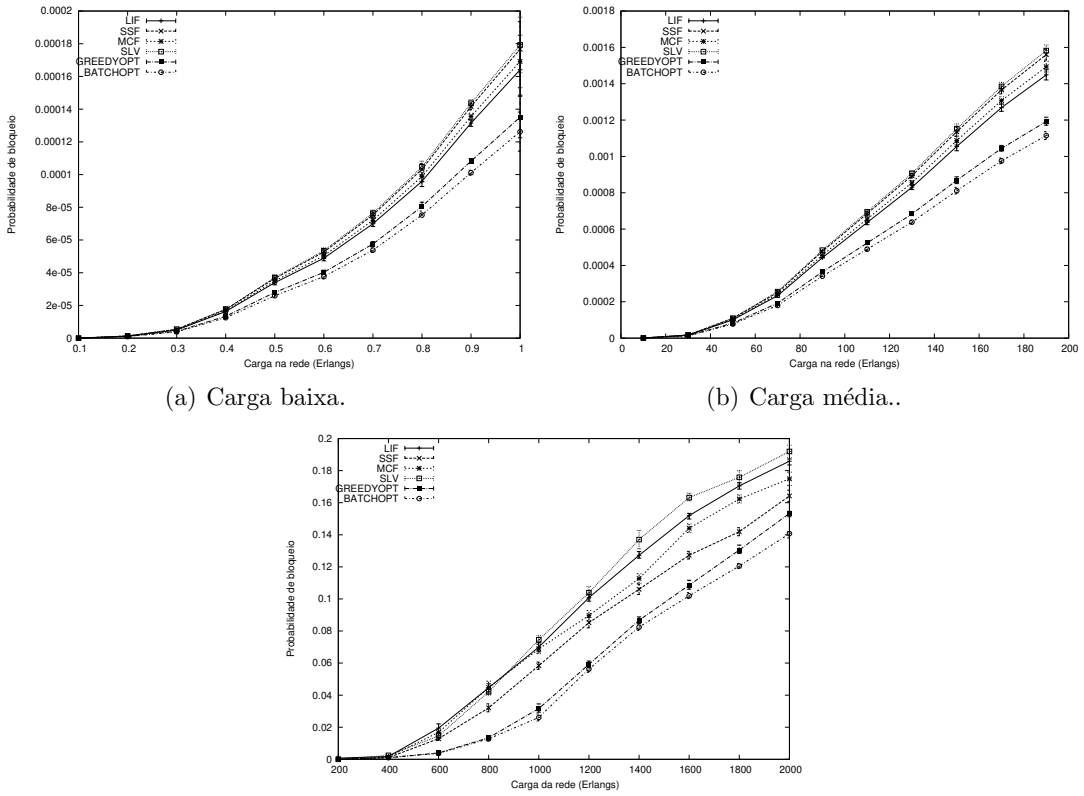
Figura 9.20: Número médio de requisições por lote em função da janela de aceitação de requisições.

Os algoritmos GreedyOPT e BATCHOPT, por outro lado, conseguem beneficiar-se do aumento do número médio de requisições em cada lote. Isso por que os algoritmos são ótimos e à medida em que mais requisições são processadas de uma vez, maior o horizonte de tempo que eles têm conhecimento sobre as requisições e mais informações possuem acerca das intersecções entre tais requisições.

Como quanto maior a janela de aceitação de requisições, melhor o desempenho dos

algoritmos GreedyOPT e BATCHOPT, na prática tal parâmetro pode ser ajustado dependendo dos requisitos temporais do tráfego transportado e determinado pela Equação 9.5. Para não impactar o desempenho de aplicações com restrições temporais e ao mesmo tempo obter uma quantidade razoável de requisições no lote, Δ foi usado com valor de $1ms$ [32].

A seguir, foram realizadas simulações com tráfego com três diferentes intensidades de carga: carga baixa (carga variando de 0.1 a 1 Erlang), carga média (carga variando de 10 a 190 Erlangs) e carga alta (carga variando de 200 a 2000 Erlangs).



(a) Carga baixa.

(b) Carga média..

(c) Carga alta..

Figura 9.21: Probabilidade de bloqueio.

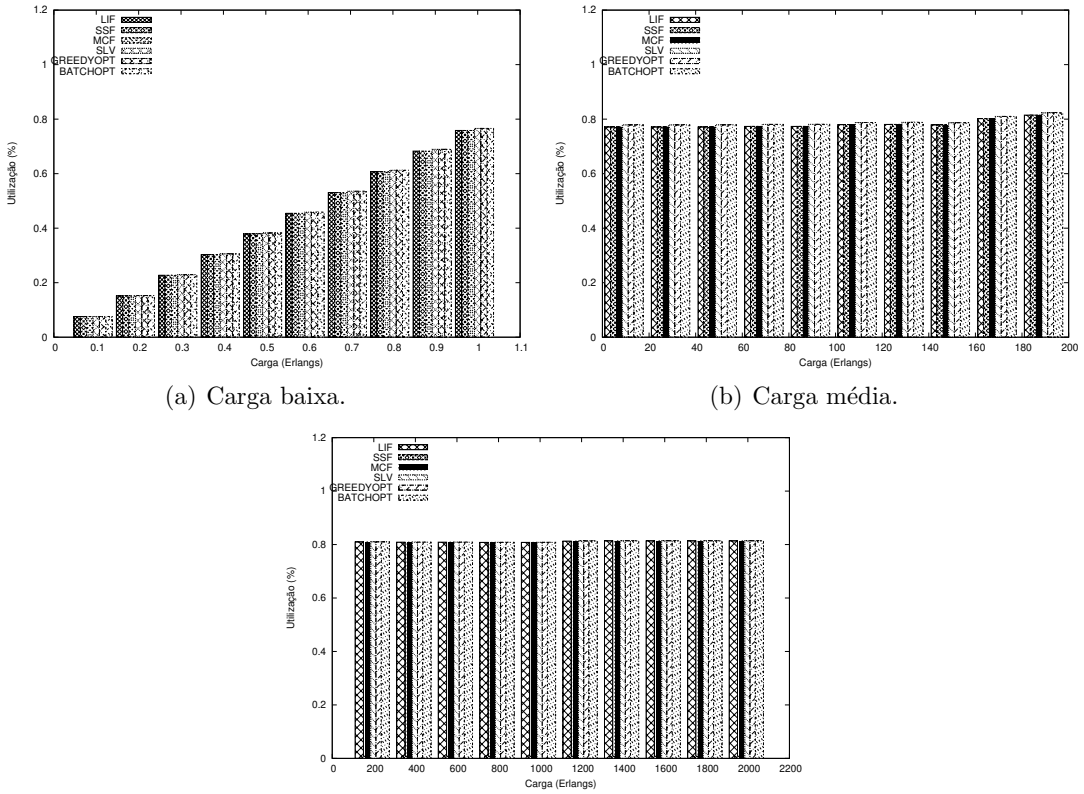
A Figura 9.21 mostra a probabilidade de bloqueio em função do aumento da carga

na rede em um cenário sem QoS, ou seja, quando todas as requisições possuem peso unitário. 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. Em relação ao pior algoritmo, o SLV, o algoritmo BATCHOPT apresentou um ganho de 42% na probabilidade de bloqueio. Isso porque, como todas as requisições têm custo 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.

O algoritmo com a segunda menor probabilidade de bloqueio foi o algoritmo GreedyOPT, com ganho de 35% em relação ao SLV. Apesar do algoritmo GreedyOPT, assim como o BATCHOPT, ser um algoritmo ótimo, o que leva à alocação de um conjunto de requisições com cardinalidade máxima, o fato do algoritmo não garantir que as requisições já escalonadas pertençam à nova solução, faz com que a probabilidade de bloqueio seja maior do que a do BATCHOPT. Isso por que as requisições já escalonadas podem ser bloqueadas pelas requisições do lote em processamento. Estas últimas, por sua vez, podem ser bloqueadas por requisições futuras e assim por diante, sem que exista garantia de que nenhuma obtenha, de fato, sucesso.

A seguir, vêm os algoritmos LIF, MCF e SSF com 12%, 8% e 2% de ganho na probabilidade de bloqueio em relação ao algoritmo SLV.

A Figura 9.22 apresenta a utilização da rede. Todos os algoritmos avaliados apresentaram níveis muito próximos de utilização. Pode-se perceber um aumento linear da utilização quando a carga de tráfego é baixa. Isso por que, à medida em que aumenta-se a carga, cresce o número de reservas realizadas. O cresimento linear é observado até a carga de 1 Erlang, quando atinge 78%. A partir desse ponto, existe uma saturação da rede, refletido pelo aumento da probabilidade de bloqueio, e o crescimento da utilização é bem menor. Da carga de 10 Erlangs até 2000 Erlangs, a utilização da rede variou de 78% a 82%.



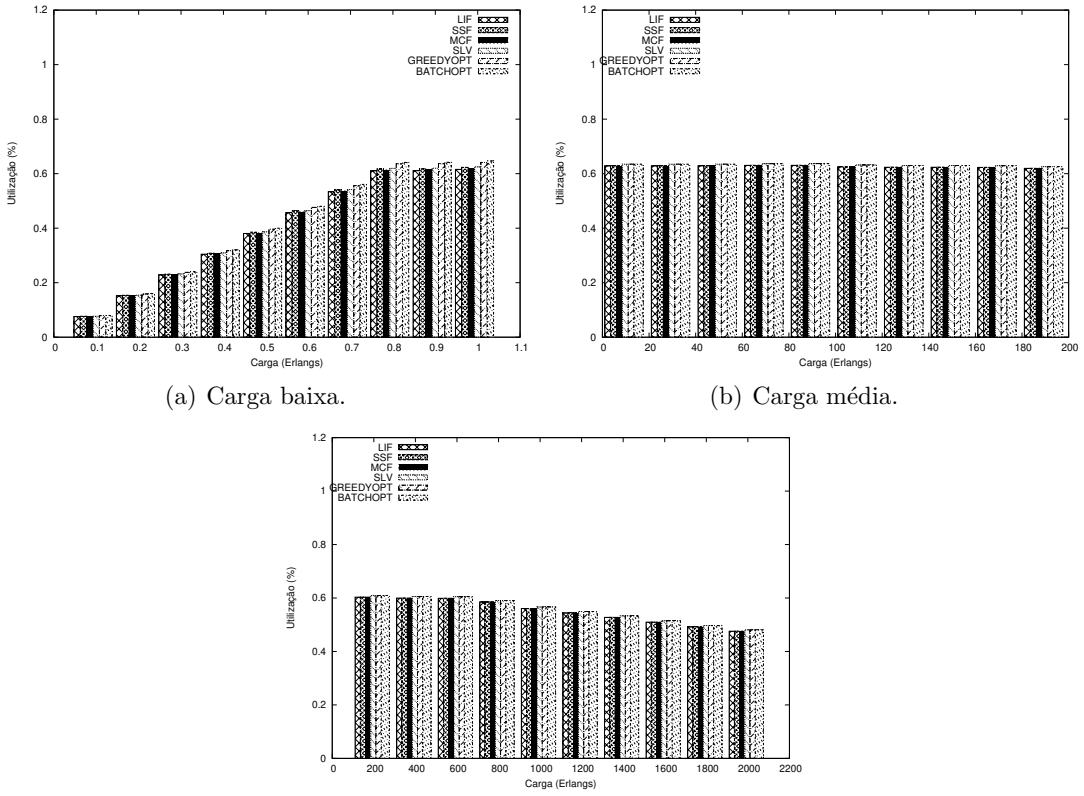
(a) Carga baixa.

(b) Carga média.

(c) Carga alta.

Figura 9.22: Utilização média da rede.

A Figura 9.23 apresenta a utilização efetiva da rede em função do aumento da carga de tráfego. É possível perceber que os algoritmos BATCHOPT e GreedyOPT produzem as duas maiores utilizações efetivas dentre os algoritmos avaliados. Uma maior utilização efetiva pode ser explicada por dois fatores: i) o algoritmo obteve menor probabilidade de bloqueio, fazendo com que um número maior de rajadas tenham sido transmitidas, e, consequentemente um maior tempo de duração das reservas bem sucedidas e ii) o algoritmo conseguiu atender rotas de tamanhos maiores, o que também se reflete em um aumento do tempo das reservas realizadas. Como pode ser observado na Figura 9.24 a diferença do tamanho médio das rotas atendidas pelos algoritmos é muito pequena, já que



(a) Carga baixa.

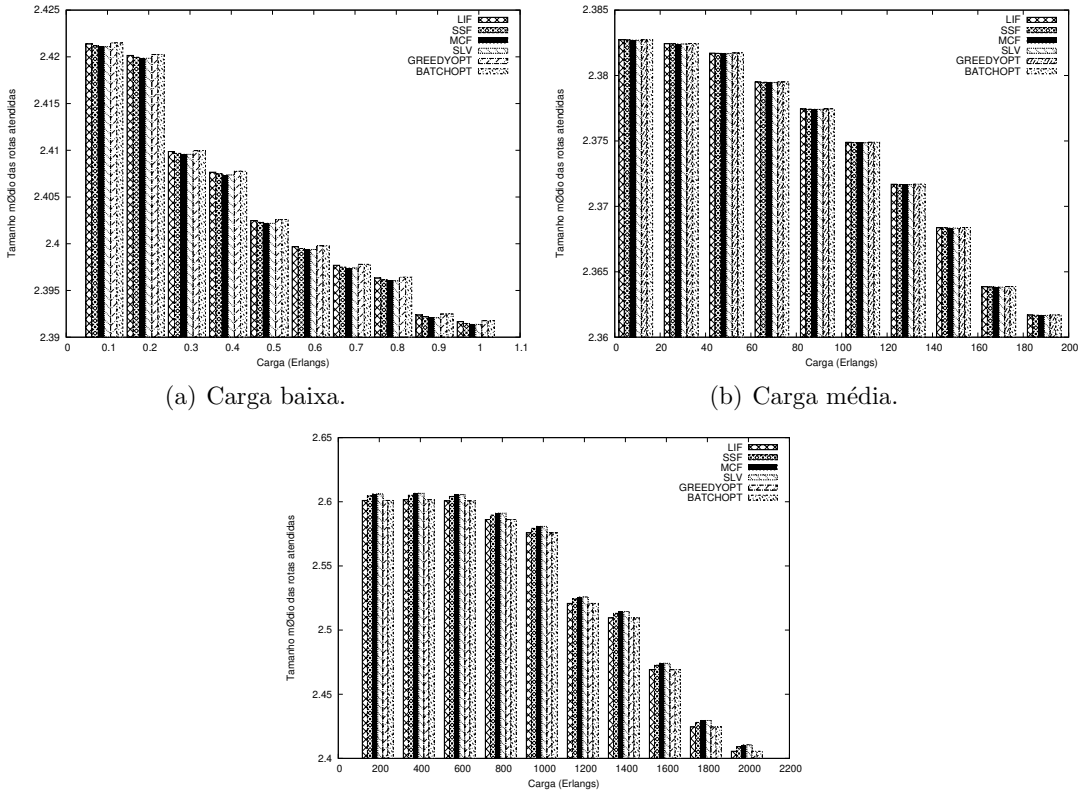
(b) Carga média.

(c) Carga alta.

Figura 9.23: Utilização efetiva média da rede.

a escolha de qual requisição será escalonada, ou qual o canal será usado, independe de fatores que possam aumentar o tamanho das rotas atendidas. Assim, conclui-se que o fator preponderante na diferença da utilização efetiva deve-se ao desempenho dos algoritmos em termos de probabilidade de bloqueio. Comparando com os algoritmos do capítulo 8, percebe-se que os algoritmos de escalonamento em lote produzem cerca de 20% a mais de utilização efetiva.

Foram realizadas, também, simulações levando-se em consideração requisitos de QoS. Para tal, foram usadas 5 classes de serviço de forma que a classe i é mais prioritária que a classe j se $i > j$. Cada requisição gerada, foi associada a uma classe de serviço de maneira



(a) Carga baixa.

(b) Carga média.

(c) Carga alta.

Figura 9.24: Tamanho médio das rotas atendidas.

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 custo das requisições de classe i.

É possível perceber através da Figura 9.25 que a probabilidade de bloqueio dos algoritmos LIF, MCF, SSF, SLV e GreedyOPT permaneceu inalterada, já que eles não levam em consideração os pesos da requisição (note que os pesos utilizados pelo algoritmo LIF correspondem ao tamanho dos intervalos e não à classe de serviço). Percebe-se, também, que a probabilidade de bloqueio do algoritmo BATCHOPT aumentou em relação ao cenário com pesos unitários. Esse é um fenômeno esperado já que o algoritmo tem uma tendência

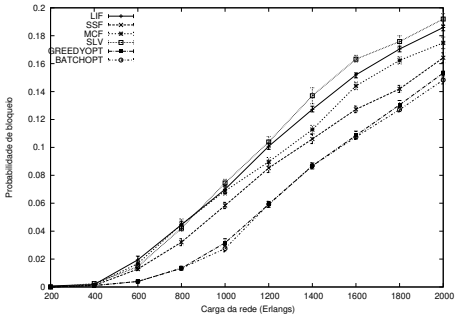


Figura 9.25: Probabilidade de bloqueio em função da carga de tráfego na rede (cenário com QoS)

a beneficiar as requisições com um maior peso (para garantir que a soma dos pesos seja máxima). Com isso, as requisições das classes de alta prioridade são mais beneficiadas pelo algoritmo BATCHOPT.

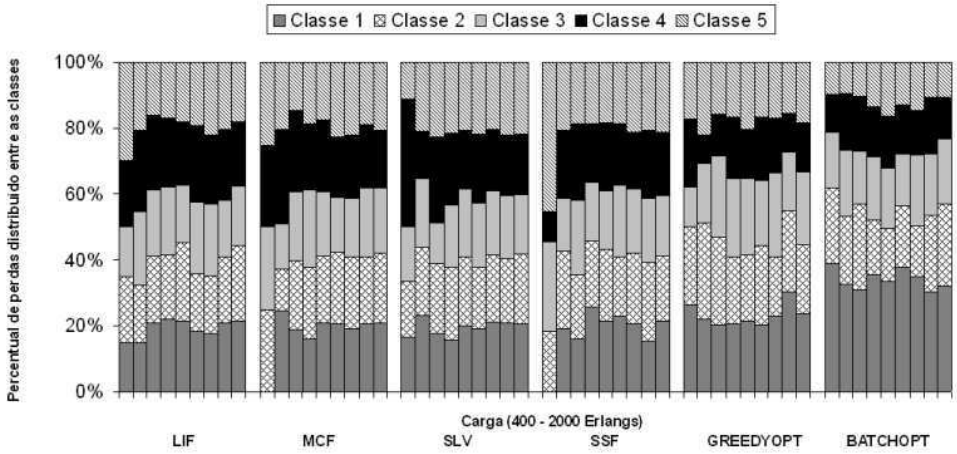


Figura 9.26: Percentual de perdas distribuído entre as classes de tráfego

Entretanto, quando se observa o percentual de perdas dividido entre as classes de serviço, ilustrado na Figura 9.26, percebe-se claramente que o algoritmo BATCHOPT penaliza menos as classes mais prioritárias, cujo peso é maior.

Além dos resultados numéricos, mediu-se também o tempo de execução dos algoritmos.

Tabela 9.1: Ganho relativo no tempo médio de execução

	Lote 1	Lote 2	Lote 3	Lote 4	Lote 5	Média
`Valor Referência`	7.2×10^{-2}	8.0×10^{-2}	6.8×10^{-2}	7.6×10^{-2}	7.2×10^{-2}	7.3×10^{-2}
`LIF`	37%	27%	22%	40%	33%	32%
`SSF`	41%	30%	25%	42%	37%	35%
`MCF`	33%	26%	26%	34%	25%	28%
`SLV`	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
`GreedyOPT`	36%	27%	25%	40%	31%	32%
`BATCHOPT`	33%	25%	23%	36%	27%	28%

As simulações foram realizadas em uma máquina Intel Pentium Core 2 Duo com 2.8Ghz e 4GB de memória RAM, executando o sistema operacional OpenSuse 11.1. O tempo de execução foi medido com o uso do comando *time*.

Foram gerados 5 lotes de requisições, cada um contendo 500 requisições geradas aleatoriamente. Para cada lote foram realizadas 10 simulações e o tempo médio de execução de cada lote calculado, bem como a média do tempo de execução em todos os lotes.

O ganho relativo no tempo de execução de cada lote é apresentado na Tabela 9.1. O ganho relativo é definido como a diferença em percentual em relação ao valor de referência, que é o tempo de execução do algoritmo mais lento. É possível observar que o algoritmo mais lento em todos os cenários foi o algoritmo SLV. O algoritmo mais rápido foi o algoritmo SSF. Pode-se também notar que o algoritmo BATCHOPT foi, no máximo 10% mais lento que o algortimo SSF (processamento do lote 5) e, em média, 32% mais rápido que o SLV e 7% mais lento que o algoritmo SSF.

O algoritmo GreedyOPT apresentou-se mais rápido que o algoritmo BATCHOPT. Ele foi, no máximo 5% mais lento que o algoritmo SSF e, em média 32% mais rápido que o algoritmo SLV e 3% mais lento que o algoritmo mais rápido, o SSF.

9.7 Resumo conclusivo

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 capítulo, foram discutidos algoritmos ótimos de escalonamento em lote para redes OBS. Os algoritmos são capazes de fornecer boas soluções para o problema de escalonamento em lote em redes OBS de forma rápida. Viu-se que o algoritmo BATCHOPT apresentou um ganho de 42% na probabilidade de bloqueio em relação ao pior algoritmo e foi em média 7% mais lento que o algoritmo mais rápido. Já o algoritmo GreedyOPT obteve 35% a menos na probabilidade de bloqueio do que o pior algoritmo e foi em média 3% mais lento do que o algoritmo mais rápido. Além disso, viu-se que o algoritmo BATCHOPT é capaz de produzir menos penalização nas rajadas das classes de alta prioridade. Assim, pode-se concluir que o algoritmo BATCHOPT é capaz de fornecer uma solução mais completa, sendo portanto indicado como a solução com melhor custo-benefício. Entretanto, se o tempo de execução é um requisito crucial, o algoritmo GreedyOPT pode ser utilizado.

Foi proposta também uma extensão do protocolo JET para atuar como estratégia de formação de lotes. Resultados de simulação mostraram que os algoritmos possuem desempenho superior aos algoritmos semelhantes existentes na literatura.

Capítulo 10

Conclusões e Trabalhos futuros

Nos últimos anos, as redes OBS têm sido alvo de intensas pesquisas. Grande parte de seu atrativo deve-se ao envio dos datagramas IP agrupados em uma única rajada, à sinalização *out-of-band*, e, sobretudo, à sua flexibilidade proporcionada pelo processo de reserva em uma via.

Entretanto, apesar de sua flexibilidade, as redes OBS ainda apresentam alta taxa de perdas, o que pode deteriorar o nível dos serviços oferecidos pela rede. Para diminuir o bloqueio e, consequentemente, oferecer melhores serviços às aplicações, a melhoria dos mecanismos de controle da rede é necessária.

O foco desta tese foi o desenvolvimento e avaliação de desempenho dos algoritmos de montagem de rajadas e escalonamento de canais, capazes de dar suporte ao tráfego IP, melhorando assim a adaptabilidade da rede OBS a uma Internet óptica de nova geração.

A primeira etapa no desenvolvimento da tese foi a análise do impacto dos mecanismos da rede. No Capítulo 5, viu-se que um tráfego multifractal que alimenta um nó OBS de ingresso pode ter suas propriedades estatísticas alteradas de multifractal para monofractal dependendo dos parâmetros utilizados nos algoritmos de montagem. Verificou-se que tal transformação pode ocasionar uma diminuição na demanda por recursos da rede, o que, como verificado através de simulações, pode significar uma redução de até 4% em termos de probabilidade de bloqueio.

Constatou-se também que a indução das transformações no tráfego multifractal necessita do conhecimento prévio acerca da escala de tempo limitante. Contudo, o método para identificação de tal escala era baseado na inspeção visual do diagrama criado a partir de um conjunto de pontos, o que torna impraticável a sua implementação.

Para solucionar este problema, foi proposto no Capítulo 6 um método não visual para detecção da escala de corte. O método possui complexidade linear em relação ao número de pontos do traço de tráfego de entrada e é baseado na observação dos valores adquiridos pelo coeficiente de determinação de uma regressão linear realizada no conjunto de pontos do plano formado pelo logaritmo do processo incremento agregado em função do logaritmo da escala de agregação. Experimentos realizados usando como entrada traços de tráfego real mostraram a aplicabilidade do método.

No Capítulo 7, foram apresentados os algoritmos de montagem de rajadas capazes de induzir transformações nas propriedades estatísticas do tráfego. Observou-se através de simulações que os algoritmos propostos são capazes de induzir transformações nas propriedades estatísticas do tráfego, reduzindo assim a demanda por recursos na rede. Verificou-se, que dentre os algoritmos propostos, o algoritmo *Proportional Fit* apresentou os melhores desempenhos em termos de vazão e atraso de montagem de rajadas.

No Capítulo 8 foi proposto um algoritmo de escalonamento de canais denominado LRC (*Least Reusable Channel*). O algoritmo LRC usa uma estratégia adaptativa na escolha de qual canal utilizar. A estratégia visa utilizar o canal com menores chances de ser utilizado por requisições futuras. A reutilização dos canais é medida pela função de reutilização, definida em função da probabilidade de inversão de rajadas e pelo tempo de vida útil dos *voids*. O desempenho do algoritmo foi avaliado, comparando-o através de simulações, com outros algoritmos propostos na literatura sob diversos modelos de tráfego. Resultados mostraram que o algoritmo produz uma diminuição de até $27,6\%$ na probabilidade de bloqueio experimentada. Além disso, o algoritmo produz maior utilização da rede e menor fator de justiça.

No Capítulo 9 foram apresentados dois algoritmos de escalonamento em lote de canais

em redes OBS. Primeiramente foi proposta uma estratégia para redução do problema de escalonamento em lote em redes OBS ao problema de *job scheduling* em máquinas idênticas. Além disso, foi proposto o algoritmo GreedyOPT para o problema de escalonamento em lote de requisições com peso unitário. O algoritmo GreedyOPT tem como principal vantagem o fato de possuir baixa complexidade computacional, o que o torna mais rápido que os demais algoritmos de escalonamento em lote. Depois, foi proposto o algoritmo BATCHOPT. O algoritmo BATCHOPT é capaz de resolver de forma ótima o problema de escalonamento em lote quando as requisições possuem diferenciação dos pesos ou prioridades. Por fim, foi proposta no mesmo capítulo, uma estratégia de formação de lote que é uma adaptação do protocolo JET para uso com algoritmos de escalonamento em lote. Simulações mostraram que o algoritmo BATCHOPT apresentou os melhores resultados em termos de probabilidade de bloqueio e utilização da rede.

10.1 Trabalhos Futuros

Nesta tese foram discutidos e apresentados mecanismos de controle para redes de comutação de rajadas ópticas. Apesar de os mecanismos terem sido vastamente avaliados e terem apresentado resultados satisfatórios, existe ainda um campo vasto de investigação de tais mecanismos em trabalhos futuros.

No Capítulo 6, foi apresentado um método não visual para detecção da escala limitante de fluxos multifractais. O método é baseado na inspeção dos valores assumidos pelo coeficiente de determinação de uma regressão linear realizada nos pontos do plano formado pelo logaritmo do processo incremento agregado, em função do logaritmo da escala de agregação. O teste do coeficiente de determinação pára quando está abaixo de um valor pré-determinado, ϵ. Os valores usados no teste foram ajustados empiricamente, assim, para garantir uma completa automatização da detecção da escala limitante, é importante o desenvolvimento de métodos capazes de realizar a estimação do valor de ϵ para qualquer traço de entrada. Outro ponto importante é que, como visto no Capítulo 6, o

método recebe como entrada o processo incremento agregado do traço. A geração do processo incremento agregado não está incorporada à detecção da escala limitante. Assim, outro ponto de análise futura, seria o desenvolvimento de métodos capazes de realizar a agregação do tráfego de entrada em função das estatísticas do tráfego multifractal.

No Capítulo 7, a avaliação de desempenho dos algoritmos de montagem foi realizada através de simulações. É importante também o desenvolvimentos de modelos analíticos que representem a operação dos mecanismos propostos para a completa validação dos resultados obtidos através das simulações.

No Capítulo 8 foi proposto um algoritmo de escalonamento que leva em consideração informações relativas ao processo de montagem de rajadas para efetuar cálculos sobre a probabilidade de inversão. Para dar mais independência ao escalonamento, é interessante o desenvolvimento de novos métodos para cálculo da probabilidade de inversão que não necessitem de informações acerca do tipo de algoritmo de montagem sendo utilizado. Outro ponto interessante para futuros trabalhos é a estimação do tempo de execução do algoritmo propostos, levando-se em consideração diversos ambientes. Outro ponto não considerado foi a presença de algoritmos de roteamento adaptativos, o que influencia a dinâmica de chegada de requisições. Além disso, como trabalho futuro, pretende-se também o desenvolvimento de um modelo analítico capaz de capturar a dinâmica da operação do algoritmo proposto, de forma que os resultados obtidos via simulação sejam completamente avaliados.

No Capítulo 9 foram discutidos algoritmos de escalonamento em lote de canais para redes OBS. Foi apresentada também uma estratégia de formação de lote. Como trabalhos futuros, pretende-se realizar o desenvolvimento de novas estratégias de formação de lote, bem como sua comparação com as estratégias discutidas neste capítulo. Além disso, pretende-se comparar os algoritmos propostos com outros algoritmos ótimos para *job scheduling* com máquinas idênticas propostos na literatura. Finalmente, pretende-se o desenvolvimento de um modelo analítico para complementar a avaliação de desempenho dos algoritmos propostos.

Referências Bibliográficas

[1] *OBS-ns Manual. Available at 02/09/2009 at http://wine.icu.ac.kr/ obsns/docs.php.*

[2] www.nlanr.net.

[3] A. J. G. Abelém and M. A. Stanton. *Inter-redes IP baseadas em Redes Ópticas*. Minicurso XX Simpósio Brasileiro de Redes de Computadores, 2002.

[4] P. Abry, R. Baraniuk, P. Flandrin, R. Riedi, and D. Veitch. The multiscale nature of network traffic: Discovery, analysis, and modelling. *IEEE Signal Processing Magazine*, 19:28–46, May 2002.

[5] P. Abry and Veitch D. Wavelet analysis of long-range dependent trafic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998.

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

[7] J. Beran. *Statistics for Long-Memory Processes*. Chapman and Hall, 1994.

[8] Khalid I. Bouzina and Hamilton Emmons. Interval scheduling on identical machines. *Journal of Global Optimization*, 9(3-4):379–393, 1996.

[9] Xiaojun Cao, Jikai Li, Yang Chen, and Chumming Qiao. Assembling TCP/IP packets in optical burst switched networks. In *IEEE GLOBECOM*, pages 2808–2812, 2002.

[10] J. E. Cavanaugh, Y. Wang, and J. W. Davis. Self-similar proccesses and their wavelet anaysis. *Handbook of Statistics 21: Stochastic Processes, Elsevier*, 2003.

[11] J. Chang and C. Park. Efficient channel scheduling algorithm in optical burst switching architecture. In *IEEE Workshop on High Performance Switching and Routing*, pages 194–198, 2002.

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

[13] Y. Chen, M. Hamdi, and D. H. K Tsang. Proportional QoS over OBS networks. In *IEEE GLOBECOM*, volume 3, pages 1510–1514, 2001.

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

[15] Kee Chaing Chua, Mohan Gurusamy, Yong Liu, and Minh Hoang Phung. *Quality of Service in Optical Burst Switched Networks*. Springer, 2006.

[16] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill, 1990.

[17] D. R. Cox. Long range dependence: A review. *Statistics*, pages 55–74, 1984.

[18] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5:835–846, 1996.

[19] A. Detti, V. Eramo, and M. Listanti. Performance evaluation of a new technique for IP support in a WDM optical network: Optical composite burst switching (OCBS). *Journal of Lightwave Technology*, 20:154–165, 2002.

[20] K. Dolzer. Assured horizon - a new combined framework for burst assembly and reservaton in optical burst switched networks. In *NOC*, 2002.

[21] M. Elhaddad, R. Melhem, T. Znati, and D. Basak. Traffic shaping and scheduling for OBS-based IP/WDM backbones. In *IEEE Opticom*, volume 5285, pages 336–345, 2003.

[22] A. Erramilli, O. Narayan, A. Neidhart, and I. Saniee. Multi-scaling models of TCP/IP and sub-frame VBR video traffic. *Journal of Communications and Networks*, 3:383–395, December 2001.

[23] Rafael Esteves, Fernando Nazareno Farias, D. Souza, and A. J. G. Abelém. Qualidade de serviço absoluta em redes OBS baseadas no GMPLS. In *Simpósio Brasileiro de Redes de Computadores*, pages 577–590, 2007.

[24] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz. The changing nature of network traffic: scaling phenomena. *SIGCOMM Computer Communication Review*, 28(2):5–29, 1998.

[25] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: a study of the role of variability and the impact of control. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 301–313, New York, NY, USA, 1999. ACM.

[26] G. B. Figueiredo, C. A. V. Melo, N. L. S. Fonseca, and M. R. Salvador. On the transformation of multifractal traffic at ingress optical burst switches. In *IEEE ICC*, pages 1040–1045, 2006.

[27] Gustavo B. Figueiredo, Nelson L. S da Fonseca, and José A. S. Monteiro. A minimum interference routing algorithm with reduced computational complexity. *Computer Networks*, 2006.

[28] An Ge, Franco Callegati, and Lakshman S. Tamil. On optical burst switching and self-similar traffic. *IEEE Communications Letters*, 4(3):98–100, 2000.

[29] S. Gowda, R. K. Shenai, K. M. Sivalingam, and H. C. Cankaya. Performance evaluation of TCP over optical burst-switched (OBS) WDM networks. In *ICC*, volume 2, pages 1433–1437, 2003.

[30] E. Haselton. A PCM Frame Switching Concept Leading to Burst Switching Network Architecture. *IEEE Communications Magazine*, 21:13–19, June 1983.

[31] G. Hu, K. Dolzer, and C. Gauger. Does burst assembly really reduce the self-similarity? In *Optical Fiber Communications Conference (OFC)*, pages 124–126, 2003.

[32] Internet2 QoS Working Group, http://qos.internet2.edu/wg/apps/fellowship/Docs/Internet2 *Network QoS Needs of Advanced Internet Applications: A Survey*, 2001.

[33] M. Izal and J. Aracil. On the influence of self-similarity on optical burst switching traffic. In *GLOBECOM*, volume 3, pages 2308–2312, 2002.

[34] Jason P. Jue and Vinod M. Vokkarane. *Optical Burst Switched Networks*. Springer, 2004.

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

[36] A. Kaheel and Alnuweiri H. Analytical evaluation of blocking probability in optical burst switching networks. In *IEEE International Conference on Communications*, 2004.

[37] Ayman Kaheel and Hussein Alnuweiri. Priority Scheme for Supporting Quality of Service in Optical Burst Switching Networks. *Journal of Optical Networking*, pages 707–719, 2004.

[38] Sungchang Kim, JimSeek Choi, and Minho Kang. Control plane architecture for QoS in OBS networks using dynamic wavelength assignment. *Lecture Notes in Computer Science*, pages 64–75, 2003.

[39] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2:1–15, 1994.

[40] J. Li and C. Qiao. Schedule bursts proactively for optical burts switched networks. *Computer Networks*, 44:617–629, 2004.

[41] W. Liao and C. Loi. Providing service diferentiation for optical-burst switched networks. *Journal of Lightwave Technology*, 22:1651–1660, 2004.

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

[43] K. Long, R. S. Tucker, and C. Wang. A new framework and burst assembly for IP diffserv over optical burst switching networks. In *GLOBECOM*, pages 3159–3164, 2003.

[44] J. Maranhão, A. Soares, and W. F. Giozza. Estudo das arquiteturas de conversão de comprimento de onda em redes WDM com comutação de rajadas Ópticas. In *XXV SBRC*, pages 133–146, 2007.

[45] D. W. Matula. Smallest-last ordering and clustering and graph coloring algorithms. *Information Processing Letters*, 30(3):217–427, July 1983.

[46] C. A. V. Melo and N. L. S. da Fonseca. Envelope process and computation of the equivalent bandwidth of multifractal flows. *Computer Networks*, 48:351–375, 2005.

[47] Cesar Augusto Viana Melo. *Modelagem e Computação de banda passante equivalente de fluxos multifractais*. PhD thesis, Instituto de Computação, Universidade Estadual de Campinas, 2004.

[48] I. M. Moraes and O. C. M. B. Duarte. Um mecanismo de controle de admissão para provisão de serviços diferenciados em redes de rajadas ópticas. In *Simpósio Brasileiro de Redes de Computadores*, 2005.

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

[50] C. Murty and M. Gurusamy. *WDM Optical Networks: Concepts, Design and Algorithms*. Prentice Hall, 2002.

[51] S. Oh and M. Kang. A burst assembly algorithm in optical burst switching networks. In *Optical Fiber Communications Conference*, pages 771–773, 2002.

[52] Stephan Olariu. An optimal greedy heuristic to color interval graphs. *Information Procesing Letters*, 1(37):21–25, 1991.

[53] A. Papoulis. *Probability, Random Variables and Sthocastic Process*, chapter 3. McGraw-Hill, 2002.

[54] Kihong Park and Walter Willinger. *Self-Similar Network Traffic and Performance Evaluation*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[55] Harry G. Perros. *An Introduction to ATM Networks*. Wiley, 2001.

[56] C. Qiao and M. Yoo. Optical burst switching (OBS) - A new paradigm for an optical Internet. *Journal of High-Speed Networks*, 8(1):69–84, Jan 1999.

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

[58] V. J. Ribeiro, Z. L. Zhang, S. Moon, and C. Diot. Small-time scaling behavior of internet backbone traffic. *Computer Network*, 48(3):315–334, 2005.

[59] Rudolf H. Riedi and Jacques Lévy Véhel. TCP traffic is multifractal : a numerical study. Technical report, INRIA, 1997.

[60] Joel J. P. C. Rodrigues, Mario M Freire, and Lorenz Pascal. 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, 2005.

[61] Robert Sedgewick. *Algorithms in C, Part 5: Graph Algorithms*. Addison-Wesley Professional, 3 edition, 2001.

[62] Web site da ferramenta coralreef. Julho 2009. disponível em ¡http://www.caida.org/tools/measurement/coralreef/¿, Julho 2009.

[63] Takuji Tachibana, Tamoya Ajima, and Shoji Kasahara. Round-robin burts assembly and constant transmission scheduling for optical burts switching networks. In *IEEE GLOBECOM*, pages 2772–2776, 2003.

[64] M. Taqqu, V. Teverovsky, and W. Willinger. Is the Ethernet data self-similar or multifractal? *Fractals*, 5:63–73, 1997.

[65] J. Turner. Terabit burts switching. *Journal of High Speed Networking*, pages 3–16, 1999.

[66] D. Veitch. D. veitch home page, julho 2009. disponível em ¡http://www.cubinlab.ee.mu.oz.au/ darryl/¿, Julho 2009.

[67] D. Veitch, N. Hohn, and P. Abry. Multifractality in TCP/IP traffic: the case agains it. *Computer Network*, 48(3):293–313, 2005.

[68] F. H. T. Vieira. *Contribuições ao cálculo da banda e probabilidade de perda para tráfego multifractal em redes*. PhD thesis, UNICAMP, 2006.

[69] V. Vokkarane, Q. Zhang, J. P. Jue, and B. Chen. Generalized burst assembly and scheduling techniques for QoS support to optical burst-switched networks. In *GLOBECOM*, 2002.

[70] V. M. Vokkarane and J. P. Jue. Prioritized burst segmentation and composite burst-assembly techniques for QoS support in optical burst-switched networks. *IEEE Journal on Selected Areas in Communications*, 21:1198–1209, 2003.

[71] X. Wang, H. Morikawa, and T. Aoyama. Priority-based wavelength assignment algorithm for optical burst switched photonic networks. In *Optical Fiber Communications Conference*, pages 765–766, 2002.

[72] Y. Xiong, M. Vandenhoute, and C. Cankaya. 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, 1999.

[73] Y. Xiong, M. Vandenhoute, and C. Cankaya. Control architecture in optical burst-switched WDM networks. *IEEE Journal of Selected Areas on Communications*, 18:1838–1851, 2000.

[74] J. Xu, C. Qiao, J. Li, and G. Xu. Efficient channel scheduling algorithms in optical burst switched networks. In *IEEE INFOCOM*, volume 3, pages 2268–2278, 2003.

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

[76] Lisong Xu, Harry G. Perros, and George N. Rouskas. A Simulation Study of Access Protocols for Optical Burst-Switched Ring Networks. *Computer Networks*, 41:143–160, 2003.

[77] M. Yoo and C. Qiao. A New Optical Burst Switching Protocol for Supporting Quality of Service. *All Optical Networking: Architecture, Control and Management Issue*, pages 396–405, 1998.

[78] Xiang Yu, Yang Chen, and Chumming Qiao. Study of traffic statistics of assembled burst traffic in optical burst switched networks. In *Opticomm*, pages 149–159, 2002.

[79] Xiang Yu, Jikai Li, Xiaojun Cao, Yang Chen, and Chumming Qiao. Traffic statistics and performance evaluation in optical burst switched networks. *Journal of Lightwave Technology*, 22(12):2722–2738, 2004.

[80] Q. Zhang, V. M. Vokkarane, J. P. Jue, and B. Chen. Absolute QoS differentiation in optical burst-switched networks. *IEEE Journal on Selected Areas in Communications*, 22:1781–1795, 2004.