PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

QUALIDADE DE SERVIÇO EM REDES INTRA-CHIP
IMPLEMENTAÇÃO E AVALIAÇÃO SOBRE A REDE HERMES

ALINE VIEIRA DE MELLO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Fernando Gehm Moraes

Porto Alegre
2006

Dados Internacionais de Catalogação na Publicação (CIP)

M527q Mello, Aline Vieira de
Qualidade de serviço em redes intra-chip : implementação e avaliação sobre a rede Hermes / Aline Vieira de Mello. - Porto Alegre, 2006.
138 f.

Diss. (Mestrado) - Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Fernando Gehm Moraes.

1. Informática. 2. Qualidade de Serviço. 3. Redes Intra-Chip. 4. Escalonamento. 5. Rede Hermes. I. Título.

CDD 004.6

Ficha Catalográfica elaborada pelo Setor de Processamento Técnico da BC-PUCRS

PUC

Campus Central
Av. Ipiranga, 6681 - prédio 16 - CEP 90619-900
Porto Alegre - RS - Brasil
Fone: +55 (51) 3320-3544 - Fax: +55 (51) 3320-3548
Email: bceadm@pucrs.br
www.pucrs.br/biblioteca

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação Intitulada "***Qualidade de Serviço em Redes Intra-Chip: Implementação e Avaliação sobre a Rede Hermes***", apresentada por Aline Vieira de Mello, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas Embarcados e Sistemas Digitais, aprovada em 16/01/2007 pela Comissão Examinadora:

Prof. Dr. Fernando Gehm Moraes – Orientador (a) — PPGCC/PUCRS

Prof. Dr. Ney Laert Vilar Calazans – PPGCC/PUCRS

Prof. Dr. Cesar Albenes Zeferino – UNIVALI

Prof. Dr. Gilles Sassatelli – CNRS/LIRMM

Homologada em 07/05/2007, conforme Ata No. 5019 pela Comissão Coordenadora.

Prof. Dr. Fernando Luís Dotti
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P. 16 – sala 106 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@inf.pucrs.br
www.pucrs.br/facin/pos

“Dar menos que seu melhor é sacrificar o dom que você recebeu.” (Steve Prefontaine)

Agradecimentos

Muitas pessoas contribuíram para a realização desta dissertação de mestrado e não poderia deixar de manifestar meu profundo agradecimento a todas elas.

Ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) pelo apoio financeiro para a realização deste trabalho possibilitando minha dedicação exclusiva ao mesmo.

Aos funcionários, alunos e professores do PPGCC pela convivência, amizade e experiências compartilhadas ao longo do curso de mestrado.

Aos colegas e amigos do grupo de apoio ao projeto de hardware (GAPH) pelo ótimo ambiente de trabalho e ótima convivência que me proporcionaram durante os seis anos que faço parte deste grupo. Especialmente, a Leandro Heleno Möller e Leonel Tedesco pelas discussões técnicas, sugestões e parceria nas publicações.

Ao Prof. Dr. Cesar Albenes Zeferino, professor da Universidade do Vale do Itajaí (UNIVALI), que gentilmente aceitou participar e colaborar com este trabalho fazendo parte da banca de avaliação.

Ao Prof. Dr. Ney Laert Vilar Calazans pelo primeiro estímulo à vida acadêmica, pelos comentários, críticas e sugestões que ajudaram a enriquecer este trabalho.

Especialmente ao meu orientador e amigo, Prof. Dr. Fernando Gehm Moraes pelo constante incentivo, pela dedicação e paciência, pelo apoio (principalmente psicológico) nos momentos de dúvidas e frustrações, pela co-autoria em vários trechos desta Dissertação. Agradeço, principalmente, por confiar em mim e no meu trabalho.

Agradeço a todos os meus amigos pessoais que, de alguma forma, me incentivaram e contribuíram para esta vitória.

A Luiz Paulo Borges de Moura pelo companheirismo, paciência e amor.

A toda a minha família que sempre me ofereceu suporte e acreditou em mim. Ao meu irmão Cristiano Vieira de Mello pela amizade e carinho. Aos meus pais Cleusa Maria Vieira de Mello e Adilson Leon de Mello pelo estímulo, confiança e amor incondicional que demonstraram durante toda a minha vida.

Muito Obrigado.

Resumo

A proposição de redes intra-chip (NoCs) para futuros e modernos sistemas embarcados baseia-se no fato de que barramentos apresentam degradação do desempenho quando compartilhados por um grande número de núcleos. Mesmo a pesquisa de NoCs sendo um campo relativamente novo, a literatura possui muitas proposições de arquiteturas de tais redes. Muitas destas proposições objetivam prover garantias de qualidade de Serviço (QoS), o que é essencial para aplicações de tempo real e multimídia. O método mais amplamente usado para obter algum grau de garantia de QoS é dividido em duas etapas. A primeira etapa é caracterizar a aplicação através da modelagem de tráfego e simulação. A segunda etapa consiste em dimensionar uma determinada rede para alcançar garantias de QoS. Projetos de NoCs destinados a atender QoS usualmente provêem estruturas especializadas para permitir ou a criação de conexões (chaveamento por circuito) ou a definição de prioridades para fluxos sem conexão. É possível identificar três desvantagens neste método de duas etapas. Primeiro, não é possível garantir QoS para novas aplicações que venham a ser executadas no sistema, se estas são definidas depois da fase de projeto da rede. Segundo, mesmo com garantias de latência fim-a-fim, métodos sem o estabelecimento de conexão podem introduzir *jitter*. Terceiro, modelar tráfego precisamente para uma aplicação complexa é uma tarefa muito difícil. Se este problema é contornado pela simplificação da fase de modelagem, erros podem ser introduzidos, conduzindo a uma parametrização da NoC pobremente adaptada para atender à QoS requerida. Este documento tem dois principais objetivos. O primeiro é avaliar o compromisso área-desempenho e as limitações do chaveamento por circuito e do escalonamento baseado em prioridades para prover QoS. Esta avaliação mostra quando tais implementações são realmente apropriadas para atender requisitos de QoS, e quando mecanismos mais elaborados são necessários. O segundo objetivo é propor o uso de um escalonamento baseado em taxas para atender requisitos de QoS, considerando o estado da NoC em tempo de execução. A avaliação do chaveamento por circuito e do escalonamento baseado em prioridades mostra que: (*i*) chaveamento por circuito pode garantir QoS somente para um pequeno número de fluxos; esta técnica apresenta baixa escalabilidade e pode desperdiçar largura de banda; (*ii*) escalonamento baseado em prioridades pode apresentar comportamento melhor esforço e, em situações de pior caso, pode conduzir a uma latência inaceitável para fluxos de baixa prioridade, além de ser sujeito a *jitter*. Por estas limitações, o escalonamento baseado em taxas surge com uma opção para melhorar o desempenho de fluxos QoS quando cenários de tráfego variáveis são usados.

Palavras Chave: Qualidade de Serviço (QoS), rede intra-chip (NoC), chaveamento por circuito, escalonamento baseado em prioridades, escalonamento baseado em taxas.

Abstract

The proposition of Networks-on-Chip (NoCs) for modern and future embedded systems capitalizes on the fact that busses present performance degradation when shared by a great number of cores. Even if NoC research is a relatively young field, the literature abounds with propositions of NoC architectures. Several of these propositions claim providing quality of service (QoS) guarantees, which is essential for e.g. real time and multimedia applications. The most widespread approach to attain some degree of QoS guarantee relies on a two-step process. The first step is to characterize application performance through traffic modeling and simulation. The second step consists in tuning a given network template to achieve some degree of QoS guarantee. These QoS targeted NoC templates usually provide specialized structures to allow either the creation of connections (circuit switching) or the assignment of priorities to connectionless flows. It is possible to identify three drawbacks in this two-step process approach. First, it is not possible to guarantee QoS for new applications expected to run on the system, if those are defined after the network design phase. Second, even with end-to-end delay guarantees, connectionless approaches may introduce jitter. Third, to model traffic precisely for a complex application is a very hard task. If this problem is tackled by oversimplifying the modeling phase, errors may arise, leading to NoC parameterization that is poorly adapted to achieve the required QoS. This work has two main objectives. The first one is to evaluate the area-performance trade-off and the limitations of circuit switching and priority scheduling to meet QoS. This evaluation shows where such implementations are really suited for achieving QoS guarantees, and when more elaborate mechanisms to meet QoS are needed. The second objective is to propose the use of a rate-based scheduling to achieve QoS requirements considering the execution time state of the NoC. The evaluation of circuit switching and priority scheduling show that: (i) circuit switching can guarantee QoS only to a small number of flows, this technique presents low scalability and can potentially waste significant bandwidth; (ii) priority-based approaches may display best-effort behavior and, in worst-case situations, may lead to unacceptable latency for low priority flows, besides being subject to jitter. In face of these limitations, rate-based scheduling arises as an option to improve the performance of QoS flows when varying traffic scenarios are used.

Keywords: Quality of Service (QoS), Network-on-Chip (NoC), circuit switching, priority-based scheduling, rate-based scheduling.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ATM	A*synchronous Transfer Mode*
BE	*Best Effort*
CBR	*Constant Bit Rate*
CRC	*Cyclic Redundancy Check*
CSIP	*Currently Serviced Input Port number*
DMA	*Direct Memory Access*
EDD	*Earliest Due Date*
EDF	*Earliest Deadline First*
EP	*End of Packet*
FIFO	*First-In First-Out*
FP	*Full Packet*
FPGA	Field-Programmable Gate Array
FQ	*Fair Queuing*
GT	*Guaranteed Throughput*
HRR	*Hierarchical Round Robin*
HTML	*HyperText Markup Language*
IP	*Internet Protocol*
ITRS	*International Technology Roadmap for Semiconductors*
Mbps	*Megabits por segundo*
MPSoC	*Multi Processor System on a Chip*
MSS	*Maximum Segment Size*
NBS	*Next Buffer State*
NoC	*Network on Chip*
OSI	*Open System Interconnection*
PGPS	*Packet-based Generalized Processor Sharing*
QoS	*Quality of Service*
RED	*Random Early Detection*
RTL	*Register Transfer Level*
RTT	*Round-Trip Time*
SLA	*Service Level Agreement*
SoC	*System on a Chip*
STDM	*Synchronous Time-Division Multiplexing*
TCP	*Transmission Control Protocol*
TDM	*Time-Division Multiplexing*
TLM	*Transaction-Level Modeling*
VBR	*Variable Bit Rate*
VBR-nrt	*Variable Bit Rate - non real time*
VBR-rt	*Variable Bit Rate - real time*
VHDL	*Very High Speed Integrated Circuit Hardware Description Language*
WFQ	*Weighted Fair Queuing*
WRR	*Weighted Round-Robin*
XML	*eXtensible Markup Language*

Sumário

1 INTRODUÇÃO

O aumento do número de transistores e da freqüência de operação, o curto tempo de projeto e a redução do ciclo de vida dos produtos eletrônicos caracterizam o cenário atual da indústria de semicondutores [INT02]. Sob essas circunstâncias, os projetistas estão desenvolvendo circuitos integrados compostos por elementos funcionais heterogêneos e complexos em um único chip, conhecidos como SoCs (do inglês *Systems on a Chip*) [BER01]. Como descrito em Gupta et al. [GUP97] e Bergamaschi et al. [BER01], o projeto de um SoC é baseado no reuso de núcleos de propriedade intelectual. Gupta e Zorian [GUP97] definem núcleo com um módulo de hardware pré-projetado e pré-verificado, que pode ser usado na construção de aplicações complexas. No entanto, núcleos não compõem SoCs sozinhos, eles devem incluir uma arquitetura de interconexão e interfaces para dispositivos periféricos [MAR01].

Usualmente, a arquitetura de interconexão é baseada em *canais ponto-a-ponto* ou *barramento compartilhado*. *Canais ponto-a-ponto* são eficientes para sistemas com um pequeno número de núcleos, mas o número de fios em torno do núcleo aumenta à medida que a complexidade do sistema cresce. Conseqüentemente, canais ponto-a-ponto têm escalabilidade e flexibilidade pobre. Um *barramento compartilhado* é um conjunto de fios comuns para múltiplos núcleos. Esta forma de interconexão possui maior escalabilidade e reusabilidade, quando comparada com canais ponto-a-ponto. Entretanto, barramentos possuem diversas desvantagens [BEN02]: (*i*) apenas uma troca de dados pode ser realizada por vez; (*ii*) existe necessidade de desenvolver mecanismos inteligentes de arbitragem do meio; (*iii*) a escalabilidade é limitada, tipicamente na ordem de poucas dezenas de núcleos conectados ao barramento; (*iv*) o uso de linhas globais em um circuito integrado com tecnologia submicrônica impõe sérias restrições ao desempenho do sistema devido às altas capacitâncias e resistências parasitas inerentes aos fios longos. Estas desvantagens podem ser parcialmente contornadas através do uso de, por exemplo, hierarquia de barramentos, onde o problema continua existindo, sendo apenas minimizado.

De acordo com vários autores, entre eles [KUM02][BEN02][GUE00], a arquitetura de interconexão baseada em barramentos compartilhados não fornecerá suporte aos requisitos de comunicação dos futuros circuitos integrados. De acordo com o ITRS, circuitos integrados serão capazes de conter bilhões de transistores, com tamanho de dispositivos em torno de 50nm e freqüências de relógio em torno de 10GHz em 2012 [INT02]. Neste contexto, uma possível solução pode ser buscada nas redes de computadores e redes de comunicação e aplicada a sistemas embarcados, criando o conceito de redes intra-chip (NoC, do inglês *Network-on-Chip*) [GUE99][KUM02].

Como descrito em [RIJ03][RIJ01], redes intra-chip estão emergindo como uma solução para as restrições existentes em arquiteturas de interconexão, devido às seguintes características: (*i*) eficiência de energia e confiabilidade [BEN01]; (*ii*) escalabilidade de largura de banda quando comparadas a arquitetura de barramento; (*iii*) reusabilidade; (*iv*) decisões distribuídas de roteamento

[BEN02][GUE00]. Nas redes intra-chip, os núcleos do sistema são interligados por meio de uma rede composta por roteadores e canais ponto-a-ponto. A comunicação entre os núcleos ocorre pela troca de mensagens transferidas por meio de roteadores e canais intermediários até atingir o seu destino [BEN02].

Atualmente, a maioria das redes provêem suporte somente ao serviço melhor esforço (BE, do inglês *Best Effort*), incluindo a rede intra-chip comercializada pela companhia Arteris [ART05]. Este serviço não oferece nenhuma garantia temporal às aplicações, há apenas o compromisso de entregar ao receptor a informação enviada pelo emissor. No serviço BE, todas as aplicações são tratadas de forma igual e o envio de pacotes pode sofrer atraso arbitrário. No entanto, de forma bastante generalizada, pode-se dizer que uma mensagem de texto enviada através de correio eletrônico não tem a mesma necessidade de prazo de entrega ao destinatário do que a transmissão de imagem e som sincronizados. O termo QoS (do inglês *Quality of Service*) refere-se à capacidade de uma rede de distinguir fluxos diferentes e prover níveis diferentes de serviço para estes fluxos. Conseqüentemente, serviços BE são inadequados para satisfazer requisitos de QoS de determinadas aplicações/módulos, como no caso, por exemplo, de fluxos multimídia.

Para atender requisitos de desempenho e por conseqüência garantir QoS, a rede necessita incluir características específicas a fim de conhecer a prioridade relativa e os requisitos de cada fluxo na rede, permitindo que recursos sejam alocados de um modo mais eficiente do que se todos os fluxos recebessem exatamente a mesma prioridade [DUA02]. Uma característica necessária, porém não suficiente, para garantir QoS é a utilização de *canais virtuais*, ou seja, o compartilhamento da largura de banda do canal físico entre l canais virtuais. Sem o uso de canais virtuais, quando um pacote aloca um canal físico, nenhum outro pacote pode usá-lo até que este pacote o libere, mesmo que o pacote aguardando possua prioridade maior. O uso de canais permite que mecanismos de alocação de recursos sejam utilizados pela rede, possibilitando que fluxos sejam tratados de forma diferenciada.

As implementações atuais de redes intra-chip que provêem suporte à QoS tentam atender os requisitos de desempenho em *tempo de projeto*. A rede é projetada de acordo com as aplicações, requerendo modelagem e simulação correta do tráfego para obter o desempenho desejado pelas aplicações alvo. Os resultados obtidos através da simulação permitem dimensionar a rede para dar suporte aos requisitos das aplicações. A síntese da rede ocorre após a simulação. Entretanto, é ainda possível que garantias de QoS não sejam conhecidas para novas aplicações. SoCs modernos, tal como telefones 3G, suportam um conjunto de aplicações que varia conforme a utilização do sistema. Projetar a rede para suportar todos os possíveis cenários de tráfego é inviável em termos de tempo de projeto, área e consumo de potência. Logo, novos mecanismos devem ser usados em *tempo de execução* para habilitar a rede a atender os requisitos de QoS para um amplo conjunto de aplicações. Exemplos destes mecanismos são freqüentemente encontrados em redes IP e ATM, como controle de admissão, controle de congestionamento e condicionamento de tráfego. A principal vantagem de usar tais mecanismos é suportar novas aplicações depois do projeto da rede, com um custo extra de área e potência dissipada.

Propostas de redes intra-chip para garantir QoS (por exemplo, [BER05][BOL03][GOO05]) adotam chaveamento por circuito e/ou escalonamento baseado em prioridades na arquitetura de seus roteadores. Entretanto, estas técnicas são implementadas em *tempo de projeto* para alguns cenários de tráfego. Para aplicações reais, onde os fluxos podem desaparecer e reaparecer aleatoriamente ou periodicamente [AND05], esta técnica pode apresentar um erro significativo em tempo de execução. No conhecimento da Autora, não existem implementações de redes intra-chip com mecanismos que atendam os requisitos de QoS em *tempo de execução*.

1.1 Objetivos

O objetivo *estratégico* da presente Dissertação é dominar a tecnologia de redes intra-chip e os mecanismos utilizados para prover QoS aos núcleos conectados a ela.

Dentre os objetivos *específicos* enumera-se:

1. Adicionar canais virtuais à rede intra-chip Hermes [MOR04], possibilitando que mecanismos para atender os requisitos de QoS (como diferenciação e priorização entre fluxos) sejam implementados sobre a rede.
2. Implementar sobre a rede Hermes mecanismos para prover QoS encontrados nas redes intra-chip atuais, como chaveamento por circuito e escalonamento baseado em prioridades.
3. Avaliar a implementação destes mecanismos, a fim de identificar onde tais implementações são realmente apropriadas para atender aos requisitos de QoS, e quando mecanismos mais elaborados são necessários.
4. Implementar sobre a rede Hermes mecanismos mais sofisticados para prover QoS a um conjunto maior de aplicações.
5. Avaliar os mecanismos mais sofisticados para prover QoS frente aos mecanismos usados pelas redes intra-chip atuais.

1.2 Organização do Documento

O presente documento está organizado da seguinte forma. Nos Capítulos 2 e 3 são apresentados os conceitos básicos e a revisão do estado da arte necessários ao entendimento do presente trabalho. No Capítulo 2 são abordados os conceitos que caracterizam as redes intra-chip, as aplicações que utilizam a rede e os serviços oferecidos pela mesma, e também são descritos os mecanismos utilizados para prover QoS. No Capítulo 3 é discutido o estado da arte em redes intra-chip que oferecem QoS e algumas destas redes são detalhadas.

Os Capítulos 4 a 8 apresentam as contribuições deste trabalho. No Capítulo 4 é detalhada a inserção de canais virtuais sobre a rede Hermes. A rede Hermes com canais virtuais é denominada Hermes-VC. Como mencionado anteriormente, o uso de canais virtuais é importante porque permite priorizar a alocação da largura de banda do canal físico, possibilitando à rede fornecer serviços

diferenciados aos fluxos. No entanto, a rede Hermes-VC usa o escalonamento *Round-robin*, que compartilha a largura de banda do canal físico de forma igual entre os canais virtuais e, portanto, sem diferenciar fluxos.

A inserção do chaveamento por circuito sobre a rede Hermes é descrita no Capítulo 5. A rede Hermes que combina chaveamento por pacote e chaveamento por circuito é denominada Hermes-CS. Na Hermes-CS, os fluxos com requisitos de desempenho (fluxos QoS) são transmitidos utilizando chaveamento por circuito, enquanto fluxos com serviço melhor esforço (fluxos BE) são transmitidos utilizando chaveamento por pacote. Desta forma, a rede Hermes-CS é a primeira implementação realizada ao longo deste trabalho que oferece algum grau de QoS.

A diferenciação entre fluxos através do escalonamento baseado em prioridades é apresentada no Capítulo 6. No escalonamento baseado em prioridades, cada fluxo possui uma prioridade e é servido conforme esta prioridade. Dois tipos de escalonamento baseado em prioridades são implementados sobre a rede Hermes: (*i*) escalonamento baseado em prioridades *fixas* (Hermes-FP) e (*ii*) escalonamento baseado em prioridades *dinâmicas* (Hermes-DP).

O Capítulo 7 apresenta a inserção de um escalonador baseado em taxas na rede Hermes (Hermes-RB). Na Hermes-RB, a largura de banda do canal é dividida entre os fluxos, compartilhando o recurso em função da taxa requerida e da taxa atualmente usada por eles. Este escalonador busca compartilhar a largura de banda do canal de um modo mais justo do que o escalonador baseado em prioridades.

O Capítulo 8 apresenta os experimentos que comparam, através da simulação funcional, as redes descritas nos Capítulos anteriores. Os seguintes parâmetros de desempenho são avaliados: latência, *jitter*, espalhamento da latência, vazão e área.

As conclusões e direcionamentos para trabalhos futuros são apresentados no Capítulo 9. O Apêndice I contém a lista das publições realizadas no período do mestrado e o Apêndice II apresenta e detalha o ambiente Atlas, o qual foi desenvolvido para integrar as ferramentas que automatizam os vários processos relacionados ao fluxo de projeto da rede Hermes.

2 CONCEITOS BÁSICOS

Este Capítulo apresenta conceitos básicos relacionados ao desenvolvimento deste trabalho. A Seção 2.1 aborda os conceitos que caracterizam as redes intra-chip. Os conceitos relacionados às aplicações que utilizam a rede e os serviços oferecidos pela mesma são apresentados na Seção 2.2. A Seção 2.3 descreve os conceitos relacionados à qualidade de serviço (QoS).

2.1 Redes Intra-Chip

Esta Seção apresenta conceitos básicos relacionados a redes. São abordados conceitos que caracterizam as redes como conectividade, roteamento, modo de chaveamento, multiplexação e controle de fluxo. Posteriormente, o modelo de referência OSI (do inglês *Open System Interconnection*) é revisado, porque muitas redes são estruturadas em camadas que encapsulam funções equivalentes às definidas por este modelo.

2.1.1 Conectividade

Uma rede de interconexão deve prover conectividade entre um conjunto de nodos. A conectividade da rede ocorre em níveis diferentes. No nível mais baixo, uma rede pode consistir de dois ou mais nodos conectados diretamente por algum meio físico [PET03]. Usualmente, este meio físico é chamado de enlace ou canal. Como ilustrado na Figura 1, canais físicos são algumas vezes limitados a um par de nodos (ponto-a-ponto), enquanto que em outros casos mais de dois nodos compartilham o mesmo canal (barramento).

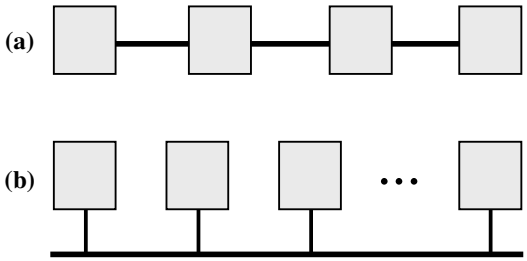


Figura 1 – Conectividade direta: (a) ponto-a-ponto; (b) barramento.

Conectividade entre dois nodos não necessariamente implica em uma conexão direta entre eles. Conectividade indireta pode ser alcançada entre um conjunto de nodos que cooperam. Considere o exemplo ilustrado na Figura 2, onde cada nodo possui um ou mais canais ponto-a-ponto. Para existir conectividade indireta, os nodos que se conectam a no mínimo dois canais (nodos no interior da nuvem) devem ser capazes de transmitir dados recebidos de um canal para outro. Os nodos com esta capacidade são comumente chamados de chaves ou roteadores. Os demais nodos (nodos no exterior da nuvem), comumente chamados de hospedeiros ou núcleos, usam a rede e suportam usuários e aplicações. Dado um conjunto de nodos conectados indiretamente, é possível que um núcleo envie mensagens para um outro através de uma seqüência de canais e roteadores.

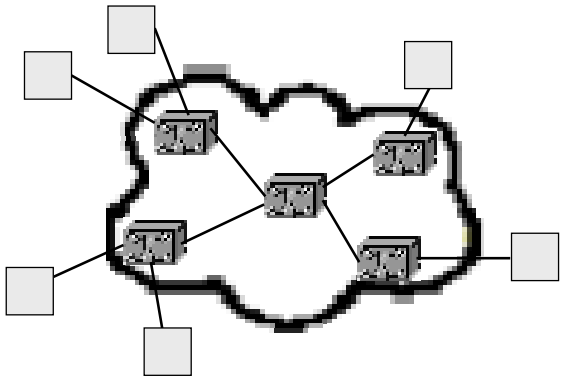


Figura 2 – Conectividade indireta.

2.1.2 Deadlock, Livelock e Starvation

Para assegurar a correta funcionalidade da rede em termos de entrega de mensagens, uma rede deve evitar *deadlock*, *livelock* e *starvation* [DUA97][PAT96]. *Deadlock* pode ser definido como uma dependência cíclica entre dois ou mais nodos requisitando acesso a um conjunto de recursos, de forma que nenhum possa obter progresso algum, independente da seqüência de eventos que ocorra. *Livelock* refere-se a mensagens circulando na rede sem fazer nenhum progresso em direção ao seu destino. Este é um problema que normalmente atinge algoritmos de roteamento adaptativos não mínimos (Seção 2.1.3). Isto pode ser evitado, restringindo o número de desvios que a mensagem pode efetuar. *Starvation* ocorre quando uma mensagem requisita um recurso, sendo bloqueada por um tempo indeterminado porque o recurso está alocado sempre a outra mensagem.

2.1.3 Roteamento

Para que um nodo pertencente a um conjunto de nodos conectados direta ou indiretamente possa se comunicar com qualquer outro nodo do conjunto, é necessário que um endereço seja atribuído a cada nodo. Um endereço identifica de forma única um nodo. Quando uma origem quer que a rede entregue uma mensagem a um determinado nodo destino, ela especifica o endereço do destino. Se os nodos origem e destino não estão conectados diretamente, então os roteadores da rede usam este endereço para decidir como a mensagem é enviada na direção do destino. O processo que determina como enviar mensagens na direção do nodo destino baseado em seu endereço é chamado roteamento. Algoritmos de roteamento podem ser classificados de acordo com vários critérios tais como: (*i*) número de destinos de cada mensagem; (*ii*) local de decisão de roteamento; (*iii*) adaptabilidade; e (*iv*) minimalidade.

2.1.3.1 Número de Destinos

Os algoritmos de roteamento podem ser classificados como *unicast* ou *multicast* de acordo com o número de destinos [DUA02]. Um algoritmo de roteamento é dito *unicast* quando as mensagens têm um único destino. Quando as mensagens podem ter mais de um destino, o algoritmo de roteamento é denominado *multicast*. Um caso particular é o *broadcast*, onde todos os possíveis destinos são destinos da mensagem.

2.1.3.2 Decisão de Roteamento

Segundo Mohapatra [MOH98], de acordo com o local de decisão de roteamento, algoritmos podem ser classificados como *roteamento na origem* ou *roteamento distribuído*. Em algoritmos de *roteamento na origem* o caminho é decidido no roteador origem, ou no núcleo, antes da mensagem ser enviada. As desvantagens desta abordagem são: (*i*) o cabeçalho da mensagem deve carregar todas as informações de roteamento; (*ii*) o método não é tolerante a falhas, ou seja, caso um caminho apresente defeito, todas as mensagens que deveriam passar por este caminho serão bloqueadas. Em algoritmos de *roteamento distribuído*, cada roteador que recebe a mensagem decide para onde irá enviá-la. A decisão é feita a partir, por exemplo, do endereço de destino presente no cabeçalho da mensagem.

2.1.3.3 Adaptabilidade

Algoritmos de roteamento também podem ser classificados como *determinísticos* ou *adaptativos*. Um algoritmo de *roteamento determinístico*, também chamado roteamento *oblivious* [MOH98], provê sempre o mesmo caminho entre um par origem-destino. Isto ocorre porque ele decide qual o caminho a ser tomado em função apenas dos endereços do roteador atual e do roteador destino. Um algoritmo de *roteamento adaptativo* decide qual o caminho a ser tomado em função do tráfego da rede [DUA02]. A vantagem desse método é que o pacote tem mais de uma alternativa para chegar ao destino. Como desvantagem pode-se citar a possibilidade de entrega de pacotes fora de ordem, caso pacotes com mesmo par origem-destino percorram caminhos distintos. O roteamento adaptativo é subdividido em *parcialmente adaptativo* e *totalmente adaptativo*. No método *parcialmente adaptativo*, apenas um subconjunto dos caminhos físicos disponíveis é alocado para a comunicação. No método *totalmente adaptativo*, todos os caminhos físicos da rede podem ser utilizados para enviar uma dada mensagem, podendo, entretanto ocorrer *deadlock*.

2.1.3.4 Minimalidade

Quanto a minimalidade, algoritmos de roteamento podem ser classificados como sendo *mínimos* ou *não mínimos*. Quando o algoritmo de roteamento é *mínimo*, a mensagem é transmitida por um dos menores caminhos entre o par origem-destino. A vantagem é a garantia que a cada chaveamento a mensagem se aproxima do destino, portanto evitando a ocorrência de *livelock*. Quando o algoritmo de roteamento é *não mínimo*, a mensagem pode escolher caminhos diferentes dos caminhos mínimos. A vantagem é a maior disponibilidade de alternativas de caminhos em relação aos roteamentos mínimos. A desvantagem é que isso pode causar *livelock*.

2.1.4 Modos de Chaveamento

Para que um roteador transfira dados de um canal da rede para outro, é necessário que um modo de chaveamento seja adotado. Os modos mais utilizados são *chaveamento por circuito* e *chaveamento por pacotes* [HWA93] [MOH98].

No *chaveamento por circuito*, uma conexão é estabelecida antes do início da transmissão dos dados. Para a conexão ser estabelecida, os recursos requisitados pela aplicação devem estar disponíveis em todos os roteadores no caminho da conexão [KUM04]. Redes que utilizam este método de chaveamento, em princípio, não necessitam de *buffers* intermediários, porque após o estabelecimento da conexão os dados não são bloqueados. Conseqüentemente, estas redes provêem QoS porque garantem que os dados são entregues ao destino com as mesmas características em que são inseridos pela origem. Porém, como a maioria das aplicações não gera dados em uma taxa constante, mas tipicamente envia dados em rajada, este método pode causar a degradação do desempenho do sistema como um todo, porque, mesmo no período em que a aplicação não está enviando dados, os recursos permanecem alocados.

No *chaveamento por pacotes*, um fluxo[1] é quebrado em frações denominadas pacotes, que são transmitidos através da rede [MOH98]. Cada pacote é individualmente roteado da origem até o destino, devendo carregar um cabeçalho que contém informações de roteamento e de controle [DUA02]. A taxa média na qual as origens transmitem dados deve ser menor do que a capacidade do canal. No entanto, a taxa máxima total na qual as origens podem transmitir dados tipicamente excede a capacidade do canal. Neste caso, os dados em excesso devem ser armazenados em um *buffer*, e tal armazenamento ocasiona atraso na transmissão dos dados (aumento da latência). Se não há espaço disponível no *buffer* para armazenar o dado, então este pode ser perdido. Quando uma rede utiliza chaveamento por pacotes, os requisitos de QoS são usualmente expressos em termos de latência máxima ou probabilidade máxima de perda de pacotes [KUM04].

O chaveamento por pacotes requer o uso de uma política de repasse de pacotes, a qual define como os pacotes se movem através dos roteadores. As três políticas mais utilizadas são *store-and-forward*, *virtual-cut-through* e *wormhole* [RIJ01][MOH98]. Na política *store-and-forward*, um roteador não pode enviar adiante um pacote até que este tenha sido completamente recebido. Cada vez que um roteador recebe um pacote, seu conteúdo é examinado para decidir o que fazer, implicando uma latência por roteador proporcional ao tamanho do pacote. Na política *virtual-cut-through*, um roteador pode enviar um pacote adiante assim que o roteador seguinte der uma garantia que o pacote será aceito completamente [RIJ01]. Logo, é necessário um *buffer* para armazenar pelo menos um pacote completo, como no *store-and-forward*. Contudo, neste caso a latência de comunicação é menor. *Wormhole* é uma variação do *virtual-cut-through* que evita a necessidade de grandes espaços de memória no roteador. Um pacote é transmitido entre roteadores em unidades chamadas *flits*[2]. Somente o *flit* de cabeçalho tem a informação de roteamento. Logo, os demais *flits* que compõem um pacote devem seguir o mesmo caminho reservado pelo cabeçalho. Freqüentemente, em redes intra-chip, o tamanho de um *flit* é igual à largura do canal físico (ou *phit*).

[1] Um *fluxo* pode ser todas as mensagens geradas por uma determinada origem, aquelas sendo transmitidas em direção a um destino comum ou todas as mensagens enviadas por uma aplicação [DUA02].
[2] Segundo Dally [DAL04], *flit* é a menor unidade de informação sobre a qual se realiza controle de fluxo.

2.1.5 Multiplexação

Para atravessar a rede, uma mensagem deve alocar recursos: canais e *buffers* [DAL04]. Tipicamente, um único *buffer* é associado a cada canal. Nestes casos, quando uma mensagem aloca um *buffer*, nenhuma outra mensagem pode utilizar o canal associado ao mesmo até que esta mensagem o libere. Alternativamente, um canal físico pode ser multiplexado em *l* canais lógicos (canais virtuais). Um canal virtual tem seu próprio *buffer*, mas compartilha a largura de banda do canal físico com outros canais virtuais [SAR00]. A Figura 3 ilustra o conceito, mostrando dois canais virtuais compartilhando o mesmo canal físico. Cada canal virtual funciona como se estivesse utilizando um canal físico distinto operando na metade da velocidade, ou seja, a taxa na qual cada mensagem é transmitida é cerca da metade da taxa alcançada quando o canal não é compartilhado [DUA02].

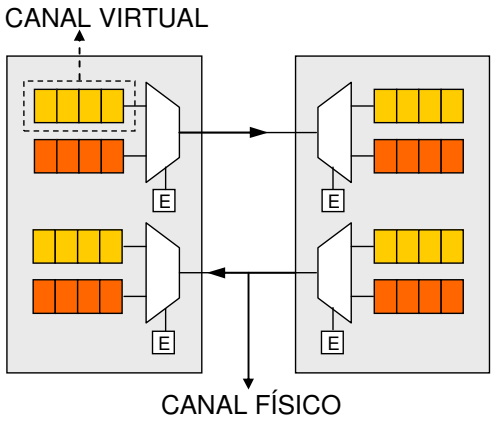


Figura 3 – Multiplexação do canal físico em dois canais virtuais. E representa o controle utilizado para determinar qual canal virtual tem acesso ao canal físico a cada instante.

Existem diferentes formas de dividir a largura de banda do canal físico entre canais virtuais. Um método comumente utilizado é a multiplexação por divisão síncrona de tempo (STDM, do inglês *Synchronous Time-Division Multiplexing*) [PET03]. O princípio do STDM é dividir o canal físico em fatias de tempo, onde cada fatia é usada por um canal virtual específico. Sua principal desvantagem é que, quando um canal virtual não tem dados para transmitir, sua capacidade é desperdiçada, mesmo se outros canais virtuais possuem dados para transmitir. Outra forma de multiplexação é a multiplexação estatística. Como STDM, a multiplexação estatística também divide o canal físico em fatias de tempo, porém os dados são transmitidos sob demanda ao invés de em fatias de tempo pré-determinadas. Logo, se somente um canal virtual tem dados para transmitir, ele transmitirá os dados sem esperar por sua fatia de tempo, podendo assim ocupar toda a largura de banda do canal físico. Na multiplexação estatística, a largura de banda do canal físico pode ser alocada usando um algoritmo de arbitragem (aleatório, *round-robin* ou prioridade) ou um escalonamento em função da idade do pacote ou do tempo limite para o pacote alcançar o destino. A possibilidade de priorizar a alocação de canais virtuais pode ser usada para prover diferentes classes de serviço, permitindo à rede prover qualidade de serviço [MUL04].

2.1.6 Controle de Fluxo

Controle de fluxo é um protocolo de sincronização para transmissão e recepção de informação [DUA02]. Este protocolo possibilita ao transmissor saber se o receptor está habilitado a receber dados (o receptor pode estar com os *buffers* de recepção cheios ou com algum problema momentâneo que o impossibilita de receber dados). Dally [DAL04] descreve três tipos de controle de fluxo como sendo os mais utilizados: (*i*) *credit based*, (*ii*) *on/off* e (*iii*) *ack/nack*.

No controle de fluxo baseado em créditos (*credit based*), quando um roteador deseja enviar um dado para um roteador vizinho, ele verifica se o canal selecionado para a transmissão dos dados possui créditos, ou seja, se o *buffer* associado ao canal possui espaço disponível para armazenamento do dado. Enquanto existir espaço disponível no *buffer*, os dados são armazenados e os créditos são decrementados no destino. Quando o número de créditos alcançar zero, nenhum dado adicional pode ser recebido. Os créditos são incrementados quando um dado é consumido pelo roteador destino, ou seja, o roteador transmite um dado do canal ao núcleo local ou a um roteador vizinho, disponibilizando espaço no *buffer*.

No controle de fluxo *On/Off*, um único bit de controle indica se o roteador pode receber dados (*on*) ou não (*off*). Um sinal é transmitido pelo roteador ao roteador vizinho quando é necessário trocar este estado (*on/off*). Um sinal *off* é transmitido quando o bit de controle é *on* e o número de espaços ocupados é maior que o limite máximo estipulado. Se o bit de controle é *off* e o número de espaços ocupados está abaixo do limite mínimo, um sinal *on* é enviado. A Figura 4 ilustra a operação do *buffer* com o uso do controle de fluxo *On/Off*.

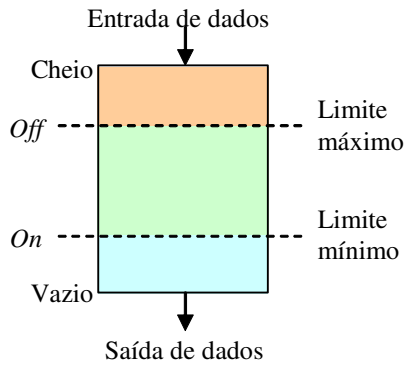


Figura 4 – Operação do *buffer* com o uso do controle de fluxo *On/Off*.

No controle de fluxo *Ack/Nack* não existe controle da disponibilidade de espaço no *buffer*. O roteador origem transmite dados no momento em que esses se tornam disponíveis. Se o roteador destino tem espaço disponível no *buffer*, o dado é aceito e o roteador destino envia um sinal de confirmação (*ack*) ao roteador origem. Se não há espaço disponível quando o dado é transmitido, o roteador destino descarta o dado e envia o sinal de confirmação negado (*nack*). O roteador origem armazena o dado até receber um *ack*. Se um *nack* é recebido, o dado é retransmitido. Segundo Dally [DAL04], o controle de fluxo *Ack/Nack* é menos eficiente no uso dos *buffers* do que o controle de fluxo baseado em créditos. Esta redução da eficiência ocorre porque os dados ficam armazenados nos *buffers* por um tempo adicional à espera do *ack*. O controle de fluxo *Ack/Nack* também é

ineficiente no uso da largura de banda quando não há espaço no *buffer* e este transmite dados que são descartados.

2.1.7 Modelo de Referência OSI

Quando uma aplicação necessita se comunicar com outra, existem muitos serviços a serem executados além do simples envio da mensagem de um nodo para o outro. Uma opção é o projetista da aplicação implementar estes serviços dentro de cada aplicação. No entanto, como muitas aplicações necessitam serviços comuns, é mais lógico implementar estes serviços comuns uma única vez e então conduzir o projetista a construir a aplicação utilizando estes serviços. O desafio do projetista da rede é identificar o conjunto correto de serviços comuns, objetivando esconder a complexidade da rede das aplicações, sem restringir demasiadamente o projetista [PET03].

O modelo de referência OSI (do inglês *Open System Interconnection*) é uma estrutura hierárquica de camadas de protocolos que define os requisitos para comunicação entre terminais de uma rede [DAY83]. Cada camada oferece um conjunto de serviços à camada superior, usando funções disponíveis na mesma camada e nas camadas inferiores. O modelo OSI possui sete camadas, como pode ser observado na Figura 5. As camadas de transporte e superiores são tipicamente implementadas apenas em terminais (núcleos), enquanto que as três camadas inferiores (física, enlace e rede) são implementadas em terminais e elementos da rede. As camadas inferiores são descritas abaixo.

Figura 5 – Camadas do modelo OSI.

As camadas inferiores possuem as seguintes características:

- *Camada física*: realiza a transferência de dados em nível de bits através de um canal.
- *Camada de enlace*: efetua a comunicação em nível de quadros (grupos de bits). Preocupa-se com o enquadramento dos dados e com a transferência desses quadros de forma confiável, realizando o tratamento de erros e o controle do fluxo de transferência de quadros.
- *Camada de rede*: faz a comunicação em nível de pacotes (grupos de quadros). Responsável pelo empacotamento das mensagens, roteamento dos pacotes entre a origem e o destino da mensagem, controle de congestionamento e contabilização de pacotes transferidos.

2.2 Aplicações e Serviços

Para projetar um sistema que sirva tanto à satisfação de QoS quanto à eficiência de utilização de recursos, é importante classificar as aplicações do sistema de acordo com seus requisitos de QoS e mapear seus dados para níveis de serviço apropriados. Este trabalho propõem a classificação ilustrada na Figura 6. Os parâmetros de desempenho utilizados para quantificar os requisitos de QoS das aplicações são apresentados na Seção 2.2.1. A classificação das aplicações é detalhada na Seção 2.2.2. A Seção 2.2.3 descreve os níveis de serviços e os acordos de nível de serviços são apresentados na Seção 2.2.4.

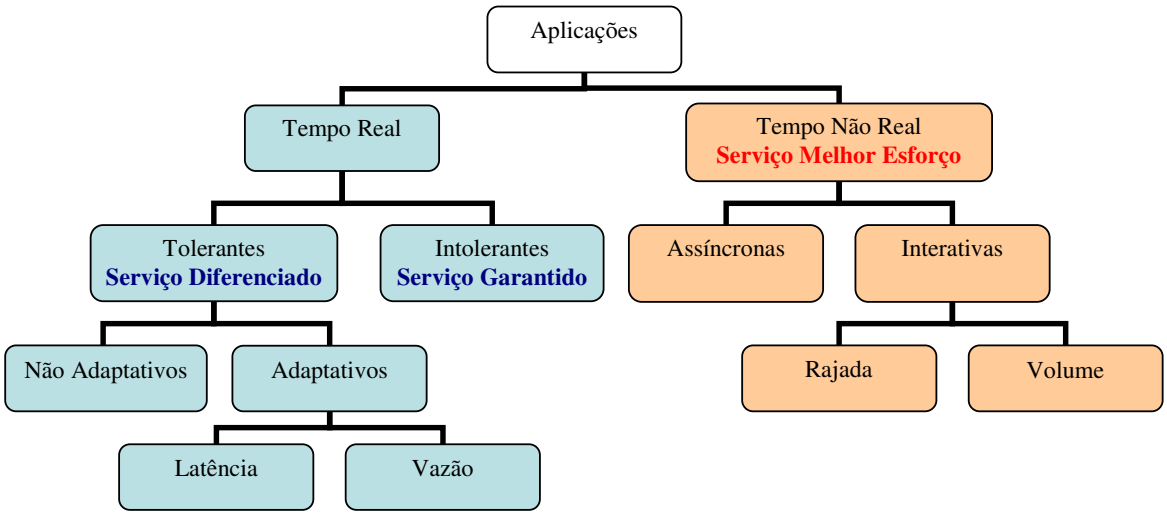


Figura 6 – Classificação das aplicações versus níveis de serviço. Em negrito, os tipos de serviço normalmente associado ao tipo de aplicação.

2.2.1 Parâmetros de Desempenho

Garantias de Serviço são quantificadas em termos de um ou mais parâmetros de desempenho. Geralmente, parâmetros de QoS incluem vazão, latência, *jitter* e taxa de perda.

A *vazão* diz respeito à taxa na qual pacotes são encaminhados pela rede [DAL04], ou seja, a quantidade de informação encaminhada por unidade de tempo. A vazão é medida através da contagem do número de bits que chegam aos destinos em um intervalo de tempo para cada fluxo (par origem-destino), computando a partir desta taxa a fração do fluxo encaminhado. Alternativamente, a vazão pode ser medida como uma fração da capacidade do canal. Uma rede com carga uniforme opera no limite de sua capacidade se o canal de comunicação é utilizado 100% do tempo [DUA02].

A *latência* é o intervalo de tempo entre o momento que a transmissão de um pacote é iniciada até este pacote ser recebido no nodo destino. Também pode ser definida como sendo o tempo necessário para um pacote atravessar a rede do roteador origem até o roteador destino [DAL04]. Segundo Duato [DUA02], a definição de latência é vaga e pode ser interpretada de maneiras diferentes. Aqui, a latência é definida com o tempo decorrido entre a injeção do pacote no

buffer do núcleo origem até o momento em que o último dado deste pacote é entregue ao núcleo destino, ou seja, o tempo em que o pacote permanece no *buffer* do núcleo origem é computado no cálculo da latência. A latência é medida em unidades de tempo.

Jitter é a variação na latência entre dois pacotes adjacentes dentro de um determinado fluxo. Todas as redes apresentam certa quantidade de *jitter*, devido aos atrasos diferenciados enfrentados pelos pacotes em cada roteador da rede. Quando o *jitter* é controlado, é possível manter a qualidade de serviço. *Jitter* baixo é freqüentemente um requisito para aplicações de tempo real, para as quais a regularidade do intervalo de dados é importante [DAL04]. A redução do efeito *jitter* requer que os pacotes sejam armazenados por tempo suficiente em *buffers*, o que é igualmente um problema. É requisito do projetista da rede determinar um ponto de equilíbrio na configuração.

Perda é a fração de pacotes descartados pela rede. Para algumas aplicações, nenhuma perda de pacote é permitida. No entanto, para outras, uma pequena fração do fluxo pode ser perdido sem afetar o desempenho [DAL04]. Do ponto de vista de aplicações em tempo real, um pacote que chega ao destino depois de um limite de latência determinado pode ser considerado como perda, pois não atende aos requisitos temporais [SHI03]. A perda pode ser conseqüência de erros provenientes do meio de transmissão físico ou do congestionamento da rede. Quanto maior o congestionamento da rede, maior o número de pacotes que poderão ser descartados.

2.2.2 Classificação das Aplicações

Como pode ser observado na Figura 6, as aplicações são classificadas em dois grandes grupos: *aplicações de tempo real* e *aplicações de tempo não real*.

2.2.2.1 Aplicações de Tempo Real

Aplicações de tempo real são aquelas que têm uma relação temporal rígida entre a origem e o destino. Uma informação recebida no destino depois de um limite de tempo determinado é considerada inútil e é descartada. Este limite de tempo deve ser conhecido tanto pela origem quanto pelo destino.

Como um exemplo concreto de aplicação de tempo real, considere uma aplicação de áudio digital. Os dados são gerados coletando amostras de um microfone e digitalizando-as usando um conversor analógico-digital. As amostras digitais são inseridas em pacotes, os quais são transmitidos através da rede e recebidos em um terminal. No destino, os dados devem ser reproduzidos em uma taxa apropriada. Por exemplo, se a amostra de voz foi coletada a uma taxa típica de uma a cada 125μs, eles devem ser reproduzidos na mesma taxa. Então, cada amostra tem um tempo particular de reprodução. No exemplo da voz, cada amostra tem um tempo de reprodução que é 125μs após a amostra precedente. Se dados chegam depois do momento apropriado de reprodução, ou porque ele foi atrasado na rede ou porque ele foi descartado e subseqüentemente retransmitido, ele é considerado inútil. A completa inutilidade de dados com atraso é que caracteriza aplicações de tempo real.

Um modo de fazer a aplicação de voz funcionar é tornar seguro que todas as amostras

levem exatamente a mesma quantidade de tempo para atravessar a rede. Deste modo, se as amostras são injetadas a uma taxa de 1 por 125μs, então elas irão ser entregues no receptor na mesma taxa, prontas para serem reproduzidas. Entretanto, é geralmente difícil garantir que todos dados atravessando uma rede chaveada por pacotes possuam exatamente a mesma latência. Pacotes são armazenados em *buffers* nos roteadores e a ocupação destes *buffers* podem variar com o tempo, significando que a latência tende a variar com o tempo e ser potencialmente diferente para cada pacote no fluxo de áudio. Uma forma de resolver este problema é armazenar os dados no receptor final e reproduzi-los somente no tempo certo.

Observar a Figura 7. Quando um pacote chega ao destino, ele é armazenado em um *buffer* até seu tempo de reprodução chegar. Quanto menor a latência do pacote na rede, maior será o tempo de espera no *buffer*. Quanto maior a latência do pacote na rede, menor será o tempo de espera no *buffer*. Logo, um contrabalanço constante é adicionado ao tempo de geração de todos os pacotes, como uma forma de garantia. Esse contrabalanço é chamado de ponto de reprodução.

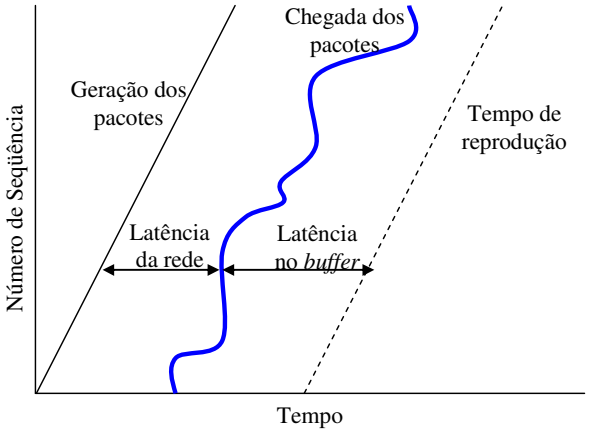


Figura 7 – Contrabalanço entre o tempo de chegada dos pacotes e o tempo de reprodução.

As aplicações de tempo real podem ser classificadas como *tolerantes* ou *intolerantes* (Figura 6) dependendo se podem ou não desrespeitar, dentro de certos limites, os requisitos de QoS. As aplicações *intolerantes* são aquelas que possuem requisitos rígidos de QoS, que uma vez violados, a aplicação produz resultados com qualidade inaceitável. Por oposição, as aplicações *tolerantes* exigem um determinado nível de QoS, mas cada parâmetro de QoS pode ter uma certa perda admissível, sem por isso inutilizar o resultado.

As aplicações tolerantes podem ser divididas em *adaptativas* e *não adaptativas*. Uma aplicação *adaptativa* é capaz de, por exemplo, variar o ponto de reprodução de acordo com a latência da rede. Se pacotes estão quase sempre chegando ao destino num limite de 300ms após serem transmitidos pela origem, então o ponto de reprodução pode ser definido de acordo com este limite. As aplicações adaptativas podem ser ajustadas de acordo com a latência, como no exemplo acima, ou de acordo com a taxa. No segundo caso, as aplicações podem manter um compromisso entre a taxa de transmissão de bits e a qualidade. Por exemplo, os parâmetros de uma aplicação podem ser definidos de acordo com uma determinada largura de banda da rede. Se mais largura de banda torna-se disponível, a aplicação pode mudar seus parâmetros para aumentar a qualidade.

2.2.2.2 Aplicações de Tempo Não Real

Diferente das aplicações de tempo real, as aplicações de tempo não real não exigem taxas rígidas para alcançar a qualidade de serviço necessária. Para estas aplicações, a recepção correta dos dados é mais importante do que a sua transferência em uma taxa constante, embora possam se beneficiar de latências menores [PET03]. Exemplos de aplicações de tempo não real são: correio eletrônico, transferência de arquivos, consultas interativas a informações e aplicações cliente/servidor tradicionais. As aplicações de tempo não real incluem aplicações *assíncronas* e *interativas*, conforme ilustrado na Figura 6.

Aplicações *assíncronas* são relativamente insensíveis ao tempo. Um exemplo de uma aplicação assíncrona é o correio eletrônico. Quando uma mensagem de correio eletrônico é enviada, a origem não recebe nenhuma confirmação do destino, exceto quando existe um erro na informação do destino ou quando a origem solicita notificação de recebimento. Aplicações assíncronas têm pouco ou nenhum efeito na arquitetura e no projeto da rede e usualmente são ignoradas [MCC03].

Aplicações *interativas* assumem alguma relação temporal entre a origem e o destino enquanto um fluxo está sendo transmitido. Entretanto, a relação temporal não é tão rígida quanto em aplicações de tempo real. Aplicações interativas exigem que seus dados sejam entregues ao destino tão rápido quanto possível.

Aplicações interativas podem ser divididas em dois tipos de interatividade: *rajada* e *volume*. A diferença entre esses dois tipos é sutil. Aplicações interativas do tipo *rajada* são aquelas que interagem rápida e freqüentemente com um usuário, onde a latência fim-a-fim da rede é a latência predominante [MCC03]. Devido à sensibilidade à latência da rede, este tipo de aplicação deve ser identificado para que a arquitetura e o projeto da rede satisfaçam seus requisitos de latência. Um exemplo de uma aplicação interativa do tipo rajada é o *telnet*. Aplicações interativas do tipo *volume* são aquelas onde o processamento realizado pela aplicação possui a latência predominante. É importante salientar que, os requisitos de latência da rede para estas aplicações tornam-se menos significativos, porque os mesmos são escondidos pela latência da aplicação. Por exemplo, quando uma aplicação tem latência de processamento alta (na ordem de segundos), a flexibilidade para suportar a latência da rede é maior do que se a latência de processamento fosse na ordem de microssegundos. Um exemplo de uma aplicação interativa do tipo volume é a transferência de arquivos (FTP).

2.2.3 Níveis de Serviço

Diferentes níveis de serviço, também chamados de modelos de serviço, envolvem diferentes compromissos entre garantias de QoS e utilização de recursos. Três níveis de serviços são descritos a seguir: *garantido, diferenciado e melhor esforço*.

2.2.3.1 Serviço Garantido

No nível de serviço *garantido*, também chamado de *serviço determinístico* ou *serviço integrado*, existe um contrato entre a rede e o cliente [DUA02]. O lado cliente do contrato

determina as características do fluxo, por exemplo, a vazão máxima oferecida. O lado rede aceita o contrato se possuir recursos suficientes para garantir o desempenho requisitado pelo cliente. Geralmente, o contrato é acertado antes da transferência de dados, durante o processo de estabelecimento de conexão[3], e é mantido ao longo do tempo de vida da conexão. Para aumentar a dinamicidade e flexibilidade, este nível de serviço pode ser estendido para permitir que o contrato seja modificado durante a existência de uma conexão. Uma conexão é estabelecida com base nas características das conexões existentes.

No serviço garantido, se a rede está funcionando e o cliente está transmitindo dados com as características estabelecidas no contrato, então o serviço é atendido. Este nível de serviço não requer que os outros clientes da rede concordem com suas características do fluxo. Por esse motivo, o serviço garantido é indicado para aplicações de tempo real intolerantes, rígidas, que não permitam perda de pacotes e que necessitam de garantias absolutas sobre o serviço que recebem [CLA92].

Serviços garantidos exigem reserva de recursos para os cenários de pior caso, o que pode ser caro. Observe a Figura 8. Para garantir vazão a um fluxo, deve-se reservar largura de banda para sua vazão máxima, mesmo quando sua vazão média é muito mais baixa. Como conseqüência, quando serviços garantidos são usados, os recursos são freqüentemente subutilizados [RIJ03]. Além da baixa utilização dos recursos, os serviços garantidos necessitam armazenar o estado de cada fluxo nos roteadores ao longo do caminho reservado. Por isto o serviço garantido possui baixa escalabilidade.

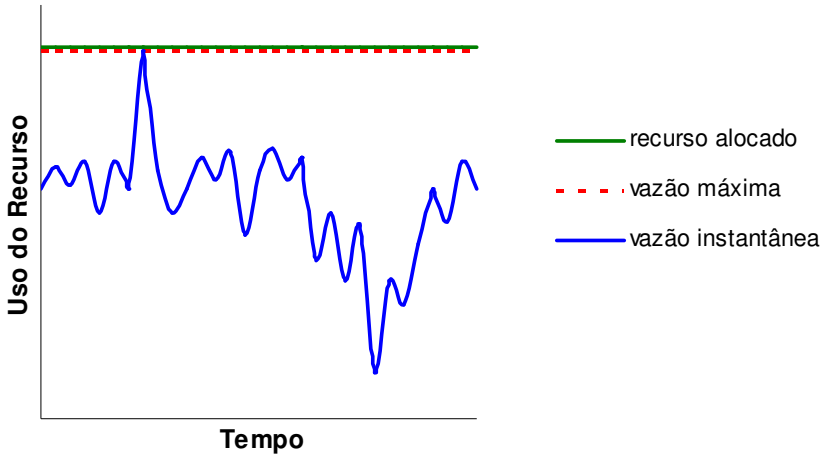


Figura 8 – Serviço garantido: alocação dos recursos versus uso dos recursos.

2.2.3.2 Serviço Diferenciado

Serviço *diferenciado* – também chamado de serviço probabilístico, serviço previsível ou serviço estatístico – explora a característica de algumas aplicações tolerantes, que são capazes de adaptar-se a variações modestas no desempenho da rede [ZHA95]. Ele permite perdas controladas. Controlada significa que o cliente admite algum nível de perda para o qual a rede fez um

[3] Neste contexto, o termo conexão não necessariamente implica em chaveamento por circuito. Ele é usado em um nível mais alto de abstração.

compromisso. Dependendo do compartilhamento do canal físico entre os canais virtuais e da proporção de perdas, a utilização dos recursos da rede pode ser alta.

Um serviço diferenciado é mais flexível e possui maior escalabilidade do que o serviço garantido, porque não necessita armazenar informação de estado para cada fluxo. O serviço diferenciado distingue classes de fluxos para as quais os recursos são alocados, ao invés de distinguir fluxos individuais [SHI03]. Pacotes são divididos em classes, e geralmente carregam a informação da classe a que pertencem em seu cabeçalho.

O serviço diferenciado possui duas importantes diferenças em relação ao serviço garantido. Primeiro, quanto ao estabelecimento da conexão. No serviço diferenciado, a conexão é estabelecida com base em medidas, ao contrário do serviço garantido, que se baseia nas características das conexões existentes. Como a carga medida da rede pode variar, o acordo de serviço para serviços diferenciados é menos confiável. Segundo, no serviço garantido, somente o cliente pode alterar as características da conexão, enquanto que no serviço diferenciado, as características da conexão são determinadas pela rede e podem variar conforme a carga desta [ZHA95].

2.2.3.3 Serviço Melhor Esforço

Serviço *melhor esforço* significa que todos os pacotes são tratados igualmente [SHI03]. Um serviço melhor esforço não possui nenhum contrato com a rede, exceto a promessa de não atrasar ou descartar pacotes desnecessariamente [CLA92]. Serviços melhor esforço usam bem os recursos porque são tipicamente projetados para cenários de caso médio em lugar de cenários de pior caso. Eles também são fáceis e rápidos para usar, porque não exigem reserva de recursos [RIJ03]. Sua principal desvantagem é que estes pacotes podem ter atraso arbitrário ou até mesmo serem descartados [DUA02]. Devido a estas características, o serviço melhor esforço normalmente é utilizado para transportar dados de aplicações de tempo não real.

2.2.4 Acordos de Nível de Serviço

Acordos de nível de serviço (SLA, do inglês *Service-Level Agreements*) são contratos formais entre a rede e um cliente. Esses contratos especificam os direitos e responsabilidades, os benefícios que serão oferecidos, as condições sobre as quais os benefícios serão garantidos, e as penalidades caso as promessas mútuas não sejam mantidas. Neste sentido, a rede tem a responsabilidade de monitorar e gerenciar os serviços, para assegurar que os clientes estão recebendo o que eles esperam e que os clientes estão cientes do que está disponível para eles [MCC03].

Os SLA especificam níveis de latência, vazão e confiabilidade[4] para vários tipos de clientes. A Tabela 1 é um exemplo para ilustrar como um SLA pode ser estruturado. Um serviço básico é aquele onde todas as características de desempenho são melhor esforço. Este serviço é o padrão e pode normalmente ser provido para qualquer cliente. Os outros níveis de serviço (prata, ouro e platina) especificam níveis crescentes de desempenho de vazão, latência e confiabilidade.

[4] Confiabilidade = 1 – taxa de perda

Dependendo dos requisitos dos clientes da rede, os SLAs podem ser mais simples, com dois ou três níveis de serviço, ou mais complexos do que o exemplo apresentado.

Tabela 1 – Exemplo de descrição de níveis de serviço [MCC03].

Serviço	Desempenho		
	Vazão	**Latência**	**Confiabilidade**
Básico	Como disponível (BE)	Como disponível (BE)	Como disponível (BE)
Prata	1,5 Mb/s	Como disponível (BE)	Como disponível (BE)
Ouro	10 Mb/s	Máximo 100 ms	Como disponível (BE)
Platina	100/10 Mb/s	Máximo 40 ms	99.999%

2.3 Qualidade de Serviço (QoS)

A implementação de QoS em redes apóia-se sobre a existência de mecanismos de condicionamento de fluxos, alocação de recursos, armazenamento e escalonamento de pacotes e controle de congestionamento, como ilustrado na Figura 9. Condicionamento de fluxo é um conjunto de mecanismos que modificam o desempenho de determinados fluxos. Mecanismos de condicionamento de fluxo são detalhados na Seção 2.3.1. Mecanismos de alocação de recursos reservam recursos da rede para que fluxos possam receber garantias de QoS. A alocação de recursos é discutida na Seção 2.3.2. Mecanismos de escalonamento determinam a ordem na qual fluxos são processados para transmissão e a ordem na qual são descartados. Tais mecanismos geralmente incluem *buffers* para armazenar os dados. Mecanismos de armazenamento e escalonamento são descritos na Seção 2.3.3. Mecanismos de controle de congestionamento necessitam ser implementados para tratar condições de sobrecarga dos *buffers* sem alterar os requisitos de QoS. Estes mecanismos são apresentados na Seção 2.3.4.

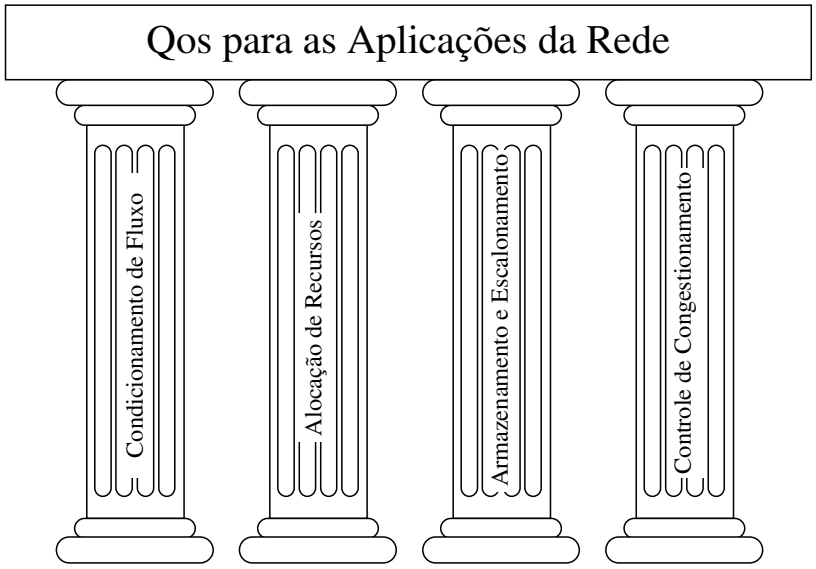


Figura 9 – Mecanismos que sustentam o QoS para as aplicações executando na rede.

2.3.1 Condicionamento de Fluxo

Uma aplicação negocia sua QoS e os requisitos de largura de banda via um contrato. O mecanismo de alocação de recursos reserva a largura de banda suficiente para atender os requisitos de QoS do fluxo. Entretanto, a alocação de recursos não é suficiente para assegurar que a QoS é atendida. Um fluxo poderia (intencionalmente ou acidentalmente) exceder o fluxo contratado e provocar degradação da QoS. Mais importante, este fluxo pode afetar a QoS de outros fluxos bem comportados. Desde modo, a rede deve garantir QoS pelo menos para os fluxos que estão de acordo com o contrato. Para isto, uma rede utiliza mecanismos de *condicionamento de fluxo*.

Condicionamento de fluxo é um conjunto de mecanismos que modificam (aumentam ou diminuem) o desempenho de determinados fluxos [MCC03]. O condicionamento de fluxo normalmente é implementado somente nos terminais da rede. Esse mecanismo envolve a classificação, medição e uma subseqüente ação, para os pacotes que não estão dentro do perfil contratado. Funções de condicionamento de fluxo são apresentadas na Figura 10.

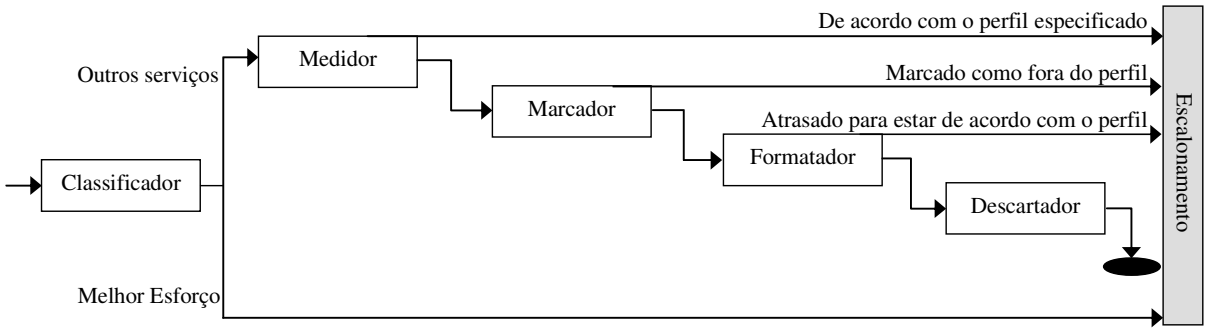


Figura 10 – Funções do condicionamento de tráfego.

A *classificação* é a habilidade de identificar fluxos. A classificação geralmente é realizada através da inspeção dos campos de cabeçalho, mas pode também se basear nos endereços de origem e destino ou no número de portas. Quando, o pacote está usando o serviço melhor esforço, ele é encaminhado sem nenhum processamento adicional. Caso o pacote pertença a outro serviço, ele é encaminhado para a fase de medição.

Medir um fluxo permite determinar se um fluxo está dentro ou fora dos limites de desempenho estipulado. Embora vários mecanismos de medição possam ser utilizados, o mais utilizado é o balde de fichas (*token bucket*), mostrado na Figura 11. Esse mecanismo é descrito por dois parâmetros: taxa de fichas r e profundidade do balde b [PET03]. Esse mecanismo funciona como segue. Para habilitar o envio de um byte, deve existir uma ficha. Para enviar um pacote de tamanho n, é necessário n fichas. O balde inicia sem nenhuma ficha e as acumula em uma taxa de r fichas por segundo. Não podem ser acumuladas mais do que b fichas. O que isto significa é que o tamanho máximo de uma rajada é b. Quando um pacote de tamanho n chega é existem n ou mais fichas no balde, o pacote é dito estar dentro dos limites de desempenho e n fichas são retiradas do balde. Se não houver fichas suficientes, então o fluxo é dito estar fora do perfil especificado. Tipicamente, nenhuma ação é realizada com os fluxos dentro dos limites de desempenho. Entretanto, quando o fluxo está fora do perfil especificado, este está sujeito à *formatação* ou

descarte. *Formatação* consiste em atrasar o fluxo para mudar as suas características de desempenho, e *descarte* é desfazer-se do fluxo. Um fluxo fora do perfil também pode ser *marcado*, quando nenhuma outra ação é realizada.

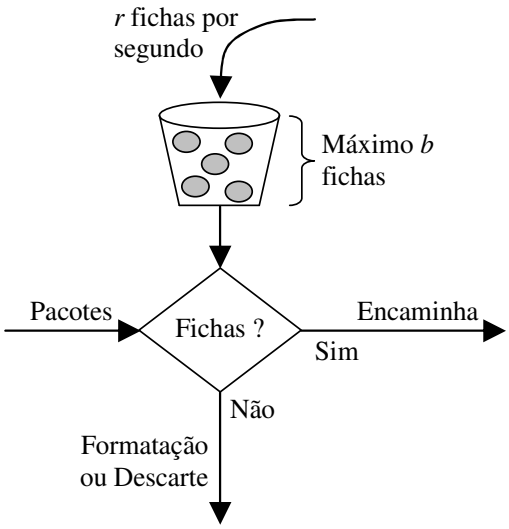


Figura 11 – Balde de fichas (*Token Bucket*).

Para formatar um fluxo fora de perfil, ele pode ser enviado para um *buffer* onde um atraso é adicionado antes do fluxo ser transmitido pela rede. Por atrasar o fluxo, um *buffer* muda o desempenho deste fluxo. Considere um acordo para um fluxo que especifica uma taxa máxima de 1,5 Mb/s. Um medidor está medindo este fluxo, calculando a taxa abaixo.

$$(200\,pacotes\,/\,segundo)\times(pacotes\,com\,1500\,bytes)\times(8\,bits\,/\,byte)=2{,}4\,Mbits\,/\,segundo$$

Esta taxa é comparada com o especificado no acordo e o fluxo é determinado como fora do perfil. Os pacotes subseqüentes são enviados para um *buffer*, sendo atrasados, e enviados a cada 10 ms (a taxa medida de 200 pacotes/segundo corresponde a um pacote a cada 5 ms). Como resultado, somente 100 pacotes podem ser transmitidos por segundo, e a taxa do fluxo torna-se:

$$(100\,pacotes\,/\,segundo)\times(pacotes\,com\,1500\,bytes)\times(8\,bits\,/\,byte)=1{,}2\,Mbits\,/\,segundo$$

A formatação continuará ou por um período específico ou até que o fluxo esteja novamente dentro do perfil.

A ação mais drástica de condicionamento de fluxo é o *descarte* de pacotes. Isto é realizado quando um fluxo está seriamente excedendo seus limites de desempenho ou quando a rede está congestionada a ponto do descarte de pacotes ser necessário.

2.3.2 Alocação de Recursos

Os recursos da rede são alocados para os fluxos conforme suas exigências. Porém, freqüentemente, não é possível atender às exigências de todos os fluxos, significando que alguns deles devem receber menos recursos de rede do que desejam. Parte do problema da alocação de recursos está em decidir quando rejeitar um fluxo [PET03].

Apresenta-se a seguir a classificação dos mecanismos de alocação de recursos. Posteriormente, discute-se o mecanismo de controle de admissão, adotado pelas redes que utilizam alocação de recursos baseado em reservas.

2.3.2.1 Classificação

Existem diversas formas de classificar mecanismos de alocação de recursos [PET03]. Três formas de caracterizar mecanismos de alocação de recursos são abordadas nesta Seção.

Mecanismos de alocação de recursos podem ser classificados em dois amplos grupos: aqueles que endereçam o problema do lado da rede (roteadores ou chaves) e aqueles que endereçam na fronteira da rede (terminais ou núcleos). Em um projeto *centrado no roteador*, cada roteador tem a responsabilidade de decidir quando os pacotes são transmitidos e selecionar quais pacotes são descartados, bem como informar aos núcleos que estão gerando o fluxo quantos pacotes eles podem enviar. No projeto *centrado no núcleo*, os núcleos observam as condições da rede (por exemplo, quantos pacotes estão sendo entregues com sucesso) e ajustam seu comportamento de acordo com estas condições. É importante ressaltar que os dois grupos não são mutuamente exclusivos.

Algumas vezes, os mecanismos de alocação de recursos também são classificados quanto ao uso ou não de reserva de recursos. No método *baseado em reservas*, o núcleo requisita à rede certa quantidade de recursos para que um fluxo seja transmitido. Cada roteador então aloca recursos suficientes para satisfazer esta requisição. Se a requisição não pode ser atendida em algum roteador, então o roteador rejeita o fluxo. No método *sem reservas baseado em retornos* (*feedback*), o núcleo começa transmitindo dados, sem primeiro reservar recursos, e em seguida ajusta sua taxa de transmissão de acordo com o retorno que ele recebe. Este retorno pode ser ou *explícito* (por exemplo, um roteador congestionado envia uma mensagem ao núcleo pedindo para que ele reduza a taxa de transmissão) ou *implícito* (por exemplo, o núcleo ajusta sua taxa transmissão de acordo com o comportamento da rede, tal como perda de pacotes).

Note que um método baseado em reservas sempre implica em um mecanismo de alocação de recursos centrado no roteador. Isto porque cada roteador é responsável por manter a quantidade de seus recursos que está atualmente reservada. Por outro lado, um método baseado em retornos pode implicar ou em um mecanismo centrado no roteador ou em um mecanismo centrado no núcleo. Tipicamente, se o retorno é *explícito*, então o roteador é envolvido na reserva de recursos. Se o retorno é *implícito*, então quase sempre o mecanismo é centrado no núcleo e os roteadores, quando se tornam congestionados, descartam pacotes silenciosamente.

A terceira forma de caracterizar mecanismos de alocação de recursos é se eles são *baseados em janelas* ou *baseados em taxas*. Esta terminologia é usada tanto para controle de fluxo quanto para mecanismos de alocação de recursos, porque ambos necessitam expressar ao transmissor a quantidade de dados que ele pode transmitir. No método *baseado em janela*, o receptor envia uma janela ao transmissor. Esta janela corresponde à quantidade de espaço no *buffer* que o receptor possui, e isto limita a quantidade de dados que o transmissor pode transmitir. No método *baseado em taxa*, o comportamento do transmissor é controlado usando uma taxa. O

receptor informa quantos bits por segundo ele é capaz de absorver, e então o transmissor se adequa a esta taxa.

2.3.2.2 Controle de Admissão

Redes que usam alocação de recursos baseada em reservas necessitam de um mecanismo de controle de admissão. O mecanismo de *controle de admissão* determina se um novo fluxo pode ser admitido pela rede sem colocar em risco as garantias de desempenho dadas aos fluxos já estabelecidos [YUM02]. Uma conexão somente pode ser aceita se os recursos de rede necessários estão disponíveis para estabelecer a conexão fim-a-fim com a qualidade de serviço requisitada.

Controle de admissão usa níveis de prioridade para mudar o comportamento de acesso à rede. Em uma rede sem controle de admissão (melhor esforço), o acesso à rede é democrático, pois todos os fluxos têm uma chance igual de receber recursos da rede [MCC03]. Com controle de admissão, o acesso é permitido, negado, ou algumas vezes atrasado, baseado na prioridade relativa do fluxo. Existem duas classes amplas de algoritmos de controle de admissão: controle de admissão *determinístico* e *estatístico* [YUM02].

Para serviços de tempo real que necessitam de limites rígidos e absolutos no atraso de cada pacote, uma admissão *determinística* é usada. Para tais serviços determinísticos, um algoritmo de controle de admissão calcula o comportamento de pior caso dos fluxos existentes em adição ao de entrada, antes de decidir se o novo fluxo deve ser admitido. Esta classe subutiliza os recursos da rede, especialmente quando os fluxos são transmitidos em rajada.

Muitas das novas aplicações tais como os fluxos de mídia não necessitam garantias de desempenho rígidas e podem tolerar uma pequena violação no limite de desempenho. Um controle de admissão estatístico pode ser usado para tais aplicações. Neste método, uma largura de banda efetiva maior do que a taxa média, mas menor do que a taxa limite é comumente usada. A largura de banda pode ser computada usando um nível *estatístico* ou uma aproximação flexível do fluxo.

Parte do problema da alocação de recursos está em decidir quando rejeitar um fluxo [PET03]. Um dos critérios usados para decidir admitir ou não um fluxo adicional na rede é reservar não mais do que 90% da largura de banda para fluxos de tempo real, permitindo aos demais fluxos ter acesso à no mínimo 10% da largura de banda do canal [CLA92]. Este valor numérico (10%) é apenas um exemplo, e experiências podem sugerir que outros valores sejam mais efetivos. Esta cota assegura que o serviço melhor esforço fique operacional todo tempo. Adicionalmente, esta cota assegura que existe capacidade de reserva suficiente para acomodar flutuações nos fluxos de serviço diferenciado.

2.3.3 Armazenamento e Escalonamento

Mecanismos de alocação de recursos necessitam que cada roteador implemente algum mecanismo de escalonamento que determine a ordem na qual fluxos são processados para transmissão e a ordem na qual são descartados. O *escalonamento* também pode afetar diretamente a latência de um pacote, por determinar quanto tempo um pacote espera para ser transmitido [PET03].

Escalonadores podem ser proprietários ou baseado em padrões. Alguns algoritmos de escalonamento padrão comumente usados incluem mecanismos de armazenamento para escolher qual o próximo pacote a ser enviado. Cada mecanismo é desenvolvido para alcançar um objetivo particular no processamento de pacotes. Por exemplo, um mecanismo de escalonamento pode tratar todos os pacotes do mesmo modo, pode selecionar pacotes aleatoriamente, ou pode favorecer determinados pacotes.

A estrutura de armazenamento e o correspondente algoritmo de escalonamento tentam alcançar os seguintes objetivos:

- *Flexibilidade*: para suportar uma variedade de serviços, e facilmente suportar novos serviços;
- *Escalabilidade*: para ser simples o bastante para permitir escalonar o número máximo de fluxos enquanto permite implementação com baixo custo;
- *Eficiência*: para maximizar utilização de recursos da rede;
- *QoS garantido*: para prover limites baixo de *jitter* e de latência fim-a-fim para fluxos de tempo real;
- *Isolamento*: para reduzir a interferência entre classes de serviço e conexões;
- *Justiça*: para permitir redistribuição rápida e justa de largura de banda que se torna dinamicamente disponível.

Os algoritmos de escalonamento podem ser classificados em quatro tipos [GIR98]: (*i*) escalonamento *baseado em prioridade*; (*ii*) escalonamento *com divisão justa work-conserving*; (*iii*) escalonamento *com divisão justa non-work-conserving*; e (*iv*) formatador de tráfego.

2.3.3.1 Escalonamento baseado em Prioridade

Um escalonador *baseado em prioridades* define uma prioridade para cada *buffer* e os serve de acordo com esta prioridade. Um *buffer* com prioridade menor é servido somente quando não existe nenhum dado esperando para ser servido nos *buffers* com prioridade maior, ou seja, dados com prioridade maior são servidos primeiro, mesmo quando dados de prioridade menor estão esperando por mais tempo. A oportunidade de transmissão dos *buffers* de prioridade baixa depende da carga dos *buffers* de prioridade alta, o qual pode variar dinamicamente. Por esta razão, os limites de latência fim-a-fim não podem ser determinados para todos os *buffers*, apenas para o *buffer* com prioridade mais alta. Conseqüentemente, é difícil suportar serviços múltiplos com QoS garantido.

Escalonamento baseado em prioridade provê uma arbitragem muito simples e eficiente entre um número pequeno de *buffers* [GIR98]. Ele pode, por exemplo, prover uma classe para fluxos de tempo real, uma classe para fluxos de tempo não real com taxa de perda garantida, e uma classe melhor esforço.

2.3.3.2 Escalonamento com Divisão Justa

Uma alternativa para o escalonamento prioritário é o escalonamento *com divisão justa*, no qual para cada *buffer* é garantida sua parte da largura de banda do canal de acordo com um peso

definido. O escalonador com divisão justa é um conceito que introduz isolamento entre os vários *buffers* em um modo que minimiza a interação entre os fluxos nos diferentes *buffers*.

No entanto, é importante salientar que a alocação de largura de banda justa é possível somente entre os *buffers* e não entre os fluxos que compartilham um *buffer*. Por exemplo, se fluxos de cada classe de serviço são agrupados em um *buffer*, então a alocação justa é possível entre as classes de serviço, mas não entre os fluxos da mesma classe.

Os algoritmos de divisão justa garantem uma determinada taxa mínima alocada entre *buffers*. Mecanismos baseados em taxas podem ser classificados dentro de duas categorias: *taxa distribuída* e *taxa controlada*.

No mecanismo baseado em *taxa distribuída*, um *buffer* pode ser servido em um taxa mais alta do que a taxa de serviço mínimo, enquanto as garantias feitas a outros serviços não forem prejudicadas. No mecanismo baseado em *taxa controlada*, nenhum *buffer* é servido em um taxa mais alta do que a taxa de serviço determinada sob alguma circunstância. A Figura 12 descreve a diferença entre os dois mecanismos. Cada *buffer* necessita uma garantia de largura de banda mínima, r_i no exemplo. O escalonador deve tentar alcançar esta garantia de largura de banda mínima para todos os *buffers*.

Esquemas baseados em taxa são também classificados como: *work-conserving* e *non-work-conserving*. Com uma disciplina *work-conserving*, o escalonador nunca está ocioso quando existem dados em qualquer um dos *buffers*. Com uma disciplina *non-work-conserving*, cada pacote recebe, explicita ou implicitamente, um tempo de elegibilidade. Mesmo quando um escalonador está ocioso, se nenhum pacote é elegível, nada será transmitido [ZHA95]. Deste modo, escalonadores de taxa controlada que não servem um *buffer* com mais do que a sua taxa alocada são classificados como *non-work-conserving*, enquanto que escalonadores com taxa distribuída são *work-conserving* [GIR98].

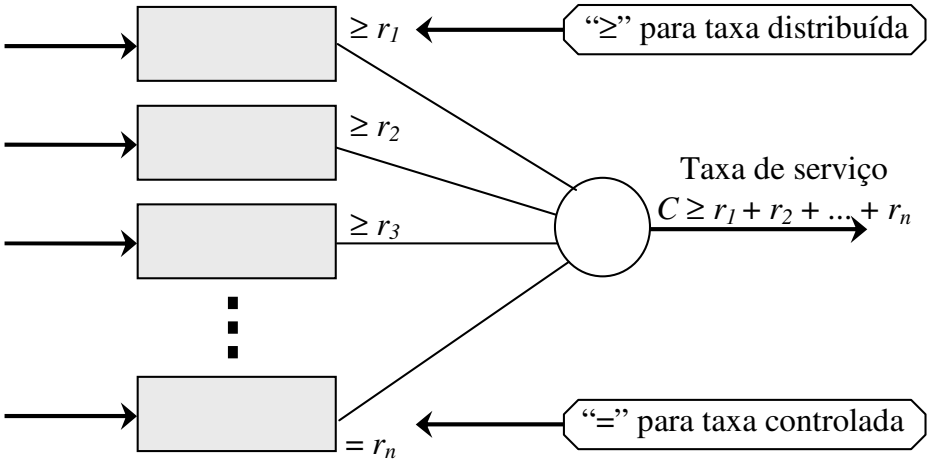


Figura 12 – Taxa controlada versus taxa distribuída. C corresponde à largura de banda ocupada do canal físico.

2.3.3.2.1 Algoritmos de Escalonamento com Divisão Justa Work-Conserving

Virtual Clock

O algoritmo de escalonamento *virtual clock* é uma emulação do STDM. STDM, como apresentado na Seção 2.1.5, é um sistema baseado em fatias temporais que não permite nenhuma multiplexação estatística e garante uma largura de banda fixa para fluxos por determinar fatias fixas. Se um *buffer* não tem nenhum dado para transmitir em determinada fatia, a fatia não é usada e a largura de banda é desperdiçada. Logo, STDM é uma disciplina *non-work-conserving*. Entretanto, com *virtual clock*, multiplexação estatística é possível e fatias são utilizadas quando algum *buffer* tem dados para transmitir. Enquanto um relógio de tempo real é usado em um sistema STDM para definir as fatias de tempo, um relógio virtual é usado para etiquetar os dados no escalonamento *virtual clock* [GIR98]. A cada dado é determinado um tempo virtual, o qual é o tempo que um dado deveria ser transmitido se o sistema STDM fosse usado. Então, dados são enviados na ordem dos tempos virtuais.

Um esboço da implementação apresentada em [ZHA91] é descrito abaixo e ilustrado na Figura 13. Para cada roteador:

1. Quando fluxos requisitam transmissão, o valor $Vtick_i = 1/AR_i$ é computado, onde AR_i é a taxa de transmissão média indicada na requisição.
2. Quando chegar o primeiro pacote do $fluxo_i$, $VirtualClock_i \leftarrow tempo\ real$
3. Quando receber cada pacote do $fluxo_i$, avançar $VirtualClock_i$ para $Vtick_i$, marcando o pacote com o valor do $VirtualClock_i$.
4. Transmitir os pacotes na ordem crescente de valores de *VirtualClock*.
5. Quando chegar um pacote e o *buffer* estiver cheio, descartar o último pacote do *buffer*.

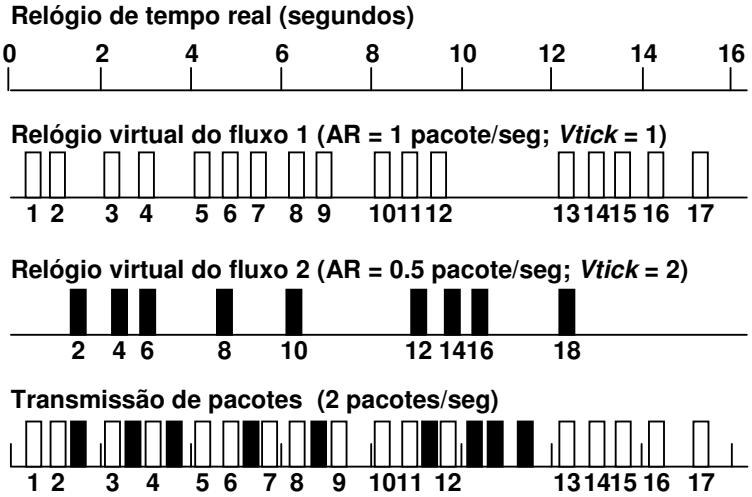


Figura 13 – Relógio real, relógio virtual e ordem de transmissão dos pacotes [ZHA91].

Observe na Figura 13 que o fluxo 1 tem *Vtick* igual a 1 (1/1) e o fluxo 2 tem *Vtick* igual a 2 (1/ 0,5). Por conseqüência, os pacotes do fluxo 1 são marcados com valores de *VirtualClock* que

possuem intervalo 1 e pacotes do fluxo 2 são marcados com valores de *VirtualClock* que possuem intervalo 2. Depois de atribuir o valor de *VirtualClock* a cada pacote, o algoritmo *virtual clock* meramente transmite os pacotes ordenados em função deste valor. Cabe ressaltar que o algoritmo de escalonamento *virtual clock* é classificado como *work-conserving*. Portanto, sempre que existe largura de banda disponível e dado para transmitir, este dado é transmitido. Por exemplo, o pacote do fluxo 2 com *VirtualClock* igual a 6 é transmitido antes do pacote do fluxo 1 com *VirtualClock* igual a 5. Isto ocorre por três razões: (1) existem recursos disponíveis; (2) o fluxo 1 não possui pacotes com *VirtualClock* menor aguardando transmissão (o pacote com *VirtualClock* igual a 5 ainda não chegou); e (3) o fluxo 2 possui pacotes para transmitir.

Weighted Fair Queuing (WFQ)

O WFQ também é conhecido como PGPS (do inglês *Packet-Based Generalized Processor Sharing*). No WFQ, um peso pode ser atribuído a cada *buffer*. Este peso controla a percentagem da largura de banda do canal que este *buffer* receberá. FQ (do inglês *Fair Queuing*) simples atribui a cada *buffer* um peso 1, o que significa que cada *buffer* recebe 1/*n* da largura de banda do canal quando existem *n buffers*. Com WFQ, entretanto, um *buffer* pode ter peso 2, um segundo *buffer* pode ter peso 1, e um terceiro pode ter peso 3. Assumindo que cada *buffer* sempre possui pacotes esperando para serem transmitidos, o primeiro *buffer* receberá 1/3, o segundo receberá 1/6 e o terceiro receberá 1/2 da largura de banda disponível.

Delay Earliest Due Date (Delay-EDD)

O algoritmo *Delay Earliest Due Date* ou *Delay-EDD* é uma extensão do escalonamento clássico *Earliest Due Date First* (EDD ou EDF). No *Delay-EDD*, o servidor negocia um contrato de serviço com cada origem. O contrato determina que se a origem obedecer a sua especificação de fluxo, então o servidor garantirá um limite de latência. O servidor atribui um prazo ao pacote que determina quando ele deve ser transmitido caso seja recebido conforme o contrato. Isto é apenas o tempo de chegada esperado adicionado ao limite de latência no servidor. Por exemplo, se um cliente assegura que ele transmite pacotes a cada 0,2 segundo, e o limite de latência em um servidor é 1 segundo, então o terceiro pacote do cliente terá um prazo de 1,6 segundo (0,2n + 1, onde n é o número de pacotes).

2.3.3.2.2 Algoritmos de Escalonamento com Divisão Justa Non-Work-Conserving

Stop-and-Go

Este esquema é baseado em uma estratégia de enquadramento, como ilustrado na Figura 14. Nesta estratégia, o eixo do tempo é dividido em quadros, que são períodos de algum comprimento constante T. *Stop-and-Go* define quadros de partida e chegada para cada canal. Em cada roteador, o quadro de chegada de cada canal de entrada é mapeado para o quadro de partida do canal de saída, introduzindo uma latência constante θ, onde $0 \leq \theta < T$. No *Stop-and-Go*, a

transmissão de um pacote que chegou a um canal *l* durante o quadro *f* deve sempre ser postergada até o início do próximo quadro. Deste modo, mesmo que existam pacotes no roteador para serem transmitidos, o canal de saída pode ficar ocioso, caracterizando a política *non-work-conserving*.

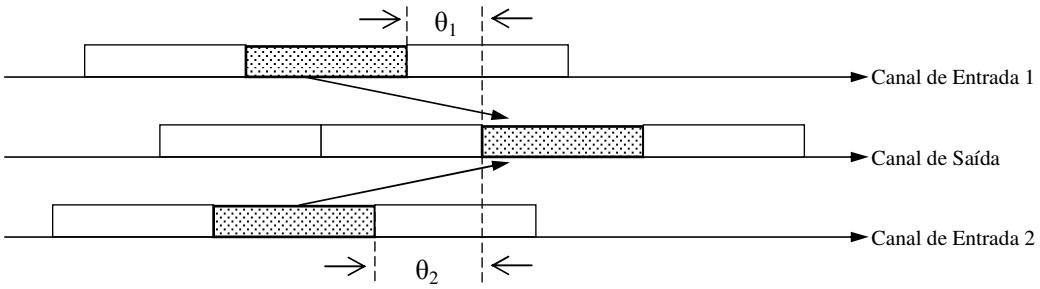


Figura 14 – Sincronização entre os canais de entrada e saída no *Stop-and-Go*.

Stop-and-Go assegura que pacotes no mesmo quadro na origem permanecerão no mesmo quadro em toda rede [ZHA95]. Por esta razão, limites de latência fim-a-fim podem ser garantidos. No entanto, a estratégia de enquadramento introduz o problema de acoplamento entre o limite da latência e a granularidade da alocação de largura de banda. A latência de um pacote em um único roteador é limitada a duas vezes a dimensão do quadro. Para reduzir a latência, um quadro menor é desejado. Entretanto, o quadro também é usado para especificar o tráfego. Logo, para ter maior flexibilidade na alocação da largura de banda, um quadro maior é preferido. Conseqüentemente, um limite baixo de latência e uma granularidade fina da alocação da largura de banda não podem ser alcançados simultaneamente em uma estratégia de enquadramento como *Stop-and-Go*.

Para resolver este problema de acoplamento, uma versão generalizada do *Stop-and-Go* com tamanhos diferentes de quadro é proposta em [ZHA95]. No *Stop-and-Go generalizado*, o eixo tempo é dividido dentro de uma estrutura de quadros hierárquicos como apresentado na Figura 15. Desta forma, é possível prover limites mais baixos de latência para alguns canais colocando-os em quadros com um tempo de quadro menor, e alocar largura de banda com granularidade fina para outros canais, colocando-os em níveis com tempo de quadro maior. Entretanto, o acoplamento entre latência e granularidade da alocação da largura de banda ainda existe dentro de cada quadro.

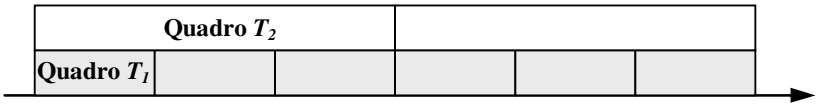


Figura 15 – Dois níveis de enquadramento com T2 = 3T1.

Jitter Earliest Due Date (EDD)

O *Jitter-EDD* é uma extensão do *delay-EDD* para prover limites de latência e *jitter*. Este método provê limites máximo e mínimo de latência. Para garantir limites de latência e *jitter*, cada nodo na rede tem que preservar o padrão de chegada do fluxo. Neste esquema, cada dado é marcado com um valor (chamado de *termo de correção*), que é a diferença do tempo entre o instante que o dado é servido e o instante que supostamente ele deveria ser servido (seu prazo). Um formatador de tráfego é usado no próximo roteador para atrasar o dado por este tempo antes de declarar que o dado

pode ser transmitido. Note que este esquema requer a participação do próximo roteador para controlar a latência.

2.3.3.3 Formatador de Tráfego

Como descrito na Seção 2.3.1, o objetivo de um formatador de tráfego é criar uma conformidade entre um fluxo e o seu perfil previamente especificado. Geralmente, o formatador de tráfego é utilizado somente nos núcleos ou interfaces de rede. No entanto, esta Seção apresenta a função de um formatador implementado em roteadores da rede.

O formatador de tráfego pode ser aplicado a um grupo de canais virtuais ou a canais virtuais individuais. Em geral, contudo, a formatação é aplicada a canais virtuais individuais usando a informação sobre o perfil do fluxo. Logo, o formatador de tráfego é pouco diferente de um escalonador com divisão justa non-work-conserving [GIR98]. Nos escalonamentos com divisão justa non-work-conserving descritos acima, o fluxo é atrasado para controlar latência e *jitter* (Stop-and-Go, EDD) ou taxa e *jitter* (HRR). Os esquemas de formatação de tráfego controlam a taxa dos fluxos com algoritmos como o balde de fichas (*token bucket*). Porém, em alguns casos, é necessária uma arquitetura que combine um formatador de tráfego e um escalonador.

Um exemplo desta arquitetura é proposto por Rexford et al. [REX97] e apresentada na Figura 17. Esta arquitetura assume que fluxos são armazenados em *buffers* independentes e policiados através dos parâmetros (σ, ρ), onde σ é o tamanho do balde de fichas e ρ é a taxa de geração de fichas. Ao invés de descartar fluxos fora do perfil, o formatador de tráfego atribui a cada fluxo um tempo de conformidade, ou seja, o tempo em que o fluxo deve ser inserido na rede a fim do fluxo ficar de acordo com o perfil. O tempo de conformidade (c) é calculado pelo algoritmo apresentado na Figura 16, onde t é o tempo de chegada do dado, X é o tempo de chegada estimado, e (σ, ρ) são parâmetros do algoritmo balde de fichas.

```
X = X + 1/ρ                 // estimativa de tempo de chegada do dado
if (X ≤ t)                  // balde de fichas cheio: reinicializa X
    c = X = t;
else if (X ≤ t + σ/ρ)       // balde de fichas parcialmente cheio
    c = t;
else                        // balde de fichas vazio: atrasa dado
    c = (X - σ/ρ)
```

Figura 16 – Algoritmo para calcular o tempo de conformidade de um dado [REX97].

O formatador usa o tempo de conformidade para escalonar os fluxos. Um classificador de tempos é usado para organizar os fluxos em ordem crescente de tempo de conformidade. Cada posição do classificador de tempos corresponde a um tempo de conformidade, e a lista encadeada de cada posição corresponde aos dados que possuem este tempo de conformidade. Na Figura 17, uma posição do classificador de tempos ocupada pela letra E (do inglês *Empty*) indica que nenhum dado foi escalonado para aquele tempo e que esta posição está vazia. Os dados no classificador de tempos são lidos no tempo atual e inseridos em uma FIFO, de onde são transmitidos.

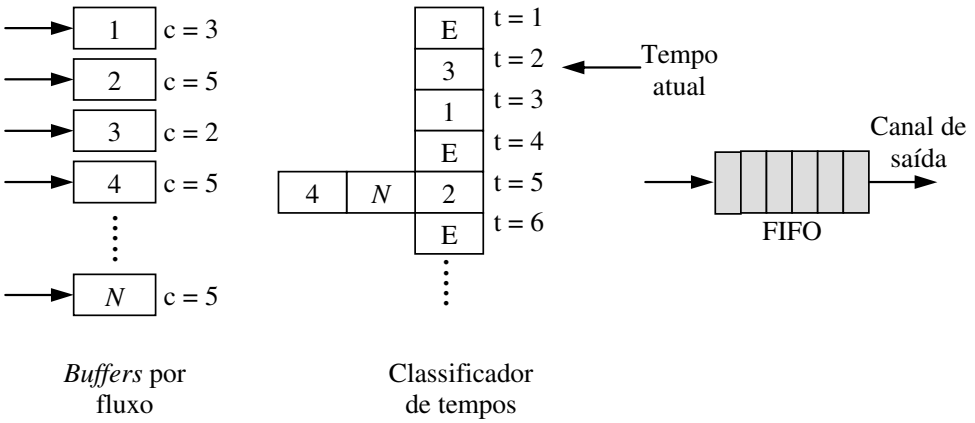


Figura 17 – Exemplo de uma arquitetura que combina um formatador de tráfego e um escalonador [REX97].

Outros exemplos de arquiteturas que combinam um formatador de tráfego e um escalonador podem ser encontrados em [GIR98].

2.3.4 Controle de Congestionamento

Congestionamento em redes é basicamente um problema de compartilhamento de recursos. Ele ocorre quando os núcleos inserem na rede uma quantidade maior de dados do que ela é capaz de tratar. Existem duas atividades distintas relacionadas ao controle ou gerenciamento de congestionamento. A primeira é a *prevenção de congestionamento*, que tenta detectar possíveis condições que levem a congestionamentos futuros e executar procedimentos para impedir que estas ocorram. A segunda é a *recuperação de congestionamento*, que atua quando um congestionamento já ocorreu para que a rede volte ao seu estado normal.

2.3.4.1 Mecanismos de Recuperação de Congestionamentos

Esta Seção descreve o exemplo predominante de controle de congestionamento fim-a-fim, que é utilizado por TCP. A estratégia essencial de TCP é enviar pacotes para rede sem reserva e então reagir à ocorrência de eventos [PET03].

A idéia do controle de congestionamento TCP é que cada origem determina quantos pacotes ela pode seguramente transmitir. Como uma determinada origem transmite muitos pacotes, ela usa a chegada de um ack como um sinal que um dos seus pacotes deixou a rede e, portanto, ela pode seguramente inserir um novo pacote sem elevar o nível de congestionamento. No entanto, determinar a capacidade disponível em um primeiro momento não é uma tarefa fácil. Adicionalmente, como conexões são estabelecidas e fechadas, a capacidade disponível da rede muda ao longo do tempo. Isto significa que alguma origem específica deve ser capaz de ajustar o número de pacotes que ela está transmitindo.

A proposta original do controle de congestionamento TCP é implementada pelos dois mecanismos descritos a seguir.

Aumento Aditivo / Redução Multiplicativa

TCP mantém uma variável de estado para cada conexão, chamada de janela de congestionamento (*CongestionWindow*), que é usada pela origem para limitar quantos dados ela pode transmitir em uma determinada janela de tempo. O mínimo entre as variáveis *CongestionWindow* e *AdvertisedWindow* determina o número máximo de dados que podem ser enviados sem o recebimento de um ack. A variável *AdvertisedWindow* corresponde ao tamanho de janela aconselhado pelo destino. O destino seleciona um valor adequado para *AdvertisedWindow* baseado na quantidade de memória alocada para bufferização. A janela efetiva de TCP é definida como segue:

$$\begin{aligned} &MaxWindow = MIN(CongestionWindow, AdvertisedWindow) \\ &EffectiveWindow = MaxWindow - (\text{Último byte enviado} - \text{Último byte que recebeu ack}) \end{aligned} \quad (1)$$

Um problema é como TCP determina um valor inicial apropriado para *CongestionWindow*. A origem TCP modifica o valor de *CongestionWindow* de acordo com o nível de congestionamento da rede. Isto envolve a redução do valor de *CongestionWindow* quando o nível de congestionamento cresce e o aumento do valor quando o nível de congestionamento diminui.

As origens TCP interpretam o estouro de prazos finais (*timeouts*) como um sinal de congestionamento e reduzem a taxa na qual transmitem dados. A cada momento em que o prazo final para o recebimento da confirmação de transmissão de dados (ACK) é atingido, a origem atribui à *CongestionWindow* a metade de seu valor. Por exemplo, suponha que o valor atual de *CongestionWindow* é 16. Se uma perda é detectada, *CongestionWindow* recebe 8. Perdas adicionais causam a redução de *CongestionWindow* para 4, então 2, e finalmente 1. *CongestionWindow* não pode ser reduzido abaixo do tamanho de um único pacote, ou, na terminologia de TCP, MSS (do inglês *Maximum Segment Size*).

Uma estratégia que somente reduza o valor de *CongestionWindow* é obviamente conservadora. O mecanismo deve ser capaz de aumentar este valor para usufruir as vantagens quando a capacidade torna-se novamente disponível. Cada vez que uma origem transmite os pacotes com sucesso, ela adiciona o equivalente a um pacote à *CongestionWindow*. Este crescimento linear é ilustrado na Figura 18(a). Nota-se que, na prática, TCP não espera pelo recebimento da confirmação (ACK) da janela inteira para adicionar um pacote à janela de congestionamento. Ao invés disto, incrementa *CongestionWindow* pela chegada de apenas um ACK. Especificamente, o valor de *CongestionWindow* é incrementado como segue, cada vez que um ACK chega:

$$\begin{aligned} &Incremento = MSS \times (MSS / CongestionWindow) \\ &CongestionWindow += Incremento \end{aligned} \quad (2)$$

Este padrão de continuamente aumentar e reduzir a janela de congestionamento continua durante todo tempo de vida de uma conexão.

Slow Start

O mecanismo de aumento aditivo como descrito é o método correto para usar quando a origem está operando perto da capacidade disponível da rede, mas não é adequado quando a origem está no início da transmissão. TCP conseqüentemente provê um segundo mecanismo, chamado de *Slow Start*, que é usado para incrementar o valor de *CongestionWindow* rapidamente do ponto inicial. *Slow Start* incrementa a janela exponencialmente, ao invés de linearmente.

Especificamente, a origem inicia atribuindo um pacote à *CongestionWindow*. Quando um ack para este pacote chega, TCP adiciona 1 à *CongestionWindow* e então envia dois pacotes. Quando recebe os correspondentes dois acks, TCP incrementa *CongestionWindow* de 2, um para cada ack, e envia quatro pacotes. A Figura 18(b) apresenta o crescimento exponencial do número de pacotes transmitidos durante o *Slow Start*. Mais detalhes sobre o *Slow Start* podem ser encontrados em [PET03].

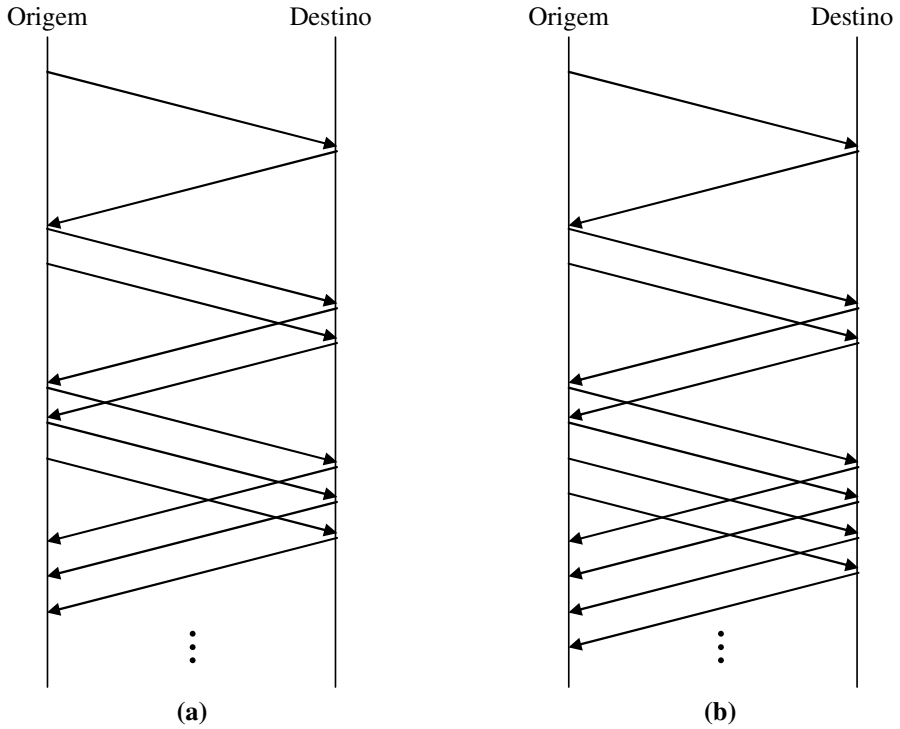


Figura 18 – (a)Transmissão de pacotes durante o aumento aditivo; (b) transmissão de pacotes durante o *Slow Start*.

2.3.4.2 Mecanismos de Prevenção de Congestionamentos

Nesta Seção, são descritos três mecanismos diferentes utilizados para evitar situações de congestionamento [PET03]. Os dois primeiros usam um método similar. Eles colocam uma quantidade pequena de funcionalidades adicionais dentro do roteador para ajudar o nodo final a antecipar o congestionamento. O terceiro mecanismo é muito diferente dos dois primeiros. Ele tenta evitar o congestionamento puramente nos nodos finais.

DECbit

O primeiro mecanismo, denominado DECbit, baseia-se na idéia de espalhar a responsabilidade pelo controle de congestionamento entre os roteadores e os nodos finais. Cada roteador monitora a ocupação média de seu *buffer* e explicitamente notifica o nodo destino quando uma situação de congestionamento está próxima de ocorrer. Esta notificação é implementada configurando um bit de congestionamento nos pacotes deste fluxo. O destino do fluxo então copia este bit de congestionamento dentro do ACK e envia de volta à origem. Finalmente, a origem ajusta sua taxa de transmissão de maneira a evitar o congestionamento.

Neste mecanismo, um único bit de congestionamento é adicionado ao cabeçalho do pacote. Um roteador ativa este bit em um pacote se o comprimento médio de seu *buffer* é maior ou igual a 1 no tempo em que o pacote chega. Este comprimento médio do *buffer* é medido sobre um intervalo de tempo que compreende o último ciclo ocupado + ciclo ocioso + ciclo ocupado atual. A Figura 19 apresenta um exemplo do comprimento do *buffer* em um roteador, em função do tempo. Essencialmente, o roteador calcula a área abaixo da curva e divide o valor pelo intervalo de tempo para computar o comprimento médio do *buffer*. Usando um comprimento de *buffer* igual a 1 para ativar o bit de congestionamento.

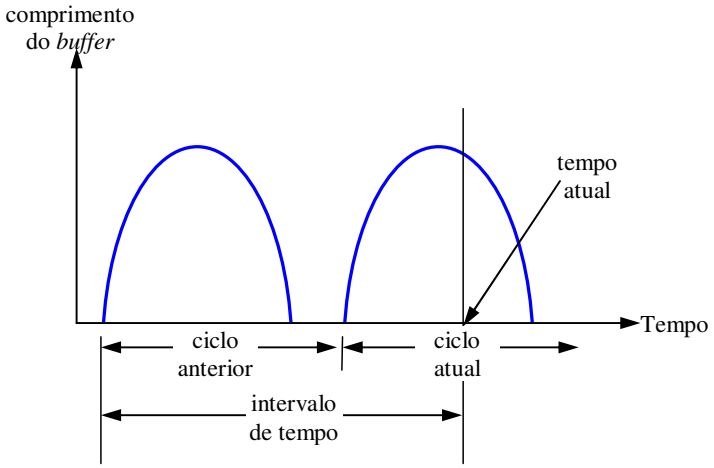


Figura 19 – Computando o comprimento médio do *buffer* em um roteador.

No lado do nodo destino, a origem armazena quantos de seus pacotes tiveram seu bit de congestionamento ativado em algum roteador. Com este objetivo, a origem mantém uma janela de congestionamento e verifica qual a fração dos pacotes da última janela que tiveram seu bit configurado. Se menos do que 50% dos pacotes tiveram seu bit configurado, então a origem aumenta sua janela de congestionamento em um pacote. Se 50% ou mais dos pacotes tiveram seu bit configurado, então a origem diminui sua janela de congestionamento em 0,875 vezes o seu valor anterior.

Random Early Detection (RED)

O segundo mecanismo, denominado RED (do inglês *Random Early Detection*), é similar ao esquema do DECbit em que cada roteador é programado para monitorar seu comprimento de

buffer, e quando detecta que um congestionamento está iminente, notifica a origem para ajustar sua janela de congestionamento. O RED difere do DECbit em dois aspectos.

O primeiro é que RED é mais comumente implementado como uma notificação implícita à origem através do descarte de um de seus pacotes. A origem é efetivamente notificada pelo fim de prazo ou ACK duplicado. Os roteadores que implementam RED descartam poucos pacotes antes de seus *buffers* estarem completamente cheios, o que causa a redução da taxa de transmissão da origem, para que muitos pacotes não tenham que ser descartados se ocorrer um congestionamento. A segunda diferença entre RED e DECbit é a escolha de quando descartar um pacote e qual pacote descartar. O algoritmo RED, originalmente proposto por Floyd e Jacobson, é descrito em [FLO93].

Baseado na Origem

Ao contrário dos dois métodos anteriores, que dependem de mecanismos nos roteadores, este método baseia-se na idéia de observar alguns sinais da rede para detectar quando situações de congestionamento estão próximas de acontecer.

Um algoritmo particular observa a rede como segue. A janela de congestionamento é incrementada como em TCP, mas a cada dois RTT[5] (do inglês *Round-Trip Time*), o algoritmo verifica se o RTT atual é maior do que a média dos RTTs mínimo e máximo. Se for, então o algoritmo diminui a janela de congestionamento de 1/8.

Um segundo algoritmo faz algo similar. A decisão se muda ou não o tamanho da janela atual é baseada no RTT e no tamanho da janela. A janela é ajustada uma vez a cada dois RTTs baseado no resultado da expressão da Equação 3. Se o resultado é positivo, a origem diminui o tamanho da janela de 1/8. Se o resultado é negativo ou zero, a origem aumenta a janela pelo tamanho máximo de pacote.

$$(JanelaAtual - JanelaAnterior) \times (RTTatual - RTTanterior) \quad (3)$$

Um terceiro esquema observa a mudança na taxa de vazão, ou mais especificamente, mudanças na taxa de transmissão. Este algoritmo, chamado de *TCP Vegas*, compara a vazão alcançada com a vazão esperada. O TCP Vegas usa a idéia de medir e controlar a quantidade de dados extras que uma conexão transmite, onde "*dados extras*" significa os dados que a origem não teria transmitido se ela tentasse se adequar exatamente à largura de banda disponível da rede. Obviamente, se uma origem está transmitindo dados extras, causará latências longas e possibilitará congestionamentos. Menos óbvio, se uma conexão está enviando poucos dados extras, não responderá suficientemente rápido ao aumento de largura de banda disponível. Ações para evitar congestionamento do TCP Vegas são baseadas em mudanças na quantidade de dados extras estimados na rede, não somente no descarte de pacotes. Detalhes sobre este algoritmo são apresentados em [PET03].

[5] RTT é a estimativa do tempo total de transmissão de ida e volta em uma determinada conexão.

3 TRABALHOS RELACIONADOS

Projetos de redes intra-chip atuais adotam um destes três métodos para prover QoS: (*i*) dimensionamento da rede para prover largura de banda suficiente para satisfazer os requisitos de todos os núcleos no sistema; (*ii*) prover suporte ao chaveamento por circuito para todos ou para núcleos selecionados; (*iii*) tornar disponível o escalonamento baseado em prioridades para transmissão de pacotes.

Harmanci et al. [HAR05] apresentam uma comparação quantitativa entre o chaveamento por circuito e o escalonamento baseado em prioridades, mostrando que a priorização de fluxos sobre uma rede sem estabelecimento de conexão é capaz de garantir latências fim-a-fim menores do que em redes com chaveamento por circuito. Entretanto, a referência não apresenta resultados numéricos. Uma possível explicação para isto é o uso de uma modelagem SystemC TLM, ao invés de modelos com precisão de ciclo de relógio utilizados neste trabalho.

O primeiro método para prover QoS mencionado acima é adotado, por exemplo, pela rede intra-chip Xpipes [BER05]. Um projetista dimensiona a Xpipes de acordo com os requisitos das aplicações, ajustando a largura de banda de cada canal a fim de atender totalmente a esses requisitos. Entretanto, quanto maior for a velocidade de processamento e a largura de banda, mais e mais novas aplicações demandando níveis superiores de desempenho de rede serão criadas [SHI03]. Adicionalmente, aplicando somente este método, não é garantido que congestionamentos locais (*hot spots*) sejam evitados, mesmo se a largura de banda é amplamente aumentada. Por estas razões, este método torna-se inadequado para satisfazer aos requisitos de um conjunto amplo de aplicações distintas.

O segundo método, chaveamento por circuito[6], provê QoS através do estabelecimento de conexões. Este método é usado nas redes intra-chip Æthereal [GOO05], aSOC [LIA00], Octagon [KAR02], Nostrum [MIL04] e SoCBUS [WIK03]. Por exemplo, a rede intra-chip Nostrum [MIL04] adota circuitos virtuais, com roteamento dos fluxos QoS decidido em tempo de projeto. As comunicações nos canais físicos são escalonadas globalmente em fatias de tempo (TDM). Circuitos virtuais garantem vazão e latência constante em tempo de execução, mesmo com taxas de inserção de tráfego variáveis. As redes com chaveamento por circuito criam conexões para todos ou para fluxos selecionados. O estabelecimento das conexões requer alocação de recursos, tais como *buffers* e/ou largura de banda do canal. Este método tem a vantagem de garantir limites temporais rígidos para fluxos individuais. No entanto, este método tem duas desvantagens: (*i*) pobre escalabilidade, pois a área do roteador cresce proporcionalmente ao número de conexões suportadas [HAR05]; (*ii*) uso ineficiente da largura de banda, devido à alocação de recursos ser baseada em cenários de pior caso, desperdiçando os recursos da rede, especialmente quando fluxos são transmitidos em rajada.

QNoC [BOL03], DiffServ-NoC [HAR05] e RSoC [VES05] são exemplos de redes intra-chip que adotam o terceiro método, chaveamento por pacote com escalonamento baseado em

[6] Neste documento, o termo chaveamento por circuito refere-se às redes provendo estruturas de nível físico entre a origem e o destino, como também às redes com chaveamento por pacote que adotam serviços de níveis mais altos (tal como circuitos virtuais) para estabelecer conexões.

prioridades. Esta técnica sem estabelecimento de conexão agrupa fluxos em classes diferentes, com níveis de serviço diferentes para cada classe. O método requer bufferização separada para manipular pacotes de acordo com os níveis de serviço. Cada nível de serviço recebe uma prioridade. A rede sempre serve primeiro *buffers* com prioridade maior e não vazios. *Buffers* com prioridade menor recebem atenção somente quando não existem dados esperando para serem servidos em *buffers* com prioridade maior. Este método oferece melhor adaptação à variação do tráfego da rede e, potencialmente, uma melhor utilização dos recursos da rede. Entretanto, vazão e latência fim-a-fim não podem ser garantidas, exceto para fluxos com prioridade maior. Também, é necessário prover alguma forma de prevenir *starvation* para fluxos com prioridade baixa. Quando fluxos compartilham recursos, mesmo fluxos com prioridade maior podem ter um comportamento imprevisível, como será demonstrado no Capítulo 8.

Nem chaveamento por circuito nem escalonamento baseado em prioridades garante QoS para *múltiplos fluxos concorrentes*. Quando se usa o chaveamento por circuito, a rede pode rejeitar um número de fluxos, devido à quantidade limitada de conexões simultaneamente suportadas, mesmo se a rede possui largura de banda disponível. Quando múltiplos fluxos com mesma prioridade competem pelos mesmos recursos, redes com escalonamento baseado em prioridades têm comportamento semelhante à rede com serviço BE, como será demonstrado no Capítulo 8. Como mencionado anteriormente, redes usando um dos três métodos descritos acima adotam técnicas em tempo de projeto para garantir QoS, através da modelagem do tráfego, dimensionamento da rede (topologia, profundidade dos *buffers*, largura do *flit*) baseado em simulação e síntese da rede. As desvantagens do dimensionamento da rede em tempo de projeto são: (*i*) a complexidade da modelagem do tráfego e da simulação do sistema é muito alta, sendo então propensas a erros; e (*ii*) a rede projetada neste modo pode não garantir QoS a novas aplicações. A primeira desvantagem pode forçar o uso de modelos simplificados, o que pode conduzir ao dimensionamento incorreto dos parâmetros da rede para a síntese. A segunda desvantagem pode surgir se novas aplicações devem executar no sistema depois de alguma implementação inicial, como ocorre em sistemas reconfiguráveis e/ou programáveis.

Os principais parâmetros de desempenho usados pelas redes revisadas acima são latência fim-a-fim e vazão. No entanto, quando QoS é considerado, outro conceito pode ser relevante, *jitter*. *Jitter* pode ser definido como a variação da latência, causada por congestionamentos na rede ou variações de caminho [DAL04]. Em redes sem estabelecimento de conexão, *buffers* introduzem *jitter*. Quando pacotes são bloqueados, a latência aumenta. Uma vez a rede podendo liberar pacotes de bloqueios, a latência diminui, devido ao envio de pacotes em rajada. Conseqüentemente, redes usando somente escalonamento baseado em prioridades não podem garantir *jitter* controlado.

Alguns outros trabalhos usam métodos diferentes para atender requisitos de QoS. Por exemplo, Andreasson e Kumar propuseram um roteamento chamado *slack-time aware routing* [AND05][KUM05], uma técnica de roteamento na origem para melhorar a utilização da rede através do controle dinâmico da inserção de pacotes BE em caminhos específicos, enquanto pacotes com vazão garantida (GT, do inglês *Guaranteed Throughput*) não estão usando estes caminhos. Entretanto, este trabalho não é diretamente relacionado ao atendimento à QoS.

As informações apresentadas acima são resumidas na Tabela 2. O método melhor esforço, onde a rede não provê nenhuma garantia temporal aos fluxos, também é apresentado nesta Tabela. As redes intra-chip Xpipes, QNoC e ÆThereal são detalhadas a seguir, como exemplos de cada uma das diferentes técnicas para prover QoS em NoCs.

Tabela 2 – Estado da arte em redes intra-chip que provêem QoS a aplicações.

Método	Rede [ano]	Vantagens	Desvantagens
BE (melhor esforço)	Hermes[2003] Arteris[2005] SPIN[2000]	• Projeto muito simples	• Nenhuma garantia temporal
Dimensionamento da rede para atender aos requisitos de QoS	Xpipes[2002]	• Garantias rígidas de QoS	• O sistema é projetado para uma determinada aplicação
Escalonamento baseado em prioridades	QNoC[2003] RSoC[2005]	• Melhor utilização dos recursos da rede	• Latência fim-a-fim não é garantida • Fluxos com mesma prioridade com comportamento BE
Chaveamento por circuito	ÆThereal[2002] ASoC[2000] SoCBUS[2002] Octagon[2001]	• Garantias rígidas de QoS	• Escalabilidade baixa • Dimensionamento da rede para o pior caso • Tempo para o estabelecimento da conexão não conhecido

A rede intra-chip Hermes [MOR04], usada como base para o desenvolvimento deste trabalho, é detalhada no próximo Capítulo. A rede Hermes, assim como as redes Xpipes, Æthereal e QNoC, possui um fluxo de projeto que é composto pelas etapas de geração da rede, geração do tráfego, simulação e avaliação de desempenho. Este fluxo de projeto é automatizado pelo ambiente Atlas (Apêndice II).

3.1 Xpipes

A rede Xpipes foi proposta por Bertozzi et al. [BER05][DAL03] para SoCs multiprocessados. Ela possui roteamento *wormhole* e faz uso do algoritmo de roteamento estático denominado *street sign routing*. Este algoritmo de roteamento permite uma implementação simples do roteador porque nenhuma decisão dinâmica tem que ser tomada no mesmo.

Uma das principais preocupações no projeto da Xpipes foi o suporte à comunicação confiável. Isto foi alcançado por meio de detecção de erro distribuída, ou seja, em cada roteador. Embora a detecção de erro distribuída cause uma sobrecarga de área nos roteadores da rede se comparada com uma solução fim-a-fim, ela é melhor capacitada para conter os efeitos da propagação de erros, por exemplo, impedindo que um cabeçalho corrompido seja transmitido para um caminho errado.

A rede intra-chip Xpipes possui alto grau de parametrização. A parametrização inclui o tamanho do *flit*, o espaço de endereçamento dos núcleos, o número máximo de roteadores entre dois núcleos, o número máximo de bits para controle de fluxo fim-a-fim, a profundidade do *buffer*, o número de canais virtuais por canal físico, entre outros.

O roteador Xpipes é ilustrado na Figura 20. No exemplo, o roteador possui 4 entradas, 4 saídas e 2 canais virtuais multiplexados sobre o mesmo canal físico. O roteador adota bufferização de saída e a arquitetura resultante consiste de múltiplas replicações do mesmo módulo de saída, apresentado na Figura 21, um para cada porta de saída do roteador. Todas as portas de entrada são conectadas a cada entrada do módulo. Os sinais de controle de fluxo gerados em cada módulo (tal como ack e nack para *flits* de entrada) são coletados por uma unidade centralizada do roteador, a qual transmite ao roteador fonte apropriado.

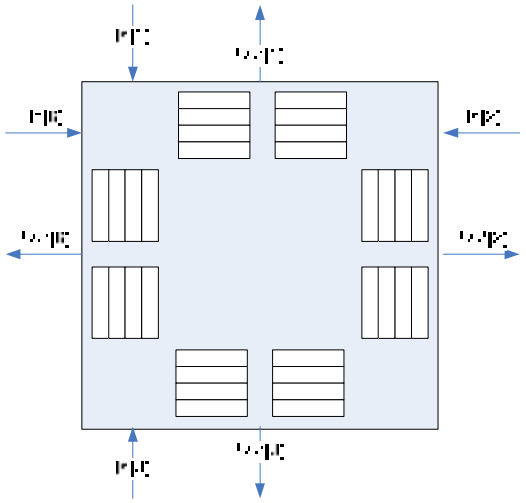


Figura 20 - Arquitetura do roteador Xpipes com 2 canais virtuais.

Como pode ser observado na Figura 21, cada módulo de saída tem 7 estágios de *pipeline* para maximizar a freqüência de operação de relógio do roteador. Os decodificadores CRC para detecção de erro trabalham em paralelo com a operação do roteador, desse modo ocultando sua latência.

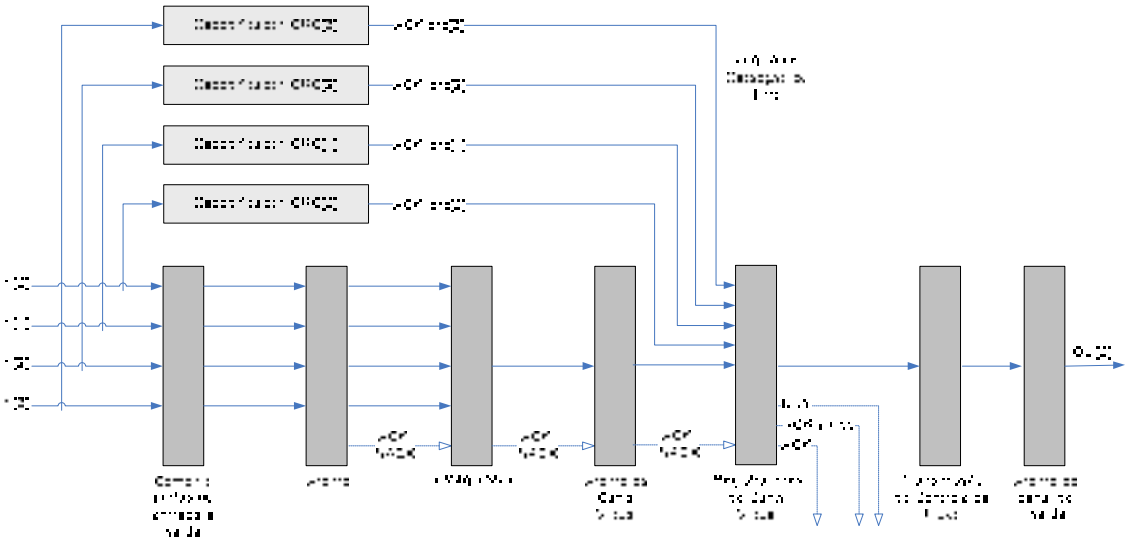


Figura 21 – Arquitetura do módulo de saída para cada porta de saída do roteador Xpipes.

O primeiro estágio do *pipeline* verifica o cabeçalho dos pacotes nas diferentes portas de entrada para determinar se os pacotes têm que ser roteados através de determinada porta de saída.

Somente pacotes direcionados para a esta porta de saída são enviados para o segundo estágio, no qual são resolvidas disputas baseadas em uma política *round-robin*. A arbitragem é realizada quando o *flit* terminador do pacote antecessor é recebido, de modo que todos os outros *flits* de um pacote possam ser propagados sem contenção referente ao atraso neste estágio. É gerado um nack para *flits* de pacotes não selecionados. O terceiro estágio possui apenas um multiplexador, o qual seleciona a porta de entrada prioritária. O estágio seguinte de arbitragem guarda o estado do *buffer* de canal virtual e determina se *flits* podem ser armazenados no *buffer* ou não. O *flit* de cabeçalho é enviado para o *buffer* com o maior número de posições livres, seguido por sucessivos *flits* do mesmo pacote. O quinto estágio é o estágio de bufferização, e a resposta ack/nack neste estágio indica se um *flit* foi armazenado corretamente ou não. O estágio seguinte cuida do envio do controle de fluxo: um *flit* é transmitido para o próximo roteador somente quando a porta de saída do roteador destino possuir posições livres disponíveis no *buffer* adequado. O último estágio de arbitragem multiplexa os canais virtuais no enlace do canal físico.

A metodologia de projeto baseado na Xpipes necessita uma ferramenta para instanciar os blocos construtivos da rede (roteadores, canais e interfaces de rede) em função de uma aplicação específica. Esta ferramenta é denominada *XpipesCompiler*. O fluxo de projeto adotado pela XpipesCompiler é mostrado na Figura 22. Partindo da especificação de uma aplicação, o projetista cria uma visão de alto nível do SoC, incluindo roteadores, canais e interfaces de rede. A informação sobre a arquitetura da rede é especificada em um arquivo de entrada para a XpipesCompiler. As tabelas de roteamento para as interfaces de rede também são especificadas. A ferramenta também utiliza a biblioteca Xpipes de componentes de rede como uma entrada adicional. A saída é uma descrição SystemC hierárquica, a qual inclui todos os roteadores, canais, interface de redes e suas conectividades. Então, a descrição final pode ser compilada e simulada com precisão ao nível de ciclo e ao nível de sinal.

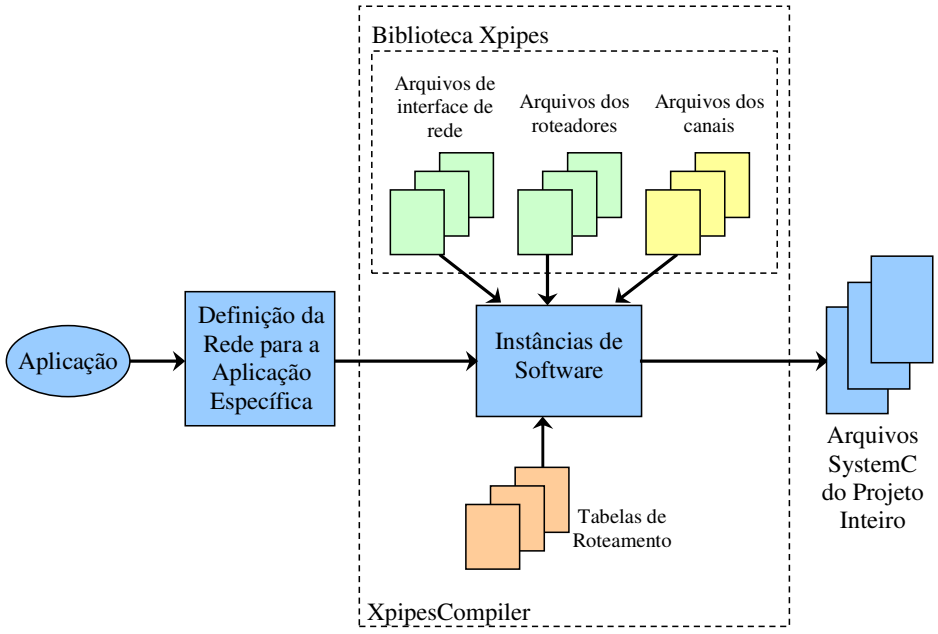


Figura 22 – Fluxo de projeto da rede Xpipes com a ferramenta XpipesCompiler.

3.2 QNoC

Bolotin et al. [BOL03] propuseram a rede intra-chip QNoC (*Quality of Service NoC*). A QNoC adota topologia malha irregular e emprega o chaveamento de pacotes *wormhole* com controle de fluxo baseado em créditos. A QNoC oferece quatro classes de serviço:

1. *Signalling*: nível de serviço com a maior prioridade na rede, para assegurar baixa latência. É utilizado por interrupções e sinais de controle.
2. *Real-time*: nível de serviço que garante largura de banda e latência para aplicações de tempo real.
3. *Read/Write* (RD/WR): nível de serviço projetado para suportar acessos curtos a memórias e registradores.
4. *Block-Transfer*: nível de serviço usado para transferência de mensagens longas e blocos grandes de dados, tal como conteúdo de *cache* e transferências DMA.

O pacote é dividido em *flits* que são classificados dentro dos seguintes tipos:

- FP (*Full Packet*): pacote com um único *flit*.
- EP (*End of Packet*): último *flit* do pacote.
- BDY(*Body*): não é o último *flit* do pacote.

O tipo do *flit* e o nível de serviço são indicados em fios separados de controle. A Tabela 3 descreve os sinais da porta de saída.

Tabela 3 – Sinais da porta de saída.

	Sinais	Largura (bit)	Descrição
Sinais de Saída	*Clk*	1	Relógio que sincroniza a transmissão de *flits*
	Data_o	Parametrizável	Dados saindo do roteador
	Type	2	Tipo do *flit*: 00: *Idle* 01: *EP* 10: *BDY* 11: *FP*
	SL	2	Nível de serviço do *flit*
Sinais de Entrada	*Buffer_Credit_SL*	4	Indica se existe espaço disponível no *buffer* para cada nível de serviço
	Buffer_Credit_valid	1	Indica se *Buffer_Credit_SL* transporta uma informação de crédito válido

A Figura 23 ilustra a arquitetura do roteador. O roteador suporta até cinco conexões: quatro para roteadores vizinhos e uma para o núcleo local. O roteador transfere pacotes das portas de entrada para as portas de saída. Dados são recebidos em *flits*. Cada *flit* que chega é primeiro

armazenado em *buffers* de entrada. Existem *buffers* separados para cada um dos quatro níveis de serviço. O algoritmo de roteamento determinístico XY é executado quando o primeiro *flit* do pacote é recebido. O primeiro *flit* contém o endereço destino do pacote e é utilizado para determinar a qual porta de saída o pacote é destinado. O número da porta de saída selecionada para a transmissão do pacote de cada nível de serviço é armazenado na tabela CRT (do inglês *Current Routing Table*). Quando um *flit* é enviado da porta de entrada para a porta de saída, uma posição no *buffer* torna-se disponível e um crédito é enviado ao roteador anterior. A porta de saída gerencia o número de posições disponíveis no *buffer* de cada nível de serviço da próxima porta de entrada. Este número é decrementado quando um *flit* é transmitido e incrementado quando um crédito do próximo roteador é recebido.

A porta de saída escalona a transmissão de *flits* de acordo com a disponibilidade de espaço no *buffer* do próximo roteador, com a prioridade do nível de serviço e com a arbitragem *round-robin* das portas de entrada esperando transmissão de pacotes dentro do mesmo nível de serviço. O número de espaços disponíveis no *buffer* do próximo roteador é armazenado na tabela NBS (do inglês *Next Buffer State*) de cada nível de serviço de cada porta de saída. A prioridade dos níveis de serviço é fixa, ordenada com *Signalling* tendo a maior prioridade, *Real-time* tendo a segunda, *RD/WR* a terceira e o *Block-Transfer* a última. O estado atual da arbitragem *round-robin* é armazenado na tabela CSIP (do inglês *Currently Serviced Input Port number*) para cada nível de serviço de cada porta de saída. Este número avança quando a transmissão de um pacote é finalizada ou se nenhuma transmissão de uma determinada porta de entrada e nível de serviço existe.

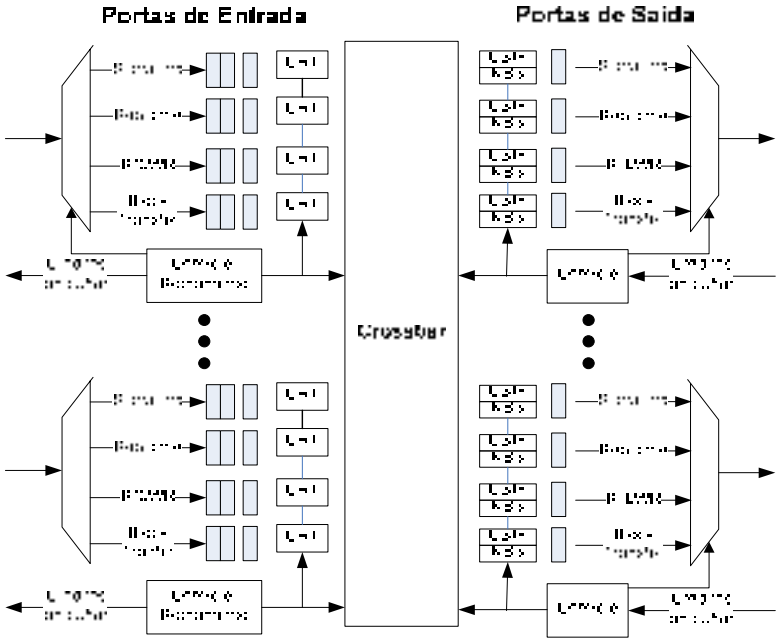


Figura 23 – Arquitetura do roteador da rede intra-chip QNoC.

Essa política de escalonamento implica que um *flit* seja transmitido pela porta de saída tão logo exista espaço disponível no *buffer* do próximo roteador e não exista nenhum pacote com prioridade maior pendente na porta de saída específica. Uma vez que um pacote com maior

prioridade chegue a uma porta de entrada, a transmissão do pacote atual é preemptada e o pacote de prioridade maior começa a ser transmitido. A transmissão de um pacote com menor prioridade é reiniciada somente depois que todos os pacotes com prioridade maior foram transmitidos.

A arquitetura da QNoC não faz bom uso do espaço disponível para armazenamento, porque particiona estaticamente o espaço do *buffer* em *T*/4, onde *T* é o espaço total de armazenamento e 4 é o número de classes de serviço. Deste modo, se um pacote de um dado serviço requisita uma porta de saída bloqueada, os *flits* do pacote são armazenados no *buffer* destinado ao serviço e o bloqueio é propagado aos demais roteadores no caminho do pacote, mesmo que exista espaço disponível no *buffer T*.

A QNoC possui um fluxo de projeto que permite que o projetista da rede altere e ajuste os parâmetros da rede a fim de satisfazer as exigências de um SoC particular. O fluxo de projeto da QNoC é apresentado na Figura 24.

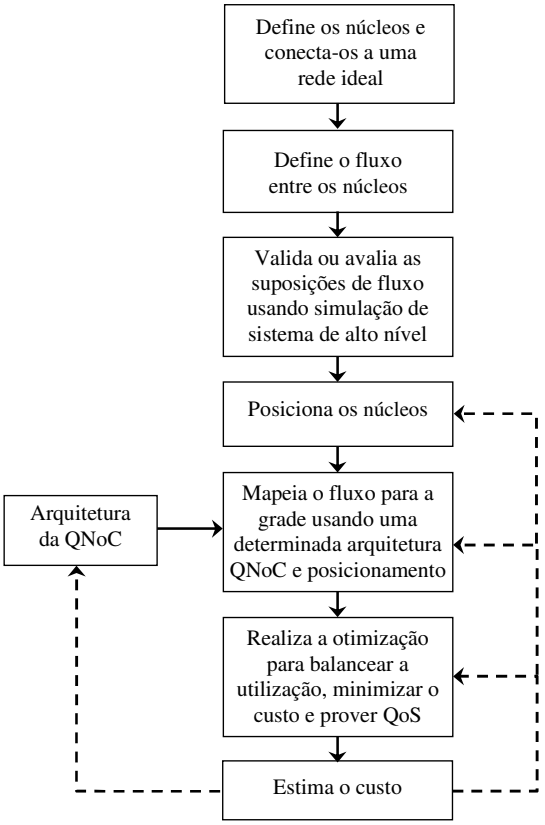


Figura 24 – Fluxo de projeto da QNoC [BOL04].

Primeiramente, os núcleos do sistema são definidos e conectados por uma infra-estrutura ideal de interconexão com largura de banda ilimitada e latência programável. Na seqüência, o fluxo entre núcleos é caracterizado. Esta caracterização é conduzida analisando os núcleos interconectados e sua especificação de fluxo. Para verificar a caracterização, o fluxo entre núcleos é medido e dividido em classes de serviço usando uma simulação de alto nível. Similarmente, as exigências de QoS são derivadas para cada classe de serviço, observando o desempenho real, bem como a avaliação através de simulação do efeito da latência e da vazão. Uma vez especificados os

padrões de tráfego, os núcleos são posicionados a fim de minimizar a densidade espacial do tráfego do sistema.

Somente após o posicionamento dos núcleos e os requisitos de comunicação entre núcleos serem determinados, a QNoC podem ser construída. A arquitetura da QNoC é finalizada, e os parâmetros arquiteturais são ajustados de acordo com o número de núcleos, a sua posição espacial, e os níveis de serviço de QoS a serem suportados. A topologia inicial é ajustada a uma *malha* e o tráfego requerido é traçado na grade de acordo com um algoritmo do roteamento, tal como o roteamento XY. Como as partes da grade podem não ser utilizadas inteiramente, alguns vértices e canais podem ser eliminados.

Uma vez que o algoritmo do roteamento é selecionado, caminhos de comunicação entre todos os pares de núcleos podem ser determinados e otimizações da largura de banda do canal podem ser realizadas. A carga média do tráfego em cada canal pode ser calculada, desde que o roteamento seja fixo e os padrões de tráfego sejam conhecidos previamente. A largura de banda do canal pode ser atribuída proporcionalmente à carga calculada nesse canal, variando o número dos fios em um canal ou sua freqüência. Desta maneira, o projetista calibra os recursos de sistema de modo que a utilização média de todos os canais na rede seja aproximadamente igual. Neste momento, o cálculo da carga média fornece somente larguras de banda de canal relativas. Para finalizar o projeto, a QNoC pode ser simulada e analisada mais precisamente por um simulador de rede. A largura de banda real pode então ser atribuída aos canais, de acordo com exigências de QoS e os resultados da simulação.

3.3 ÆThereal

Goossens et. al. propuseram a rede intra-chip Æthereal [GOO05][GOO03][GOO02]. Ela oferece serviços diferenciados com conexão. Uma conexão descreve a comunicação entre um núcleo origem e um ou mais núcleos destino, com um nível de serviço associado. Conexões devem ser criadas expressando o nível de serviço requisitado. A aceitação da conexão pode incluir a reserva de recursos na rede, tais como *buffers* ou o percentual de largura de banda do canal. Depois do uso, a conexão é finalizada e os recursos liberados. Diferentes conexões podem ser criadas ou finalizadas independentemente.

A rede intra-chip Æthereal possui um roteador que combina vazão garantida (GT, do inglês *Guaranteed Throughput*) e melhor esforço (BE), como ilustrado na Figura 25. Desse modo é garantido desempenho, mesmo quando os fluxos estão transmitindo com vazão máxima (cenário de pior caso), aliado a uma boa média de uso de recursos. O serviço GT oferece uma latência fim-a-fim fixa e tem prioridade mais alta, forçada pelo árbitro. O serviço BE usa toda a largura de banda que não está reservada ou não é usada pelo tráfego GT. Recursos são sempre utilizados quando existem dados disponíveis.

As conexões GT são criadas ou removidas através de pacotes BE que reservam um caminho na rede, dependendo da disponibilidade de recursos (linha "*program*" na Figura 25). Os pacotes BE usam roteamento *wormhole* na origem, e os fluxos GT usam chaveamento por circuito

com STDM. Roteadores usam *buffers* virtuais de saída para o fluxo BE. O fluxo GT é fortemente escalonado para minimizar a bufferização. Roteadores não descartam ou ordenam pacotes. Uma variante do algoritmo de escalonamento *iSLIP* [KUM04] é usada para o fluxo BE.

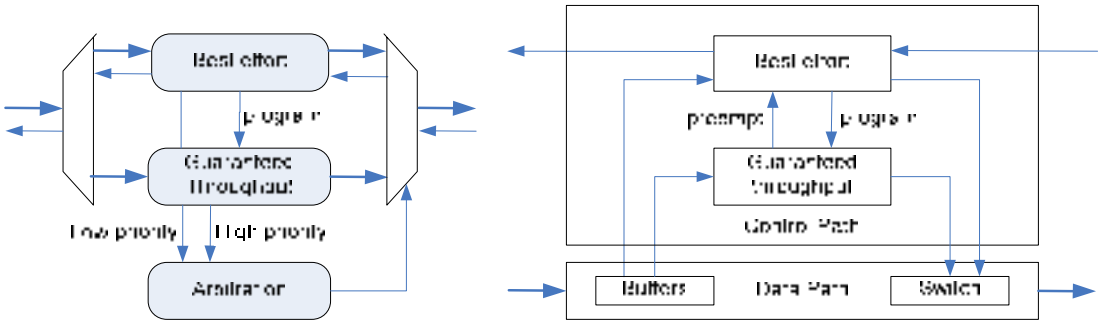


Figura 25 – Duas visões do roteador da Æthereal, que combina fluxos BE e GT.

A construção de uma rede Æthereal baseia-se em um fluxo de projeto que possui como etapas principais a geração, configuração e a verificação/simulação, como ilustrado na Figura 26. Segundo os proponentes desta rede, tal divisão traz como benefícios: (*i*) simplificação para se adotar heurísticas, aumentando o controle do usuário sobre o processo; (*ii*) redução da complexidade para otimização da rede; e (*iii*) facilidade para o usuário personalizar, adicionar ou substituir partes do fluxo de forma a melhorar seu desempenho. No fluxo de projeto Æthereal, a construção da rede (geração de topologia e mapeamento dos núcleos) é realizada ao mesmo tempo que o seu balanceamento (dimensionamento dos canais com menor carga), diferentemente do que acontece na QNoC [BOL03], onde a construção da rede é feita, seguida da análise interna de desempenho, para posteriormente otimizar a rede.

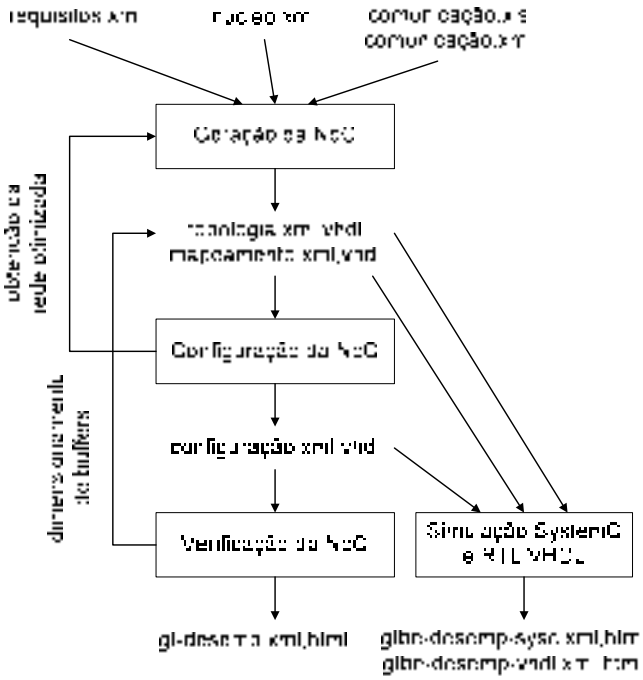
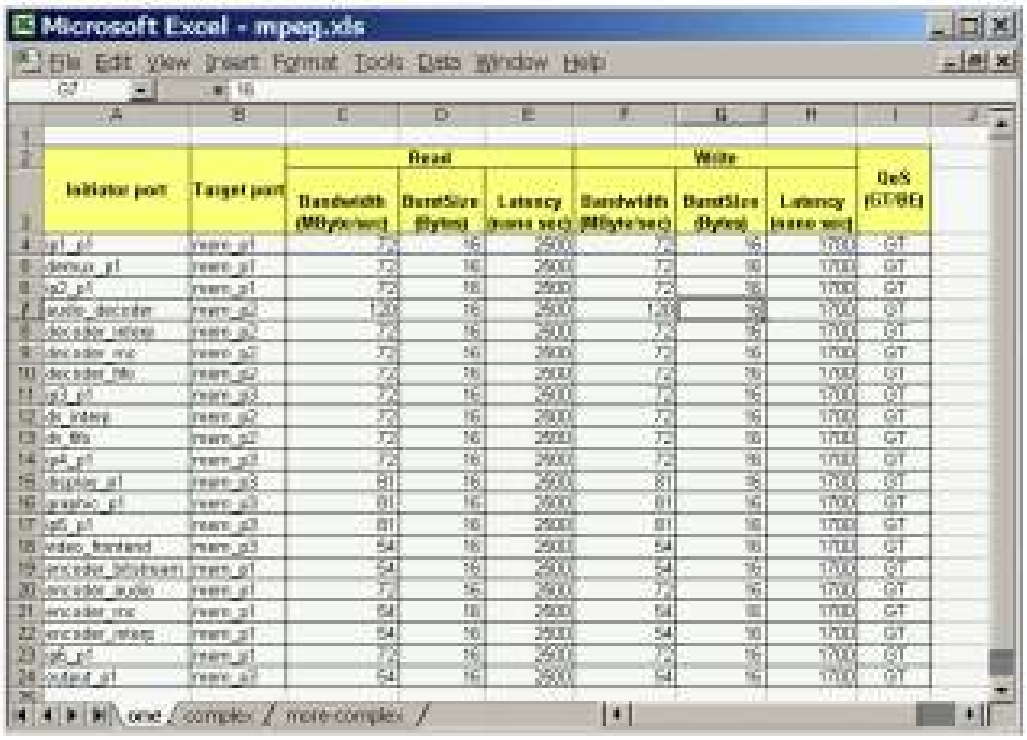


Figura 26 – Fluxo de projeto da rede Æthereal.

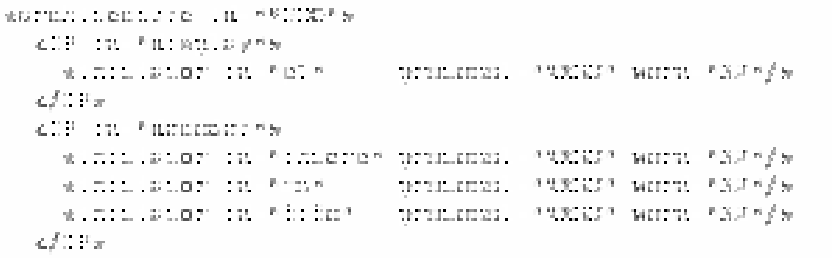
A etapa de geração da rede intra-chip recebe como dados de entrada arquivos formatados em XML contendo:

- Requisitos da aplicação (requisitos.xml e comunicação.xml/xls), onde são especificados: uma lista de conexões, sendo que cada conexão especifica qual mestre se comunicará com qual escravo (padrão espacial de tráfego), a largura de banda mínima a ser utilizada, a latência máxima permitida, o tamanho da rajada para leitura ou escrita de dados e a classe de serviço que deve atender o fluxo (podendo ser BE ou GT). Um exemplo de especificação da aplicação (arquivo de comunicação) é mostrado na Figura 27(a);
- Especificação dos núcleos conectados à rede (núcleo.xml): onde cada porta deve especificar o protocolo de comunicação com a rede e a largura da palavra de dados. Um exemplo de especificação de núcleos é mostrado na Figura 27(b).

Ao final da etapa de geração são criados os arquivos XML descrevendo a topologia e o mapeamento dos núcleos, os quais são entradas para a etapa de configuração.



(a)



(b)

Figura 27 – (a) Especificação das aplicações; (b) Especificações dos núcleos.

Na etapa de configuração da rede é gerado um arquivo com informações para síntese do hardware contendo, para cada conexão, o caminho do mestre até o escravo e o número de créditos para controle de fluxo. Um exemplo de arquivo de configuração gerado é mostrado na Figura 28.

A etapa de verificação da rede recebe como entrada o arquivo de configuração e são realizados cálculos analíticos para determinar a vazão mínima, a latência máxima, e a quantidade máxima de espaços em *buffer* utilizados. Os resultados obtidos são comparados aos requisitos das aplicações (descritos nos arquivos de comunicação) e mostrados em uma tabela indicando através de cores se os requisitos foram atendidos (verde) ou não (vermelho). Um exemplo de arquivo de verificação é mostrado na Figura 29.

Figura 28 – Exemplo de arquivo de configuração.

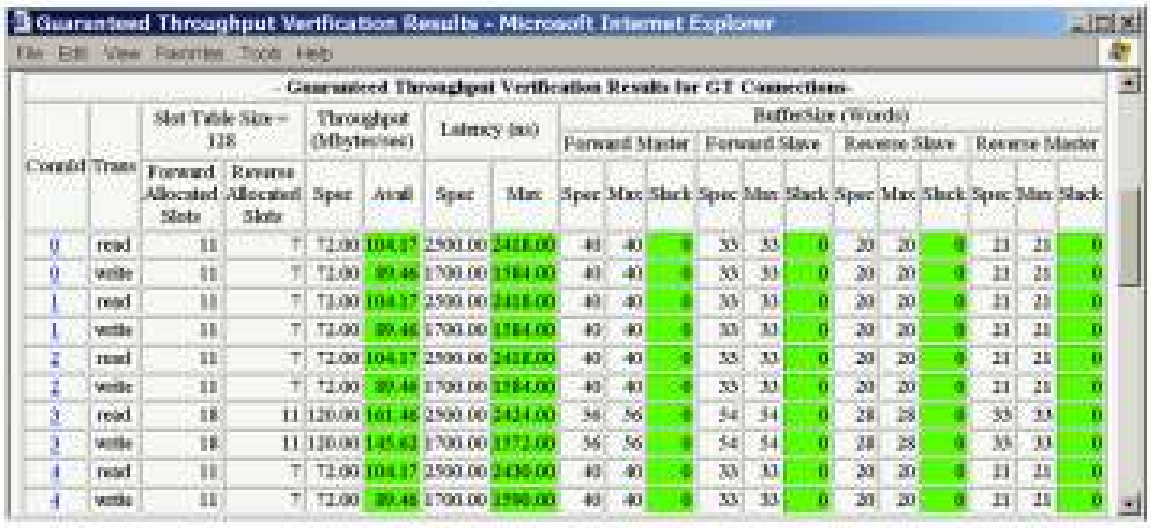
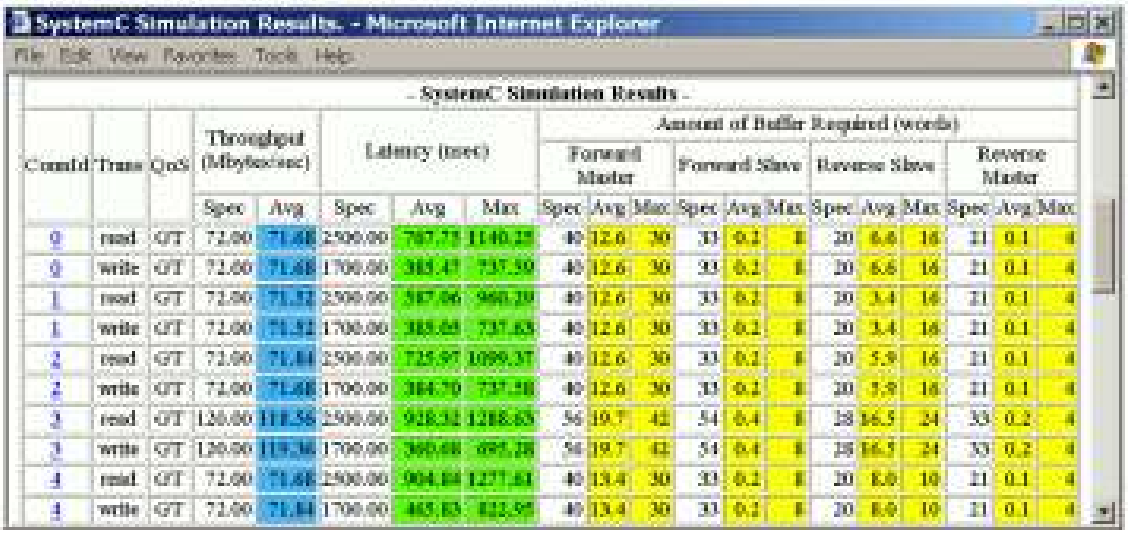


Figura 29 – Exemplo de arquivo de verificação.

A etapa de verificação oferece suporte apenas a comunicações GT, e calcula os valores de pior caso para vazão, latência e utilização de *buffers*, sem considerar casos médios. A etapa de simulação oferece suporte à avaliação de desempenho no caso médio para conexões tanto BE quanto GT. Dois tipos de simulação podem ser adotados: RTL VHDL (com precisão de bit e de ciclo) e SystemC (nível de *flit*). Esta etapa recebe como entrada o mesmo arquivo de configuração recebido pela verificação. Como saída a etapa de simulação gera uma tabela (Figura 30) semelhante à gerada no processo de verificação, tendo como resultados adicionais as medições para valores médios de vazão, latência e alocação de *buffers*.



SystemC Simulation Results

CommId	Trans	QoS	Throughput (Mbytes/sec)		Latency (nsec)			Amount of Buffer Required (words)											
								Forward Master			Forward Slave			Reverse Slave			Reverse Master		
			Spec	Avg	Spec	Avg	Max	Spec	Avg	Max	Spec	Avg	Max	Spec	Avg	Max	Spec	Avg	Max
0	read	GT	72.00	71.68	2500.00	787.75	1140.25	40	12.6	30	33	0.2	8	20	6.6	16	21	0.1	4
0	write	GT	72.00	71.68	1700.00	383.47	737.59	40	12.6	30	33	0.2	8	20	6.6	16	21	0.1	4
1	read	GT	72.00	71.52	2500.00	787.66	960.29	40	12.6	30	33	0.2	8	20	3.4	16	21	0.1	4
1	write	GT	72.00	71.52	1700.00	388.05	737.63	40	12.6	30	33	0.2	8	20	3.4	16	21	0.1	4
2	read	GT	72.00	71.84	2500.00	725.97	1099.37	40	12.6	30	33	0.2	8	20	5.9	16	21	0.1	4
2	write	GT	72.00	71.68	1700.00	384.70	737.38	40	12.6	30	33	0.2	8	20	5.9	16	21	0.1	4
3	read	GT	120.00	119.58	2500.00	928.32	1288.63	56	19.7	42	54	0.4	8	28	16.5	24	33	0.2	4
3	write	GT	120.00	119.36	1700.00	360.68	695.28	56	19.7	42	54	0.4	8	28	16.5	24	33	0.2	4
4	read	GT	72.00	71.68	2500.00	904.84	1277.61	40	13.4	30	33	0.2	8	20	8.0	10	21	0.1	4
4	write	GT	72.00	71.84	1700.00	465.83	822.95	40	13.4	30	33	0.2	8	20	8.0	10	21	0.1	4

Figura 30 – Exemplo de arquivo com resultados de simulação.

4 CANAIS VIRTUAIS NA REDE INTRA-CHIP HERMES

Este Capítulo apresenta a primeira contribuição deste trabalho, a inserção de canais virtuais na rede Hermes (Hermes-VC). A inserção de canais virtuais permite que mecanismos de alocação de recursos sejam utilizados pela rede, possibilitando que fluxos sejam tratados de forma diferenciada. No entanto, é importante ressaltar que a rede Hermes-VC fornece apenas o serviço BE, ou seja, todos os fluxos são tratados de forma igual. A diferenciação entre fluxos é apresentada nos próximos Capítulos. A Seção 4.1 descreve o projeto da rede Hermes-VC. A validação funcional do roteador Hermes-VC é apresentada na Seção 4.2.

As redes descritas neste Capítulo e nos Capítulos seguintes usam o mesmo conjunto básico de características: topologia malha bidirecional, chaveamento por pacote *wormhole*, roteamento XY (determinístico e distribuído), e canais físicos multiplexados em pelo menos dois canais virtuais. Isso certamente não cobre todas as possíveis características encontradas nas arquiteturas de redes propostas na literatura. Entretanto, todo ou um conjunto dessas características são encontradas na maioria das redes [BJE06][MOR04].

4.1 Hermes-VC

A rede Hermes [MOR04] é uma infra-estrutura para a geração de redes intra-chip com topologia malha bidirecional, chaveamento por pacote *wormhole* e baixa sobrecarga de área. Diz-se infra-estrutura porque não se trata de uma única rede intra-chip. Existe um conjunto de parâmetros definíveis pelo usuário (Apêndice II), tais como: (*i*) o controle de fluxo; (*ii*) a dimensão da rede; (*iii*) a largura do canal de comunicação; (*iv*) a profundidade dos *buffers*; e (*v*) o algoritmo de roteamento. O uso da topologia malha bidirecional é justificado por facilitar as tarefas de posicionamento e de roteamento em circuitos integrados e/ou FPGAs. Nessa topologia, cada roteador tem um número diferente de portas, dependendo de sua posição no que diz respeito aos limites da rede, conforme apresentado na Figura 31. Por exemplo, o roteador central tem cinco portas. Entretanto, cada roteador do limite da rede tem somente três ou quatro portas.

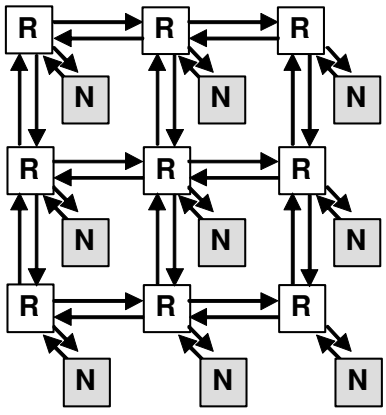


Figura 31 - Topologia malha bidirecional 3x3. *N* representa os núcleos IP. *R* representa os roteadores.

O chaveamento por pacote *wormhole* implica a divisão do pacote em *flits*. O tamanho do *flit* na infra-estrutura Hermes é parametrizável, e o número máximo de *flits* em um pacote é fixado em $2^{(\text{tamanho do } flit \text{ em bits})}$-1. O primeiro e o segundo *flit* de um pacote são informações de cabeçalho, sendo respectivamente o endereço do roteador destino e o número de *flits* do corpo do pacote. O chaveamento por pacote *wormhole* permite que cada canal físico seja multiplexado em *l* canais virtuais. Embora a multiplexação de canais físicos aumente o desempenho do chaveamento [RIJ01], é importante manter o compromisso entre o desempenho, a complexidade e a área do roteador.

A rede Hermes suporta duas estratégias de controle de fluxo: (*i*) *handshake* e (*ii*) *credit based*. O controle de fluxo baseado em créditos (*credit based*) é utilizado ao longo do presente trabalho. Trata-se de um protocolo síncrono que possui melhor desempenho se comparado ao protocolo *handshake* [CAR04]. O protocolo de controle de fluxo, para permitir o compartilhamento do canal físico por *l* canais virtuais, deve ser capaz de distinguir entre os *l* canais virtuais usando o canal físico.

A interface de comunicação entre roteadores Hermes-VC vizinhos é apresentada na Figura 32. Os seguintes sinais compõem a porta de saída: (1) *Clock_tx*: sincroniza a transmissão de dados; (2) *Tx*: indica disponibilidade de dado; (3) *Lane_tx*: indica o canal virtual transmitindo dado; (4) *Data_out*: dado a ser transmitido; (5) *Credit_in*: informa disponibilidade de *buffer* no roteador vizinho, para cada canal virtual. O número de canais virtuais (*l lanes*) e a largura do barramento de dados (*n bits*) são parametrizáveis em função dos recursos de roteamento disponíveis e da memória disponível para esquemas de bufferização.

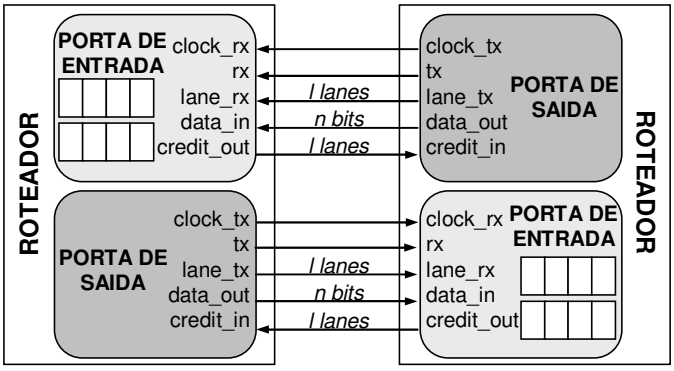


Figura 32 – Interfaces entre roteadores Hermes-VC.

O roteador Hermes possui uma lógica de controle de chaveamento centralizada e até 5 portas bidirecionais: East, West, North, South e Local. A porta Local estabelece a comunicação entre o roteador e seu núcleo local. As outras portas do roteador são conectadas aos roteadores vizinhos. Cada porta unidirecional (de entrada ou de saída) do roteador corresponde a um canal físico. A Figura 33 apresenta a arquitetura do roteador Hermes-VC com dois canais virtuais por canal físico.

Cada porta de entrada tem um *buffer* para diminuir a perda de desempenho com o bloqueio de *flits*. A perda de desempenho ocorre porque quando um *flit* é bloqueado em um dado roteador, os *flits* seguintes do mesmo pacote também são bloqueados neste, em outros roteadores ou no núcleo

local. Com a inserção de um *buffer*, o número de roteadores afetados pelo bloqueio dos *flits* potencialmente diminui. O *buffer* inserido no roteador Hermes funciona como uma fila FIFO (do inglês *First In First Out*) circular, cuja profundidade *p* é parametrizável. Quando o canal físico é dividido em *l* canais virtuais, um *buffer* com profundidade *p*/*l* é associado a cada canal virtual. Por exemplo, o roteador apresentado na Figura 33 possui um espaço de armazenamento de 8 *flits* por porta de entrada. Como cada canal físico é multiplexado em dois canais virtuais, o *buffer* associado a cada canal virtual tem profundidade de 4 *flits* (8/2).

A porta de entrada recebe *flits* e armazena-os no *buffer* do canal virtual indicado pelo sinal *lane_rx* (Figura 32). Depois, o número de créditos (espaços livres para armazenamento) do canal virtual é decrementado. Quando a porta de saída transmite um *flit*, este *flit* é removido do *buffer* e o número de créditos é incrementado. Os créditos disponíveis alcançam o roteador vizinho através do sinal *credit_out* (Figura 32).

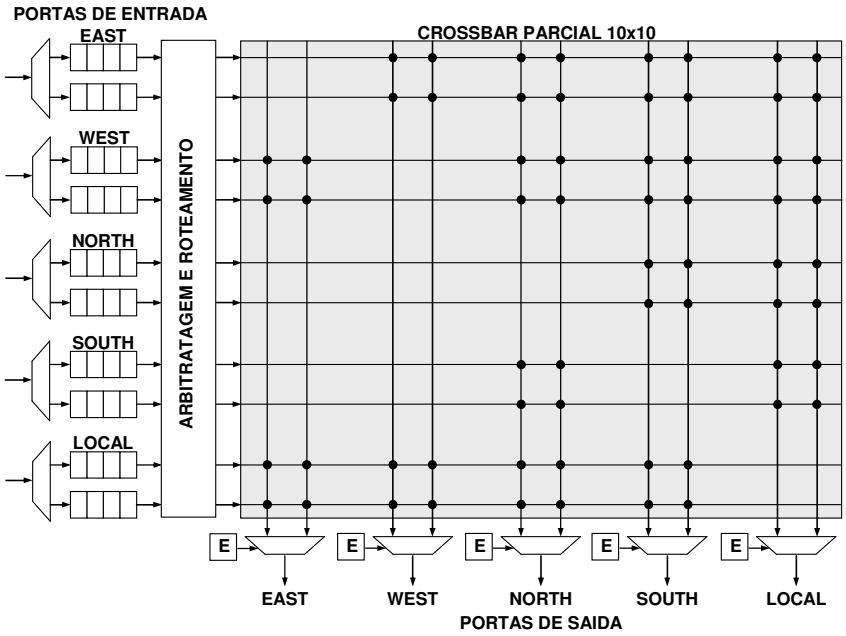


Figura 33 – Arquitetura do roteador Hermes-VC com dois canais virtuais e roteamento XY.

Múltiplos pacotes podem chegar simultaneamente em um dado roteador. Uma arbitragem *round-robin* centralizada garante acesso a um único canal virtual de entrada quando um ou mais canais virtuais requisitam simultaneamente roteamento. Esta arbitragem utiliza um esquema de prioridades dinâmico, proporcionando um serviço mais justo que a prioridade estática. Em geral, políticas mais justas possuem um melhor desempenho global. A prioridade atribuída a um canal virtual é determinada pelo último canal virtual a ter uma requisição de acesso ao roteamento atendida. Por exemplo, se o canal virtual de entrada L2 da porta Local foi o último a ter uma requisição de roteamento atendida, o canal virtual de entrada L1 da porta East terá a maior prioridade, sendo seguido pelo canal virtual de entrada L2 da porta East, como apresentado na Figura 34. Esse método garante que todas as requisições serão atendidas em algum momento, prevenindo a ocorrência de *starvation*.

Porta	East		West		North		South		Local	
Canal Virtual	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2
Prioridade	9	8	7	6	5	4	3	2	1	0

Figura 34 – Prioridade dos canais virtuais quando o canal virtual de entrada L2 da porta Local foi o último a ter a requisição de roteamento atendida, onde valores mais altos correspondem a prioridade mais alta.

Se o pedido de roteamento do pacote é atendido pelo árbitro, o algoritmo de roteamento XY é executado para conectar o canal virtual de entrada a um canal virtual de saída. O algoritmo de roteamento XY encaminha os pacotes primeiro na direção X e depois na direção Y. O comportamento deste algoritmo permite a implementação de um *crossbar* parcial, como ilustrado na Figura 33. Pacotes recebidos pelas portas Local, East ou West podem ser transmitidos por qualquer porta, exceto pela porta de recebimento. Pacotes recebidos pela porta North podem ser transmitidos somente pelas portas Local e South, enquanto que pacotes recebidos pela porta South podem ser transmitidos somente pela porta Local e North. A adoção de um *crossbar* parcial reduz a área do roteador em 3% comparado a um *crossbar* completo.

O algoritmo de roteamento XY pode retornar uma porta de saída livre ou ocupada. Quando o algoritmo retorna uma porta de saída ocupada, todos os *flits* deste pacote são bloqueados. Quando o algoritmo retorna uma porta de saída livre, a conexão entre o canal virtual de entrada e o canal virtual de saída é estabelecida e uma tabela de chaveamento é atualizada. No roteador Hermes-VC, uma porta (canal físico) é considerada ocupada apenas quando todos os seus canais virtuais estão ocupados. No algoritmo de roteamento XY não é necessário estabelecer uma ordem entre os canais virtuais, porque a restrição de percorrer primeiro a coordenada X e depois a coordenada Y é suficiente para evitar a ocorrência de *deadlock*.

A Figura 35 ilustra o caminho percorrido por quatro pacotes em uma rede Hermes-VC 8x8 com dois canais virtuais (L1 e L2). Assume-se que: (*i*) L2 somente é utilizado quando L1 está ocupado; (*ii*) alguns canais virtuais podem estar bloqueados (**x** na Figura 35); (*iii*) um pacote usa sempre o mesmo canal virtual em um mesmo canal físico, porém pode usar canais virtuais diferentes em canais físicos diferentes, como ocorre nos caminhos 2, 3 e 4.

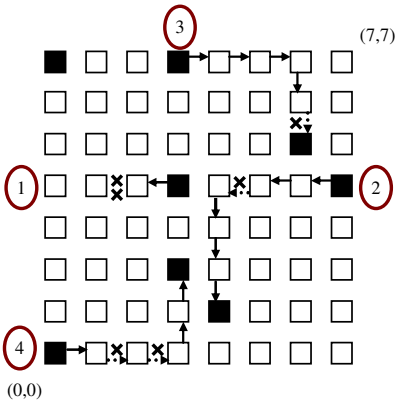


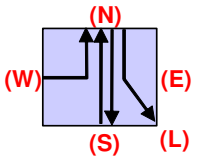
Figura 35 – Caminho percorrido por 4 pacotes em uma rede Hermes-VC 8x8 com dois canais virtuais (L1 e L2), onde quadrados representam roteadores, quadrados pretos representam os roteadores origem ou destino de algum pacote, setas contínuas representam o canal virtual L1, setas pontilhadas representam o canal virtual L2 e x representam canais bloqueados.

A informação de ocupação dos canais virtuais das portas de um roteador é coletada em uma tabela de chaveamento, cuja estrutura é ilustrada na Figura 36. A tabela de chaveamento é composta por três vetores: *in*, *out* e *free*. O vetor *in* descreve conexões entre um canal virtual de entrada (índice de *In*) e um canal virtual de saída (conteúdo de *In*). O vetor *out* descreve conexões de um canal virtual de saída (índice de *Out*) a um canal virtual de entrada (conteúdo de *Out*). O vetor *free* armazena o estado dos canais virtuais de saída: livre (1) ou ocupado (0). Os vetores *In* e *Out* são preenchidos por um identificador único (*id*) construído pela combinação do número da porta (*np*), do número de canais virtuais por canal físico (*ncv*) e pelo número do canal virtual (*cv*), como apresentado na Equação 4:

$$id = (np \times ncv) + cv \tag{4}$$

As portas East, West, North, South e Local são numeradas de 0 a 4, respectivamente. Os canais virtuais L1, L2 ... L*n* de cada porta são numeradas de 0 a *n*-1, respectivamente.

Considere um roteador com dois canais virtuais por canal físico, o canal virtual L1 da porta North possui o identificador 4 ($(2\times2)+0$) e o canal virtual L2 possui o identificador 5 ($(2\times2)+1$). A tabela de chaveamento apresentada na Figura 36(b) representa o estado de chaveamento ilustrado na Figura 36(a). Considere a porta North. O canal virtual de saída L1 está ocupado (*free* = 0) e conectado ao canal virtual de entrada L1 da porta West (*out* = 2). O canal virtual de saída L2 está ocupado (*free* = 0) e conectado ao canal virtual de entrada L1 da porta South (*out* = 6). O canal virtual de entrada L1 está conectado ao canal virtual de saída L1 da porta Local (*in* = 8) e o canal virtual de entrada L2 está conectado ao canal virtual de saída L1 da porta South (*in* = 6). A tabela de chaveamento apresenta informações redundantes com o objetivo de aumentar o desempenho do algoritmo de roteamento.



(a)

Porta	E		W		N		S		L	
VC	0 – L1	1 – L2	2 – L1	3 – L2	4 – L1	5 – L2	6 – L1	7 – L2	8 – L1	9 – L2
Free	1	1	1	1	0	0	0	1	0	1
In			4		8	6	5			
Out					2	6	5		4	

(b)

Figura 36 – (a) Exemplo de estado de chaveamento. (b) Tabela de chaveamento correspondente.

Depois do roteamento, um escalonador é responsável por alocar a largura de banda entre os *l* canais virtuais de cada porta de saída. Cada canal virtual, se tem *flits* para transmitir e créditos para transmissão usa no mínimo $1/l$ da largura de banda do canal físico. Se um único canal virtual satisfaz esta condição, ele usa toda largura de banda do canal físico. A Figura 37 ilustra um exemplo de alocação de banda de um canal físico. O canal físico é compartilhado por dois canais virtuais, onde um transmite o pacote A e o outro o pacote B. Quatro situações de alocação são apresentadas na Figura. A primeira situação corresponde à alocação da metade da largura da banda para cada canal virtual, supondo que ambos possuem créditos para transmissão. A segunda e a quarta situações ilustram a alocação de toda a largura de banda para apenas um dos canais virtuais, porque o outro não possui créditos para transmissão. Na terceira situação, a largura de banda não é utilizada, porque nenhum dos canais virtuais possui créditos para transmissão.

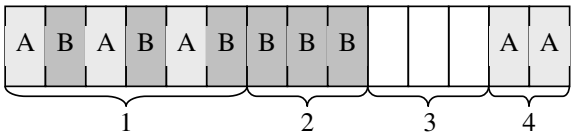


Figura 37 – Exemplo de alocação da largura de banda de um canal físico.

Depois que todos os *flits* do pacote são transmitidos, a conexão entre o canal virtual de entrada e o canal virtual de saída deve ser encerrada. Isto pode ser feito de dois modos diferentes: por um *trailer* ou usando um contador de *flits*. Um *trailer* requer um ou mais *flits* para serem usados como terminador de pacote e uma lógica adicional é necessária para detectar o *trailer*. O roteador Hermes-VC possui um contador para cada canal virtual de entrada. O contador de um canal virtual específico é inicializado quando o segundo *flit* do pacote é recebido, indicando o número de *flits* que compõe a carga útil do pacote. O contador é decrementado a cada *flit* transmitido com sucesso. Quando o valor do contador alcança zero, a posição do vetor *free* correspondente ao canal virtual de saída vai para um (*free* = 1), encerrando a conexão.

4.2 Validação Funcional do Roteador Hermes-VC

O roteador Hermes-VC foi descrito em VHDL e validado por simulação funcional. A Figura 38 mostra a simulação funcional de um roteador recebendo um pacote pelo canal virtual L1 da porta West e transmitindo-o pelo canal virtual L1 da porta East. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 38.

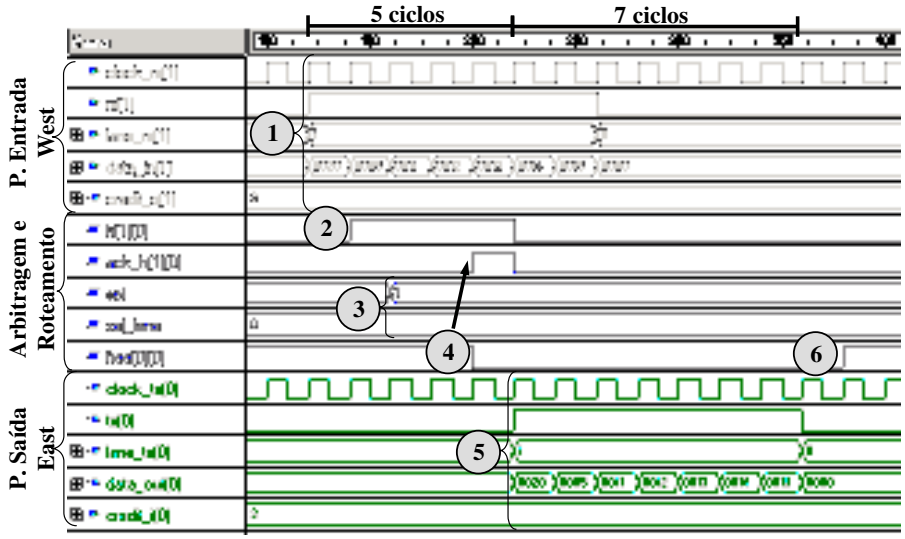


Figura 38 – Simulação do roteador Hermes-VC recebendo um pacote pela porta de entrada West e transmitindo-o pela porta de saída East.

1. O roteador recebe um pacote pelo canal virtual L1 da porta West (índice 1).
2. O canal virtual L1 da porta de entrada West requisita roteamento ativando o sinal *h* (*h*(1)(0)).
3. A arbitragem é executada e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada West (*sel* = 1) recebe permissão de roteamento.

4. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(0) = 0), e o sinal *ack_h* é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta de saída East foi estabelecida.
5. O roteador começa a transmitir o pacote pelo canal virtual L1 da porta de saída East (índice 0).
6. Todos os *flits* do pacote são transmitidos e a conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta e saída East é encerrada (*free*(0)(0) = 1).

A latência ideal em ciclos de relógio para transferir um pacote da origem para o destino é dada pela Equação 5:

$$latência\ ideal = 5n + P \tag{5}$$

onde: 5 é o número de ciclos de relógio requerido pelo algoritmo de arbitragem/roteamento para cada roteador, *n* é o número de roteadores no caminho da comunicação (origem e destino inclusive), *P* é o tamanho do pacote em *flits*.

Na simulação apresentada na Figura 38, observa-se que o pacote é transmitido pelo roteador com latência igual a ideal, 13 ciclos de relógio, onde 5 ciclos de relógio são gastos pelo algoritmo de arbitragem/roteamento e 7 ciclos de relógio são gastos na transmissão dos sete *flits* que compõem o pacote.

Uma simulação funcional do compartilhamento da largura de banda do canal de saída East por dois pacotes oriundos das portas de entrada Local e West é apresentada na Figura 39. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 39.

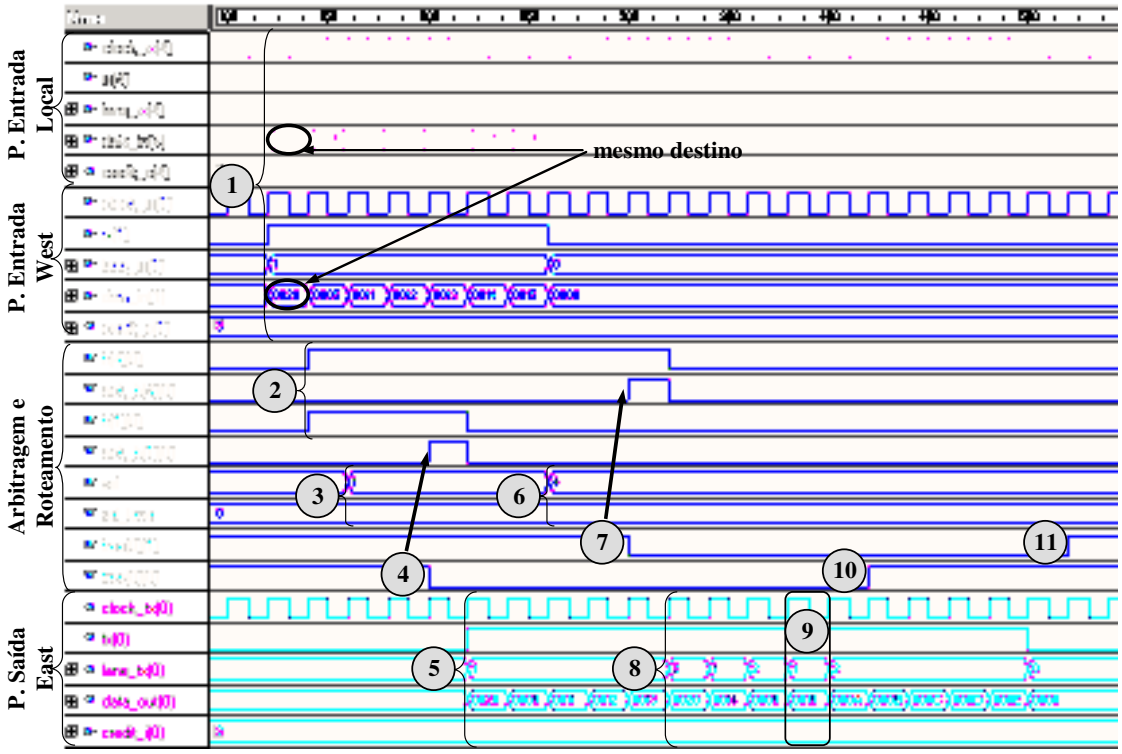


Figura 39 – Simulação do roteador recebendo simultaneamente dois pacotes, um pela porta de entrada Local e outro pela porta de entrada West. Ambos têm como destino a porta de saída East.

1. O roteador recebe simultaneamente um pacote pela porta de entrada Local (índice 4) e outro pela porta de entrada West (índice 1), ambos destinados à porta de saída East (índice 0).
2. Ambos requisitam roteamento ativando o sinal *h*.
3. A arbitragem é executada e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada West (*sel* = 1) recebe permissão de roteamento.
4. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(0) = 0), e o sinal *ack_h* é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta de saída East foi estabelecida.
5. O roteador começa a transmitir o pacote oriundo do canal virtual L1 da porta de entrada West pelo canal virtual L1 da porta de saída East (índice 0).
6. A arbitragem é executada novamente e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada Local (*sel* = 4) recebe permissão de roteamento.
7. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(1) = 0), e o sinal *ack_h* é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada Local e o canal virtual L2 da porta de saída East foi estabelecida. O canal virtual L2 é selecionado para transmissão porque o canal virtual L1 já está ocupado.
8. A largura de banda da porta de saída East começa a ser **compartilhada** pelos canais virtuais L1 e L2.
9. O canal virtual L1 envia o último *flit* do pacote e a largura de banda da porta de saída East passa a ser totalmente alocada para o canal virtual L2.
10. A conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta de saída East é encerrada (*free*(0)(0) = 1).
11. A conexão entre o canal virtual L1 da porta de entrada Local e o canal virtual L2 da porta de saída East é encerrada (*free*(0)(1) = 1).

Na simulação apresentada na Figura 39, observa-se as quatro situações de compartilhamento de largura de banda ilustradas na Figura 37. No instante inicial da simulação, nenhum canal virtual possui *flits* para transmitir. No intervalo de tempo entre 220ns e 320ns, o canal virtual L1 começa a transmitir *flits* utilizando a largura de banda total do canal físico. Entre o intervalo de 320ns e 400ns, os canais virtuais L1 e L2 compartilham o canal físico, cada canal virtual alocando a metade da largura de banda. Em 400ns, o canal virtual L1 termina de transmitir seus *flits* e o canal virtual L2 passa a alocar toda largura de banda. Em 500ns o canal virtual L2 termina de transmitir seus *flits* e a condição inicial é retomada.

5 CHAVEAMENTO POR CIRCUITO SOBRE A REDE HERMES

Este Capítulo apresenta a segunda contribuição deste trabalho, a diferenciação entre fluxos através do estabelecimento de conexão. Neste projeto, a rede oferece um serviço com vazão garantida (GT) aos fluxos com requisitos de QoS e um serviço BE aos fluxos sem requisitos de QoS. Este método, GT mais BE, é similar ao implementado na rede Æthereal. A Seção 5.1 descreve o projeto da rede Hermes com chaveamento por circuito (Hermes-CS). A validação funcional desta rede é apresentada na Seção 5.2.

5.1 Hermes-CS

Este projeto parte de uma versão da Hermes com dois canais virtuais, L1 e L2. O canal virtual L1 transporta dados utilizando chaveamento por circuito, enquanto o canal virtual L2 transporta dados utilizando chaveamento por pacote. Fluxos GT têm maior prioridade do que fluxos BE, com garantia de latência fim-a-fim. Quando um fluxo GT deixa um canal físico livre, fluxos BE podem usar este canal, sem com isto penalizar o fluxo GT. A Figura 40 ilustra o compartilhamento da largura de banda da porta North por um fluxo GT (oriundo da porta South) e um fluxo BE (oriundo da porta West). Observe que o fluxo BE utiliza a largura de banda não utilizada pelo fluxo GT.

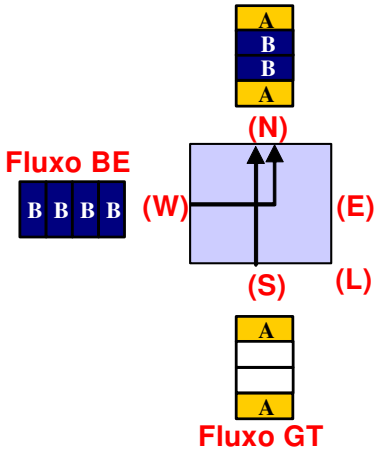


Figura 40 – Exemplo do compartilhamento da largura de banda em um roteador Hermes-CS, onde o fluxo BE utiliza a largura de banda não utilizada pelo fluxo GT.

A Figura 41 ilustra a interface entre os roteadores deste projeto. Os seguintes sinais compõem a porta de saída: (1) *clock_tx*: sincroniza a transmissão de dados; (2) *tx*: indica dado disponível; (3) *lane_tx*: indica o canal virtual transmitindo o dado; (4) *data_out*: dado a ser enviado; (5) *credit_in*: indica existência de espaço disponível no *buffer* do canal virtual L2; (6) *ack_out*: indica a criação "1" ou remoção "0" de uma conexão.

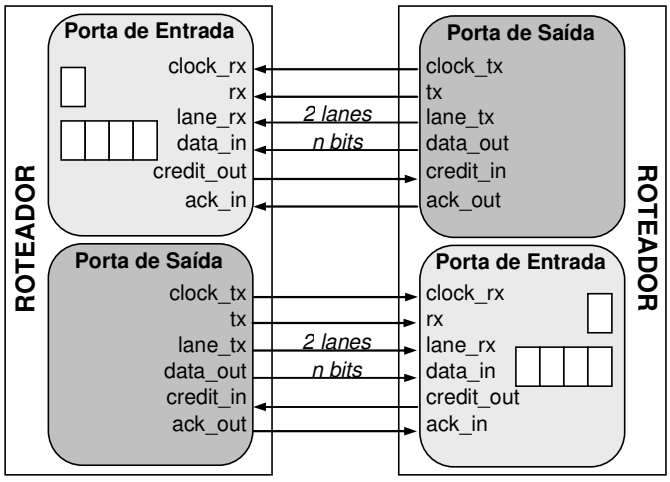


Figura 41 – Interface física entre roteadores Hermes-CS.

Um fluxo GT requer o estabelecimento de uma conexão antes da transmissão dos dados. Uma conexão entre um núcleo origem e um núcleo destino requer a reserva do canal virtual L1 ao longo do caminho entre seus respectivos roteadores. Esta reserva de caminho evita outras conexões no mesmo caminho.

A implementação do chaveamento por circuito é mais simples do que o chaveamento por pacote, permitindo que: (*i*) um registrador seja usado no canal virtual L1 ao invés de um *buffer*; e (*ii*) o controle de fluxo no canal virtual L1 seja simplificado, não requerendo nem *handshake* nem controle baseado em créditos. Algumas redes, como a Æthereal, armazenam em uma tabela de dados a largura de banda requerida pelos fluxos GT. Esta tabela aumenta significativamente a área do roteador. Utilizando essa tabela, a rede Æthereal permite que múltiplos fluxos GT utilizem o mesmo canal físico, multiplexando a largura de banda via TDM. Na Hermes-CS, somente uma conexão pode ser estabelecida por canal físico, não requerendo essa área adicional.

A Figura 42 ilustra o protocolo usado para transmitir dados dos fluxos GT. Os passos deste protocolo são descritos abaixo, onde a numeração tem correspondência na Figura 42.

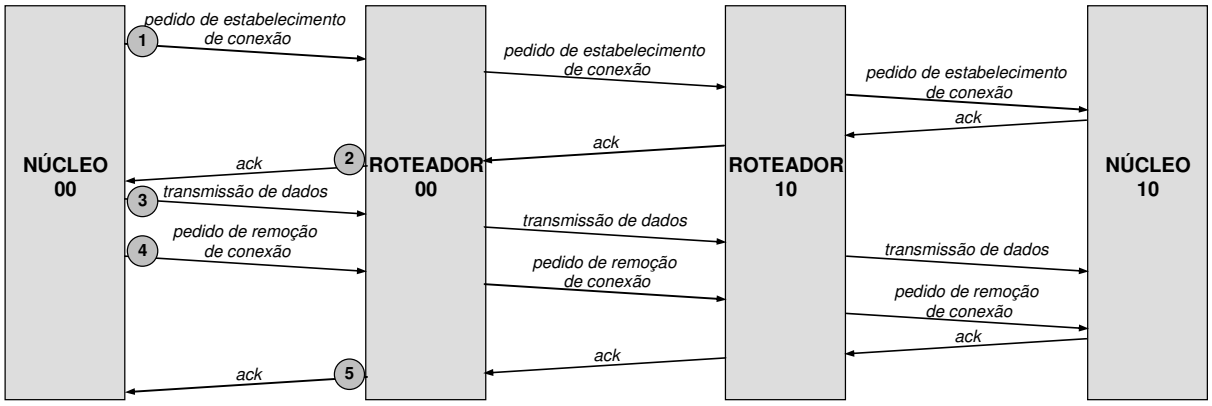


Figura 42 – Protocolo de envio de dados de fluxos GT.

1. O núcleo origem envia um pacote BE de controle (através do canal virtual L2) ao núcleo destino, solicitando a criação de uma conexão.

2. O sinal *ack* é ativado quando o pacote de controle chega ao núcleo destino. O sinal *ack* é propagado até o núcleo origem em *r* ciclos de relógio, onde *r* é o número de roteadores no caminho da conexão (destino e origem inclusive).
3. Quando o sinal *ack* chega ao núcleo origem, a conexão é efetivamente estabelecida e os dados podem ser transmitidos através do canal virtual L1.
4. Quando a origem não tem mais dados para transmitir, ela envia um pacote BE de controle (através do canal virtual L2) solicitando a remoção da conexão.
5. Quando o pacote de controle chega ao núcleo destino, o sinal *ack* é desativado, sendo esta informação propagada até a origem em *r* ciclos de relógio. Quando o sinal *ack* desativado chega ao núcleo origem, a conexão é efetivamente removida.

Em resumo, conexões são estabelecidas ou removidas usando pacotes BE de controle. Estes pacotes são diferenciados dos pacotes BE de dados pelo bit mais significativo do primeiro *flit* de cabeçalho. Quando este bit é '1', o pacote BE tem função de controle e o segundo *flit* do pacote informa o comando a ser executado (atualmente, dois pacotes de controle são definidos, estabelecimento e remoção da conexão). Pacotes BE de controle não possuem corpo. A Figura 43 apresenta um pacote BE de controle que solicita o estabelecimento de uma conexão entre um núcleo qualquer conectado à rede Hermes-CS e o núcleo destino 22.

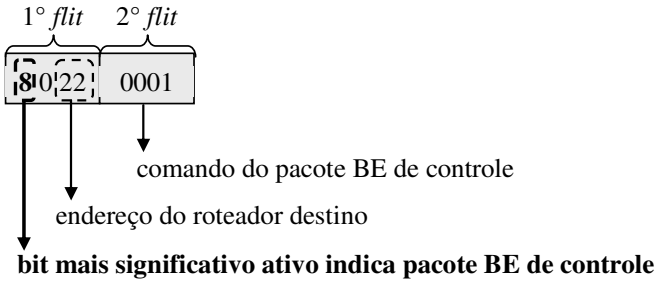


Figura 43 – Campos do pacote BE de controle.

A implementação atual de chaveamento por circuito na rede Hermes pode ser aprimorada pela adição de um mecanismo de prazo final para o estabelecimento de uma conexão. Este mecanismo pode evitar a reserva de um caminho por um longo período quando alguma porção da rede apresenta congestionamento. Mesmo que fluxos BE possam usar um caminho reservado sem uso, como relatado acima, a reserva de caminhos (sem efetivo estabelecimento da conexão) pode evitar o estabelecimento de outras conexões.

5.2 Validação Funcional da Rede Hermes-CS

A rede Hermes-CS foi descrita em VHDL e validada por simulação funcional. A Figura 44 mostra a simulação funcional do estabelecimento de uma conexão entre os núcleos 00 e 20. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 44.

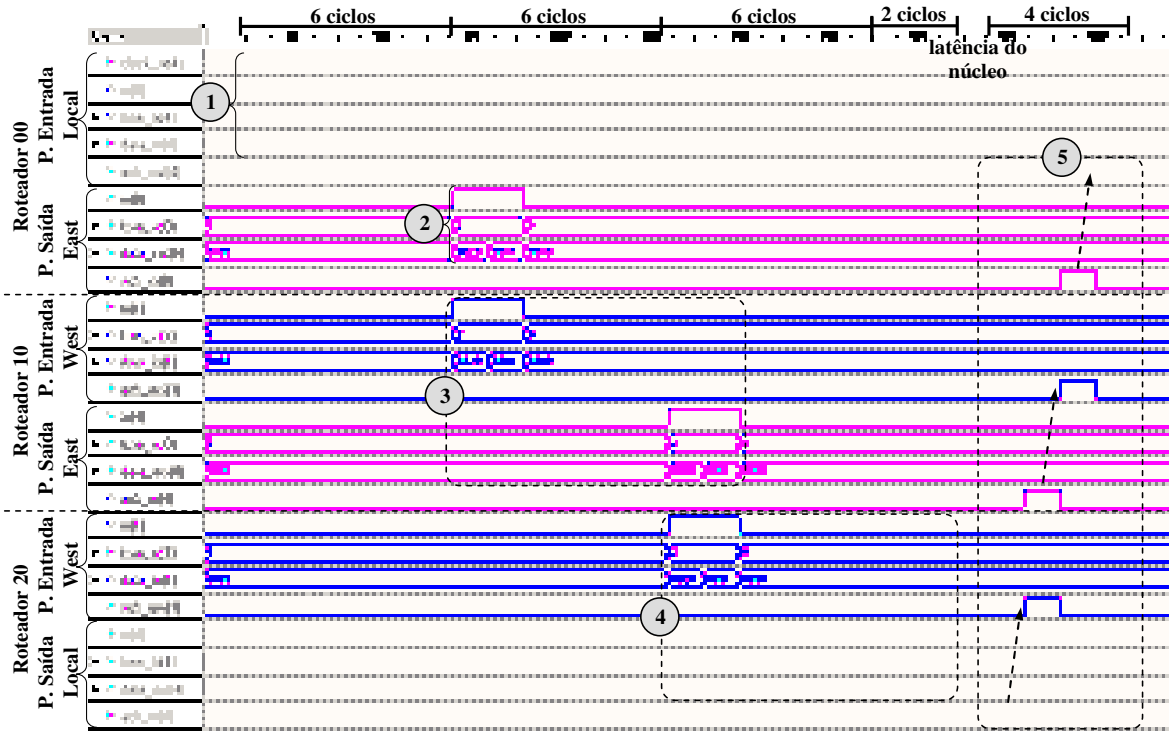


Figura 44 – Simulação do estabelecimento de uma conexão entre os núcleos 00 e 20.

1. O roteador 00 recebe um pacote pelo canal virtual L2 da porta Local (índice 4). O bit mais significativo do primeiro *flit* (8020) está ativo, indicando pacote BE de controle. Conseqüentemente, o pacote contém os campos ilustrados na Figura 43: a parte baixa do primeiro *flit* indica o endereço do roteador destino (20) e o segundo *flit* indica o comando (01 - estabelecimento de uma conexão).
2. O roteador 00 realiza o roteamento, estabelece a conexão entre o canal virtual L2 da porta de entrada Local e o canal virtual L2 da porta de saída East (índice 0) e então transmite o pacote.
3. O roteador 10 recebe o pacote pelo canal virtual L2 da porta West (índice 1), realiza o roteamento, estabelece a conexão entre o canal virtual L2 da porta de entrada West e o canal virtual L2 da porta de saída East (índice 0) e então transmite o pacote.
4. O roteador 20 recebe o pacote pelo canal virtual L2 da porta West (índice 1), realiza o roteamento, estabelece a conexão entre o canal virtual L2 da porta de entrada West e o canal virtual L2 da porta de saída Local (índice 4) e então transmite o pacote.
5. O núcleo destino recebe o pedido de estabelecimento de conexão e então ativa o sinal *ack*, informando que a conexão foi estabelecida com sucesso. O sinal *ack* é propagado até a origem, consumindo um ciclo de relógio em cada roteador no caminho da conexão. Quando o sinal *ack* chega ao núcleo origem, o núcleo pode começar a enviar os dados.

A latência ideal em ciclos de relógio para estabelecer e remover uma conexão é dada pela Equação 6. Após o estabelecimento da conexão, a latência para transmitir dados QoS do núcleo origem ao núcleo destino é igual a *n*, ou seja, um ciclo de relógio por roteador no caminho da conexão.

$$latência\ \ ideal = 6n + 2 + LA + (n+1) \quad (6)$$

onde: 6 é o número de ciclos de relógio requerido pelo algoritmo de arbitragem/roteamento para cada roteador, *n* é o número de roteadores no caminho da comunicação (origem e destino inclusive), 2 é o tamanho do pacote BE de controle, *LA* é a latência para o núcleo destino enviar o *ack* e (*n* + 1) representa o número de ciclos de relógio necessários para a propagação do *ack* do núcleo destino em direção ao núcleo origem.

Na simulação apresentada na Figura 44, observa-se que o núcleo destino ativa o sinal *ack* um ciclo de relógio após o recebimento do pacote (*LA* = 1). No entanto, este tempo pode variar de acordo com a implementação da interface de rede que conecta o núcleo à rede.

O envio de dados pela conexão estabelecida entre os núcleos 00 e 20 é apresentado na Figura 45. Esta Figura ilustra a continuação da simulação apresentada na Figura 44.

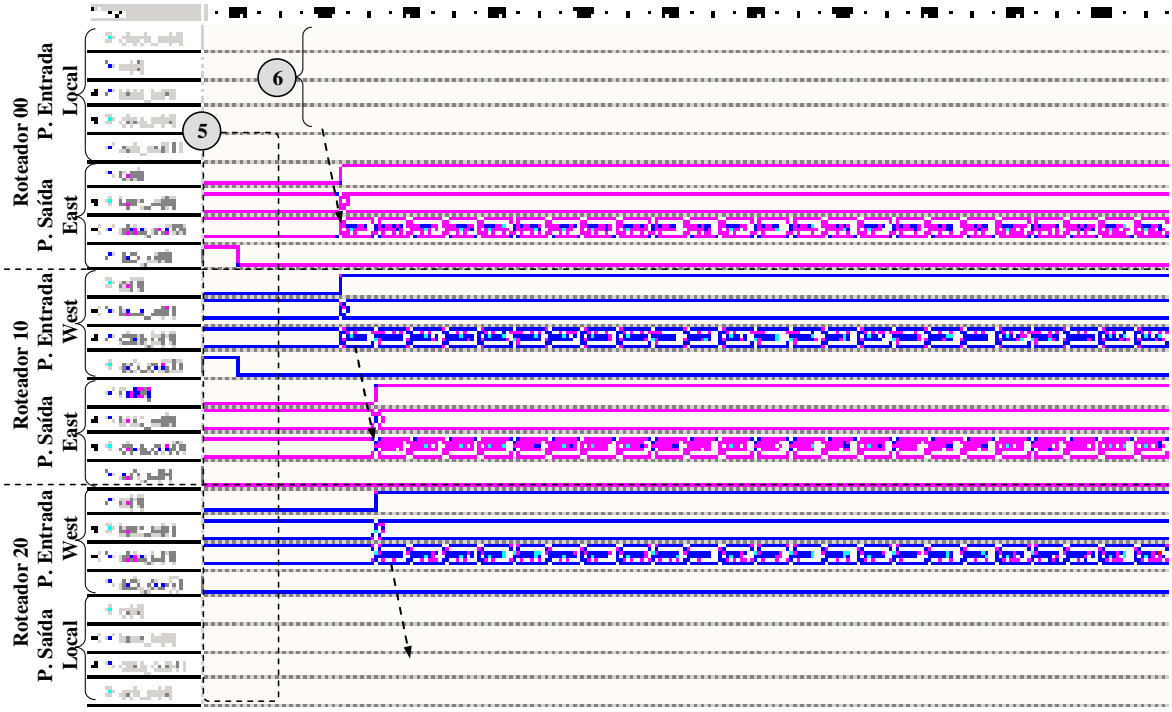


Figura 45 – Simulação do envio de dados pela conexão estabelecida entre os núcleos 00 e 20.

6. O roteador 00 recebe dados pelo canal virtual L1 da porta Local (índice 4). O canal virtual L1 é usado para transmitir dados com requisitos de QoS. Todos os dados transmitidos por este canal virtual são dados úteis, ou seja, estes dados não possuem cabeçalho nem terminador. O caminho dos dados é definido durante a fase de estabelecimento da conexão.

Depois do envio de todos os dados QoS, o núcleo origem deve enviar um outro pacote BE de controle, removendo a conexão. Este procedimento é semelhante ao apresentado na Figura 44, à exceção do segundo *flit* do pacote que deve conter o comando 02 (remoção da conexão).

A simulação apresentada na Figura 46 mostra um exemplo de compartilhamento da largura de banda entre dados de uma conexão QoS e um pacote BE. Os dados da conexão QoS são

transmitidos em rajadas de 5 *flits* com taxa de inserção de 50%. O pacote BE é composto por nove *flits* (dois *flits* de cabeçalho e sete *flits* de dados úteis) transmitidos em uma única rajada. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 46.

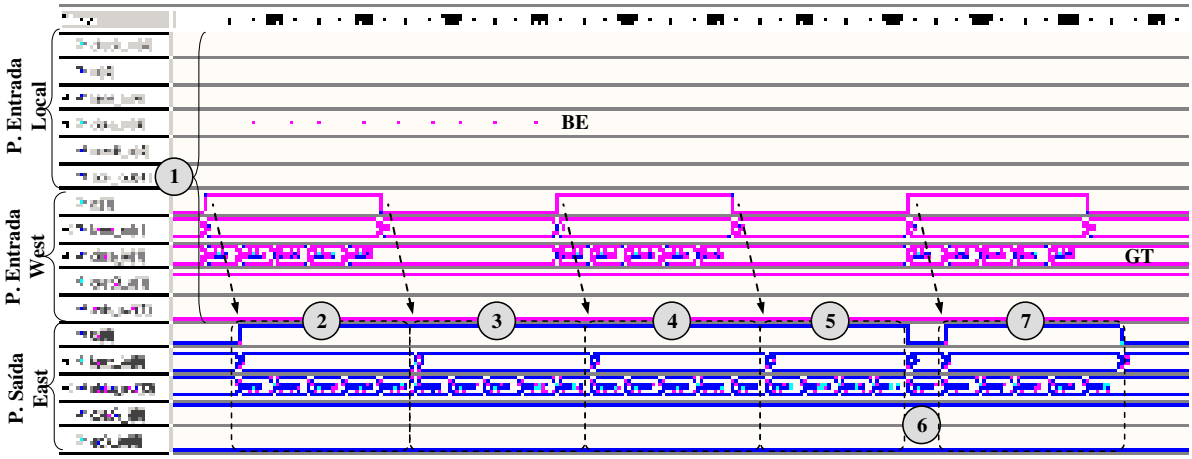


Figura 46 – Simulação do compartilhamento da largura de banda entre dados QoS e um pacote BE.

1. O roteador recebe simultaneamente um pacote BE pelo canal virtual L1 da porta Local (índice 4) e dados QoS pelo canal virtual L2 da porta West (índice 1), ambos destinados a porta de saída East (índice 0).
2. Como os dados QoS têm prioridade maior em relação ao pacote BE, eles são transmitidos pela porta de saída East usando toda a largura de banda do canal. É importante observar que um dado QoS possui latência de um ciclo de relógio entre a recepção pela porta de entrada e a subseqüente transmissão pela porta de saída.
3. Quando a conexão QoS não possui dados para transmitir, os *flits* do pacote BE são transmitidos.
4. A conexão QoS possui dados para transmitir, conseqüentemente aloca toda a largura de banda da porta de saída East.
5. A conexão QoS não possui dados para transmitir, então o restante dos *flits* do pacote BE são transmitidos.
6. O pacote BE terminou de transmitir os seus *flits* e a conexão QoS não possui dados para transmitir, conseqüentemente a largura de banda da porta de saída East fica ociosa.
7. A conexão QoS possui novamente dados para transmitir, então aloca toda a largura de banda da porta de saída East.

A latência ideal para transmitir pacotes BE na rede Hermes-CS é a mesma da rede Hermes-VC (Equação 5, página 75).

6 ESCALONAMENTO BASEADO EM PRIORIDADES SOBRE A REDE HERMES

Este Capítulo apresenta a terceira contribuição deste trabalho, a inserção de mecanismos de alocação de recursos baseados em prioridades na rede Hermes. Estes mecanismos de alocação permitem a diferenciação dos fluxos de acordo com suas necessidades de desempenho. Dois mecanismos baseados em prioridades são implementados: (*i*) mecanismo baseado em prioridades *fixas* e (*ii*) mecanismo baseado em prioridades *dinâmicas*. O projeto da rede Hermes com mecanismo baseado em prioridades fixas (Hermes-FP) é descrito na Seção 6.1 e a validação funcional do roteador desta rede é apresentada na Seção 6.2. A Seção 6.3 descreve o projeto da rede Hermes com o mecanismo baseado em prioridades dinâmicas (Hermes-DP) e a validação do roteador desta rede é apresentada na Seção 6.4.

6.1 Hermes-FP

O objetivo deste projeto é adicionar à rede Hermes a possibilidade de prover serviços diferenciados aos fluxos, utilizando um mecanismo de alocação de recursos baseado em prioridades fixas, semelhante à QNoC [BOL03]. Este mecanismo modifica a política de arbitragem e o escalonamento do roteador, sem modificar a interface do roteador Hermes-VC (Figura 32). A Figura 47 ilustra a arquitetura do roteador Hermes-FP, salientando as modificações realizadas em relação à arquitetura do roteador Hermes-VC (Figura 33).

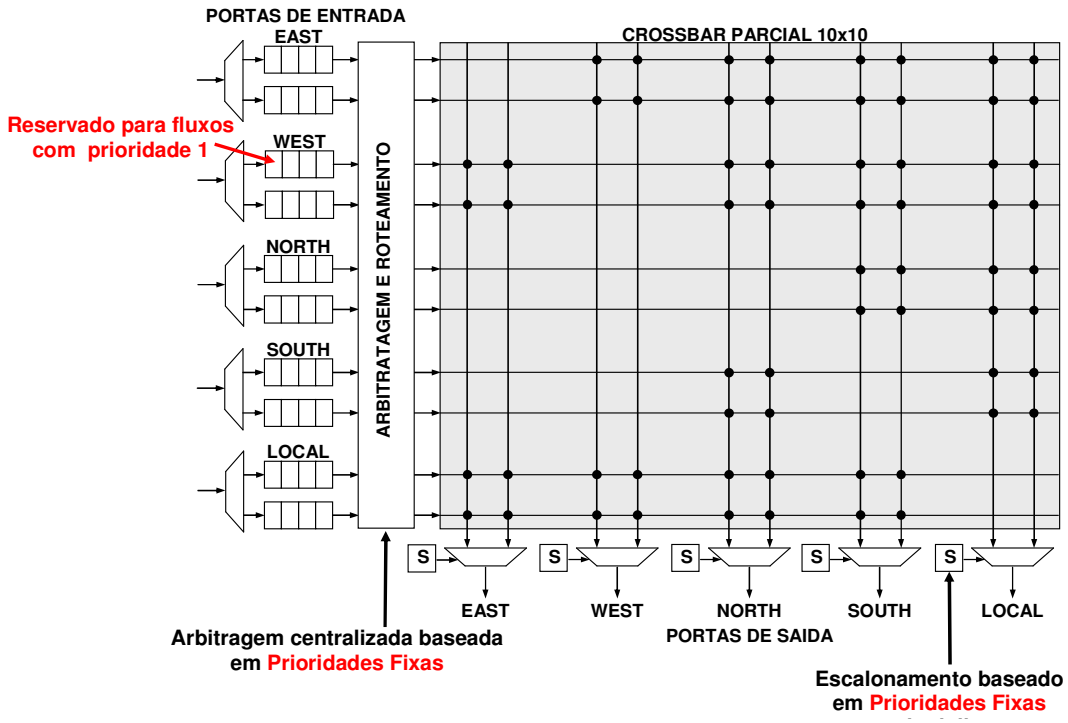


Figura 47 – Arquitetura do roteador Hermes-FP com dois canais virtuais.

No projeto Hermes-FP, cada canal virtual é associado a uma prioridade estática e é servido conforme esta prioridade. A prioridade de cada canal virtual é dada por seu índice, como definido pela Equação 7. Deste modo, esta rede é capaz de diferenciar até *n* fluxos, onde *n* é o número de canais virtuais por canal físico.

$$prioridade\ de\ L_i = i - 1 \tag{7}$$

Para diferenciar fluxos, um novo campo, denominado *priority*, é inserido na parte alta do primeiro *flit* do pacote, conforme ilustrado na Figura 48. Este campo determina qual canal virtual é usado para transmissão do pacote. Por exemplo, o canal virtual L2 transmite pacotes com prioridade 1. O usuário deve atribuir ao campo *priority* um valor entre zero e (*n*-1), sendo zero atribuído aos pacotes de menor prioridade e (*n*-1) aos de maior prioridade. Somente o roteador origem verifica o campo *priority*. Os demais roteadores transmitem pacotes usando o mesmo canal virtual alocado pelo roteador origem.

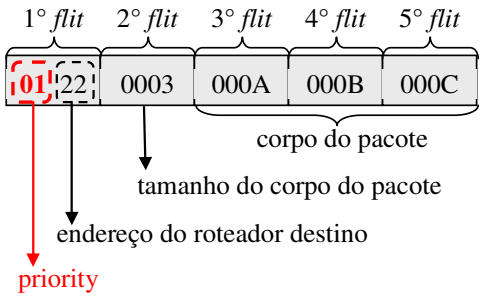


Figura 48 – Campos dos pacotes transmitidos pela rede Hermes-FP.

Como mencionado anteriormente, a atribuição de prioridades fixas aos canais virtuais requer modificações na política de arbitragem e escalonamento do roteador. Em ambos, o pacote associado ao canal virtual com prioridade maior é servido primeiro, mesmo se os outros pacotes estão esperando há mais tempo. Logo, a transmissão dos dados dos canais virtuais com prioridade menor depende da carga dos canais virtuais com prioridade maior, a qual pode variar dinamicamente. Por esta razão, os limites de latência fim-a-fim não podem ser determinados para todos os pacotes, apenas para os pacotes transmitidos pelo canal virtual com prioridade mais alta. Quando pacotes com a mesma prioridade competem por recursos, o mecanismo de prioridades fixas não provê garantias rígidas a nenhum dos pacotes. Conseqüentemente, é difícil dar suporte a serviços múltiplos com QoS garantida [GIR98].

A Figura 49 ilustra a alocação de banda de um canal físico utilizando o escalonamento baseado em prioridades fixas. O canal físico é compartilhado por dois canais virtuais onde um transmite o pacote A com prioridade 1 e o outro o pacote B com prioridade 0. Três situações de alocação são apresentadas na Figura. A primeira situação corresponde à alocação de toda a largura de banda para o pacote A porque este tem prioridade maior do que o pacote B. Na segunda situação, o pacote B utiliza toda a largura de banda porque o pacote A não possui créditos para transmissão. Na terceira situação, a largura de banda não é utilizada porque nenhum dos pacotes possui créditos.

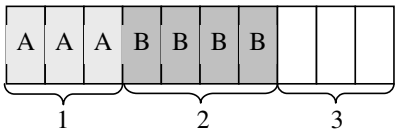


Figura 49 – Alocação da largura de banda de um canal físico utilizando escalonamento baseado em *prioridades fixas*.

Arbitragem e escalonamento baseado em prioridades fixas são eficientes para um número pequeno de canais virtuais [GIR98]. Por exemplo, é possível reservar um canal virtual para fluxos de tempo real, um segundo para fluxos de tempo não real com taxa de perda garantida, e um terceiro para fluxos melhor esforço. A desvantagem deste método é o fato da área do roteador aumentar aproximadamente com o quadrado do número de canais virtuais [MEL05].

6.2 Validação Funcional do Roteador Hermes-FP

O roteador Hermes-FP foi descrito em VHDL e validado por simulação funcional. A Figura 50 mostra a simulação do roteador Hermes-FP recebendo dois pacotes simultaneamente. Um pacote (QoS) é recebido pelo canal virtual L2 da porta Local e outro pacote (BE) é recebido pelo canal virtual L1 da porta West. Os dois pacotes são transmitidos pela porta de saída East. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 50.

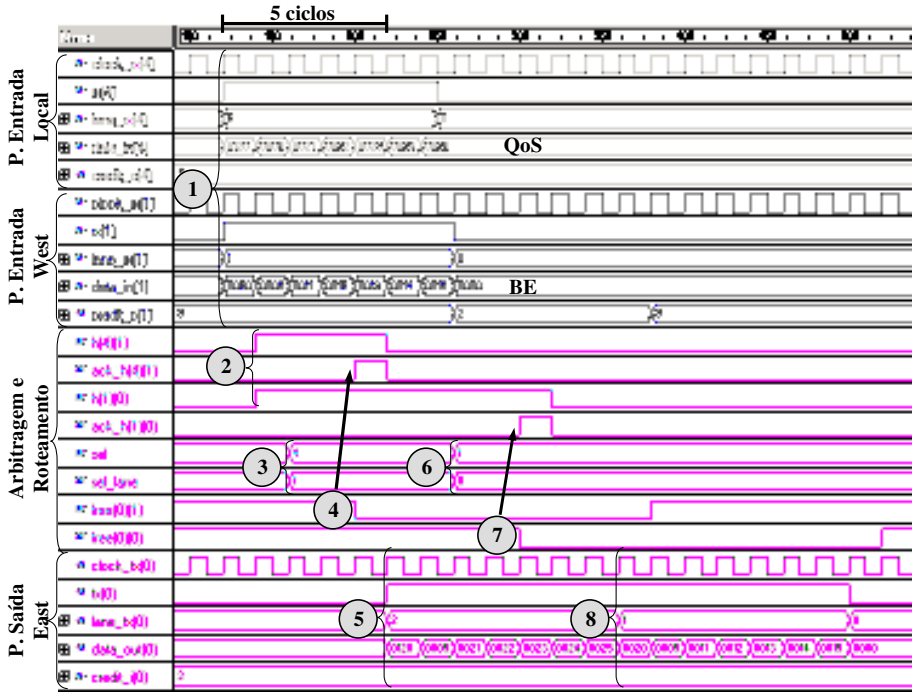


Figura 50 – Simulação do roteador Hermes-FP recebendo dois pacotes com prioridades diferentes. Prioridade 1 corresponde a pacote QoS. Prioridade 0 corresponde a pacote BE.

1. O roteador recebe simultaneamente um pacote com prioridade 1 (QoS) pelo canal virtual L2 da porta Local (índice 4) e um pacote com prioridade 0 (BE) pelo canal virtual L1 da porta West (índice 1). A prioridade de cada pacote pode ser observada na parte alta do primeiro *flit*.
2. Ambos requisitam roteamento, ativando os sinais *h*(4)(1) e *h*(1)(0).

3. A arbitragem é executada e o pacote recebido pelo canal virtual L2 (*sel_lane* = 1) da porta de entrada Local (*sel* = 4) recebe permissão de roteamento, porque o canal virtual L2 tem prioridade maior do que o outro canal requisitando roteamento.
4. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(1) = 0), e o sinal *ack_h*(4)(1) é ativado, indicando que a conexão entre o canal virtual L2 da porta de entrada Local e o canal virtual L2 da porta de saída East foi estabelecida.
5. O roteador começa a transmitir o pacote oriundo do canal virtual L2 da porta de entrada Local pelo canal virtual L2 da porta de saída East (índice 0). É importante lembrar que na Hermes-FP os pacotes são transmitidos usando o mesmo canal virtual utilizado na recepção dos mesmos.
6. A arbitragem é executada novamente e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada West (*sel* = 1) recebe permissão de roteamento.
7. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(0) = 0), e o sinal *ack_h*(1)(0) é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta de saída East foi estabelecida.
8. O último *flit* do pacote QoS é transmitido pelo canal virtual L2 da porta de saída East. Imediatamente após, o pacote BE começa a ser transmitido pelo canal virtual L1 desta porta.

Na simulação apresentada na Figura 50, observa-se que a conexão entre a porta de entrada West e a porta de saída East é estabelecida em 300ns, semelhante ao que ocorre na simulação do roteador Hermes-VC, ilustrada na Figura 39. No entanto, aqui a largura de banda do canal físico não é compartilhada pelos dois canais virtuais, porque o canal virtual L2 tem prioridade maior.

A simulação funcional quando dois pacotes com mesma prioridade competem pelos mesmos recursos é apresentada na Figura 51. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 51.

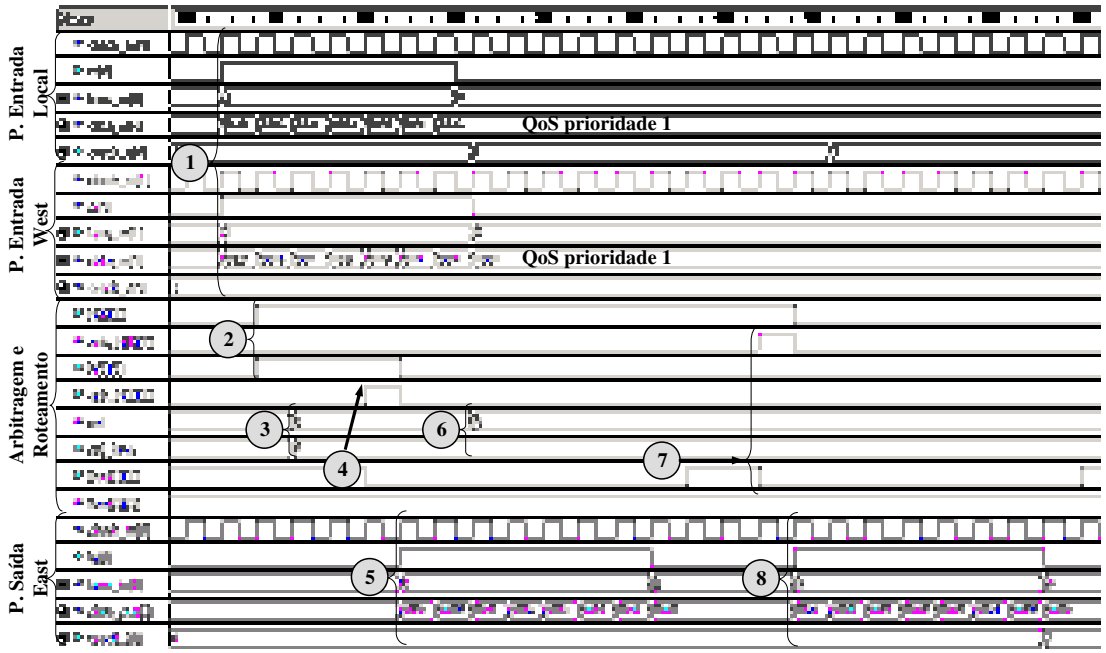


Figura 51 – Simulação do roteador Hermes-FP recebendo dois pacotes com a mesma prioridade.

1. O roteador recebe simultaneamente dois pacotes com mesma prioridade (*priority* = 1) pelas portas de entrada Local (índice 4) e West (índice 1).
2. Ambos requisitam roteamento ativando o sinal *h*.
3. Como os dois pacotes requisitando roteamento possuem a mesma prioridade, a arbitragem *Round-robin* é usada para determinar qual terá acesso ao roteamento. Neste caso, a requisição do canal virtual L2 (*sel_lane* = 1) da porta de entrada West (*sel* = 1) é atendida.
4. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(1) = 0), e o sinal *ack_h*(1)(1) é ativado, indicando que a conexão entre o canal virtual L2 da porta de entrada West e o canal virtual L2 da porta de saída East foi estabelecida.
5. O roteador começa a transmitir o pacote oriundo do canal virtual L2 da porta de entrada West pelo canal virtual L2 da porta de saída East (índice 0).
6. A arbitragem é executada novamente, e o pacote recebido pelo canal virtual L2 (*sel_lane* = 1) da porta de entrada Local (*sel* = 4) recebe permissão de roteamento. O algoritmo de roteamento é executado. No entanto, o canal virtual L2 da porta de saída East está ocupado. Conseqüentemente, a conexão não pode ser estabelecida.
7. A solicitação de roteamento permanece ativa até que o canal virtual L2 seja liberado. Quando isto ocorre (*free*(0)(1) = 1), o roteamento é executado com sucesso, a tabela de chaveamento é atualizada (*free*(0)(1) = 0) e o sinal *ack_h*(4)(1) é ativado, indicando que a conexão entre o canal virtual L2 da porta de entrada Local e o canal virtual L2 da porta de saída East foi estabelecida.
8. O roteador começa a transmitir o pacote oriundo do canal virtual L2 da porta de entrada Local pelo canal virtual L2 da porta de saída East.

Na rede Hermes-FP, a latência ideal em ciclos de relógio para transferir um pacote (BE ou QoS) da origem para o destino é a mesma da rede Hermes-VC (veja na Equação 5, página 75).

6.3 Hermes-DP

Este projeto é semelhante à Hermes-FP, porém um mecanismo de alocação de recursos baseado em prioridades *dinâmicas* é utilizado para prover serviços diferenciados aos fluxos. Este mecanismo, como o anterior, não necessita alterações na interface do roteador (Figura 32).

No projeto Hermes-DP, a prioridade de cada canal virtual varia conforme a prioridade do pacote que o mesmo está transmitindo. Isto permite: (*i*) a transmissão de pacotes por qualquer canal virtual livre; (*ii*) a transmissão de um pacote por canais virtuais diferentes em roteadores diferentes; e (*iii*) a transmissão de pacotes com mesma prioridade por canais virtuais diferentes em um mesmo roteador.

Neste projeto, o campo *priority* também é incluído no cabeçalho do pacote, conforme ilustrado na Figura 48. No entanto, o usuário pode atribuir a este campo um valor entre zero e (2^t-1), onde *t* é a largura em bits de um *flit*. Conseqüentemente, a Hermes-DP pode diferenciar um número

maior de fluxos (2^t) do que a Hermes-FP.

A alocação de recursos baseada em prioridades dinâmicas requer modificações na política de arbitragem, no roteamento e no escalonamento do roteador, conforme ilustrado na Figura 52.

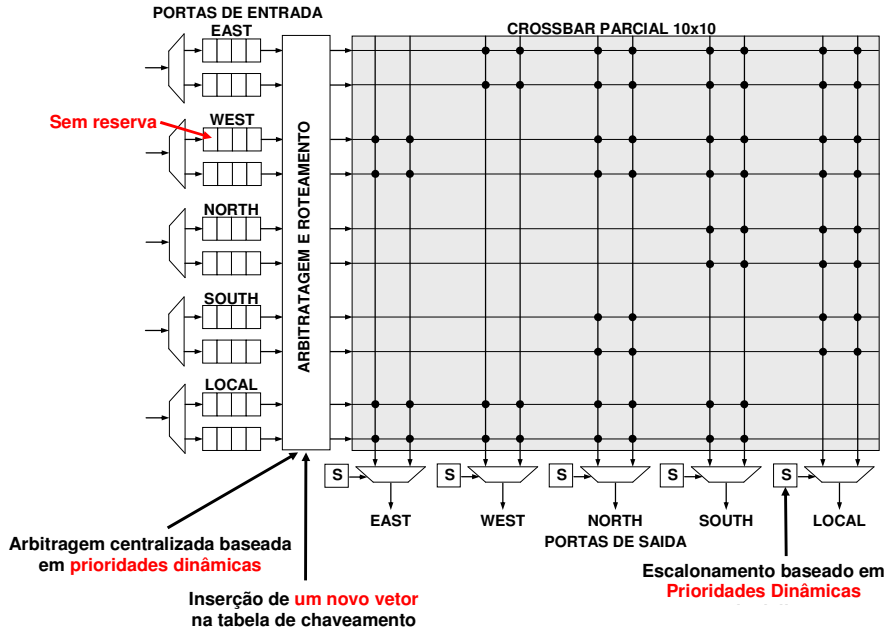


Figura 52 – Arquitetura do roteador Hermes-DP com dois canais virtuais.

A política de arbitragem é a mesma adotada pela Hermes-FP, ou seja, o pacote com prioridade maior é servido primeiro. Em caso de empate, uma política de arbitragem round-robin é usada. No entanto, como neste projeto a prioridade está associada somente ao pacote, é necessário um circuito mais complexo para determinar qual pacote tem prioridade maior e, portanto, terá acesso ao roteamento. Este circuito utiliza comparadores em cadeia conforme ilustrado na Figura 53. A inclusão deste circuito aumenta em dois ciclos de relógio o tempo consumido pela arbitragem. Entretanto, o número de ciclos excedentes depende do número de canais virtuais. Quanto maior o número de canais virtuais, maior o número de níveis da cadeia de comparadores e por conseqüência maior o número de ciclos consumidos.

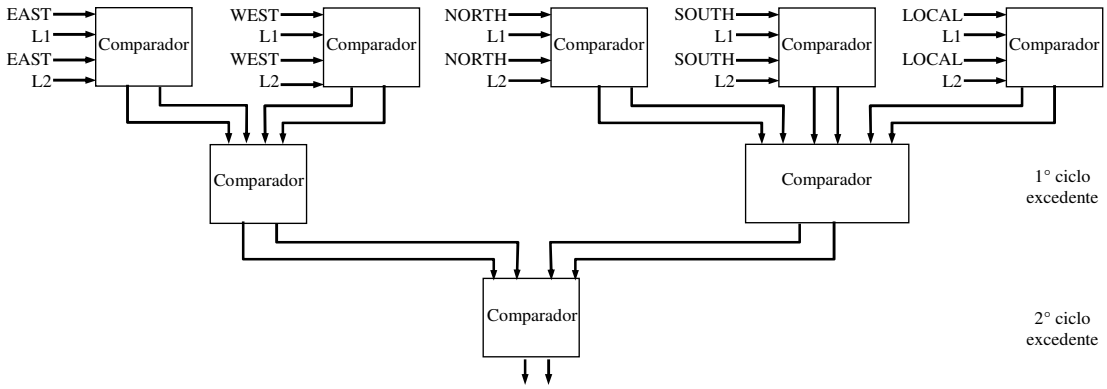
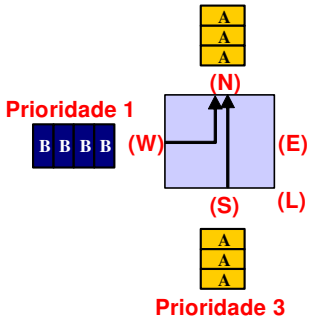


Figura 53 – Circuito com comparadores em cadeia que decide qual a porta e canal virtual possui o pacote com maior prioridade.

No roteamento, a tabela de chaveamento é estendida por um novo vetor, denominado *priority*. O vetor *priority* informa a prioridade dos canais virtuais de saída. Durante o roteamento, a posição do vetor *priority* correspondente à porta e canal virtual de saída selecionados é preenchida com o valor do campo *priority* do pacote que está sendo roteado. A tabela de chaveamento apresentada na Figura 54(b) representa os chaveamentos ilustrados na Figura 54(a). Considere a porta North. O canal virtual de saída L1 está ocupado (*free* = 0) e conectado ao canal virtual de entrada L1 da porta West (*out* = 2). O canal virtual de saída L2 está ocupado (*free* = 0) e conectado ao canal virtual de entrada L1 da porta South (*out* = 6). A posição do vetor *priority* correspondente ao canal virtual de saída L1 é preenchida com a prioridade do pacote B (1), enquanto que a posição correspondente ao canal virtual de saída L2 é preenchida com a prioridade do pacote A (3).



(a)

Porta	E		W		N		S		L	
VC	0 – L1	1 – L2	2 – L1	3 – L2	4 – L1	5 – L2	6 – L1	7 – L2	8 – L1	9 – L2
Free	1	1	1	1	0	0	1	1	1	1
In			4				5			
Out					2	6				
Priority					1	3				

(b)

Figura 54 – (a) Chaveamento. (b) Tabela de chaveamento.

No escalonamento, o vetor *priority* é verificado para determinar qual canal virtual tem o pacote com maior prioridade. No exemplo apresentado na Figura 54, o pacote A tem maior prioridade e por esta razão usa toda a largura de banda do canal físico. Em caso de empate, uma das seguintes abordagens é adotada: (*i*) *round-robin*, onde os *x* pacotes com maior prioridade recebem 1/*x* da largura de banda do canal físico; ou (*ii*) um dos pacotes com maior prioridade recebe 100% da largura de banda do canal físico. A abordagem utilizada pelo escalonador é parametrizável e deve ser definida na fase de projeto da rede. O empate pode ocorrer porque pacotes com a mesma prioridade podem ser transmitidos por canais virtuais diferentes, o que não é possível na Hermes-FP.

A principal desvantagem deste projeto é a falta de reserva de recursos, o que permite que pacotes BE (prioridade 0) ocupem todos os canais virtuais, impedindo a transmissão de fluxos com prioridade maior. Esta desvantagem é ilustrada na Figura 55, onde o pacote C possui prioridade maior do que os pacotes A e B, porém não pode ser transmitido porque os pacotes A e B foram chaveados primeiro. Logo, o pacote C deve aguardar que um dos canais virtuais seja liberado para depois ser transmitido.

É importante ressaltar que o chaveamento por pacote *wormhole* dificulta o uso de preempção de pacotes. O método *wormhole* funciona como um *pipeline*, onde os *flits* de cabeçalho (contendo informações do destino) se movem pela rede e todos os *flits* de dados (*payload*) os seguem. Depois que a conexão entre o canal virtual de entrada e o canal virtual de saída é estabelecida, todos os flits do pacote devem ser transmitidos antes da conexão ser encerrada.

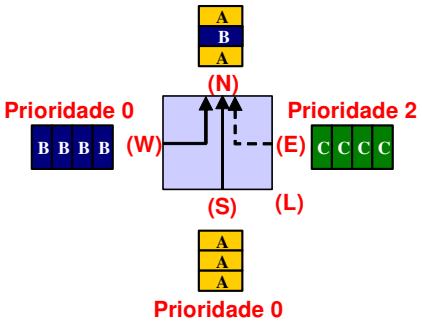


Figura 55 – Chaveamento em um roteador Hermes-DP com dois canais virtuais. A linha contínua identifica um chaveamento efetivado. A linha pontilhada identifica um chaveamento pendente.

6.4 Validação Funcional do Roteador Hermes-DP

O roteador Hermes-DP foi descrito em VHDL e validado por simulação funcional. A Figura 56 mostra a simulação funcional do roteador recebendo simultaneamente dois pacotes. Um pacote é recebido pelo canal virtual L1 da porta entrada Local e o outro é recebido pelo canal virtual L1 da porta de entrada West. Ambos têm como destino a porta de saída East. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 56.

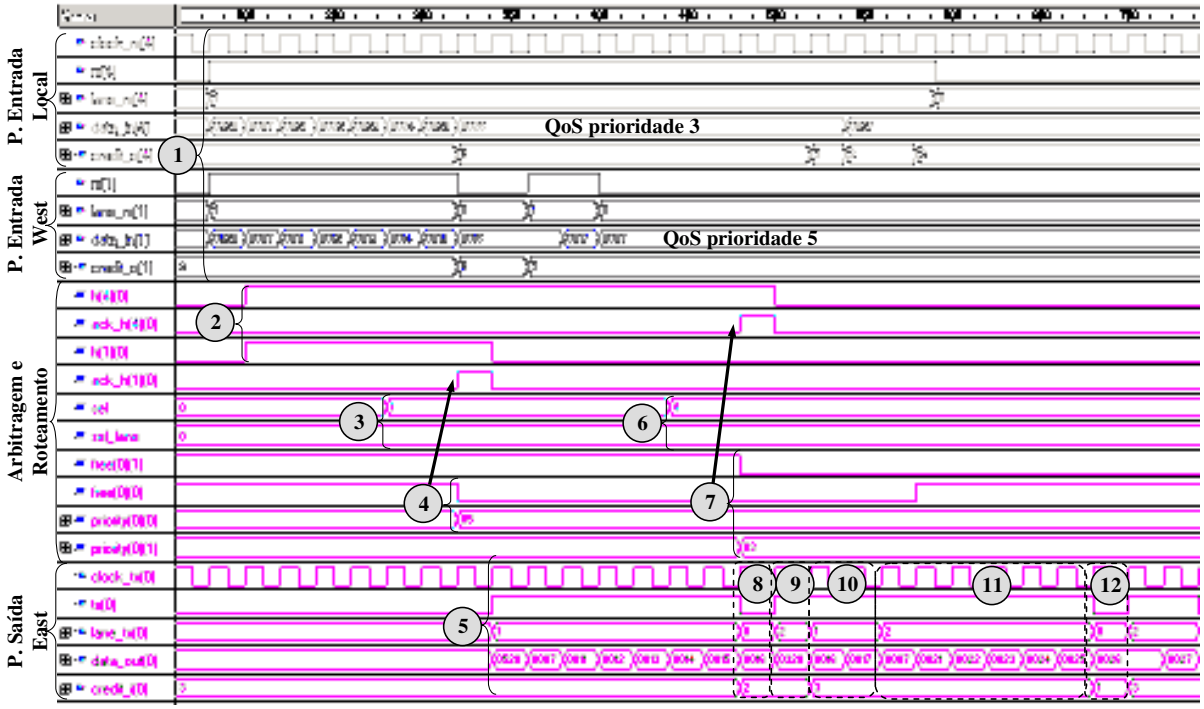


Figura 56 – Simulação do roteador Hermes-DP recebendo simultaneamente dois pacotes com prioridades diferentes.

1. O roteador recebe simultaneamente um pacote com prioridade 3 pelo canal virtual L1 da porta de entrada Local (índice 4) e um pacote com prioridade 5 pelo canal virtual L1 da porta de entrada West (índice 1). A prioridade de cada pacote pode ser observada na parte alta do primeiro *flit*.

2. Ambos requisitam roteamento ativando o sinal *h*.
3. A arbitragem é executada e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada West (*sel* = 1) recebe permissão de roteamento, porque o pacote associado a esta porta tem prioridade maior.
4. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(0) = 0), e o sinal *ack_h*(1)(0) é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta de saída East foi estabelecida. É importante lembrar que, durante o roteamento, a posição do vetor *priority* correspondente à porta e canal virtual de saída selecionados é preenchida com o valor do campo *priority* do pacote que está sendo roteado (*priority*(0)(0) = 5).
5. O roteador começa a transmitir o pacote oriundo do canal virtual L1 da porta de entrada West pelo canal virtual L1 da porta de saída East (índice 0).
6. A arbitragem é executada novamente e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada Local (*sel* = 4) recebe permissão de roteamento.
7. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(1) = 0), e o sinal *ack_h*(4)(0) é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada Local e o canal virtual L2 da porta de saída East foi estabelecida.
8. Quando o canal virtual selecionado possui créditos para recepção de dados (*credit_i* = 1), os *flits* são transmitidos em um ciclo de relógio. Caso contrário (*credit_i* = 0), o canal virtual libera o canal físico para ser utilizado por outro canal virtual. Se nenhum canal virtual pode usar o canal físico, o sinal *tx* é desativado. Aqui, o canal físico fica ocioso porque o canal virtual L1 não possui créditos e o canal virtual L2 não possui dados para transmitir. A falta de dados no canal virtual L2 deve-se ao fato que a conexão com este canal virtual é estabelecida no tempo 480ns e, portanto, somente no ciclo de relógio seguinte (500ns) esta conexão começará a transmitir dados.
9. O canal virtual L2 começa a transmitir seus dados, aproveitando que o canal virtual L1 (que possui prioridade maior) não possui créditos para transmissão.
10. O canal virtual L1 (com prioridade maior) volta a ter créditos e, por conseqüência, volta a ocupar toda a largura de banda do canal físico.
11. O canal virtual L1 termina de transmitir seus dados. Conseqüentemente, o canal virtual L2 passa novamente a transmitir seus dados.
12. O canal físico fica ocioso porque o canal virtual L1 não possui dados para transmitir e o canal virtual L2 não possui créditos para a transmissão.

A simulação funcional quando dois pacotes com mesma prioridade competem pelos mesmos recursos é apresentada na Figura 57. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 57. Nesta simulação, o escalonador adotada a abordagem *round-robin* em caso de empate entre pacotes com mesma prioridade.

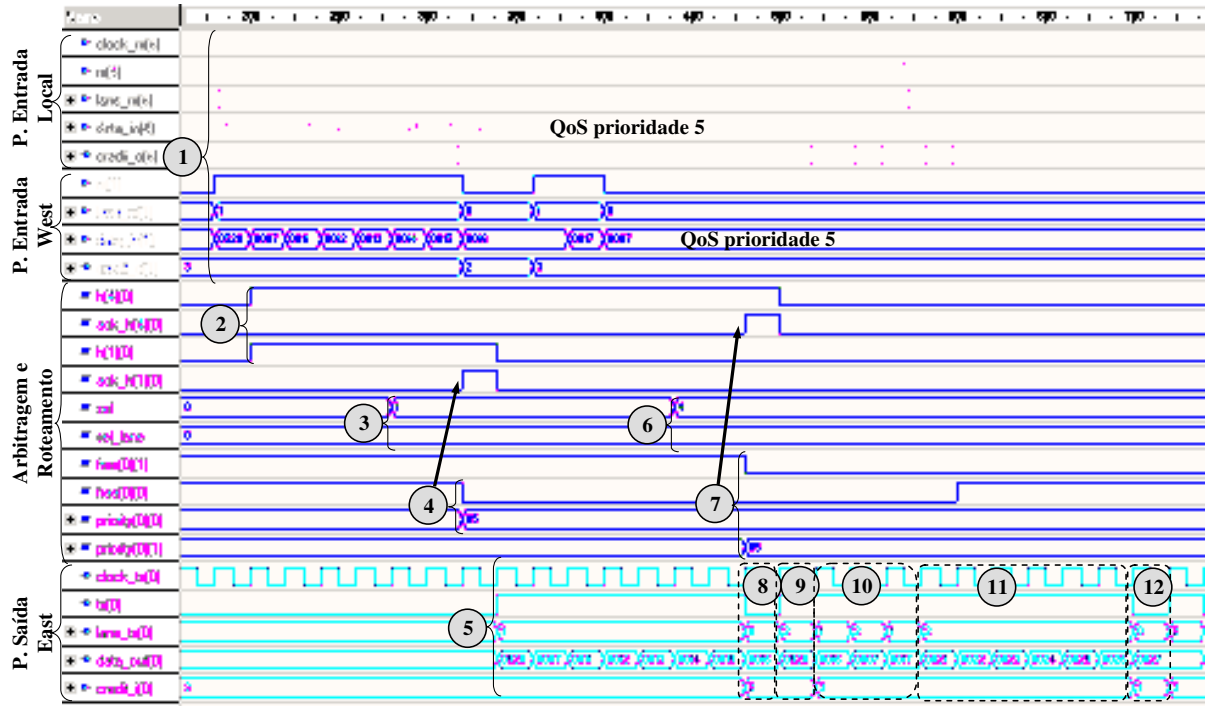


Figura 57 – Simulação do roteador Hermes-DP recebendo simultaneamente dois pacotes com a mesma prioridade.

1. O roteador recebe simultaneamente dois pacotes com prioridade 5 pelas portas de entrada Local (índice 4) e West (índice 1). A prioridade de cada pacote pode ser observada na parte alta do primeiro *flit*.
2. Ambos requisitam roteamento ativando o sinal *h*.
3. Como os dois pacotes requisitando roteamento possuem a mesma prioridade, a arbitragem *Round-robin* é usada para determinar qual terá acesso ao roteamento. Neste caso, a requisição do canal virtual L1 (*sel_lane* = 0) da porta de entrada West (*sel* = 1) é atendida.
4. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(0) = 0), e o sinal *ack_h*(1)(0) é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada West e o canal virtual L1 da porta de saída East foi estabelecida.
5. O roteador começa a transmitir o pacote oriundo do canal virtual L1 da porta de entrada West pelo canal virtual L1 da porta de saída East (índice 0).
6. A arbitragem é executada novamente e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada Local (*sel* = 4) recebe permissão de roteamento.
7. O algoritmo de roteamento é executado, a tabela de chaveamento é atualizada (*free*(0)(1) = 0), e o sinal *ack_h*(4)(0) é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada Local e o canal virtual L2 da porta de saída East foi estabelecida.
8. O canal físico fica ocioso porque o canal virtual L1 não possui créditos e o canal virtual L2 não possui dados para transmitir. O motivo da falta de dados no canal virtual L2 é o mesmo apresentado na simulação Figura 56, ou seja, a conexão com este canal virtual é estabelecida no tempo 480ns e somente no ciclo de relógio seguinte (500ns) ela começará a transmitir dados.

9. O canal virtual L2 começa a transmitir seus dados. Neste momento, o canal virtual L1 continua sem créditos para transmissão.
10. O canal físico começa a ser **compartilhado** pelos canais virtuais L1 e L2, que possuem a mesma prioridade.
11. O canal virtual L1 termina de transmitir seus dados. Conseqüentemente, o canal virtual L2 passa a ocupar toda a largura de banda do canal físico.
12. O canal físico fica ocioso porque o canal virtual L1 não possui dados para transmitir e o canal virtual L2 não possui créditos para a transmissão.

Na simulação apresentada na Figura 57, observa-se o compartilhamento da largura de banda do canal físico entre dois pacotes com a mesma prioridade. Este compartilhamento não é possível na rede Hermes-FP, porque esta não permite a transmissão de pacotes com mesma prioridade por canais virtuais diferentes.

A principal desvantagem do mecanismo baseado em prioridades dinâmicas, a falta de reserva de recursos, é observada na simulação funcional apresentada na Figura 58. Esta simulação funcional mostra um pacote com prioridade maior aguardando roteamento porque todos os canais virtuais estão ocupados. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 58.

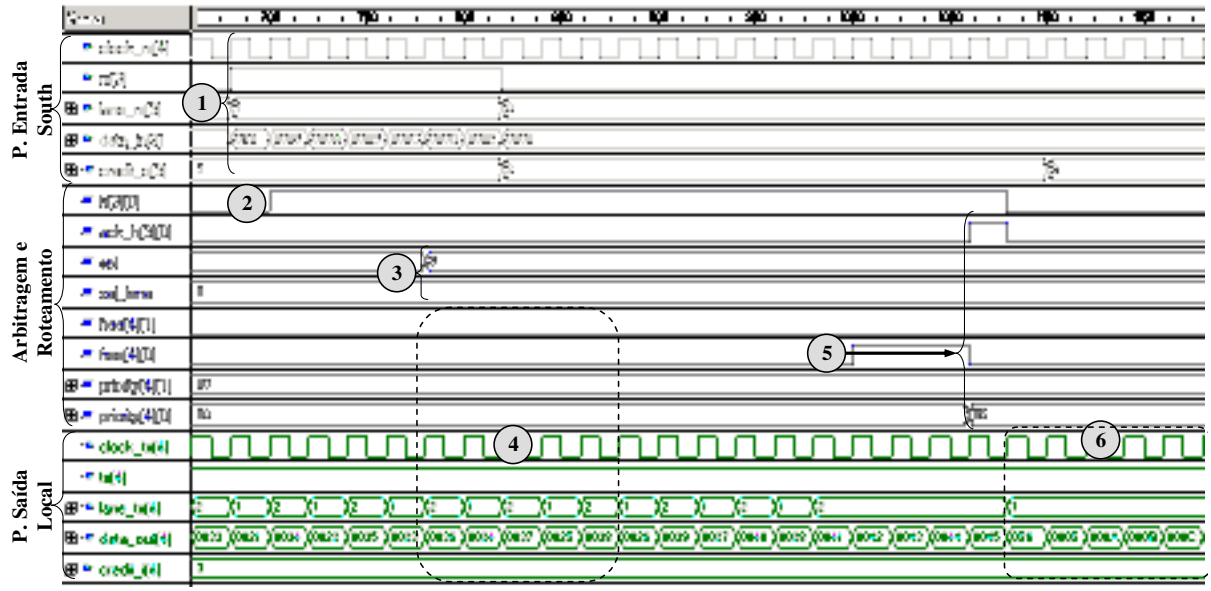


Figura 58 – Simulação da principal desvantagem do mecanismo baseado em prioridades dinâmicas - a falta de reserva de recursos.

1. O roteador recebe um pacote com prioridade 5 pelo canal virtual L1 da porta de entrada South (índice 3). A prioridade do pacote pode ser observada na parte alta do primeiro *flit*.
2. O canal virtual L1 da porta South requisita roteamento ativando o sinal *h* (*h*(3)(0)).
3. A arbitragem é executada e o pacote recebido pelo canal virtual L1 (*sel_lane* = 0) da porta de entrada South (*sel* = 3) recebe permissão de roteamento.

4. O algoritmo de roteamento é executado. No entanto, como todos os canais virtuais da porta de saída escolhida pelo roteamento (porta Local) estão ocupados (*free*(4)(0) = 0 e *free*(4)(1) = 0), a conexão não é estabelecida.
5. A solicitação de roteamento permanece ativa até que um canal virtual seja liberado. Quando isto ocorre (*free*(4)(0) = 1), o roteamento é executado com sucesso, a tabela de chaveamento é atualizada (*free*(4)(0) = 0) e o sinal *ack_h*(3)(0) é ativado, indicando que a conexão entre o canal virtual L1 da porta de entrada South e o canal virtual L1 da porta de saída Local foi estabelecida.
6. A largura de banda da porta de saída Local é totalmente alocada para o canal virtual L1, porque a prioridade do pacote associado a este canal é maior.

Na simulação apresentada na Figura 58, observa-se que a prioridade dos pacotes que estão ocupando os canais virtuais da porta de saída Local (*priority*(4)(0) = 0 e *priority*(4)(1) = 0) é menor do que a do pacote que está esperando o roteamento (*priority* = 5). Isto revela que: se todos os canais virtuais da porta de saída escolhida pelo roteamento estão ocupados, não importa a prioridade do pacote aguardando roteamento, ele deverá esperar até que um canal virtual seja liberado. Não existe reserva de recursos para pacotes com prioridades como na rede Hermes-FP.

Outra diferença da Hermes-DP em relação a Hermes-PF está no cálculo da latência ideal, ilustrada na Equação 8. Note que a rede Hermes-DP consome dois ciclos de relógio a mais para realizar o algoritmo de arbitragem/roteamento em cada roteador. Esse acréscimo refere-se à necessidade de um circuito mais complexo para determinar qual pacote possui prioridade maior.

$$latência\ ideal = 7n + P \tag{8}$$

onde: 7 é o número de ciclos de relógio requerido pelo algoritmo de arbitragem/roteamento para cada roteador, n é o número de roteadores no caminho da comunicação (origem e destino inclusive), P é o tamanho do pacote em *flits*.

7 ESCALONAMENTO BASEADO EM TAXAS SOBRE A REDE HERMES

Neste Capítulo é proposto o projeto de rede que utiliza mecanismos de alocação de recursos baseados em taxas, denominado Hermes-RB. Este mecanismo define a prioridade de cada fluxo dinamicamente, de acordo com a taxa requerida e taxa usada pelo mesmo. A Seção 7.1 descreve o projeto da rede Hermes com escalonamento baseado em taxas (Hermes-RB). A validação funcional da rede Hermes-RB é apresentada na Seção 7.2.

7.1 Hermes-RB

Redes de telecomunicação têm adotado políticas de escalonamento baseadas em taxas para controle de congestionamento. Exemplos de tais políticas são: *Virtual Clock* (Seção 2.3.3.2.1, página 45), *Weighted Fair Queuing* (Seção 2.3.3.2.1, página 45) e o método proposto em [LEE95].

A política de escalonamento baseada em taxas proposta aqui assume as seguintes características de rede intra-chip: chaveamento por pacote *wormhole*, roteamento determinístico, e canais físicos multiplexados em no mínimo dois canais virtuais. Fluxos BE são transmitidos usando somente um canal virtual, enquanto fluxos QoS podem usar qualquer canal virtual. Esta reserva de recursos para fluxos QoS é necessária para evitar que múltiplos fluxos BE impeçam que um fluxo QoS use o canal.

A Figura 59 ilustra a interface entre os roteadores deste projeto. A interface entre os roteadores é semelhante à interface da Hermes-CS, diferenciando-se pelos sinais *credit* e *ack*, os quais servem a todos os canais virtuais. Os seguintes sinais compõem a porta de saída: (1) *clock_tx*: sincroniza a transmissão de dados; (2) *tx*: indica dado disponível; (3) *lane_tx*: indica o canal virtual transmitindo o dado; (4) *data_out*: dado a ser enviado; (5) *credit_in*: informa disponibilidade de espaço no *buffer* do roteador vizinho, para cada canal virtual; (6) *ack_out*: indica a criação ou remoção de uma conexão virtual; (7) *nack_out*: indica a impossibilidade de criar ou remover uma conexão virtual.

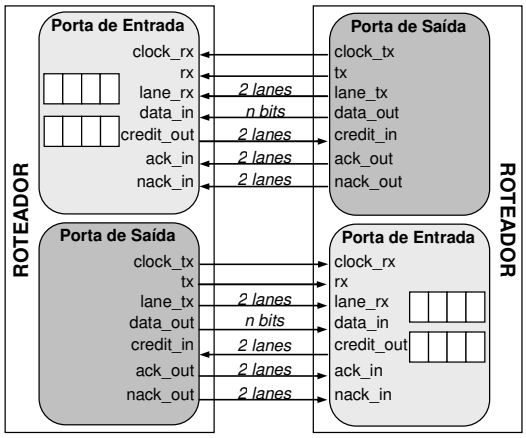


Figura 59 – Interface física entre roteadores Hermes-RB.

A política de escalonamento baseada em taxas proposta compreende duas etapas: controle de admissão seguido por um escalonamento dinâmico.

A etapa de admissão determina se um novo fluxo pode ser admitido pela rede sem colocar em risco as garantias de desempenho dadas a outros fluxos QoS. A etapa de admissão começa pelo envio de um pacote de controle do roteador origem ao roteador destino, contendo a taxa requerida pelo núcleo, conforme ilustrado na Figura 60. O fluxo QoS é admitido pela rede se e somente se todos os roteadores no caminho até o destino podem transmitir a taxa requerida. Quando o pacote de controle chega ao destino, um sinal de confirmação (*ack*) é propagado até o roteador origem. Este processo é similar ao estabelecimento de conexão no chaveamento por circuito, mas diferentemente do chaveamento por circuito não existe nenhuma reserva estática.

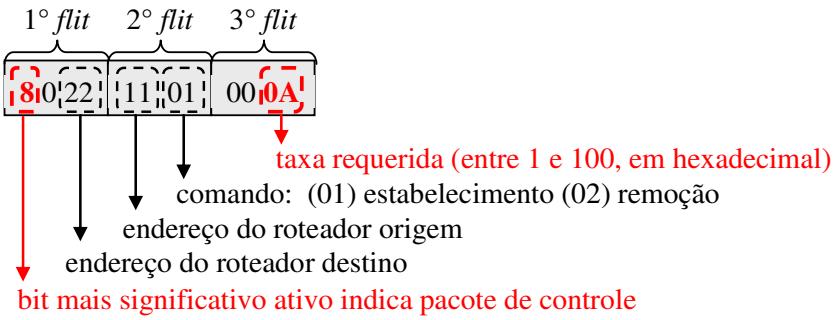


Figura 60 – Campos dos pacotes de controle.

Quando o fluxo QoS é admitido, um *conexão virtual* é estabelecida entre o roteador origem e destino, como em redes ATM [GIR98]. Esta conexão virtual corresponde a uma linha na *tabela de fluxos* (veja na Figura 62) de cada roteador no caminho da conexão. Cada linha da tabela de fluxos identifica um fluxo QoS usando os seguintes campos: roteador origem, roteador destino, taxa requerida e taxa usada. A profundidade da tabela de fluxos determina quantos fluxos QoS simultâneos podem ser admitidos em cada roteador. Depois da transmissão completa, a conexão virtual é removida pelo roteador origem através do envio de outro pacote de controle.

Uma vez o caminho virtual estabelecido, o roteador origem pode começar a enviar pacotes de dados do fluxo QoS, conforme ilustrado na Figura 61. Quando os pacotes de dados chegam a uma porta de entrada de um roteador, eles são armazenados nos *buffers* de entrada, arbitrados e roteados para uma porta de saída (Figura 62). Pacotes roteados para a mesma porta de saída são servidos de acordo com a política de escalonamento proposta.

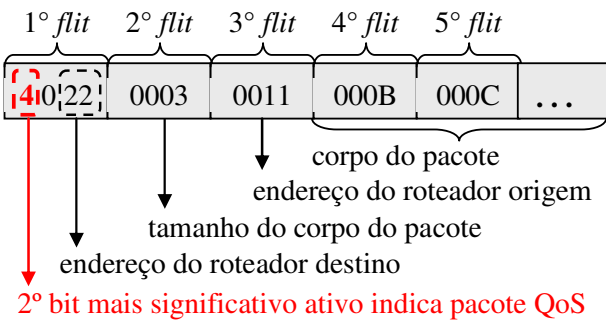


Figura 61 – Campos dos pacotes de dados.

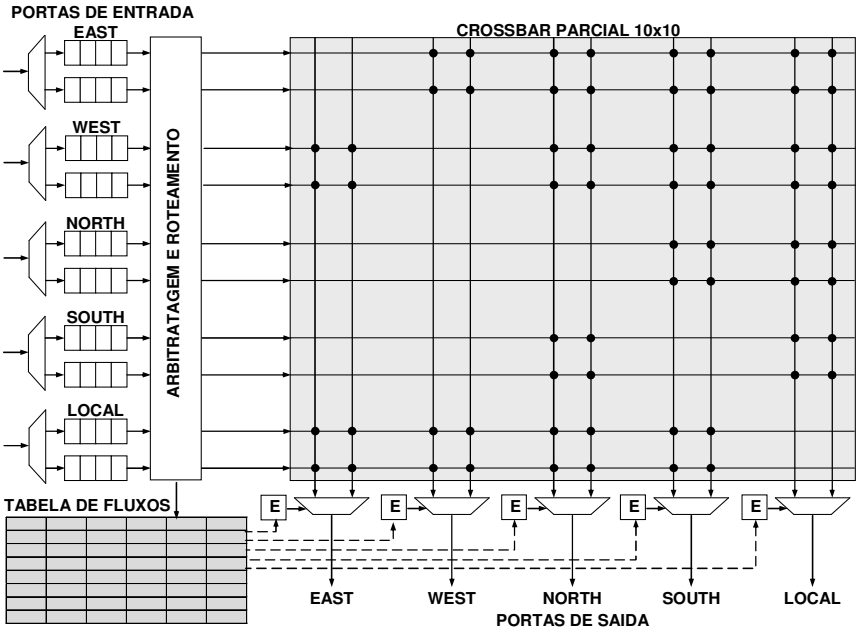


Figura 62 – Arquitetura do roteador para suportar o escalonamento baseado em taxas proposto.

Na política de escalonamento implementada, fluxos BE são transmitidos somente quando nenhum fluxo QoS requer o canal físico. Quando dois ou mais fluxos QoS competem, o fluxo com maior prioridade é escalonado primeiro.

Como ilustrado na Figura 62, a tabela de fluxos é lida pelo escalonador (blocos nomeados E, na Figura 62) para encontrar a prioridade de cada fluxo QoS roteado para uma mesma porta de saída. A *prioridade* de um fluxo QoS é a diferença entre a taxa requerida e a taxa atualmente usada pelo fluxo QoS. A prioridade do fluxo é atualizada periodicamente de acordo com a Equação 9, onde *i* é o instante de medição da taxa. Uma prioridade positiva significa que, no período de amostragem considerado, o fluxo está usando menos largura de banda do que a requerida. Uma prioridade negativa significa que, no período de amostragem considerado, o fluxo está violando a taxa admitida.

$$prioridade_i = taxa\ requerida_i - taxa\ usada_i \tag{9}$$

A taxa requerida é fixada durante a etapa de controle de admissão. A taxa usada (TU_i) é computada *periodicamente* de acordo com a Equação 10:

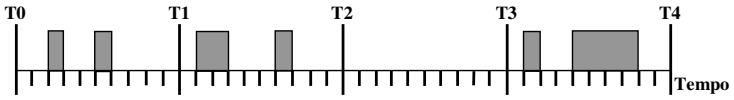
$$TU_i = \begin{cases} TA_i, & if\ TU_i = 0 \\ \dfrac{TU_{i-1} + TA_i}{2}, & if\ TU_i \neq 0. \end{cases} \tag{10}$$

Na Equação 10:

- TA_i é a taxa atual usada durante o período atual;
- TU_i é a média entre a taxa usada no período anterior (TU_{i-1}) e a taxa atual (TA_i).

A Figura 63 ilustra pacotes de um dado fluxo sendo transmitidos. Os tempos T0 a T4 designam quando as taxas são amostradas, assumindo, no exemplo, 10 unidades de tempo em cada

intervalo. A tabela na Figura corresponde ao comportamento de uma linha na tabela de fluxos de T0 a T4.



Tempo	T0	T1	T2	T3	T4
Roteador origem	01	01	01	01	01
Roteador destino	55	55	55	55	55
Taxa requerida	25%	25%	25%	25%	25%
Taxa atual (**TA**)	0%	20%	30%	0%	50%
Taxa usada (**TU**)	0%	20%	25%	12%	31%
Taxa real	0%	20%	25%	16%	25%
Prioridade	**25**	**5**	**0**	**13**	**-6**

Figura 63 – Exemplo do comportamento de uma transmissão de pacotes de um fluxo QoS.

Neste exemplo, a 4ª linha da tabela contém a taxa requerida (25%) para este fluxo. No tempo T1, a taxa atual (5ª linha) é 20%, correspondendo à largura de banda usada pelo fluxo no intervalo anterior (T0 a T1). De acordo com a Equação 10 é possível obter a taxa usada (6ª linha da tabela). A 8ª linha da tabela contém a prioridade do fluxo, a qual é atualizada de acordo com a Equação 9.

O intervalo entre os tempos de amostragem é um parâmetro importante do método. A 7ª linha contém a taxa real do fluxo (apresentada aqui somente com o objetivo de comparação, não está presente na tabela de fluxos). Se o intervalo escolhido é muito curto, a taxa usada computada pode não corresponder à taxa real, reduzindo a eficiência do método. Se o intervalo é muito longo, a taxa usada computada será correta, mas a prioridade do fluxo permanecerá fixa por um longo período, também prejudicando o método.

Para minimizar o erro induzido pelo período de amostragem, o método adota dois intervalos de amostragem. No exemplo previamente apresentado, considere uma segunda taxa atual (*TA2*) e um intervalo de amostragem 4 vezes maior do que o original. Neste exemplo, *TA2* será igual a 100% (somatório de *TA* de T0 a T4) em T4. Dividindo *TA2* por 4, a taxa usada correta é obtida (*TUC*, na Equação 11). Pode ser observado que aplicando *TUC* a *TU* a cada *n* intervalos (4 neste exemplo), o erro é minimizado.

$$TUC = \frac{TA2}{n} = \frac{\sum_{i=0}^{n-1} TA_i}{n} \tag{11}$$

Conseqüentemente, na Equação 10, TU_i recebe *TUC* quando *i mod n* é igual a zero, onde *n* corresponde ao resultado da divisão do valor do intervalo de amostragem longo pelo valor do intervalo de amostragem curto. É importante mencionar que se somente a taxa usada é considerada no cálculo da prioridade ($prioridade_i = 100 - TU_i$), a política de escalonamento tende a balancear o uso do canal físico. Isto implica em desrespeitar o fato que fluxos QoS distintos podem requerer taxas distintas.

7.2 Validação Funcional da Rede Hermes-RB

A rede Hermes-RB foi descrita em VHDL e validada por simulação funcional. A Figura 64 mostra a simulação funcional do estabelecimento de uma conexão virtual entre os núcleos 00 e 20. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 64.

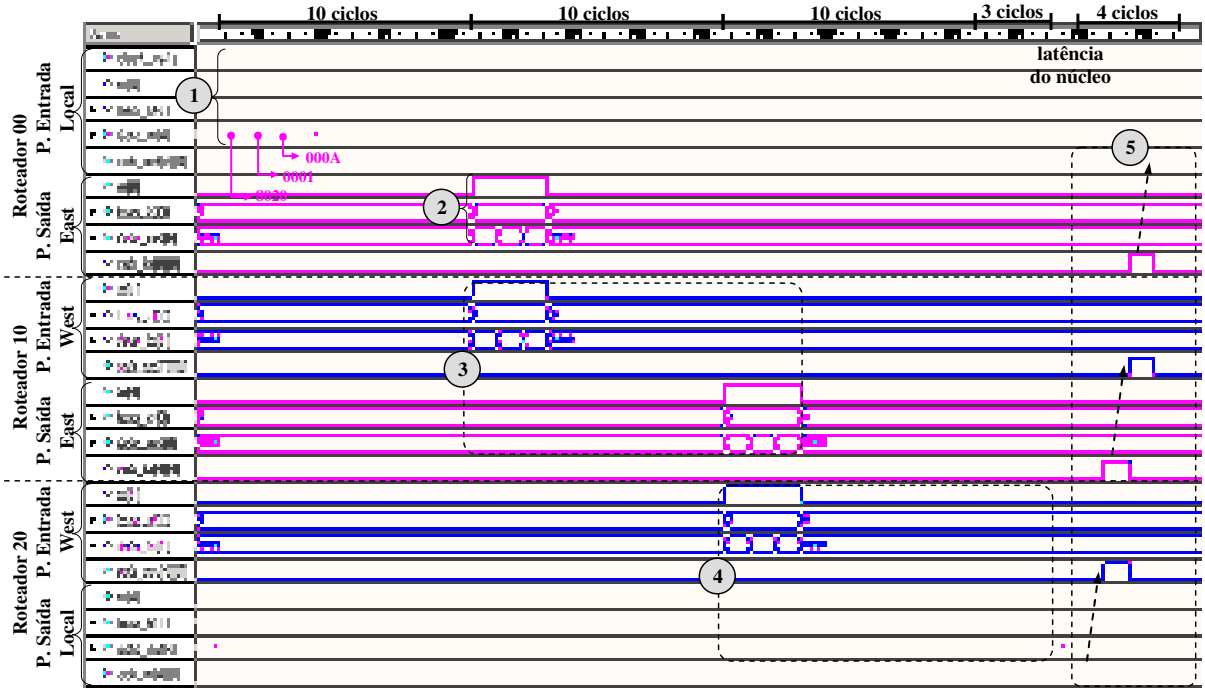


Figura 64 – Simulação do estabelecimento de uma conexão virtual entre os núcleos 00 e 20.

1. O roteador 00 recebe um pacote pelo canal virtual L1 da porta Local (índice 4). O bit mais significativo do primeiro *flit* (8020) está ativo, indicando pacote BE de controle. Conseqüentemente, o pacote contém os campos ilustrados na Figura 60: a parte baixa do primeiro *flit* indica o endereço do roteador destino (20), a parte alta do segundo *flit* indica o endereço do roteador origem (00), a parte baixa do segundo *flit* indica o comando (01 - estabelecimento de uma conexão) e a parte baixa do terceiro *flit* indica a taxa requerida (0A, ou seja, 10% da largura de banda disponível).
2. O roteador 00 realiza o roteamento do pacote, verifica se a porta de saída selecionada (neste caso a porta East) possui a largura de banda requerida, insere a conexão na tabela de fluxos e então transmite o pacote pelo canal virtual L1 da porta East (índice 0).
3. O roteador 10 recebe o pacote pelo canal virtual L1 da porta West (índice 1), realiza o roteamento, verifica se a porta de saída selecionada (neste caso a porta East) possui a largura de banda requerida, insere a conexão na tabela de fluxos e então transmite o pacote pelo canal virtual L1 da porta East (índice 0).
4. O roteador 20 recebe o pacote pelo canal virtual L1 da porta West (índice 1), realiza o roteamento, verifica se a porta de saída selecionada (neste caso, a porta Local) possui a largura de banda requerida, insere a conexão na tabela de fluxos e então transmite o pacote pelo canal

virtual L1 da porta Local (índice 4).

5. O núcleo destino recebe o pedido de estabelecimento de conexão e então ativa o sinal *ack*, informando que a conexão foi estabelecida com sucesso. O sinal *ack* é propagado até a origem, consumindo um ciclo de relógio em cada roteador no caminho da conexão. Quando o sinal *ack* chega ao núcleo origem, o núcleo pode começar a enviar os dados.

A latência ideal em ciclos de relógio para estabelecer e remover uma conexão é dada pela Equação 12:

$$latência\ ideal = 10n + 3 + LA + (n+1) \tag{12}$$

onde: 10 é o número de ciclos de relógio requerido pelo algoritmo de arbitragem/roteamento para cada roteador, n é o número de roteadores no caminho da comunicação (origem e destino inclusive), 3 é o tamanho do pacote BE de controle, LA é a latência para o núcleo destino enviar o *ack* e $(n + 1)$ representa o número de ciclos de relógio necessários para a propagação do *ack* do núcleo destino em direção ao núcleo origem.

Na simulação apresentada na Figura 64, observa-se que o núcleo destino ativou o sinal *ack* um ciclo de relógio após o recebimento do pacote (LA = 1). No entanto, este tempo pode variar de acordo com a implementação da interface de rede que conecta o núcleo à rede.

O envio de dados pela conexão estabelecida entre os núcleos 00 e 20 é apresentada na Figura 65. Esta Figura ilustra a continuação da simulação apresentada na Figura 64. Os passos da simulação são descritos abaixo, onde a numeração tem correspondência na Figura 65.

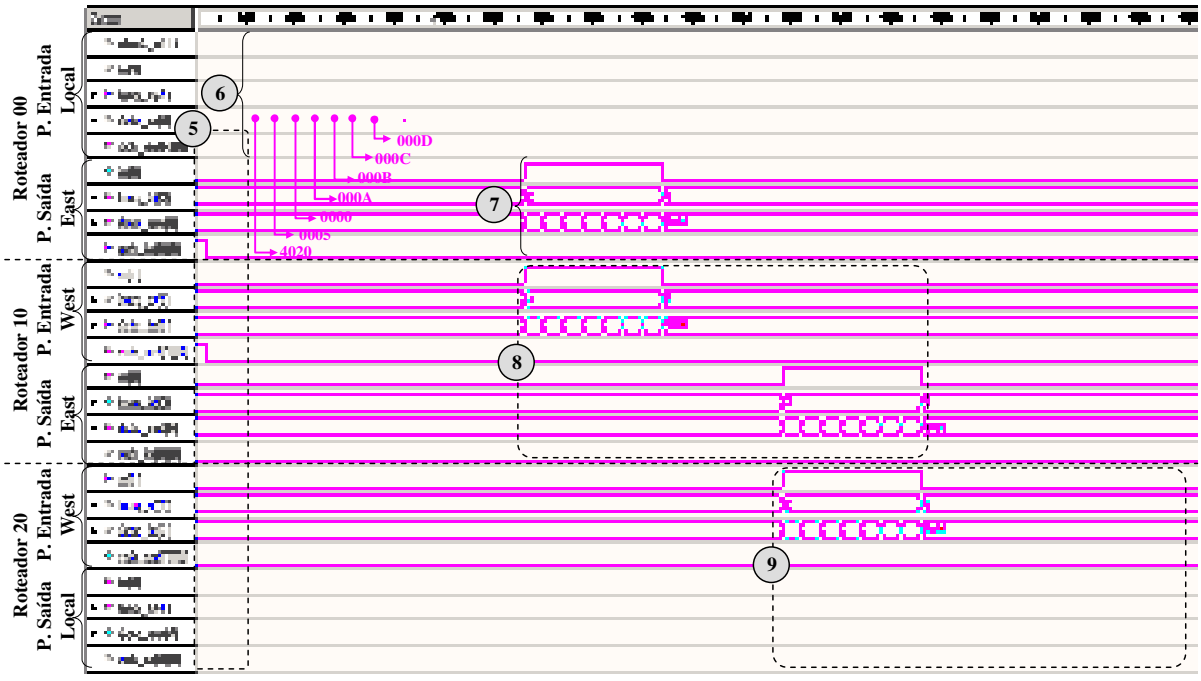


Figura 65 – Simulação do envio de dados pela conexão virtual estabelecida entre os núcleos 00 e 20.

6. O roteador 00 recebe um pacote pelo canal virtual L2 da porta Local (índice 4). O segundo bit mais significativo do primeiro *flit* (4020) está ativo, indicando pacote de dados QoS.

Conseqüentemente, o pacote contém os campos ilustrados na Figura 61: a parte baixa do primeiro *flit* indica o endereço do roteador destino (20), o segundo *flit* indica o número de *flits* do pacote, o terceiro *flit* indica o endereço do roteador origem (00) e os *flits* seguintes correspondem ao corpo do pacote.

7. O roteador 00 realiza o roteamento do pacote, a tabela de fluxos é acessada para que a prioridade do pacote seja consultada e então o pacote é transmitido pelo canal virtual L2 da porta East (índice 0).
8. O roteador 10 recebe o pacote pelo canal virtual L2 da porta West (índice 1), realiza o roteamento do pacote, a tabela de fluxos é acessada para que a prioridade do pacote seja consultada e então o pacote é transmitido pelo canal virtual L2 da porta East (índice 0).
9. O roteador 20 recebe o pacote pelo canal virtual L2 da porta West (índice 1), realiza o roteamento do pacote, a tabela de fluxos é acessada para que a prioridade do pacote seja consultada e então o pacote é transmitido pelo canal virtual L2 da porta Local (índice 4).

Embora a simulação apresentada na Figura 65 ilustre o envio de apenas um pacote, um fluxo QoS normalmente é composto por vários pacotes. Não faz sentido estabelecer uma conexão para enviar apenas um pacote. Depois do envio do último pacote de um fluxo, o núcleo origem deve enviar um outro pacote BE de controle, removendo a conexão. O procedimento de envio do pacote BE de controle para remover a conexão é igual ao apresentado na Figura 64, à exceção da parte baixa do segundo *flit* que agora contém o comando 02 (remoção da conexão).

A latência ideal em ciclos de relógio para transferir um pacote de **dados** da origem para o destino é dada pela Equação 13.

$$latência\ ideal = (AR \times n) + P \tag{13}$$

onde: *AR* é o número de ciclos de relógio requerido pelo algoritmo de arbitragem/roteamento para cada roteador, *n* é o número de roteadores no caminho da comunicação (origem e destino inclusive), *P* é o tamanho do pacote em *flits*. A constante *AR* possui o valor 5 para pacotes BE e o valor 13 para pacotes QoS.

Nota-se que a latência ideal para pacotes BE de dados é igual a das redes Hermes-VC e Hermes-FP. No entanto, a latência ideal para pacotes QoS aumenta significativamente. Primeiro, porque é necessário o estabelecimento e a remoção de uma conexão. Segundo, porque cada roteador no caminho da conexão consome 2,6 vezes mais tempo para arbitrar e rotear um pacote QoS do que para arbitrar e rotear um pacote BE de dados.

A simulação apresentada na Figura 64 ilustra o estabelecimento bem sucedido de uma conexão. Entretanto, quando o roteador não possui a largura de banda requerida pela conexão, ele deve rejeitá-la. O procedimento de rejeição de uma conexão é apresentado na Figura 66. Os passos da simulação são descritos a seguir, onde a numeração tem correspondência na Figura 66.

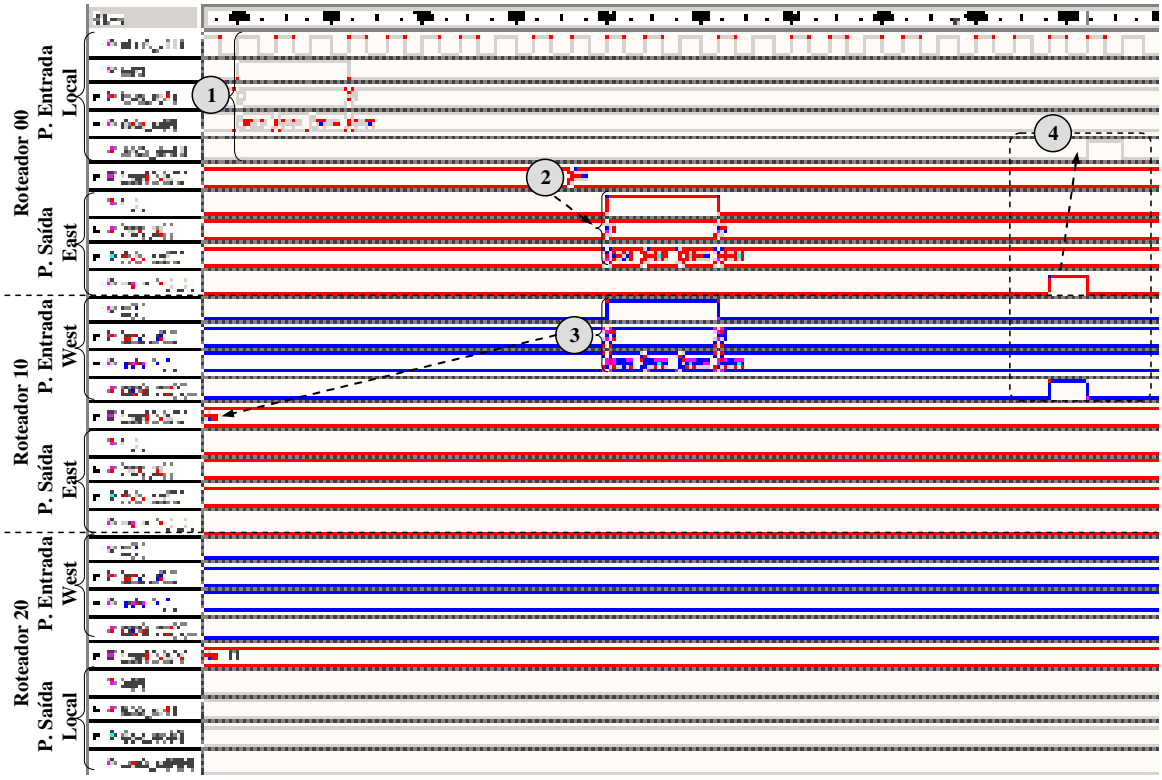


Figura 66 – Simulação de um pedido de estabelecimento de conexão rejeitado.

1. O roteador 00 recebe um pacote pelo canal virtual L1 da porta Local (índice 4). O bit mais significativo do primeiro *flit* (8020) está ativo, indicando pacote BE de controle. Conseqüentemente, o pacote contém os campos ilustrados na Figura 60: a parte baixa do primeiro *flit* indica o endereço do roteador destino (20), a parte alta do segundo *flit* indica o endereço do roteador origem (00), a parte baixa do segundo *flit* indica o comando (01 - estabelecimento de uma conexão) e a parte baixa do terceiro *flit* indica a taxa requerida (0A, ou seja, 10% da largura de banda disponível).
2. O roteador 00 realiza o roteamento do pacote, verifica se a porta de saída selecionada (neste caso a porta East) possui a largura de banda requerida, aloca a taxa requerida (*busyRate*(0) = 0A) e então transmite o pacote pelo canal virtual L1 da porta East (índice 0).
3. O roteador 10 recebe o pacote pelo canal virtual L1 da porta West (índice 1), realiza o roteamento e verifica se a porta de saída selecionada (neste caso, a porta East) possui a largura de banda requerida. Como a porta de saída East possui 96% da largura de banda ocupada (*busyRate*(0) = 60, em hexadecimal), o roteador 10 deve negar o pedido de estabelecimento da conexão e não enviar o pacote ao próximo roteador.
4. O roteador 10 rejeita o pedido de estabelecimento da conexão ativando o sinal *nack*. Este sinal é propagado até a origem, consumindo um ciclo de relógio em cada roteador no caminho da conexão. Quando o sinal *nack* chega ao núcleo origem, o núcleo deve enviar um pacote BE de controle para remover a conexão. Este pacote é necessário porque o sinal *nack* não carrega informações suficientes para remover a conexão.

8 AVALIAÇÃO DOS MECANISMOS PARA PROVER QOS

É difícil definir um modelo genérico para avaliar o desempenho de uma rede. O comportamento destas possui diferenças consideráveis de uma arquitetura para outra e de uma aplicação para outra. Por exemplo, existem aplicações que enviam mensagens longas, enquanto outras transmitem mensagens muito curtas. Segundo Duato e Yalamanchili [DUA02], o desempenho da rede geralmente é mais afetado pelas condições de tráfego do que pelos parâmetros de projeto.

Este Capítulo apresenta os experimentos que comparam, através da simulação VHDL funcional, as redes descritas nos Capítulos anteriores. Os parâmetros fixos de projeto de rede são: rede malha 8x8; roteamento XY; *flit* com largura igual a 16 bits; 2 canais virtuais; *buffer* com profundidade de 8 posições para cada canal virtual de entrada. A Seção 8.1 descreve a configuração dos experimentos utilizados na avaliação das redes. Os resultados da avaliação do uso de canais virtuais sobre a rede Hermes (Hermes-VC) é realizada na Seção 8.2. A Seção 8.3 apresenta os resultados da avaliação do mecanismo de chaveamento por circuito (Hermes-CS). Os resultados da avaliação do mecanismo baseado em prioridades fixas (Hermes-FP) e do mecanismo baseado em prioridades dinâmicas (Hermes-DP) são apresentados nas Seções 8.4 e 8.5, respectivamente. A Seção 8.6 mostra os resultados da avaliação do mecanismo baseado em taxas (Hermes-RB). Os resultados de área dos roteadores destas redes, mapeados para ASIC e FPGA, são apresentados na Seção 8.7.

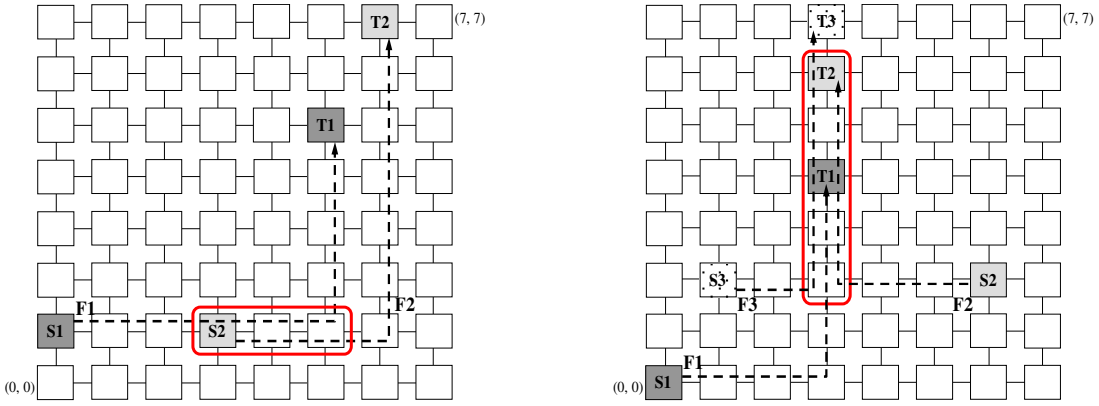
8.1 Configuração dos Experimentos

A Tabela 4 apresenta os fluxos usados nos experimentos. O fluxo A é caracterizado como um serviço CBR (do inglês *Constant Bit Rate*), isto é, um fluxo transmitindo em uma taxa fixa durante todo o tempo de vida [KUM04]. O fluxo B é caracterizado como um serviço VBR (do inglês *Variable Bit Rate*), isto é, um fluxo com taxa de bit variável [KUM04]. Este fluxo VBR é modelado usando distribuição Pareto com a mesma modelagem proposta em [PAN05]. Segundo [PAN05], a distribuição Pareto é observável no fluxo em rajada entre módulos intra-chip nas aplicações de vídeo MPEG-2 e aplicações de rede. Os fluxos A e B possuem requisitos de QoS, como latência e *jitter*. Núcleos gerando fluxos A e B transmitem 200 pacotes. Porém, para que as fases de carga e descarga da rede não interfiram nos resultados do experimento, os primeiros e os últimos 50 pacotes de cada fluxo são desconsiderados na análise. O fluxo C é um fluxo BE, também modelado usando distribuição Pareto. Este fluxo é usado para perturbar os fluxos com requisitos de QoS (A e B), sendo considerado um fluxo de ruído. Por esta razão, resultados do fluxo C não são discutidos. Este documento apresenta os resultados somente para um número pequeno de pacotes (200), porém um comportamento semelhante foi observado em experimentos (não discutidos aqui) com um número maior de pacotes (2000).

Tabela 4 – Caracterização dos fluxos usados nos experimentos.

Tipo	Serviço	QoS	Distribuição	Número de Pacotes	Tamanho do Pacote	Destino
A	CBR	Sim	Uniforme (20%)	200	50	Único
B	VBR	Sim	Pareto (40% em ON)	200	50	Único
C	BE	Não	Pareto (20% em ON)	Aleatório	20	Aleatório

Dois cenários de avaliação são definidos para estes fluxos. No primeiro, dois fluxos QoS (F1 e F2) originados em diferentes núcleos compartilham parte do caminho até os destinos. No segundo cenário, três fluxos QoS (F1, F2 e F3) compartilham parte do caminho até os destinos. Os demais núcleos da rede transmitem fluxos C, perturbando os fluxos QoS. A Figura 67 apresenta a distribuição espacial dos núcleos origem e destino.



(a) Cenário I **(b) Cenário II**

Figura 67 – Distribuição espacial dos núcleos origem e destino dos fluxos QoS. *S* indica o núcleo origem de cada fluxo. *T* indica o núcleo destino de cada fluxo. As linhas pontilhadas indicam o caminho de cada fluxo. Os retângulos arredondados destacam a área onde estes fluxos competem por recursos da rede.

A distribuição espacial do tráfego e os cenários foram escolhidos para ressaltar as limitações dos projetos de rede quando recursos são compartilhados entre fluxos QoS. Modelos CBR (por exemplo, vídeo não compactado) e VBR (por exemplo, MPEG) são artificiais, porém o comportamento destes modelos corresponde ao comportamento de aplicações reais.

A Tabela 5 resume os experimentos usados para avaliar os projetos de rede. A coluna prioridade não tem significado para as redes Hermes-VC e Hermes-RB. Na Hermes-CS, fluxos com prioridade 1 são fluxos GT e fluxos com prioridade 0 são fluxos BE.

Tabela 5 – Configuração dos experimentos. *NA* significa Não Aplicável.

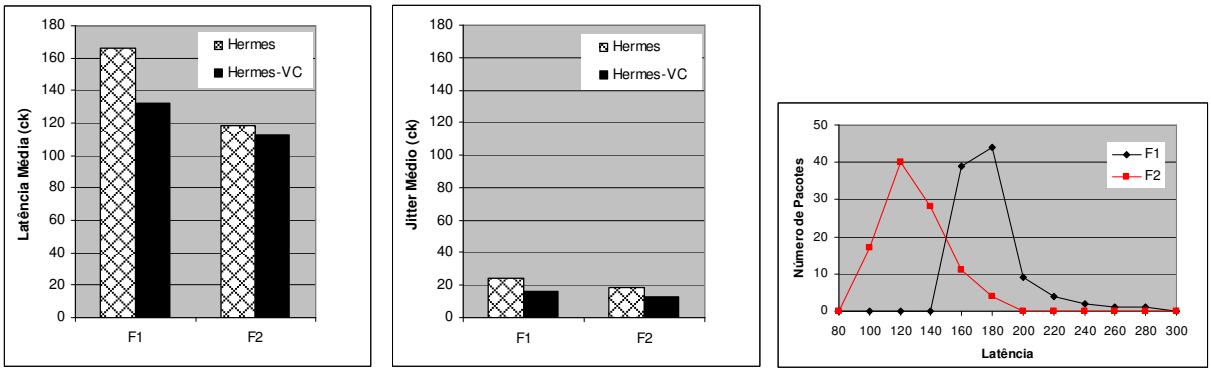
Experimento	Cenário	Fluxo F1 (QoS)		Fluxo F2 (QoS)		Fluxo F3 (QoS)		Fluxos de Ruído (BE)	
		Tipo	Prioridade	Tipo	Prioridade	Tipo	Prioridade	Tipo	Prioridade
I	I	A (CBR)	1	A (CBR)	0	NA	NA	C	0
II	I	A (CBR)	1	A (CBR)	1	NA	NA	C	0
III	I	B (VBR)	1	B (VBR)	1	NA	NA	C	0
IV	II	A (CBR)	1	A (CBR)	1	A (CBR)	0	C	0
V	II	B (VBR)	1	B (VBR)	1	B (VBR)	0	C	0

O número de canais virtuais define o número de fluxos competindo por recursos da rede no mesmo canal físico. Como todos os projetos de rede têm dois canais virtuais, existem três opções quando mais de um fluxo QoS deve ser transmitido: todos os fluxos com prioridade baixa (todos os experimentos usando a rede Hermes-VC), alguns fluxos com prioridade alta (experimentos I, IV e V, usando todas redes, à exceção da Hermes-VC), ou todos os fluxos com prioridade alta (experimentos II e III, usando todas redes, à exceção da Hermes-VC).

8.2 Avaliação do Uso de Canais Virtuais

Esta Seção compara a rede Hermes sem canais virtuais (Hermes) e com canais virtuais (Hermes-VC). Os seguintes parâmetros de desempenho são avaliados: latência, *jitter*, espalhamento da latência e vazão.

A Figura 68 ilustra a latência média, o *jitter* e o espalhamento da latência para o experimento I. As redes Hermes e Hermes-VC oferecem apenas o serviço BE, onde a latência média e o *jitter* do pacote dependem das condições de tráfego durante a transmissão. Deste modo, nenhuma das redes oferece garantias a nenhum fluxo. No entanto, é possível observar que o fluxo F2 tem latência média menor do que o fluxo F1. Isto ocorre porque F1 e F2 são fluxos CBR. Portanto, eles inserem pacotes na rede em intervalos fixos. Como o núcleo origem de F2 está mais próximo da área disputada pelos fluxos, ele sempre é servido primeiro. É possível observar também que o uso de canais virtuais reduz a latência média e o *jitter* dos fluxos. Esta redução está associada (1) ao compartilhamento da largura de banda do canal físico entre os canais virtuais e (2) a possibilidade de executar o algoritmo de arbitragem/roteamento de um fluxo enquanto outro fluxo está sendo transmitido. Sem o uso de canais virtuais, quando um pacote encontra um canal físico ocupado, ele aguarda o canal físico ser liberado para somente depois realizar o algoritmo de arbitragem/roteamento. Com o uso de canais virtuais, um pacote irá aguardar para realizar o algoritmo de arbitragem/roteamento somente se todos os canais virtuais estiverem ocupados. Caso exista algum canal virtual livre, o algoritmo de arbitragem/roteamento é executado e a largura de banda do canal físico passa a ser compartilhada entre os canais virtuais ocupados. Cabe lembrar que os recursos são compartilhados não somente pelos fluxos F1 e F2, mas também pelos fluxos tipo C.

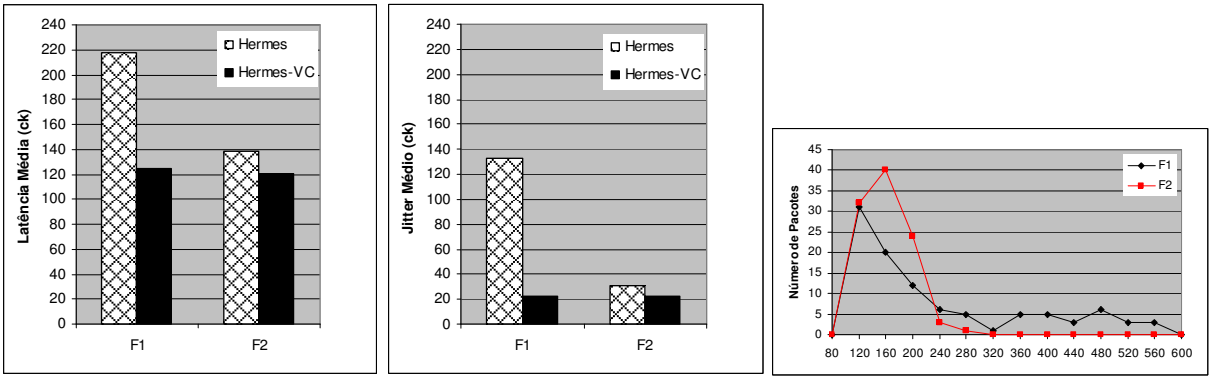


(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-VC**

Figura 68 – Fluxos F1 e F2, experimento I, fluxos CBR.

As redes Hermes e Hermes-VC não possuem diferenciação entre fluxos. Portanto, os experimentos I e II, onde a única diferença é a prioridade dos fluxos F1 e F2, são idênticos para estas redes.

A Figura 69 ilustra a latência média, o *jitter* e o espalhamento da latência para o experimento III, onde F1 e F2 são fluxos VBR. Neste experimento, os pacotes são inseridos na rede com intervalos variáveis usando uma carga de 40% no período ON. Comparando os resultados dos experimentos III e I, observa-se que as redes Hermes e Hermes-VC apresentam comportamentos diferentes. Na rede Hermes, a latência média e o *jitter* de ambos os fluxos aumentam. Isto ocorre porque a latência desta rede tende a aumentar significativamente com a disputa por recursos, provocando congestionamentos. Por conseqüência, nos períodos OFF, onde os fluxos não deveriam inserir pacotes, pacotes bloqueados estão sendo inseridos. Em oposição, na rede Hermes-VC, o benefício do compartilhamento de banda é mais visível com fluxos VBR. Em fluxos VBR, o momento de colisão dos pacotes é aleatório, e por conseqüência não existe um fluxo que seja sempre servido primeiro. Isto possui duas conseqüências: (*i*) o *jitter* e o espalhamento da latência de ambos os fluxos aumenta e (*ii*) devido à duração dos períodos OFF, as latências médias tendem a ser semelhantes.



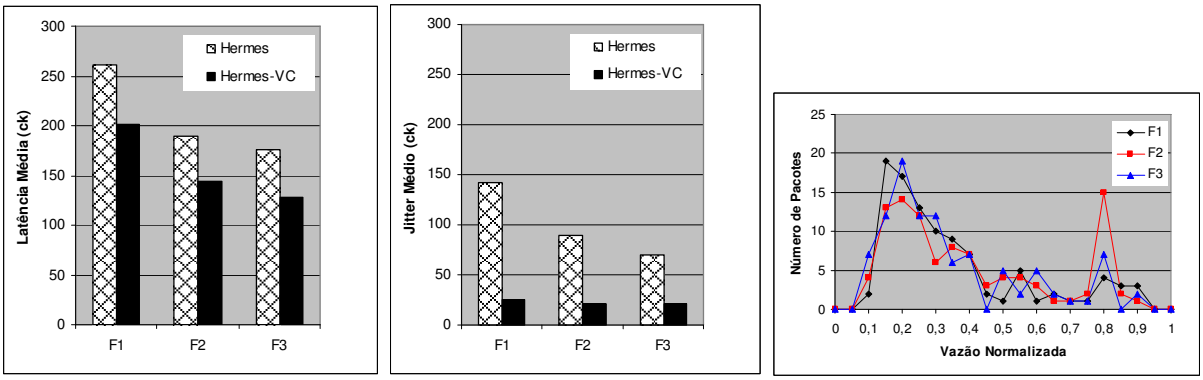
(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-VC**

Figura 69 – Fluxos F1 e F2, experimento III, fluxos VBR.

As Figuras 69 e 70 mostram o comportamento do uso de canais virtuais quando fluxos competem por recursos da rede. No caso de fluxos CBR (experimento I), um dos fluxos tem latência menor, porque está mais perto da área de disputa por recursos e sempre é servido primeiro. No caso de fluxos VBR (experimento III), o uso de canais virtuais reduz a latência de ambos os fluxos, porque, devido à variação no intervalo de inserção de pacotes, não existe um fluxo que seja sempre servido primeiro. Entretanto, os valores de *jitter* são mais altos.

A Figura 70 ilustra a latência média, o *jitter* e a vazão para o experimento IV. Aqui, três fluxos CBR competem por recursos. É possível observar que a latência média e o *jitter* dos fluxos têm o mesmo comportamento da Figura 68. A proximidade do fluxo em relação à área de disputa por recurso determina quão grandes ou pequenos serão estes valores. No cenário II, F3 está mais próximo da área de disputa por recursos, seguido por F2 e depois por F1. Portanto, F3 tem a menor

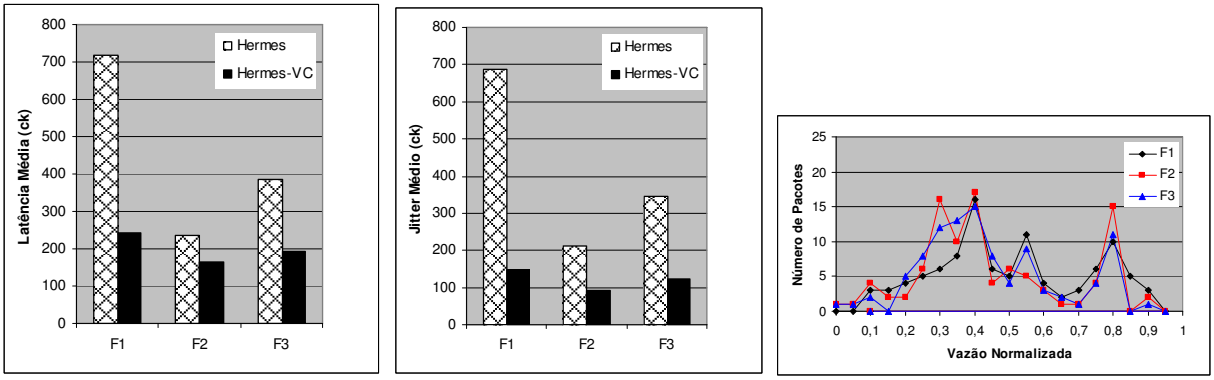
latência média, seguido por F2 e depois por F1. A vazão de todos os fluxos apresenta grande variação, havendo pacotes com taxa bem superior à taxa de inserção (20%). Isto ocorre devido às situações de congestionamento, onde pacotes são transmitidos em rajada depois da liberação da condição de bloqueio.



(a) Latência média **(b) *Jitter* médio** **(c) Vazão da rede Hermes-VC**

Figura 70 – Fluxos F1, F2 e F3, experimento IV, fluxos CBR.

A Figura 71 apresenta os resultados do experimento V, onde três fluxos VBR competem por recursos. Comparando o experimento V ao experimento III, observa-se que a latência média e o *jitter* dos fluxos apresentam aumento significativo quando mais fluxos competem pelos mesmos recursos. Observa-se também que a vazão de todos os fluxos apresenta variação maior para fluxos VBR do que para fluxos CBR (experimento IV).



(a) Latência média **(b) *Jitter* médio** **(c) Vazão da rede Hermes-VC**

Figura 71 – Fluxos F1, F2 e F3, experimento V, fluxos VBR.

A avaliação do uso de canais virtuais demonstrou que a adição de canais virtuais melhora o desempenho da rede, devido ao compartilhamento de recursos, mas nenhuma garantia é oferecida aos fluxos.

8.3 Avaliação do Mecanismo de Chaveamento por Circuito

Se um fluxo QoS tem que ser transmitido sem competição com outros fluxos QoS, o mecanismo de chaveamento por circuito é eficiente para garantir QoS. A Figura 72 ilustra a quantidade de tempo necessário para o estabelecimento da conexão, transmissão dos dados e

remoção da conexão, usando fluxos do experimento II, com F1 e F2 sendo fluxos GT, competindo pelo mesmo canal virtual. A quantidade de tempo para estabelecer e remover uma conexão (pequena neste experimento) pode variar de acordo com o tráfego da rede, pois estas ações são controladas por pacotes BE.

Ambos os fluxos estão transmitindo 10.000 *flits*, equivalente a 200 pacotes com 50 *flits* cada. Como a taxa dos fluxos é 20% da largura de banda disponível, o tempo total para transmitir todos os 10.000 *flits* é 50.000 ciclos de relógio. Como ilustrado na Figura 72, o fluxo F2 estabelece sua conexão primeiro. O fluxo F2 consome 148 ciclos de relógio para criar a conexão, mais 50.000 ciclos de relógio para transmitir dados, e 73 ciclos de relógio para remover a conexão. O fluxo F1 espera todos estes ciclos de relógio para começar a transmissão. Conseqüentemente, o tempo de transmissão total para ambos os fluxos é aproximadamente 100.000 ciclos de relógio.

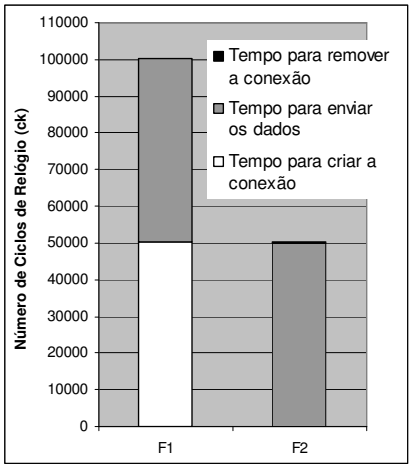


Figura 72 – Tempo para estabelecimento da conexão, transmissão dos dados e remoção da conexão para os fluxos F1 e F2, experimento II, fluxos CBR.

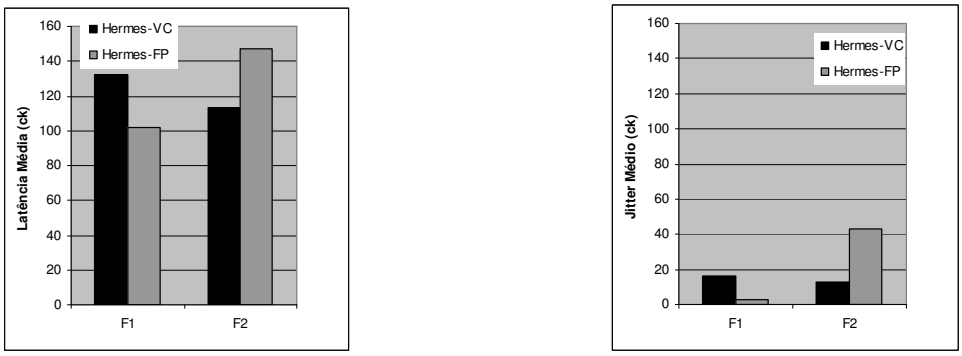
Se o chaveamento por pacote é usado, canais são compartilhados entre fluxos resultando em um tempo menor para entregar todos os flits (neste caso, aproximadamente 50.000). Isto revela a principal desvantagem do chaveamento por circuito: reserva de recursos estática. Esta desvantagem pode ser parcialmente minimizada usando multiplexação por divisão de tempo (TDM), alocando a largura de banda do canal físico em fatias de tempo de tamanho fixo. Entretanto, quando se usa TDM, um comportamento regular do tráfego é requerido (como fluxos CBR) para ajustar a taxa de transmissão dos dados às fatias de tempo reservadas. Por outro lado, o risco de desperdiçar largura de banda continua presente, juntamente com o risco de perder dados.

8.4 Avaliação do Mecanismo de Prioridade Fixa

Esta Seção compara as redes Hermes-VC e Hermes-FP. Os seguintes parâmetros de desempenho são avaliados: latência, *jitter*, espalhamento da latência e vazão.

A Figura 73 ilustra a latência média e o *jitter* para o experimento I. Como apresentado na Seção 8.2, a Hermes-VC não oferece garantias a nenhum fluxo, a latência média e o *jitter* do pacote dependem das condições de tráfego durante a transmissão. Na Hermes-FP, o fluxo F1 com maior

prioridade tem latência média próxima à latência ideal (5(10)+50) e *jitter* perto de zero. Isto ocorre porque F1 tem prioridade maior e uso exclusivo do canal virtual L2. Conseqüentemente, sempre que F1 tem dados para transmitir, ele tem acesso ao canal físico. Por outro lado, F2 é sempre bloqueado enquanto F1 está entregando *flits*. F2 apresenta uma latência média de aproximadamente 50 ciclos de relógio maior do que a latência ideal e seu *jitter* é aproximadamente 40 ciclos de relógio, representando 80% do tamanho do pacote.

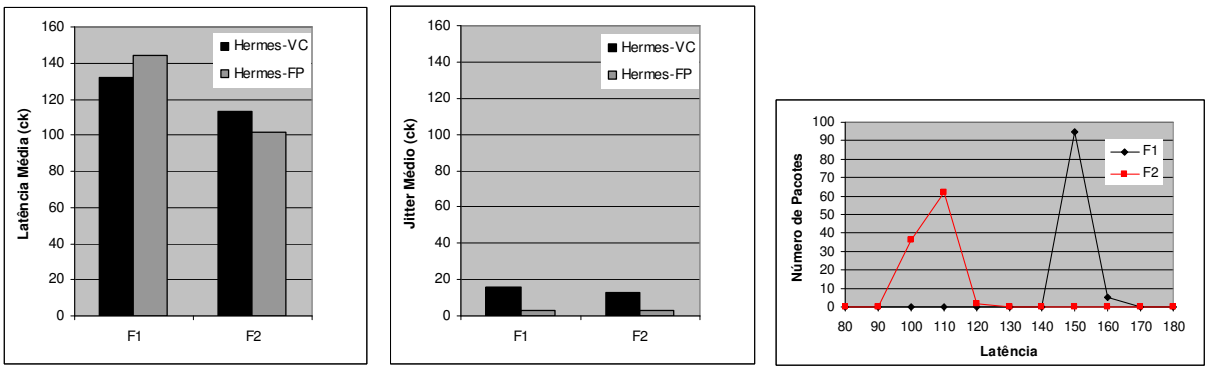


(a) Latência média **(b) *Jitter* médio**

Figura 73 – Fluxos F1 e F2, experimento I, fluxos CBR com prioridades diferentes.

Este experimento mostra que, mesmo quando existem fluxos perturbando, um mecanismo baseado em prioridades fixas é eficiente para atender aos requisitos de QoS, desde que não haja competição entre fluxos com a mesma prioridade.

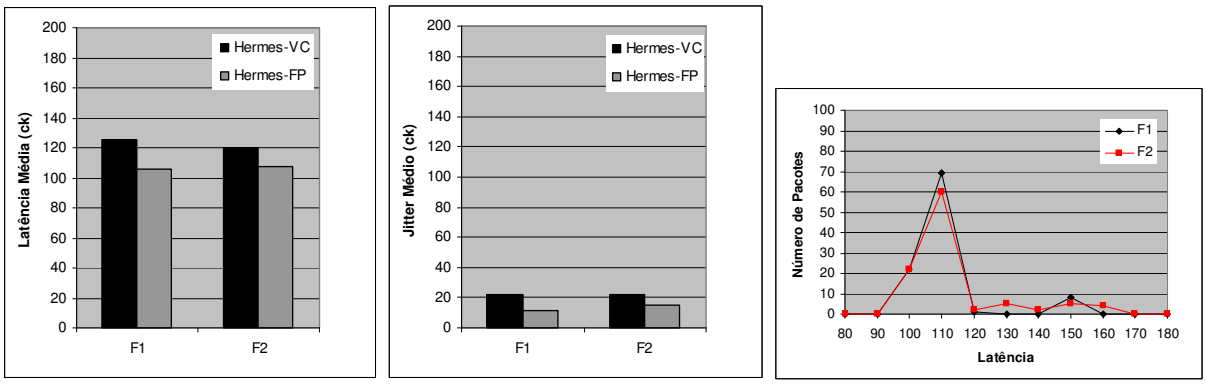
A Figura 74 ilustra a latência média, o *jitter* e o espalhamento da latência para o experimento II. Aqui, F1 e F2 têm a mesma prioridade. Logo, competem pelo canal virtual L2. É possível observar que F2 tem latência média próxima à ideal, enquanto a latência de F1 é aproximadamente 50% maior do que a ideal. Isto ocorre porque F1 e F2 são fluxos CBR. Portanto, eles inserem pacotes na rede em intervalos fixos. Como o núcleo origem de F2 está mais próximo da área disputada pelos fluxos, ele sempre é servido primeiro. Por esta razão, F1 e F2 têm *jitter* perto de zero e espalhamento da latência pequeno. O pequeno espalhamento da latência pode ser constatado pela comparação entre os resultados apresentados na Figura 74 (Hermes-FP) e na Figura 68 (Hermes-VC).



(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-FP**

Figura 74 – Fluxos F1 e F2, experimento II, fluxos CBR com mesma prioridade.

No entanto, quando F1 e F2 são fluxos VBR (experimento III) os resultados são bem diferentes, como mostrado na Figura 75. Neste experimento, os pacotes são inseridos na rede com intervalos variáveis usando uma carga de 40% no período ON, o que representa uma carga média de 20%. Assim, não existe um fluxo que é sempre servido primeiro. Isto tem duas conseqüências: (*i*) o *jitter* de ambos os fluxos aumenta, e (*ii*) devido à duração dos períodos OFF, ambas latências tendem à latência ideal.

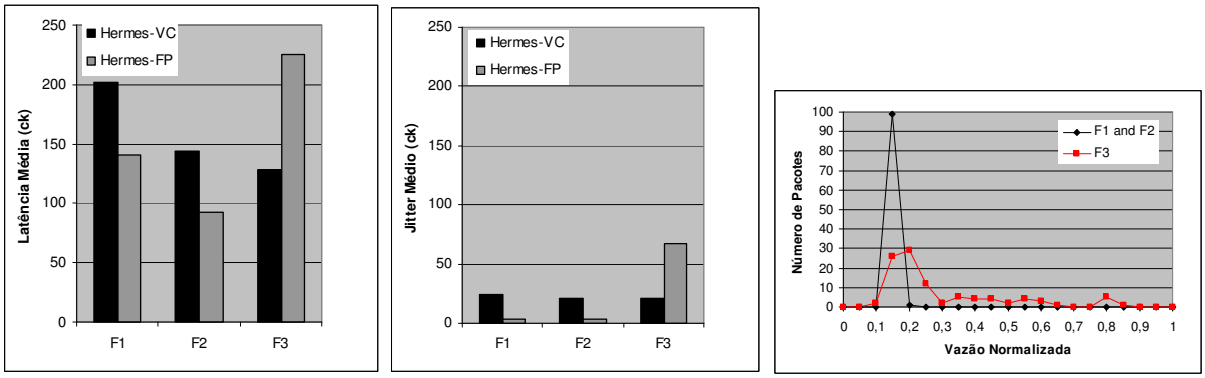


(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-FP**

Figura 75 – Fluxos F1 e F2, experimento III, fluxos VBR.

As Figuras 75 e 76 mostram o comportamento do mecanismo baseado em prioridades fixas quando fluxos com a mesma prioridade competem por recursos da rede. No caso de fluxos CBR (experimento II), um dos fluxos tem comportamento imprevisível, semelhante a um fluxo BE. No caso de fluxos VBR (experimento III), o mecanismo baseado em prioridades fixas garante latências perto da ideal para os fluxos com prioridade maior. Entretanto, estes apresentam valores mais altos de *jitter*. Dependendo dos parâmetros que especificam QoS para os fluxos, o uso do mecanismo baseado em prioridades fixas pode ser limitado a situações específicas, onde a competição entre fluxos de igual prioridade é evitada ou é mínima.

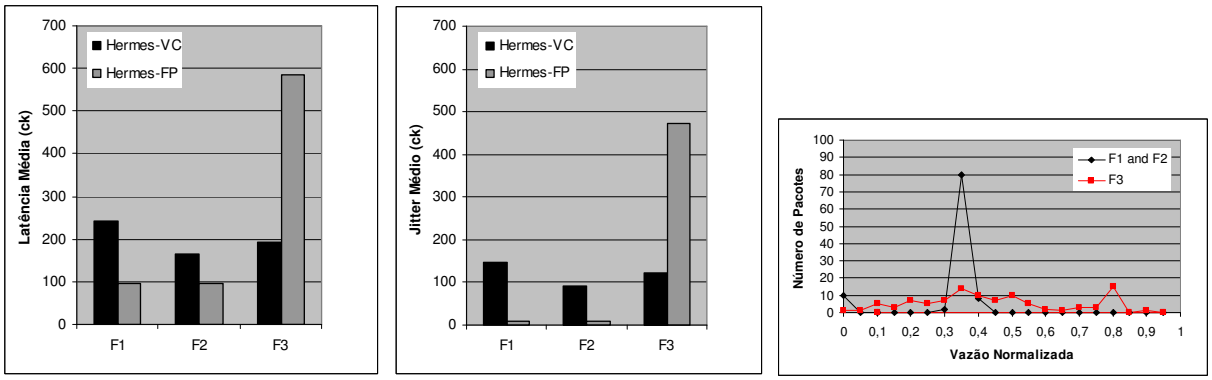
A Figura 76 ilustra a latência média, o *jitter* e a vazão para o experimento IV. Aqui, dois fluxos com prioridade alta competem por recursos com um terceiro fluxo com prioridade baixa. É possível observar que a latência média e o *jitter* dos fluxos prioritários (F1 e F2) têm comportamento similar ao da Figura 74. Estes fluxos têm 99% dos pacotes com vazão entre 15% e 20%, condizendo com suas taxas de inserção. O pequeno espalhamento da vazão para fluxos prioritários pode ser constatado pela comparação entre os resultados apresentados na Figura 76 (Hermes-FP) e na Figura 70 (Hermes-VC). Entretanto, o fluxo de prioridade baixa (F3) possui latência média alta (aproximadamente 2,5 vezes a latência ideal (5(8)+50)) e *jitter* igualmente alto. A vazão dos pacotes de F3 apresenta grande variação, tendo pacotes com taxa superior à taxa de inserção. Isto ocorre porque pacotes são transmitidos em rajada depois da liberação da condição de bloqueio. Se F3 tem vazão como requisito de QoS, usar um mecanismo baseado em prioridades fixas é inadequado.



(a) Latência média **(b) *Jitter* médio** **(c) Vazão da rede Hermes-FP**

Figura 76 – Fluxos F1, F2 e F3, experimento IV, fluxos CBR.

A Figura 77 apresenta os resultados do experimento V, onde três fluxos VBR competem por recursos. Fluxos prioritários são transmitidos com latência próxima à ideal (90 ciclos de relógio) e *jitter* perto de zero. Estes fluxos têm 90% dos pacotes com vazão entre 35% e 45%, e 10% dos pacotes com vazão entre 0% e 5%. Isso ocorre porque fluxos VBR usam uma distribuição Pareto ON-OFF. Conseqüentemente, o primeiro pacote de um período ON possui vazão extremamente baixa, porque o tempo do período OFF também é contabilizado. Aqui, o fluxo com prioridade baixa (F3) é penalizado, apresentando latência e *jitter* altos, e vazão irregular (espalhamento da vazão excessivo).



(a) Latência média **(b) *Jitter* médio** **(c) Vazão da rede Hermes-FP**

Figura 77 – Fluxos F1, F2 e F3, experimento V, fluxos VBR.

A Tabela 6 resume a avaliação do mecanismo de prioridade fixa.

Tabela 6 – Resumo da avaliação da rede Hermes-FP (todos os fluxos competem com pacotes BE).

Experimento	Descrição	QoS - Garantia de:			
		Latência	*Jitter*	Vazão	Explicação
I	Um fluxo CBR com prioridade ***sem*** competição	Sim	Sim	Sim	Quando o fluxo prioritário tem dados para transmitir, ele tem acesso ao canal físico.
II	Dois fluxos CBR com prioridade ***com*** competição	Não	Sim	Sim	A inserção de pacotes na rede em intervalos fixos faz com que o fluxo mais próximo da área disputada seja sempre servido primeiro. Conseqüentemente, este fluxo tem latência média próxima à ideal, enquanto o outro tem um aumento significativo na latência.
III	Dois fluxos VBR com prioridade ***com*** competição	Sim	Não	Sim	A inserção de pacotes na rede em intervalos variáveis permite que a latência média dos fluxos tenda à latência ideal. No entanto, faz com que o *jitter* aumente.
IV	Dois fluxos CBR ***com*** prioridade ***competindo*** com	Não	Sim	Sim	Fluxos prioritários com comportamento idêntico ao Experimento II.
	um fluxo CBR ***sem*** prioridade	Não	Não	Não	Fluxo sem prioridade sem nenhuma garantia.
V	Dois fluxos VBR ***com*** prioridade ***competindo*** com	Sim	Não	Sim	Fluxos prioritários com comportamento idêntico ao Experimento III
	um fluxo VBR ***sem*** prioridade	Não	Não	Não	Fluxo sem prioridade sem nenhuma garantia.

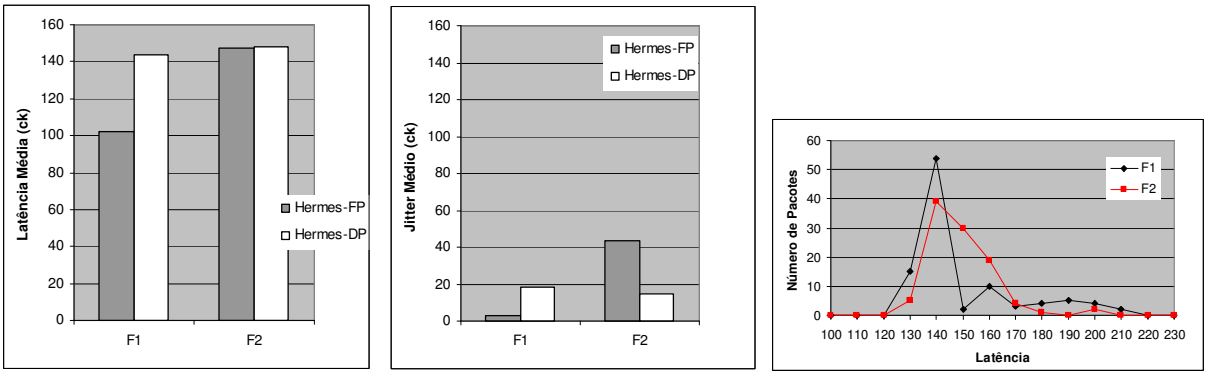
8.5 Avaliação do Mecanismo de Prioridade Dinâmica

Nesta Seção, a rede Hermes-DP é comparada à rede Hermes-FP. Os seguintes parâmetros de desempenho são avaliados: latência, *jitter* e espalhamento da latência.

A Figura 78 mostra os resultados obtidos no experimento I, onde um fluxo QoS é transmitido sem competição com outro fluxo QoS, mas competindo com fluxos BE. O desempenho da rede Hermes-DP é inferior à rede Hermes-FP. Duas razões podem explicar este resultado:

1. A latência ideal para a rede Hermes-FP é 100 ciclos de relógio (5(10)+50, onde 5 é o número de ciclos de relógio requerido pelo algoritmo de arbitragem/roteamento em cada roteador, 10 é o número de *hops* e 50 é o tamanho do pacote), enquanto para a rede Hermes-DP é 120 ciclos de relógio (7(10)+50, devido aos dois ciclos de relógio extra requeridos pelo algoritmo de arbitragem/roteamento).
2. Mesmo que a rede Hermes-DP privilegie os fluxos com prioridade maior, não existe reserva de canal virtual para pacotes prioritários. Conseqüentemente, se não existe pacote com prioridade maior sendo transmitido, fluxos BE podem ser transmitidos por todos os canais virtuais, bloqueando pacotes prioritários até que um canal virtual seja liberado.

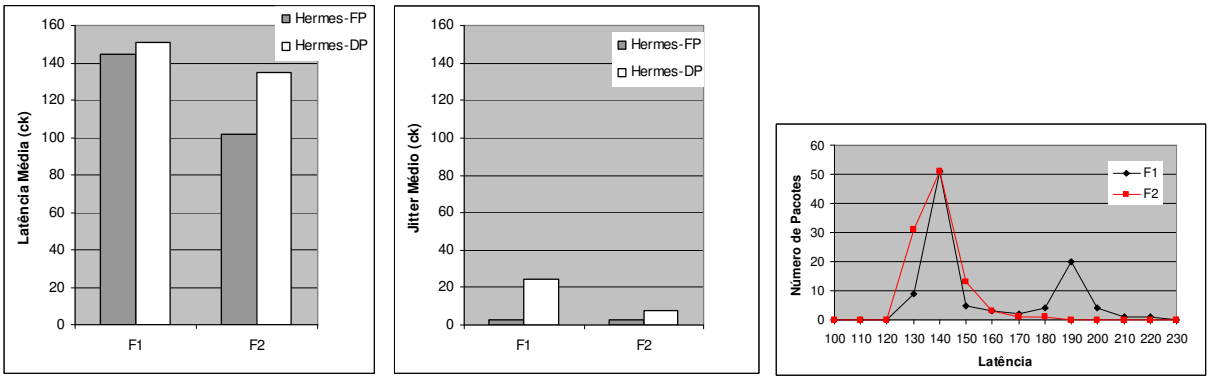
Este comportamento conduz a valores imprevisíveis de *jitter*. Neste experimento, o *jitter* médio de F2 (sem prioridade) é menor do que o de F1 (com prioridade). A segunda razão apresentada acima conduz a um importante espalhamento na latência de F1. Quando os roteadores no caminho do fluxo possuem algum canal virtual de saída livre, os pacotes são transmitidos com latência perto da ideal. Por outro lado, quando estes roteadores possuem todos os canais virtuais de saída ocupados, ocorre um bloqueio que aumenta significativamente a latência dos pacotes.



(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-DP**

Figura 78 – Fluxos F1 e F2, experimento I, fluxos CBR.

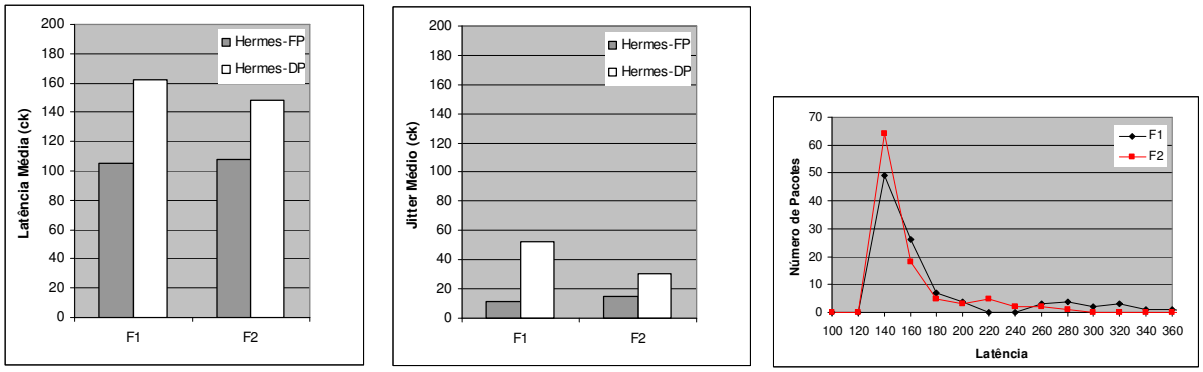
As Figuras 80 e 81 mostram os resultados obtidos quando fluxos com mesma prioridade competem pelos os mesmos recursos (canais de saída). Como mencionado na Seção 6.3, a rede Hermes-DP pode transmitir pacotes com a mesma prioridade por canais virtuais diferentes. Portanto, não se privilegia um dos fluxos em detrimento do outro, como na rede Hermes-FP. No entanto, para ambos os experimentos, o desempenho da rede Hermes-DP é inferior ao da rede Hermes-FP pela mesma razão: ausência de reserva de recursos. A competição pela ocupação dos canais virtuais de saída ocorre entre todos os fluxos, não importando qual a prioridade do mesmo. Logo, os fluxos não são perturbados apenas por fluxos com mesma prioridade, mas por todo o tráfego da rede. Por esta razão, o tipo do fluxo (CBR ou VBR) não influencia diretamente o desempenho do fluxo.



(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-DP**

Figura 79 – Fluxos F1 e F2, experimento II, fluxos CBR.

Aumentar o número de canais virtuais do roteador pode, a princípio, melhorar o desempenho dos fluxos. No entanto, a área do roteador é penalizada. Cabe ressaltar ainda que mesmo aumentando o número de canais virtuais do roteador, um determinado pacote com prioridade pode ser bloqueado ao encontrar todos os canais virtuais da porta de saída ocupados, e conseqüentemente não ter seus requisitos de QoS atendidos.



(a) Latência média **(b) *Jitter* médio** **(c) Espalhamento da latência da rede Hermes-DP**

Figura 80 – Fluxos F1 e F2, experimento III, fluxos VBR.

Embora o mecanismo baseado em prioridades dinâmicas tenha se mostrado ineficiente para garantir QoS, a avaliação deste mecanismo demonstrou a importância da reserva de recursos para que a rede ofereça QoS.

8.6 Avaliação do Escalonamento Baseado em Taxas

Nesta Seção, a rede Hermes-RB é comparada à rede Hermes-FP. A Tabela 7 apresenta os valores de latência, *jitter* e vazão para o experimento II (dois fluxos CBR com mesma prioridade). Ambas as políticas de escalonamento garantem vazão perto da taxa de inserção (20%). Analisando-se os resultados da rede Hermes-FP, o fluxo F2 tem latência média perto da ideal, enquanto o fluxo F1 tem latência maior (a latência média é 77% maior do que a ideal). F1 e F2 são fluxos CBR, competindo pelos mesmos recursos. Eles inserem pacotes na rede em intervalos fixos. Como o núcleo origem do fluxo F2 está mais próximo à região de disputa, ele é sempre servido primeiro. Este experimento demonstra que o mecanismo baseado em prioridades fixas (Hermes-FP) é ineficiente para atender requisitos de QoS quando fluxos com mesma prioridade competem pelos mesmos recursos. No escalonamento baseado em taxas (Hermes-RB), a prioridade é dinamicamente atualizada de acordo com a taxa usada, não em função do tempo de chegada dos pacotes no roteador. Conseqüentemente, como ambos os fluxos tem a mesma taxa requerida, a largura de banda é igualmente dividida entre os fluxos, resultando em valores muito parecidos de latência para ambos os fluxos, próximo aos valores ideais. O valor de *jitter* é levemente aumentado quando comparado ao escalonamento baseado em prioridades, devido à latência ideal maior do roteador Hermes-RB. Este resultado demonstra a eficiência do método.

A Tabela 8 mostra os resultados para o experimento III, onde F1 e F2 são fluxos VBR. Neste experimento, pacotes são inseridos na rede em intervalos variáveis, usando uma carga de 40% no período ON. O modelo de tráfego ON-OFF aleatoriza os instantes de inserção de pacotes, o que insere *jitter*. Por esta razão, o *jitter* não é apresentado na Tabela 8. Em ambos os métodos de escalonamento, F1 tem latência média perto da latência ideal. No escalonamento baseado em prioridade (Hermes-FP), F2 tem latência média 56% maior do que a latência ideal, e no

escalonamento baseado em taxas (Hermes-RB) somente 33% maior. Apesar do fato dos métodos terem comportamento semelhante, o mecanismo baseado em taxas é superior ao mecanismo baseado em prioridades, porque é capaz reduzir a diferença entre a latência média e a latência ideal.

Tabela 7 – Resultados para os fluxos F1 e F2, experimento II, fluxos CBR.

Figuras de Desempenho		Hermes-FP		Hermes-RB	
		F1	F2	F1	F2
Latência	Ideal (ck)	250,00	250,00	330,00	330,00
	Mínima (ck)	441,00	250,00	330,00	330,00
	Média (ck)	**443,40**	**251,86**	**333,54**	**332,42**
	Máxima (ck)	450,00	258,00	350,00	346,00
Jitter (ck)		2,14	1,78	4,07	3,01
Vazão Média (%)		19,80	19,80	19,80	19,80

Tabela 8 – Resultados para os fluxos F1 e F2, experimento III, fluxos VBR.

Figuras de Desempenho		Hermes-FP		Hermes-RB	
		F1	F2	F1	F2
Latência	Ideal (ck)	250,00	250,00	330,00	330,00
	Mínima (ck)	250,00	250,00	330,00	330,00
	Média (ck)	**253,40**	**351,96**	**337,58**	**440,00**
	Máxima (ck)	266,00	390,00	477,00	545,00
Vazão Média (%)		38,82	39,26	38,86	39,40

8.7 Resultados de Área

A Tabela 9 detalha a área do roteador mapeado para a biblioteca de células padrão CMOS 0,35µm (TSMC). A área do roteador é semelhante para as redes Hermes-VC e Hermes-FP. A área do roteador Hermes-DP é superior a dos outros roteadores devido à inclusão de um circuito mais complexo na lógica de arbitragem/roteamento. O roteador Hermes-CS tem uma área menor, porque os *buffers* de entrada do canal virtual L1 são substituídos por simples registradores. Os resultados apontam para o fato que mecanismos baseados em prioridades fixas (Hermes-FP) e chaveamento por circuito (Hermes-CS) não aumentam a área, comparando à Hermes-VC. Tais mecanismos podem ser usados para forçar a rede a respeitar requisitos de QoS, não influenciando significativamente na área final. A área para todas as implementações é dominada pelos *buffers*. É recomendado usar geradores de memória para otimizar a área final. Considerando núcleos reais (com no mínimo 200.000 portas), é esperado uma sobrecarga de área em torno de 10% por núcleo, devido ao roteador.

Tabela 9 – Resultados de área do roteador para a biblioteca de células padrão CMOS 0,35µm.

	Hermes-VC	Hermes-FP	Hermes-DP	Hermes-CS
Número de portas equivalentes	18.657	18.621	21.080	12.792
Freqüência de relógio estimada (MHz)	160	168	147	175

A Tabela 10 apresenta a área do roteador, obtida com a ferramenta de síntese Synplify, objetivando dispositivos FPGA. Os resultados são equivalentes ao mapeamento ASIC, com uma penalidade pequena para a Hermes-DP, e uma área menor para a Hermes-CS.

Tabela 10 – Resultados de área do roteador para o FPGA XC2V1000.

Recursos	Mapeamento para o dispositivo FPGA XC2V1000 da Xilinx								
	Usado				disponível	Usado /Disponível			
	Hermes-VC	Hermes-FP	Hermes-DP	Hermes-CS		Hermes-VC	Hermes-FP	Hermes-DP	Hermes-CS
Slices	1071	1158	1383	967	5.120	20,92%	22,62%	27,01%	18,89%
LUTs	1984	2150	2529	1622	10.240	19,38%	21,00%	24,70%	15,84%
Flip Flops	513	479	646	467	11.212	4,56%	4,27%	5,76%	4,17%

A área do roteador Hermes-RB não foi ainda avaliada, porque a descrição HDL não esta otimizada para síntese. É esperado um pequeno acréscimo na área, porque somente uma tabela pequena e poucos contadores foram adicionados ao roteador Hermes-VC.

9 CONCLUSÃO E TRABALHOS FUTUROS

9.1 Conclusões

Este trabalho avaliou diferentes métodos para prover QoS a redes intra-chip. O mecanismo baseado em prioridades dinâmicas mostrou-se ineficiente para garantir QoS, devido à ausência da alocação de recursos. Mecanismos baseados em prioridades fixas e chaveamento por circuito podem garantir QoS. Entretanto, ambos apresentam limitações, especialmente quando fluxos com requisitos de QoS competem por recursos da rede. Como apresentado no experimento I (Seção 8.1), se nenhum fluxo com mesma prioridade compete por recursos, o mecanismo baseado em prioridades fixas é eficiente. Quando fluxos com a mesma prioridade competem por recursos, o mecanismo baseado em prioridades fixas não provê garantias rígidas a nenhum dos fluxos. É possível observar um *jitter* mínimo quando fluxos CBR são usados, mas a conseqüência é a penalização da latência de um dos fluxos (como apresentado no experimento II). Quando fluxos VBR são usados, latências próximas à ideal aparecem, mas com incremento do *jitter* (experimento III). Uma alternativa para isto, aumentar o número de prioridades, implica em aumentar o número de canais virtuais, o que pode ser proibitivo em termos de área de silício. Quando se usa chaveamento por circuito, todos os requisitos de QoS são garantidos após o estabelecimento da conexão. No entanto, se algum outro fluxo QoS que compartilhe parte do caminho da conexão possuir prazo final para o estabelecimento da conexão, então este mecanismo não será capaz de garantir esse requisito.

O estado da arte em redes intra-chip ainda não apresenta soluções eficientes para prover QoS às aplicações quando o tráfego da rede não é conhecido previamente. O mecanismo baseado em taxas proposto ajusta a prioridade do fluxo de acordo com a taxa requerida e a taxa atualmente usada pelo fluxo. Bons resultados foram obtidos com fluxos CBR, com latências dos fluxos perto dos valores ideais. O mecanismo baseado em taxas supera o problema de fluxos com mesma prioridade competindo por recursos, porque aloca os recursos aos fluxos de acordo com a taxa requerida por eles. Com fluxos VBR, onde pacotes são injetados aleatoriamente dentro da rede, o método proposto é também superior ao escalonamento baseado em prioridades fixas. Entretanto, neste caso, o escalonamento baseado em taxas não alcança a latência ideal quando fluxos QoS competem.

9.2 Trabalhos Futuros

Enumeram-se três trabalhos a serem desenvolvidos em curto prazo: otimização da rede Hermes-RB, avaliação do desempenho das redes desenvolvidas com outros experimentos e prototipação das mesmas.

Reduzir a latência para entregar pacotes QoS no roteador Hermes-RB é um dos primeiros trabalhos que devem ser desenvolvidos. A elevada latência para arbitrar e rotear um pacote QoS no

roteador Hermes-RB prejudica a comparação da Hermes-RB com as demais redes. Atualmente, este roteador consome 2,6 vezes mais tempo para arbitrar e rotear um pacote QoS do que o roteador Hermes-FP.

Outro trabalho que deve ser desenvolvido é a utilização de outros experimentos para avaliar o desempenho das redes, como por exemplo: (*i*) com mais de uma área de disputa por recursos; (*ii*) com mais fluxos disputando os recursos; e (*iii*) utilizando fluxos de aplicações reais de MPSoCs.

Deve-se também otimizar a rede Hermes-RB para síntese, e avaliar as redes desenvolvidas em plataformas de prototipação. O objetivo desta prototipação é duplo: (*i*) validar as descrições; (*ii*) avaliar o desempenho da rede para um elevado números de pacotes, através de emulação de tráfego.

Os mecanismos implementados nesta dissertação melhoram o desempenho da rede, no que se refere à latência, *jitter* e vazão. Entretanto, nenhuma garantia rígida é oferecida aos fluxos (à exceção da Hermes-CS, que provê garantias rígidas a um pequeno número de fluxos). Logo, a médio prazo, deve-se desenvolver serviços em camadas superiores à camada de rede que permitam incluir a especificação dos requisitos do tráfego, e que seja possível a verificação se estes requisitos estão sendo atendidos. Exemplos destes serviços são: estabelecimento do acordo de serviço, condicionamento de tráfego e controle de congestionamento.

A longo prazo, os serviços de QoS devem ser integrados ao *kernel* de um sistema operacional. Realizando-se essa integração, aplicações podem negociar serviços com a rede, e realizar a transferência com garantias de serviço.

REFERÊNCIAS BIBLIOGRÁFICAS

[AND05] Andreasson, D.; Kumar, S. "*Improving BE Traffic QoS Using GT Slack in NoC Systems*". In: 23th IEEE Norchip Conference (NORCHIP'05), 2005, pp. 44-47.

[ART05] Arteris. "*Arteris Network on Chip Company*". Available at http://www.arteris.net. Last access: September 2006.

[BEN01] Benini, L.; De Micheli, G. "*Powering networks on chips: energy-efficient and reliable interconnect design for SoCs*". In: 14th International Symposium on System Synthesis (ISSS'01), 2001, pp. 33-38.

[BEN02] Benini, L.; De Micheli, G. "*Networks on chips: a new SoC paradigm*". Computer, 35(1), 2002, pp. 70-78.

[BER01] Bergamaschi, R.; et al. "*Automating the design of SOCs using cores*". IEEE Design & Test of Computers, 18(5), 2001, pp. 32-45.

[BER05] Bertozzi, D.; Benini, L. "*Xpipes: A Network-on-chip Architecture for Gigascale Systems-on-Chip*". IEEE Circuits and Systems Magazine, 4(2), 2004, pp. 18-31.

[BJE06] Bjerregaard, T.; Mahadevan, S. "*A survey of research and practices of Network-on-chip*". ACM Computing Surveys, 38(1), 2006, pp. 1-51.

[BOL03] Bolotin E. et al. "*QNoC: QoS architecture and design process for network on chip*". Journal of Systems Architecture, 50(2-3), 2004, pp. 105-128.

[BOL04] Bolotin, E. et al. "*Automatic hardware-efficient SoC integration by QoS network on chip*". In: 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS'04), 2004, pp. 479-482.

[CAR04] Carara, E. "*Arquiteturas para Roteadores de Redes Intra-chip*". Trabalho de Conclusão, FACIN-PUCRS, 2004, 62p.

[CLA92] Clark, D.; Shenker, S.; Zhang, L. "*Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*". In: Proceedings of the Conference on Communications Architecture & Protocols (SIGCOMM'92), 1992, pp. 14-26.

[DAL03] Dall'Osso, M.; Biccari, G.; Giovannini, L.; Bertozzi, D.; Benini, L. "*Xpipes: a Latency Insensitive Parameterized Network-on-chip Architecture for Multi-Processor SoCs*". In: 21st International Conference on Computer Design (ICCD'03), Oct. 2003, 536 p.

[DAL04] Dally, W. J.; Towles, B. "*Principles and Practices of Interconnection Networks*". Morgan Kaufmann Publishers, 2004, 550 p.

[DAY83] Day, J. D.; Zimmermman, H. "*The OSI reference model*". Proceedings of IEEE, 71(12), 1983, pp. 1334-1340.

[DUA97] Duato, J.; Yalamanchili, S.; Ni, L "*Interconnection Networks*". IEEE Computer Society Press, 1997, 515 p.

[DUA02] Duato, J.; Yalamanchili, S.; Ni, L. "*Interconnection Networks*". Elsevier Science, 2002, 600 p.

[FLO93] Floyd, S.; Jacobson, V. "*Random Early Detection Gateways for Congestion Avoidance*". IEEE/ACM Transactions on Networking, 1(4), 1993, pp. 397-413.

[GIR98] Giroux, N.; Ganti, S. "*Quality of Service in ATM Networks: State-of-Art Traffic Management*". Prentice Hall, 1998, 252 p.

[GOO02] Goossens, K.; van Meerbergen, J.; Peters, A.; Wielage, P. "*Networks on Silicon: Combining Best-Effort and Guaranteed Services*". In: Design, Automation and Test in Europe (DATE'02), 2002, pp. 423-425.

[GOO03] Goossens, K.; Dielissen, J.; van Meerbergen, J.; Poplavko, P.; Radulescu, A.; Rijpkema, E.; Waterlander, E.; Wielage, P. "*Guaranteeing the Quality of Service in Networks on Chip*". In Nurmi, J.; Tenhunen, H.; Isoaho, J.; Jantsch, A., editors, Networks on Chip, Kluwer Academic Publishers, 2003, pp. 61-82.

[GOO05] Goossens, K.; Dielissen, J.; Radulescu, A. "*Æthereal Network on Chip: Concepts, Architectures, and Implementations*". IEEE Design and Test of Computers, 22(5), 2005, pp. 414-421.

[GUE99] Guerrier. P.; Greiner. A. "*A Scalable Architecture for System-On-Chip Interconnections*". In: Sophia Antipolis Forum on MicroElectronics (SAME'99), 1999, pp. 1-4.

[GUE00] Guerrier. P.; Greiner. A. "*A generic architecture for on-chip packet-switched interconnections*". In: Design, Automation and Test in Europe (DATE'00), 2000, pp. 250-256.

[GUP97] Gupta, R.; Zorian, Y. "*Introducing Core-Based System Design*". IEEE Design & Test of Computers, 14(4), 1997, pp. 15-25.

[HAR05] Harmanci, M. D.; Escudero, N. P.; Leblebici, Y.; Ienne, P. "*Quantitative Modelling and Comparison of Communication Schemes to Guarantee Quality-of-Service in Networks-on-Chip*". In: IEEE International Symposium on Circuits and Systems (ISCAS'05), 2005, pp. 1782-1785.

[HWA93] Hwang, K. "*Advanced Computer Architecture: Parallelism, Scalability, Programmability*". New York: McGraw-Hill, 1993, pp. 213-256.

[INT02] International Sematech. "*International Technology Roadmap for Semiconductors – 2002*". Available at http://public.itrs.net. Last Access: April 2006.

[KAR02] Karim, F.; Nguyen, A.; Dey S. "*An interconnect architecture for network systems on chips*". IEEE Micro, 22(5), 2002, pp. 36-45.

[KUM02] Kumar, S.; Jantsch, A.; Soininen, J. P.; Fonsell, M. "*A Network on Chip Architecture and Design Methodology*". In: IEEE Computer Society Annual Symposium on VLSI. (ISVLSI'02), 2002. pp. 117-123.

[KUM04] Kumar, A.; Manjunath, D.; Kuri, J. "*Communication Networking: An Analytical Approach*". Morgan Kaufman Publishers, 2004, 929 p.

[KUM05] Kumar, S.; Andreasson, D. "*Slack-Time Aware Routing in NoC Systems*". In: IEEE International Symposium on Circuits and Systems (ISCAS'05), 2005, pp. 2353-2356.

[LEE95] Lee, J. W.; Kim, C. K.; Lee, C. W. "*Rate-based scheduling discipline for packet switching networks*". IEE Electronics Letters, 31(14), 1995, pp. 1130-1131.

[LIA00] Liang, J.; Swaminathan, S.; Tessier, R. "*aSOC: A Scalable, Single-Chip communications Architecture*". In: IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT'00), 2000, pp. 37-46.

[MAR01] Martin, G.; Chang, H. "*System on Chip Design*". In: 9th International Symposium on Integrated Circuits, Devices & Systems (ISIC'01), Tutorial 2, 2001. pp. 12-17.

[MCC03] McCabe, J. "*Network Analysis, Architecture, and Design*". 2ª Edição, Morgan Kauffmann Publishers, 2003, 501 p.

[MEL05] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "*Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC*". In: 18th Symposium on Integrated Circuits and Systems (SBCCI'05), 2005, pp. 178-183.

[MIL04] Millberg, M.; Nilsson, E.; Thid, R.; Jantsch, A. "*Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the NOSTRUM Network on Chip*". In: Design, Automation and Test in Europe (DATE'04), 2004, pp. 890-895.

[MOH98] Mohapatra, P. "*Wormhole Routing Techniques for Directly Connected Multicomputer Systems*". ACM Computing Surveys, 30(3), 1998, pp. 374-410.

[MOR04] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. "*Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip*". Integration the VLSI Journal, 38(1), 2004, pp. 69-93.

[MUL04] Mullins, R.; West, A.; Moore, S. "*Low-Latency Virtual-Channel Routers for On-Chip Networks*". In: 31st Annual International Symposium on Computer Architecture (ISCA'04), 2004, pp. 188-197.

[PAN05] Pande, P.; Grecu, C.; Jones, M.; Ivanov, A.; Saleh, R. "*Performance Evaluation and design Trade-Offs for Network-on-Chip Interconnect Architectures*". IEEE Transactions on Computers, 54 (8), 2005, pp. 1025-1040.

[PAT96] Patterson, D.; Hennessy, J. L. "*Computer Architecture: A Quantitative Approach*". Morgan Kaufmann Publishers, 1996, 760 p.

[PET03] Peterson, L.; Davie, B. "*Computer Networks: A System Approach*". 3ª Edição, Morgan Kauffmann Publishers, 2003, 813 p.

[REX97] Rexford, J.; Greenberg, A.; Wong. A. "*Scalable Architectures for Integrated Traffic Shaping and Link Scheduling in High-Speed ATM Networks*". In: IEEE Journal on Selected Areas in Communications, 15 (5), 1997, pp. 938-950.

[RIJ01] Rijpkema, E.; et al. "*A Router Architecture for Networks on Silicon*". In: 2nd Workshop PROGRESS on Embedded Systems (PROGRESS'01), Netherlands, 2001, pp. 1-8.

[RIJ03] Rijpkema, E.; Goossens, K.; Rădulescu, A. "*Trade Offs in the Design of a Router with both Guaranteed and Best-Effort Services for Networks on Chip*". In: Design, Automation and Test in Europe (DATE'03), 2003, pp. 350-355.

[SAN03] Santos, F.; Zeferino, C.; A. Susin. "*Modelos Parametrizáveis de Árbitros Centralizados para a Síntese de Redes-em-Chip*". Hífen, 2003, pp. 91-96.

[SAR00] Sarbazi-Azad, H; Mackenzie; L. M.; Ould-Khaoua, M. "*The Effect of the Number of Virtual Channels on the Performance of Wormhole-Routed Mesh Interconnection*

Networks". In: 16th Annual UK Performance Engineering Workshop (UKPEW'00), 2000, pp. 95-102.

[SHI03] Shin, J.; Lee, D.; Kuo, C.-C. "*Quality of Service for Internet Multimedia*". Prentice Hall, 2003, 204 p.

[TED06] Tedesco, L.; Mello, A.; Giacomet, L.; Calazans, N.; Moraes, F. "*Application Driven Traffic Modeling for NoCs*". In: 19th Symposium on Integrated Circuits and Systems (SBCCI'06), 2006, pp. 62-67.

[VES05] Véstias, M.; Neto, H. "*A Reconfigurable SoC Platform Based on a Network on Chip Architecture with QoS*". In: XX Conference on Design of Circuits and Integrated Systems (DCIS'05), 2005, 6 p.

[WIK03] Wiklund, D.; Liu D. "*SoCBUS: Switched Network on Chip for Hard Real Time Systems*". In: 17th International Parallel and Distributed Processing Symposium (IPDPS'03), 2003, 8p.

[YUM02] Yum, K. H.; Kim, E. J.; Das, C. R.;Yousif, M.; Duato, J. "*Integrated Admission and Congestion Control for QoS Support in Clusters*". In: Proceedings of IEEE International Conference on Cluster Computing (CLUSTER'02), 2002, pp. 325-332.

[ZHA91] Zhang, L. "*Virtual Clock: A new traffic control algorithm for packet-switched networks*". ACM Transactions on Computer Systems, 9(2), 1991, pp. 101-124.

[ZHA95] Zhang, H. "*Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks*". In: Proceedings of the IEEE, 83 (10), 1995, pp. 1374-1996.

APÊNDICE I – PUBLICAÇÕES ACEITAS E SUBMETIDAS

As publicações realizadas no período do mestrado demonstram a aceitação do presente trabalho na comunidade científica, nacional e internacional. As publicações aceitas e submetidas são listadas abaixo.

Ost, L.; Mello, A.; Palma, J.; Calazans, N.; Moraes, F. "*MAIA - A Framework for Networks on Chip Generation and Verification*". In: Asia South Pacific Design Automation Conference (ASP-DAC'05), 2005, pp. 18-21.

Carara, E.; Mello, A.; Calazans, N.; Moraes, F. "*Canais Virtuais em Redes Intra-Chip - Implementação na rede HERMES*". In: XI Workshop IBERCHIP (IBERCHIP'05), 2005, pp. 320-321.

Mello, A.; Möller, L.; Calazans, N.; Moraes, F. "*MultiNoC: A Multiprocessing System Enabled by a Network on Chip*". In: Design, Automation and Test in Europe (DATE'05), 2005, pp. 234-239.

Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F.; "*Virtual Channels in Networks on Chip: Inplementantion and Evaluation on Hermes NoC*". In: 18th Symposium on Integrated Circuits and Systems (SBCCI'05), 2005, pp. 178-183.

Tedesco, L.; Mello, A.; Calazans, N.; Moraes, F. "*Traffic Generation and Performance Evaluation for Mesh-based NoCs*". In: 18th Symposium on Integrated Circuits and Systems (SBCCI'05), 2005, pp. 184-189.

Mello, A.; Möller, L.; Moraes, F. "*Redes Intra-chip: Projeto da Rede Hermes*". In: Ricardo Reis; Sandro Sawicki; Rafael Santos. (Org.). Advanced Topics on Microelectronics. 1 ed. Porto Alegre: Doravante Publishers, 2006, pp. 121-144.

Tedesco, L.; Mello, A.; Calazans, N.; Moraes, F. "*Application Driven Traffic Modeling for NoCs*". In: 19th Symposium on Integrated Circuits and Systems (SBCCI'06), 2006, pp. 62-67.

Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "*Evaluation of Current QoS Mechanisms in Networks on Chip*". In: International Symposium on System-on-Chip 2006 (SoC'06), 2006, pp. 115-118.

Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "*Quality of Service in Networks-on-Chip Beyond Priority and Circuit Switching Techniques*". Integration the VLSI Journal, major revision submitted in November 3rd, 2006.

APÊNDICE II – AMBIENTE ATLAS

O ambiente Atlas surge da necessidade de integrar ferramentas onde são automatizados os vários processos relacionados ao fluxo de projeto da rede Hermes: geração da rede, geração de tráfego, simulação e avaliação de desempenho. Inicialmente a rede é gerada, sendo configurados parâmetros da construção da rede, como largura dos canais, profundidade de *buffers*, utilização de canais virtuais, estratégias de controle de fluxo e adoção de mecanismos de QoS. Cenários de tráfego são gerados para caracterizar aplicações que executarão sobre a rede. Na simulação, os dados de tráfego são injetados na rede, ocorrendo, nesta etapa, a efetiva comunicação entre os núcleos. Após a simulação, é possível a avaliação de desempenho das comunicações ocorridas, onde são detectados, por exemplo, pontos da rede com congestionamento e caminhos críticos para passagem de dados. São gerados gráficos, tabelas, mapas e relatórios que auxiliam na análise dos resultados obtidos.

A Figura 81 ilustra a interface principal do ambiente Atlas, a qual permite invocar as ferramentas que executam todas as etapas do fluxo de projeto da rede.

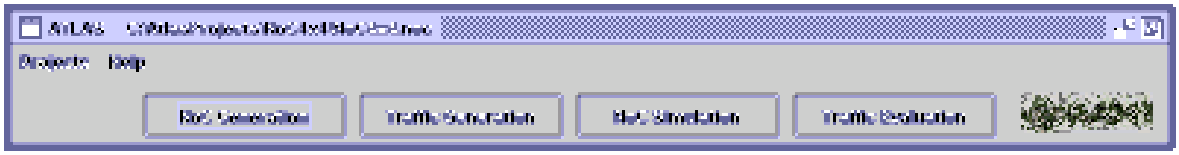
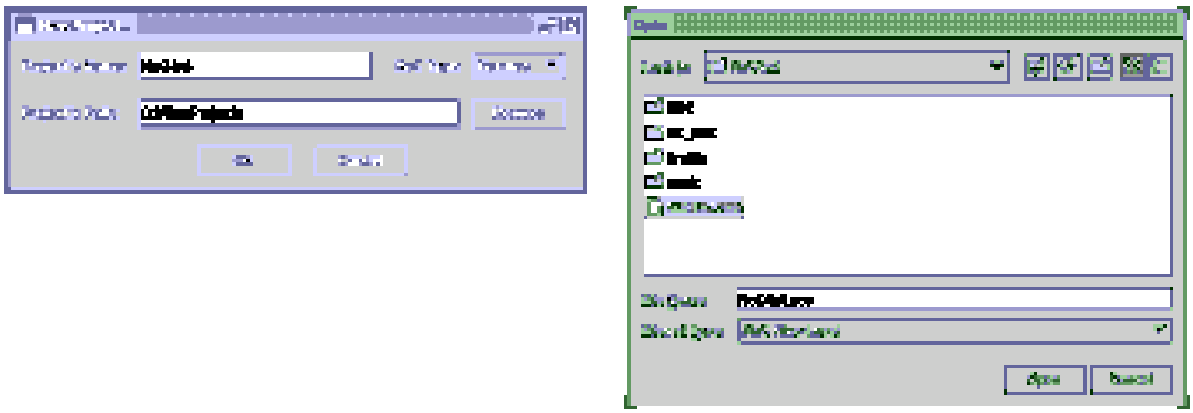


Figura 81 – Interface principal do ambiente Atlas.

O primeiro passo no ambiente Atlas é criar ou abrir um projeto de rede. Sem este passo, as ferramentas do ambiente não podem ser acessadas pelo usuário. Quando o usuário opta por criar um novo projeto, a interface ilustrada na Figura 82(a) é apresentada. Nesta interface, o usuário deve informar o nome do novo projeto, informar o tipo da rede, selecionar o caminho onde o diretório do projeto será criado e clicar sobre o botão *Ok*. Quando o usuário opta por abrir um projeto de rede pré-existente, a interface ilustrada na Figura 82(b) é apresentada. Nesta interface, o usuário deve selecionar o projeto e clicar sobre o botão *Open*.



(a) Criar **(b) Abrir**

Figura 82 – Interfaces para criar e abrir um projeto de rede.

Após a criação ou abertura de um projeto de rede, o usuário pode acessar as ferramentas que compõem o ambiente Atlas. São elas:

- ***NoC Generation*** : gera a rede a partir da configuração dos seguintes parâmetros: controle de fluxo (*handshake* ou *credit based*); número de canais virtuais; tipo de escalonamento (*Round Robin* ou *Priority*); dimensão da rede; largura do canal de comunicação; profundidade do *buffer*; algoritmo de roteamento; e testbench (descrito em SystemC). Ao solicitar a geração da rede, a ferramenta gera todos módulos que compõem os roteadores e realiza a interconexão destes. A rede é descrita em VHDL.
- ***Traffic Generation***: gera o tráfego que será transmitido através da rede. Esta etapa gera para cada roteador um arquivo com a descrição do fluxo que este deve inserir na rede.
- ***NoC Simulation***: invoca um simulador VHDL externo (ModelSim®), utilizando a descrição da rede e os testbenchs (gerados pela ferramenta *NoC Generation*), e o cenário de tráfego (gerado pela ferramenta *Traffic Generation*). Os resultados da simulação são gravados em arquivos.
- ***Traffic Evaluation***: captura os dados dos arquivos gerados na fase de simulação e permite a análise de várias métricas fundamentais na entrega de mensagens, como vazão e latência. A ferramenta possibilita gerar vários tipos de gráficos, tabelas e resumos para uma detalhada avaliação de desempenho.

A ferramenta *NoC Generation* é detalhada na Seção 9.3. A Seção 9.4 detalha a ferramenta *Traffic Generation*. A ferramenta *NoC Simulation* é detalhada na Seção 9.5 e a ferramenta *Traffic Evaluation* na Seção 9.6.

9.3 NoC Generation

A rede Hermes é uma infra-estrutura para a geração de redes intra-chip. Diz-se infra-estrutura porque não existe uma única rede intra-chip. Existe um conjunto de módulos, como árbitro, *buffers*, portas de entrada/saída, todos parametrizáveis em função das restrições do projeto. A parametrização dos roteadores e a geração manual da rede a partir destes é um processo passível de erros, devido à grande quantidade de fios que ligam os roteadores entre si e ao núcleo local, como ilustra a Figura 83. A automatização desse processo foi a principal motivação para o desenvolvimento da ferramenta *NoC Generation*.

Além de automatizar o processo de interconexão dos roteadores, a ferramenta permite a configuração de parâmetros como: (*i*) largura dos canais de dados; (*ii*) profundidade das *buffers* de entrada dos roteadores e (*iii*) dimensão da rede. Pretende-se, a partir da configuração desses parâmetros pré-definidos, gerar redes intra-chip que respeitem os requisitos de uma aplicação especifica. Outra possibilidade é utilizar a ferramenta para gerar diversas configurações, possibilitando a análise de qual configuração tem melhor desempenho para uma dada aplicação.

A ferramenta *NoC Generation* também otimiza o roteador através da remoção de *buffers* de entrada de canais não utilizados, o que reduz a área da Hermes em hardware.

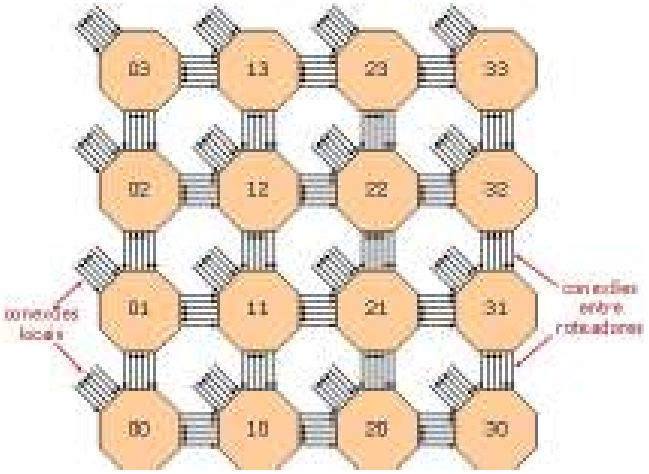


Figura 83 – Interconexões entre roteadores em uma rede intra-chip 4x4.

A interface da ferramenta *NoC Generation* é apresentada na Figura 84. Esta interface é dividida em três blocos básicos: (1) barra de parametrização, (2) área de visualização e (3) geração da rede.

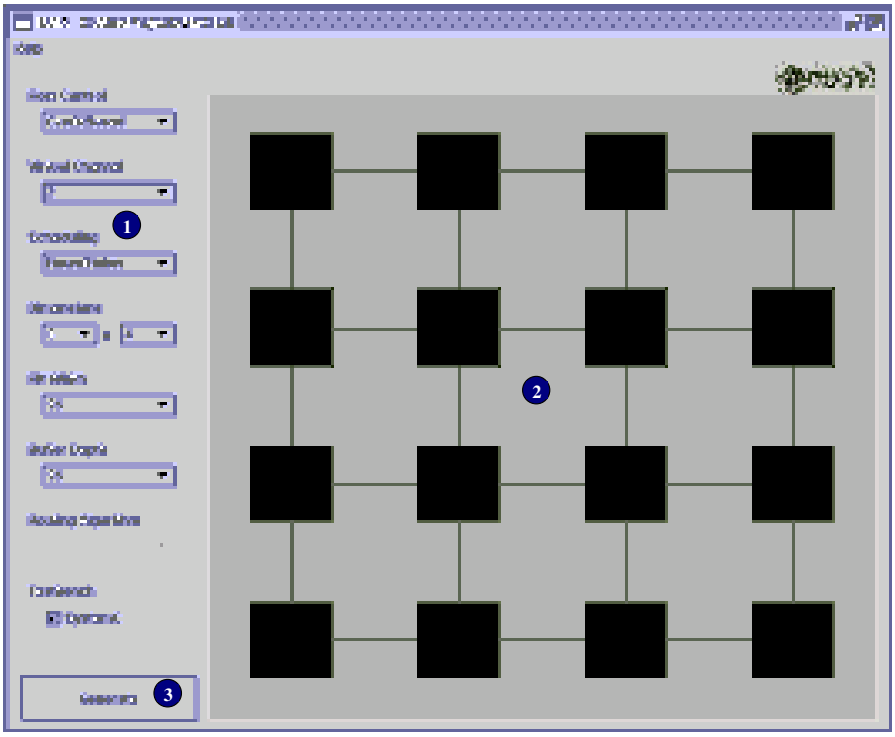


Figura 84 – Interface da ferramenta *NoC Generation*.

O usuário parametriza a rede a partir dos campos contidos na *barra de parametrização* (índice 1 na Figura 84). A partir desses campos é possível configurar: (*i*) o controle de fluxo; (*ii*) o número de canais virtuais; (*iii*) o tipo de escalonamento; (*iv*) a dimensão da rede; (*v*) a largura do canal de comunicação; (*vi*) profundidade dos *buffers* de entrada dos roteadores; (*vii*) o algoritmo de roteamento; e (*vi*) o testbench. A Figura 85 ilustra o local da arquitetura do roteador que é modificado de acordo com a configuração do parâmetro tipo de escalonamento (*Scheduling*).

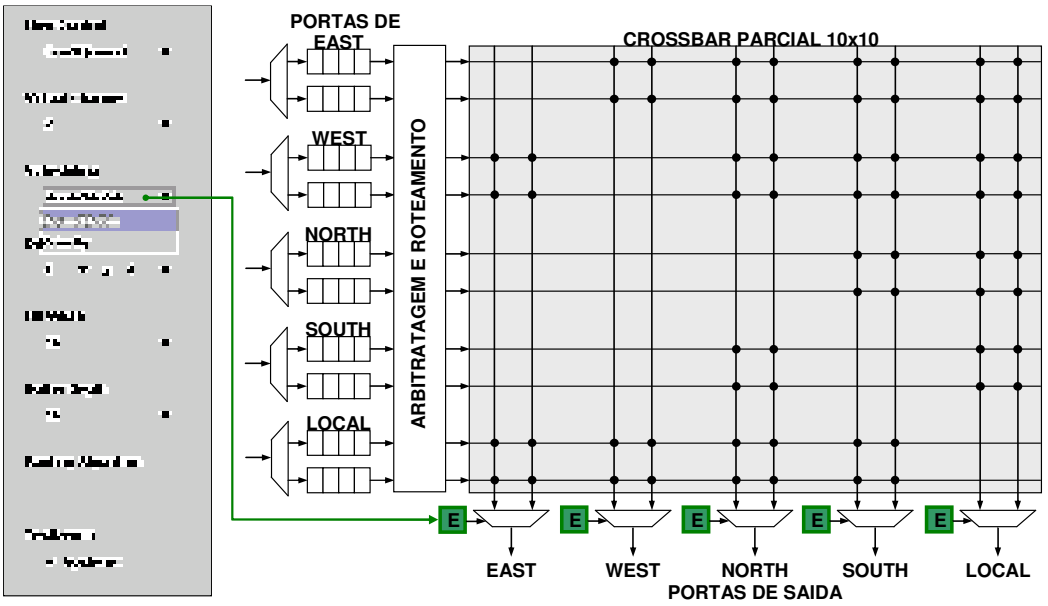


Figura 85 – Influência do parâmetro *tipo de escalonamento* na arquitetura roteador.

A *área de visualização* (índice 2 na Figura 84) desenha a rede de acordo com os valores configurados pelo usuário na barra de parametrização. Com isso, é possível visualizar a dimensão da rede, os endereços e as conexões dos roteadores. Na Figura 84, a área de visualização apresenta uma rede malha com dimensão 4x4.

A rede é gerada quando o usuário clica sobre o botão *Generate* (índice 3 na Figura 84). A estrutura de diretórios apresentada na Figura 86 é construída durante o processo de geração da rede. Nesta estrutura:

- O diretório *NoC* contém os arquivos VHDL que descrevem a rede;
- O diretório *SC_NoC* contém os testbenchs descritos em SystemC;
- O arquivo *NoC4x4.noc* descreve a rede e é usado pelas ferramentas do ambiente Atlas;
- O arquivo *simulate* é um script para simulação no simulador ModelSim®;
- O arquivo *topNoC* é um arquivo VHDL que integra a rede aos testbenchs.

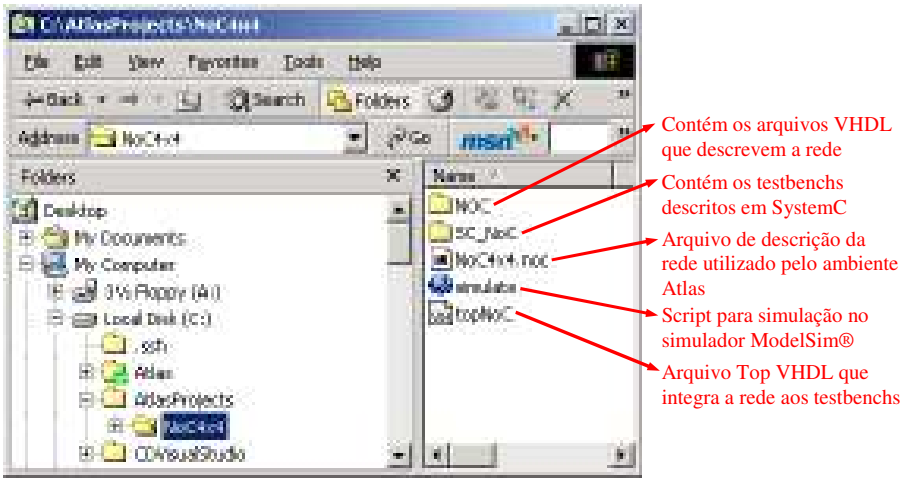


Figura 86 – Estrutura de diretórios.

9.4 Traffic Generation

A ferramenta *Traffic Generation* gera os arquivos de tráfego (descrevendo um comportamento) que serão transmitidos através da rede durante a simulação. A interface principal desta ferramenta é apresentada na Figura 87. Esta interface é dividida em três blocos: (1) barra de menus; (2) área de visualização; e (3) geração do tráfego da rede.

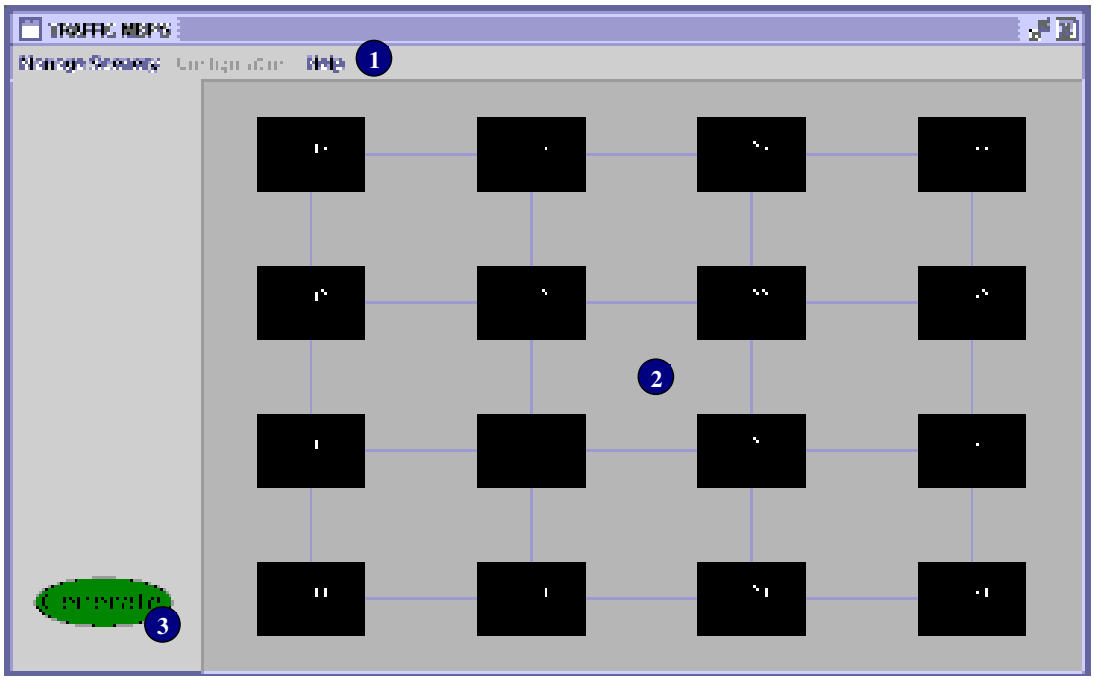
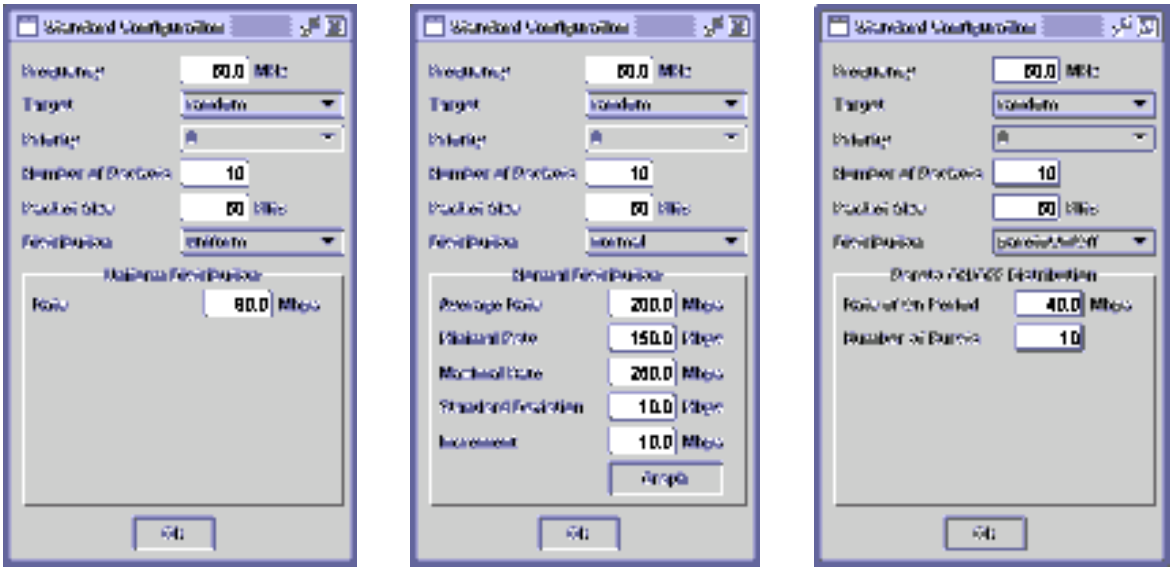


Figura 87 – Interface principal da ferramenta *Traffic Generation*.

A barra de menus possui três menus: *Manage Scenery*, *Configuration* e *Help*. O menu *Manage Scenery* permite o usuário criar ou abrir um cenário de tráfego. O menu *Configuration* permite o usuário definir a configuração padrão dos parâmetros utilizados pelas interfaces de rede durante a geração do tráfego. Esta configuração é chamada de padrão porque é atribuída a todas interfaces dos roteadores da rede. O menu *Help* contém a documentação detalhada da ferramenta *Traffic Generation*.

A Figura 88 apresenta os três possíveis formatos de interface para a escolha da configuração padrão. Os seis parâmetros apresentados na parte superior de todas as interfaces são independentes. Enquanto, os parâmetros apresentados na parte inferior das interfaces dependem da escolha do tipo de distribuição (uniforme, normal ou Pareto on/off). Em versões futuras da Atlas, é provável que outras distribuições de tráfego estejam disponíveis além destas três.

A configuração de cada roteador pode ser alterada individualmente através da seleção do roteador na área de visualização (índice 2 na Figura 87). Para selecionar um roteador, o usuário deve clicar sobre o desenho do roteador desejado. Por exemplo, na Figura 89, o usuário clicou sobre o roteador com endereço 22. Os parâmetros de tráfego de um roteador individual são inicialmente idênticos aos da configuração padrão (Figura 88).



(a) **(b)** **(c)**

Figura 88 – Interface apresentada para a configuração padrão dos parâmetros para (a) tráfego uniforme, (b) tráfego normal e (c) tráfego pareto on/off.

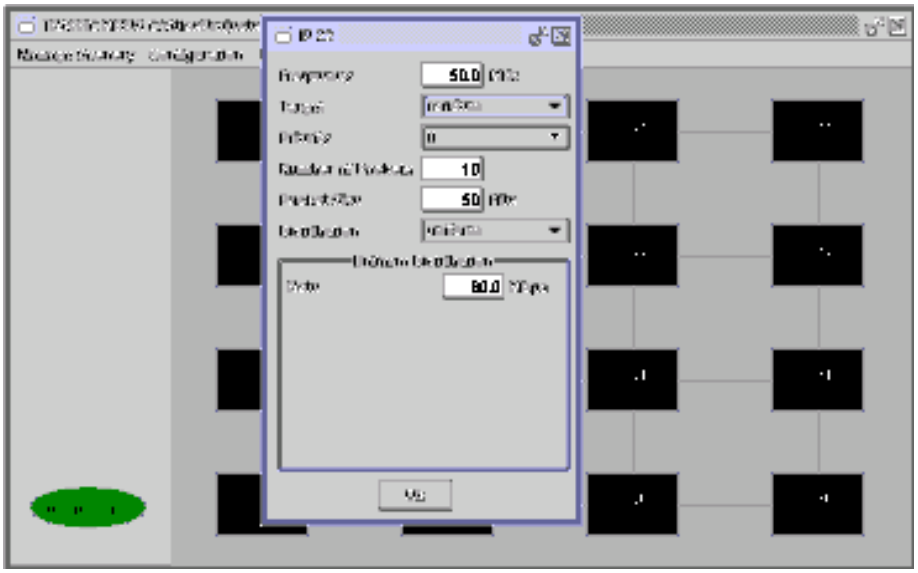


Figura 89 – Interface para a configuração individual dos parâmetros de tráfego do roteador 22.

Uma vez que pelo menos um roteador da rede tenha sido configurado para gerar tráfego para a simulação, o usuário pode clicar sobre o botão *Generate* (índice 3 na Figura 87). Durante a geração dos arquivos de tráfego, a árvore de subdiretórios apresentada na Figura 90 é construída. Nesta estrutura:

- O diretório *Traffic* contém todos os cenários de tráfego do projeto;
- O diretório *TrafficComplement* contém todos os arquivos do cenário de tráfego. O nome do diretório corresponde ao nome do cenário de tráfego;
- O arquivo *TrafficComplement.traffic* descreve o cenário de tráfego. Este arquivo é usado nas fases de simulação e avaliação;

- O diretório *In* contém os arquivos de entrada da simulação os quais são gerados pela ferramenta *Traffic Generation*;
- O diretório *Out* é criado para conter os arquivos de saída gerados depois do passo de simulação.

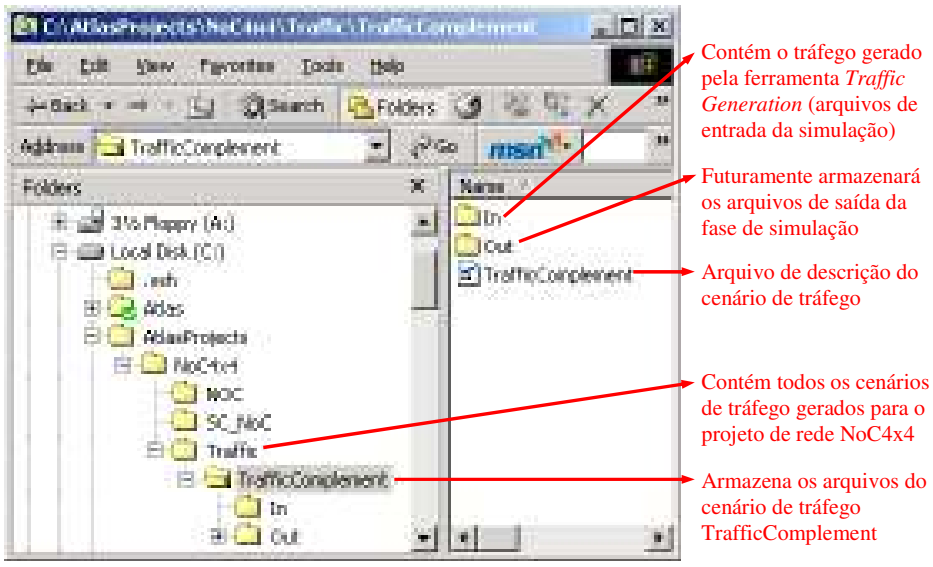


Figura 90 – Árvore de subdiretórios criada após a geração do tráfego.

9.5 NoC Simulation

A fase de simulação verifica o correto funcionamento da rede gerada. No ambiente Atlas, a simulação da rede é executada usando o simulador ModelSim®. Se o usuário não tem acesso a este simulador ou se ele deseja utilizar outro simulador, a fase de simulação não deve ser executada dentro do ambiente Atlas. Tudo que é necessário é um ambiente de simulação que aceite VHDL e SystemC. Ajuda se o simulador aceitar scripts Tcl, assim os scripts gerados automaticamente dentro da Atlas podem também ser utilizados. Aceitar outros simuladores ou generalizar os scripts de simulação é considerado pela equipe de desenvolvimento como trabalho futuro.

A interface usada para controlar a simulação é apresentada na Figura 91. Quatro parâmetros devem ser configurados na interface da ferramenta *NoC Simulation*:

- ***Scenery's name***: indica o nome do cenário a ser simulado.
- ***Modelsim's path***: indica o caminho do diretório onde o simulador *ModelSim*® foi instalado. Este diretório deve conter o subdiretório "*win32*", o qual possui o arquivo "*modelsim.exe*".
- ***Simulation Time***: indica o tempo de simulação para a rede (um número inteiro em milisegundos).
- ***Internal Evaluation***: indica se os resultados da simulação devem gerar dados para permitir a avaliação *interna* da rede. A avaliação *interna* permite a análise dos dados referentes às portas e canais internos da rede (aqueles conectando algum par de roteadores), enquanto a avaliação *externa* consiste na análise somente do comportamento das portas locais da rede.

Figura 91 – Interface da ferramenta NoC Simulation.

A simulação começa quando o usuário clica sobre o botão *Ok* (Figura 91). A *NoC Simulation* utiliza scripts para manipular os arquivos do cenário de tráfego, invocar o simulador *ModelSim*®, compilar e simular a rede. A tela exibida na Figura 92 é apresentada depois do início da simulação. Se tudo for bem, no fim da simulação esta tela é fechada e os arquivos de saída da simulação estão no diretório Out do cenário de tráfego, como ilustrado na Figura 93.

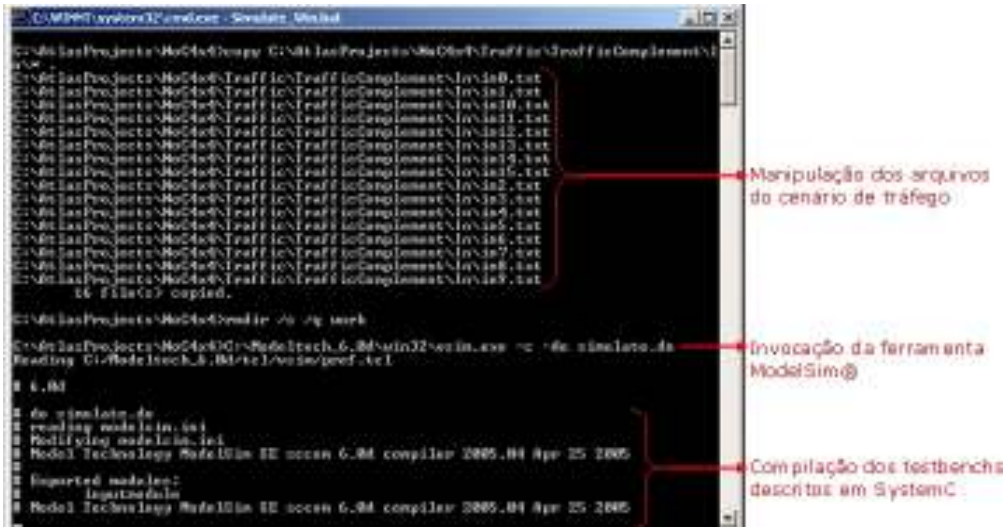


Figura 92 – Tela mostrando uma visão parcial da execução do script de simulação.

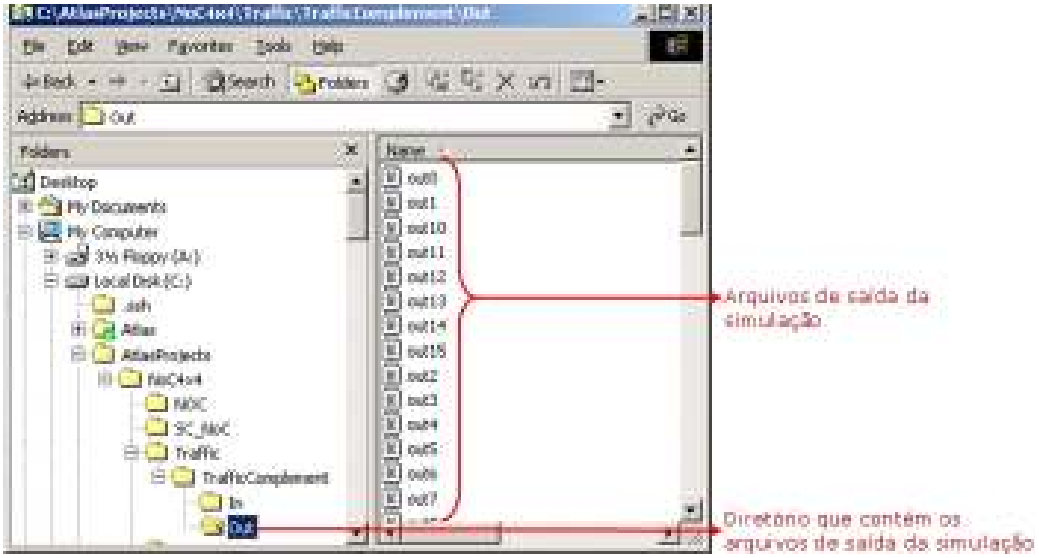


Figura 93 – Conteúdo do diretório Out depois da simulação, mostrando os arquivos gerados.

Observação: a função interna de SystemC no ModelSim® mostrou freqüentemente ser irregular. Ás vezes, em algumas máquinas, durante a compilação de arquivos SystemC chamados de dentro dos scripts do Atlas, o ModelSim® simplesmente pendura. Veja as linhas na Figura 92 contendo a string **sccom**. Estas correspondem a chamada da versão interna do compilador **gcc** no ModelSim®. Se o ModelSim® pendurar, abortar o processo inteiro e começar a simulação novamente usualmente resolve o problema. Depois de compilar todos os arquivos SystemC, o script começa a compilar os arquivos VHDL da rede. Abaixo tem um exemplo desta transação onde chamadas a **sccom** são seguidas por chamadas a **vcom**, o compilador VHDL do ModelSim®. Se este passo é alcançado, a simulação vai sempre até o final. Note também que dependendo do tempo de simulação e o tamanho da rede, este passo pode ser muito longo, de muitos minutos a horas ou mesmo dias. A Memória do computador pode também ser um problema aqui. No mínimo 1Gbyte de memória deve estar disponível para simulações longas.

```
# Exported modules:
# outmodulerouter
# Model Technology ModelSim SE sccom 6.2b compiler 2006.07 Jul 31 2006
# Model Technology ModelSim SE vcom 6.2b Compiler 2006.07 Jul 31 2006
# -- Loading package standard
```

9.6 Traffic Evaluation

A ferramenta *Traffic Evaluation* foi desenvolvida com o objetivo de facilitar a análise dos resultados gerados durante a simulação da rede. A primeira etapa na execução da ferramenta é a seleção do cenário de tráfego a ser analisado, conforme apresentado na Figura 94. O usuário seleciona o cenário de tráfego ao clicar sobre o botão *Open*.

Figura 94 – Interface para seleção do cenário de tráfego a ser avaliado.

Após a seleção do cenário de tráfego, a interface ilustrada na Figura 95 é apresentada. Esta interface é dividida em três blocos básicos: (1) barra de menus; (2) barra de parametrização; e (3) área de visualização. A *barra de menus* (índice 1 na Figura 95) permite o usuário selecionar um cenário de tráfego para ser analisado, gerar relatórios que consideram todo o cenário de tráfego e obter ajuda sobre a ferramenta *Traffic Evaluation*. A *barra de parametrização* (índice 2 na Figura 95) permite a avaliação de fluxos individuais e avaliação de parâmetros internos da rede. A *área de visualização* (índice 3 na Figura 95) permite identificar a dimensão da rede, os endereços e as conexões dos roteadores. Na Figura 95, a área de visualização apresenta uma rede malha com dimensão 4x4.

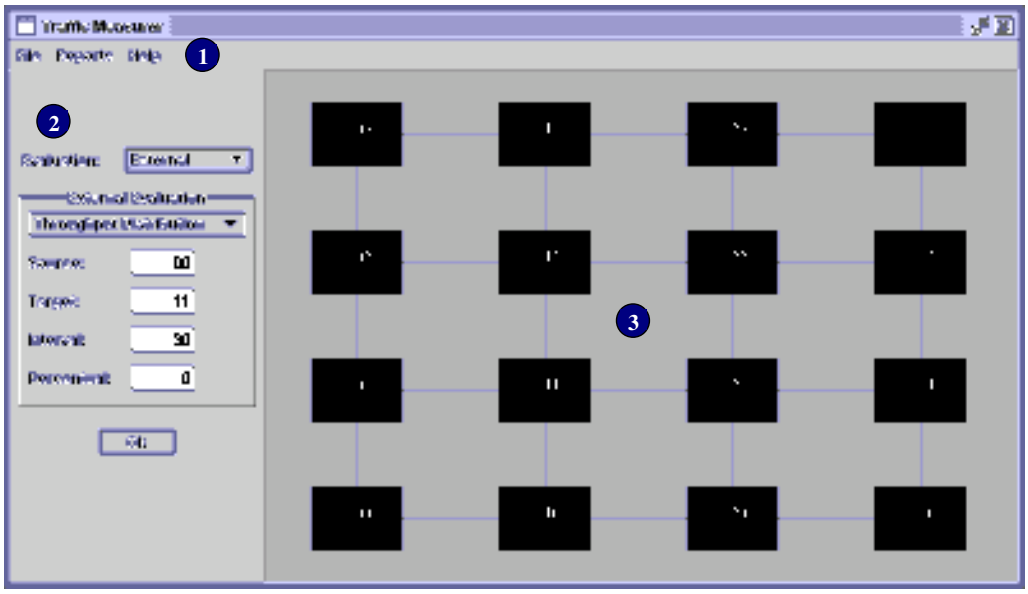


Figura 95 – Interface principal da ferramenta *Traffic Evaluation*.

Três relatórios podem ser gerados considerando o cenário de tráfego completo:

1. ***Links Analysis Report***: exibe um relatório contendo informações (por exemplo, número de pacotes transmitidos, número de bits por ciclo de relógio) sobre todos os canais da rede. A interface de apresentação deste relatório é exibida na Figura 96. Nesta figura, podem ser observadas as informações sobre os canais leste e norte do roteador 00. ***Observação***: o relatório contendo as informações sobre os canais da rede faz parte da avaliação interna. Portanto, o subitem *Links Analysis Report* somente é disponibilizado quando a opção *Internal Evaluation* é selecionada na fase de simulação.

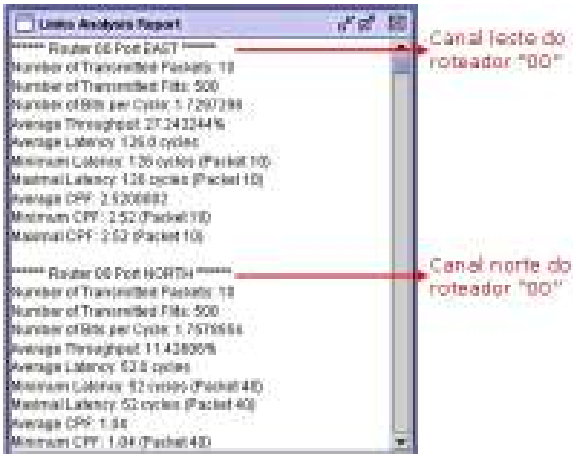


Figura 96 – Relatório da análise dos canais da rede.

2. ***Latency Analysis Report***: exibe uma lista ordenada dos pacotes com maior ou menor latência. O relatório de latência é dividido em dois blocos: tabela (índice 1 da Figura 97) e área de visualização (índice 2 da Figura 97). A *tabela* informa a latência, o número de seqüência e os núcleos origem e destino dos pacotes que compõem a lista. A *área de visualização* possibilita ao usuário verificar o caminho percorrido pelo pacote selecionado na tabela. Para selecionar um

pacote, o usuário deve clicar sobre a linha da tabela que contém as informações sobre mesmo.

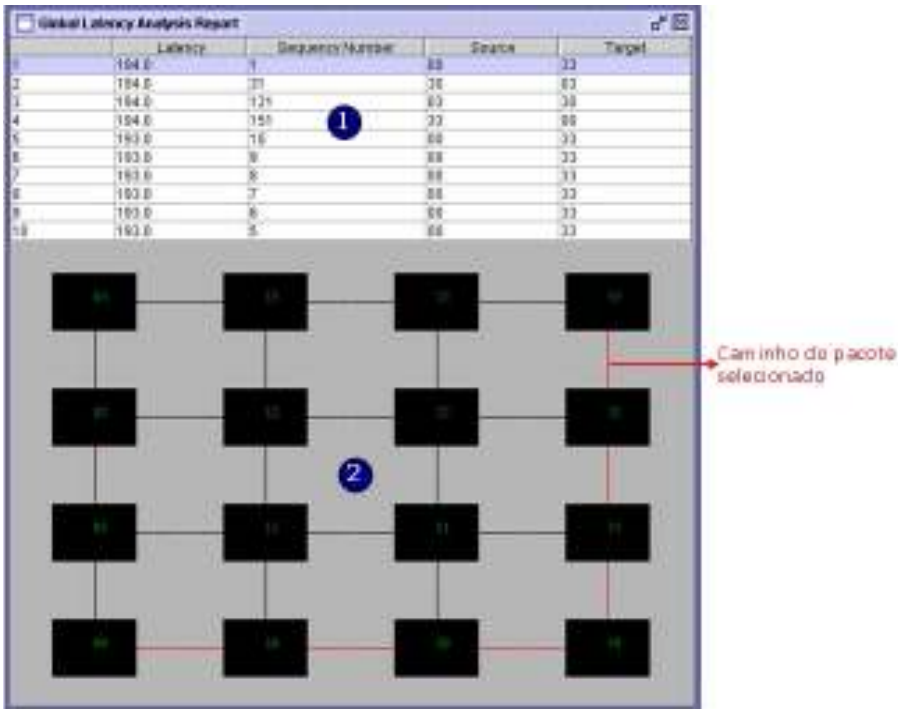


Figura 97 – Relatório de pacotes ordenados pela latência.

3. ***Global Report***: exibe um resumo da avaliação externa. Este resumo é apresentado no formato html, conforme apresentado na Figura 98. Os resultados da avaliação externa são agrupados em fluxos (conjunto de pacotes que possuem o mesmo núcleo origem e destino). O relatório global é dividido nas seguintes partes:
   - ***Informações sobre a rede*** (índice 1 da Figura 98): conjunto de informações obtidas a partir do arquivo gerado pela ferramenta *NoC Generation*.
   - ***Identificação dos fluxos*** (índice 2 da Figura 98): endereço dos núcleos origem e destino de cada fluxo. No relatório global, cada linha corresponde a um fluxo distinto.
   - ***Dados gerados*** (índice 3 da Figura 98): conjunto de informações obtidas a partir dos arquivos gerados pela ferramenta *Traffic Generation*, que tem o objetivo de servir como referência aos resultados obtidos durante a simulação.
   - ***Dados coletados*** (índice 4 da Figura 98): conjunto de informações obtidas a partir da análise dos arquivos gerados durante a simulação. Estas informações possibilitam ao usuário verificar, por exemplo: (*i*) o correto funcionamento da rede, ou seja, se todos os pacotes gerados foram entregues corretamente; (*ii*) se a vazão obtida no núcleo destino corresponde a taxa de inserção dos pacotes; (*iii*) se a latência dos pacotes está próxima à latência ideal, ou seja, se a rede está ou não congestionada.
   - ***Gráficos*** (índice 5 da Figura 98): permite a visualização de dois gráficos: (*i*) gráfico da distribuição de latência dos pacotes e (*ii*) gráfico da distribuição de vazão dos pacotes.

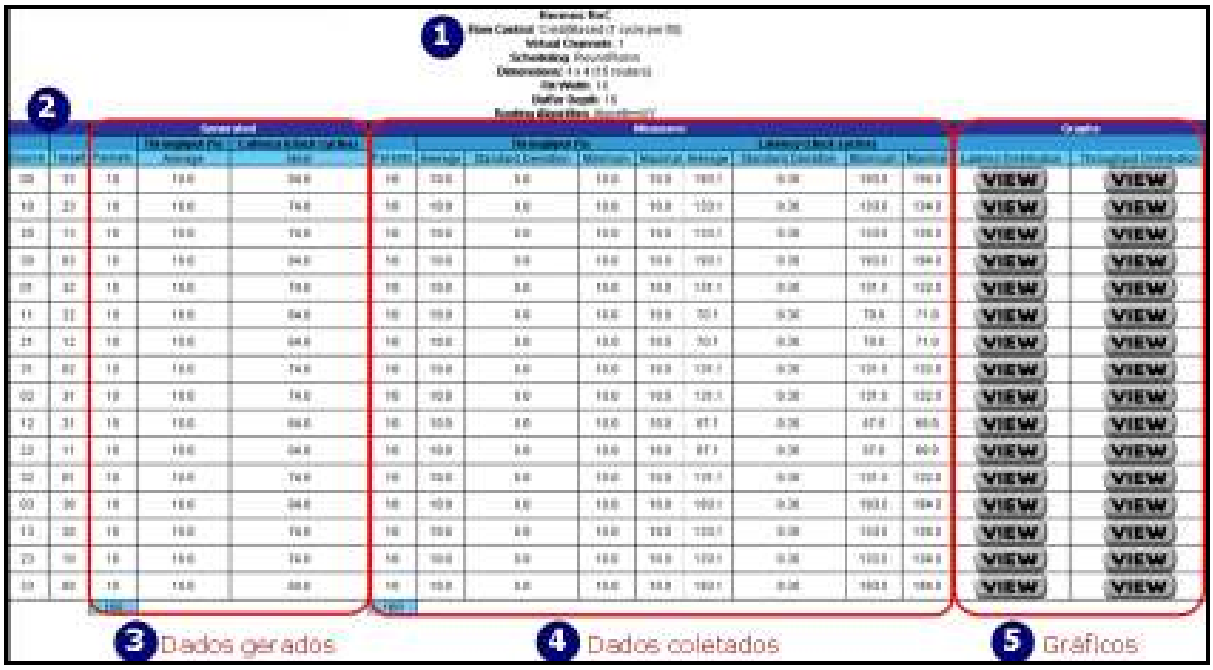


Figura 98 – Relatório global.

Além dos relatórios que permitem a avaliação de todo o cenário de tráfego, o usuário pode avaliar fluxos individuais através da configuração dos campos contidos na *barra de parametrização* (índice 2 na Figura 95). O parâmetro principal a ser configurado é o tipo de avaliação (*Evaluation*). Este parâmetro é dito principal porque a configuração dos outros parâmetros depende dele. Existem dois tipos de avaliação: *avaliação externa* e *avaliação interna*.

Avaliação Externa

A avaliação externa analisa arquivos referentes às portas locais dos roteadores. Os resultados da avaliação externa são agrupados em fluxos. Portanto, avaliação externa necessita da configuração dos seguintes parâmetros:

- ***Source***: indica o endereço do núcleo origem do fluxo a ser avaliado.
- ***Target***: indica o endereço do núcleo destino do fluxo a ser avaliado.
- ***Interval***: indica o intervalo para a exibição das taxas no gráfico. Por exemplo, quando a taxa mínima é 150 Mbps, a taxa máxima é 250 Mbps e o intervalo escolhido é 10 Mbps, os pacotes serão exibidos com uma das seguintes taxas: 150, 160, 170, 180, 190, 200, 210, 220, 230, 240 ou 250 Mbps.
- ***Percentual***: indica qual o percentual de pacotes que não serão analisados. Por exemplo, se o percentual escolhido for 10% então 10% dos pacotes com maior taxa e 10% dos pacotes com menor taxa não serão analisados.

Dois tipos de gráficos podem ser gerados a partir dos parâmetros da avaliação externa: (1) *gráfico da distribuição de vazão de um determinado fluxo* e (2) *gráfico da distribuição de latência de um determinado fluxo*.

A Figura 99 exibe o gráfico da distribuição de vazão para um fluxo que possui 10 pacotes. Observe que 9 (nove) pacotes são apresentados no gráfico. Isto ocorre porque a vazão é calculada pela seguinte Equação:

$$\frac{(100 \times packetSize \times ciclesPerFlit)}{(time - lastTime)} \quad (14)$$

Onde, *packetSize* é o número de *flits* do pacote, *cyclesPerFlit* é o número de ciclos gastos pelo controle de fluxo para transmitir um *flit*, *time* é o tempo de chegada do último *flit* do pacote atual e *lastTime* é o tempo de chegada do último *flit* do pacote anterior. Conseqüentemente, não é possível calcular a vazão para o primeiro pacote.

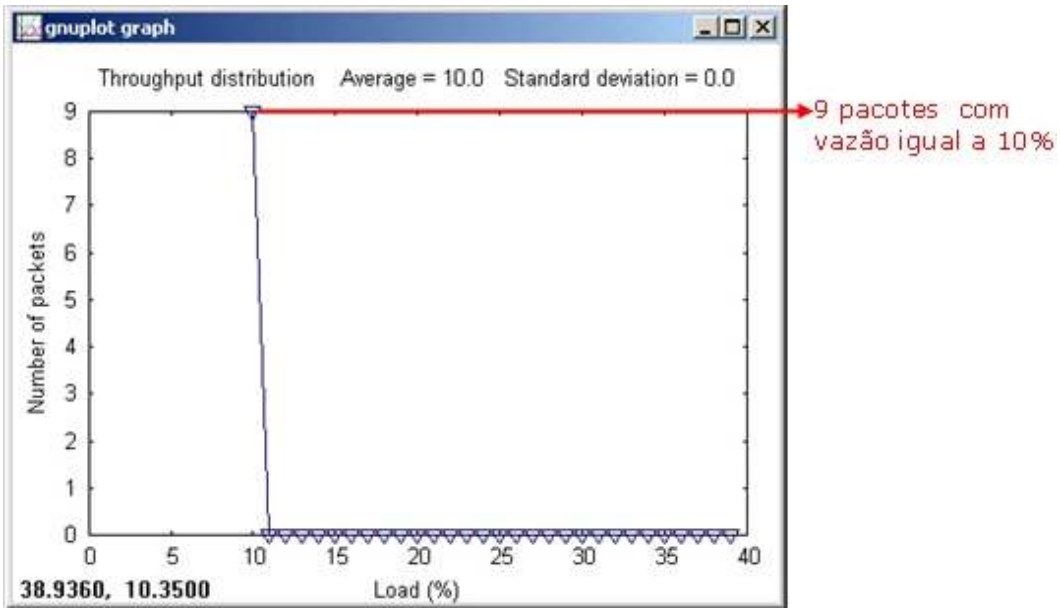


Figura 99 – Gráfico da distribuição de vazão de um fluxo com 10 pacotes.

Avaliação Interna

A avaliação interna analisa arquivos referentes às portas e canais internos à rede (aqueles conectando algum par de roteadores). Estes arquivos são gerados durante a simulação quando a opção *Internal Evaluation* é selecionada. Caso esta opção não seja selecionada durante a simulação, a avaliação interna é desabilitada na *Traffic Evaluation*.

Cinco critérios podem ser analisados na avaliação interna: (*i*) número de flits transmitidos, (*ii*) número de ciclos por flit, (*iii*) utilização dos canais da rede, (*iv*) vazão dos canais da rede e (v) vazão dos canais no caminho de um fluxo. O critério *vazão dos canais no caminho de um fluxo* é exibido no formato de gráficos 2D, um gráfico para cada canal no caminho do fluxo, conforme ilustrado na Figura 100. Os demais critérios podem ser visualizados no formato textual, conforme apresentado na Figura 101(a), ou no formato gráfico 3D, conforme apresentado na Figura 101(b).

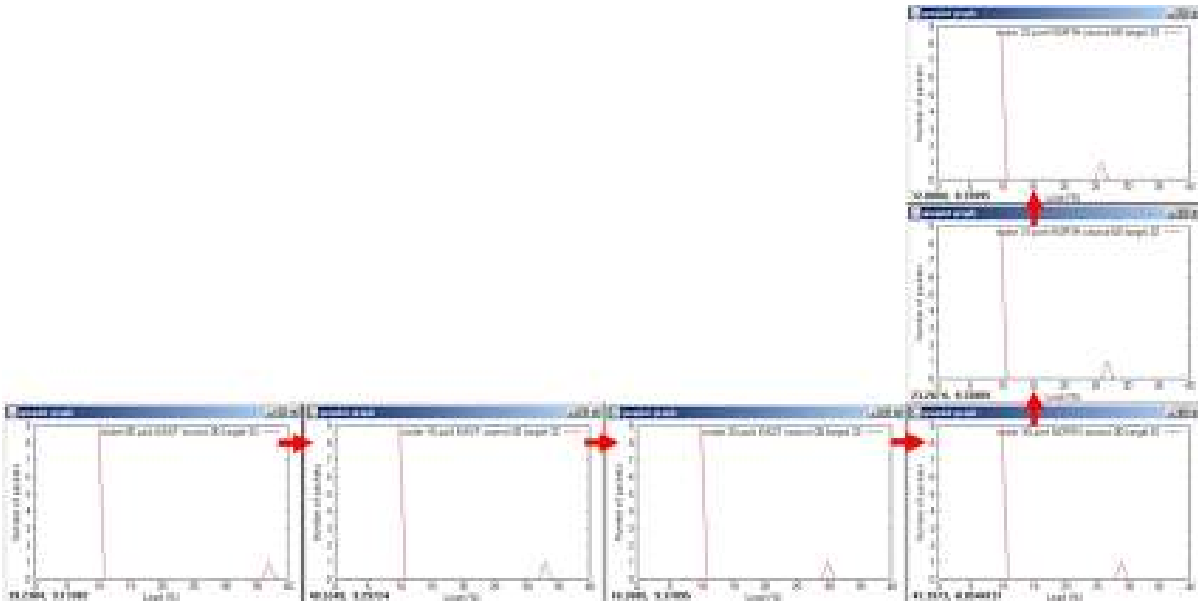
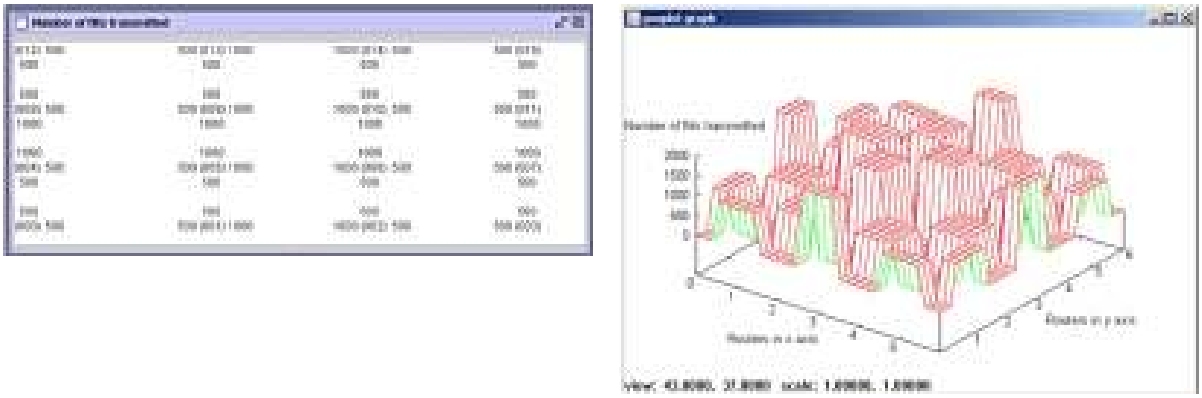


Figura 100 – Gráfico de vazão para cada canal no caminho do fluxo entre os roteadores origem (endereço 00) e destino (endereço 33).



(a) Gráfico textual **(b) Gráfico 3D**

Figura 101 – Gráficos para o critério número de *flits* transmitidos.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo