PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA PROPOSTA PARA GERAÇÃO DE TRÁFEGO E AVALIAÇÃO DE DESEMPENHO PARA NOCS

por

LEONEL PABLO TEDESCO

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

Prof. Dr. Fernando Gehm Moraes
Orientador

Porto Alegre, Novembro de 2005.

Resumo

O projetista de um SoC (*System on Chip*) que conecta núcleos de propriedade intelectual através de uma NoC (*Network on Chip*) necessita de métodos que ofereçam suporte para avaliação do desempenho das aplicações que executam nesse tipo de estrutura de comunicação. Dois aspectos fundamentais que tais métodos devem considerar são a geração e avaliação de tráfego. A geração de tráfego corresponde à injeção de pacotes na rede de acordo com especificações das aplicações, como carga oferecida e latência mínima. A avaliação de desempenho auxilia no cálculo de métricas como latência e tráfego aceito nos canais e interfaces da rede, auxiliando na identificação de pontos de congestionamento e de contenção. Neste trabalho são apresentados métodos genéricos para geração de tráfego e avaliação de desempenho nas transações que ocorrem em NoCs. A validação dos conceitos de geração de tráfego e avaliação de desempenho é realizada através de estudos de caso executados na infra-estrutura de comunicação HERMES. Dois parâmetros são considerados na geração de tráfego: distribuição espacial e carga oferecida pelos núcleos. Dois tipos de métodos para avaliação de desempenho são propostos: (*i*) avaliação externa, uma estratégia bastante utilizada por diversos grupos de pesquisa, onde a rede é considerada uma caixa preta e os resultados de tráfego são obtidos apenas a partir das interfaces de rede externas; (*ii*) avaliação interna, onde o desempenho é computado em cada canal da rede. Através do conjunto de métodos aqui apresentados, o projetista obtém condições estabelecer diferentes cenários de tráfego e avaliar o desempenho das comunicações que ocorrem nestes cenários.

Palavras Chave: NoC (*Network-on-Chip*), SoC (*System-on-a-Chip*), geração de tráfego, avaliação de desempenho.

Abstract

The designer of a system on a chip (SoC) that connects intellectual property cores through a network on chip (NoC) needs methods to support application performance evaluation. Two key aspects these methods have to address are the generation and evaluation of network traffic. Traffic generation allows injecting packets in the network according to application constraint specifications such as offered load and end-to-end latency. Performance evaluation helps in computing latency and accepted traffic at network channels and interfaces, as well as to identify congestion and hot-spots. This dissertation proposes general methods for both aspects in NoCs. Case studies validate the methods presented, using the HERMES NoC communication infrastructure. Two parameters are considered here to define traffic generation: packet spatial distribution and offered load. Two types of methods to evaluate performance in NoCs are discussed: (*i*) external evaluation, a common strategy found in related works, where the network is considered as a black box and traffic results are obtained only from the external network interfaces; (*ii*) internal evaluation, where performance is computed in each network channel. The set of methods presented in this dissertation offers to the designer conditions to establish different traffic scenarios and evaluate performance of the communications that occur in these scenarios.

Keywords: NoC (Network-on-Chip), SoC (System-on-a-Chip), traffic generation, performance evaluation.

Agradecimentos

Nesta etapa extremamente importante da minha vida não poderia deixar de manifestar meu profundo agradecimento a todas pessoas que de alguma forma contribuíram para que essa Dissertação se tornasse uma realidade.

Primeiramente gostaria de agradecer ao meu orientador, o Prof. Dr. Fernando Gehm Moraes, pela paciência e dedicação a mim prestados durante todo o período de mestrado. Professor, podes ter certeza de que a nossa amizade é algo que guardarei para o resto de minha vida. Obrigado por tudo!

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), que foi o órgão que ofereceu suporte financeiro para o desenvolvimento deste trabalho.

Ao Prof. Dr. Ney Laert Vilar Calazans por ter me co-orientado durante o mestrado. Tuas contribuições nos Trabalhos Individuais, Plano de Estudo e Pesquisa, Seminário de Andamento, escrita de artigo e nossas discussões envolvendo NoCs foram de extrema valia para o direcionamento desta Dissertação.

Aos professores Fabiano Hessel e Eduardo Bezerra pelo aprendizado adquirido nas cadeiras relacionadas a Sistemas Digitais e Arquitetura de Computadores, onde pude ler diversos artigos na área e conhecer conceitos acerca de sistemas embarcados.

Aos bolsistas de iniciação científica Diego Garibotti, Leonardo Luigi Giacomet (que me auxiliaram no desenvolvento das ferramentas de geração de tráfego e avaliação de desempenho em NoCs) e Everton Carara (que me ofereceu suporte técnico para inserção de mecanismos de medição de tráfego em FPGA).

Aos meus colegas, mas acima de tudo, AMIGOS que fiz no GAPH. À Aline Vieira de Mello, pelo seu auxílio técnico, sugestões e críticas (sempre muito construtivas) para o desenvolvimento dos métodos de geração de tráfego e avaliação de desempenho para a NoC HERMES. Aos meus grandes amigos Leandro Heleno Möller, Edson Ifarraguirre Moreno, Ewerson Luiz de Souza Carvalho, Luis Henrique Leal Ries César Augusto Missio Marcon e Luciano Copello Ost pela convivência extremamente agradável, especialmente no futz semanal sagrado, onde o Ost insistia que dava aula ☺.

Aos colegas de pós-graduação Odorico Mendizabal, Rafael Soares, Melissa Vetromille, Virginia Cunha, Joseane Pedroso, Carlos Adail Scherer Jr. e Érico Bastos pela ótima convivência que me proporcionaram.

Aos meus parceiros de fé Guilherme Rohde, Fábio Pasini e André Barros. Vocês sabem o quanto são importantes para mim, e assim será sempre. Nossa amizade é imortal, como dito por vocês mesmos em meu aniversário. Obrigado por existirem, meus irmãos!!!

À Camila Pires Carvalho, pelo amor, carinho e dedicação. Tua presença ilumina a minha vida. Te amo demais!

À minha família que, mesmo estando em Pelotas, sempre me ofereceu suporte e acreditou que eu poderia realizar este meu sonho de concluir o mestrado. Ao meu irmão Leonardo Heitor Tedesco, que foi e será para sempre meu amigo de todas as horas. Te amo, Mano!!! Aos meus pais Marina Izabel Tedesco e Mauro Heitor Tedesco. O amor, carinho e compreensão de vocês amenizam o efeito causado pelo fato de eu estar longe de casa desde os tempos em que cursava Engenharia de Computação na FURG. Essa minha vitória também é de vocês! Eu os amo demais!!!

Finalmente agradeço a Deus por ter me dado saúde e colocado no meu caminho pessoas que me fazem um ser humano feliz.

Muito Obrigado.

Sumário

1 INTRODUÇÃO 1
1.1 NOCS – FLUXO DE PROJETO E PARÂMETROS ESTRUTURAIS 2
1.2 OBJETIVOS 7
1.3 ORGANIZAÇÃO DO DOCUMENTO 7

2 GERAÇÃO DE TRÁFEGO E AVALIAÇÃO DE DESEMPENHO – CONCEITOS BÁSICOS 9
2.1 GERAÇÃO DE TRÁFEGO 9
2.1.1 Padrões para distribuição espacial de tráfego 9
2.1.2 Carga oferecida 14
2.1.3 Modelagem de tráfego 14
2.1.4 Categorias de serviço 16
2.1.5 Serviços de comunicação 18
2.2 AVALIAÇÃO DE DESEMPENHO 19
2.2.1 Vazão 20
2.2.2 Latência 20
2.3 CONCLUSÕES 21

3 ESTADO DA ARTE 23
3.1 SPIN 23
3.2 QNOC 26
3.3 ÆTHEREAL 31
3.4 GENKO ET AL. 34
3.5 OUTROS TRABALHOS RELACIONADOS 37
3.5.1 Hu e Marculescu 37
3.5.2 Santi et al. 38
3.5.3 Yum et al. 38
3.6 POSICIONAMENTO DA DISSERTAÇÃO EM RELAÇÃO AO ESTADO-DA-ARTE 39

4 GERAÇÃO DE TRÁFEGO PARA NOCS 41
4.1 PACOTE 41
4.2 GERAÇÃO DE PADRÕES PARA DISTRIBUIÇÃO ESPACIAL DE TRÁFEGO 42
4.3 GERAÇÃO DE TRÁFEGO VARIANDO TAXAS DE INJEÇÃO 43
4.3.1 Parametrização do tráfego 44
4.3.2 Momento de criação do pacote 45
4.3.3 Geração de tráfego a uma taxa específica 47
4.3.4 Modelagem de tráfego 49
4.3.4.1 Modelagem de tráfego com taxas de injeção pré-estabelecidas 49
4.3.4.2 Modelagem de tráfego com taxas de injeção estabelecidas de maneira aleatória 52
4.4 CONCLUSÕES 53

5 AVALIAÇÃO DE DESEMPENHO PARA NOCS 55
5.1 LATÊNCIA 58
5.1.1 Obtenção da latência 58
5.1.2 Avaliação da latência 59
5.2 TEMPO MÉDIO PARA TRANSMISSÃO DE FLITS 61
5.2.1 Obtenção do avcpf 61
5.2.2 Avaliação de avcpf 62
5.3 TRÁFEGO ACEITO 63

5.3.1 Obtenção do tráfego aceito 63
5.3.2 Avaliação do tráfego aceito 64
5.4 UTILIZAÇÃO MÉDIA DE CANAIS 66
5.4.1 Obtenção da utilização média dos canais 67
5.4.2 Avaliação da utilização média dos canais 68
5.5 VERIFICAÇÃO DO ATENDIMENTO DE REQUISITOS DE CARGA OFERECIDA E LATÊNCIA IDEAL .. 70
5.6 CONCLUSÕES 71

6 VALIDAÇÃO DOS MÉTODOS PARA GERAÇÃO DE TRÁFEGO E AVALIAÇÃO DE DESEMPENHO 73
6.1 ESTUDO DE CASO 1 74
6.1.1 Geração de tráfego 74
6.1.2 Avaliação de desempenho 75
6.2 ESTUDO DE CASO 2 78
6.2.1 Geração de tráfego 78
6.2.2 Avaliação de desempenho 79
6.3 ESTUDO DE CASO 3 85
6.3.1 Geração de tráfego 86
6.3.2 Avaliação de desempenho 86
6.4 ESTUDO DE CASO 4 88
6.4.1 Padrões de tráfego espacial e quantidade de pacotes gerados 89
6.4.2 Coleta de dados de geração de tráfego e para avaliação de desempenho 90
6.4.3 Resultados obtidos 91
6.5 CONCLUSÕES 93

7 CONCLUSÕES 95
7.1 CONTRIBUIÇÕES NA GERAÇÃO DE TRÁFEGO 95
7.2 CONTRIBUIÇÕES NA AVALIAÇÃO DE DESEMPENHO 95
7.3 CONCLUSÕES E TRABALHOS FUTUROS 96

8 REFERÊNCIAS BIBLIOGRÁFICAS 99

9 ANEXO I: A APLICAÇÃO COMPARADOR DE SEQUÊNCIAS 103
9.1.1 O algoritmo de Smith-Waterman 103
9.1.2 Exemplo de comparação de duas seqüências de caracteres 104
9.1.3 Preenchimento da matriz de comparação em um sistema multiprocessado 105
9.1.4 Implementação utilizando uma NoC 106

Lista de Figuras

Figura 1 - Fluxo de projeto de NoCs. 3
Figura 2 – Nodos de (a) processamento e (b) de roteamento. 4
Figura 3 – Exemplos de topologias regulares: (a) malha 2D; (b) árvore gorda e (c) anel cordal. 4
Figura 4 – Pontos de coleta para avaliação de desempenho, exemplificada para uma NoC topologia malha 3x3. 6
Figura 5 - Padrão de tráfego bit-reversal. 11
Figura 6 - Padrão de tráfego perfect shufle. 11
Figura 7 - Padrão de tráfego butterfly. 12
Figura 8 - Padrão de tráfego matrix transpose. 12
Figura 9 - Padrão de tráfego complemento. 13
Figura 10 - Padrão de tráfego complemento, em uma malha 4x4, utilizando o algoritmo de roteamento XY. 13
Figura 11 – Parâmetros para o dimensionamento da carga oferecida. 14
Figura 12 – Cadeia de Markov de 2 estados. 15
Figura 13 – Arquitetura genérica de comunicação. 24
Figura 14 – (a) Requisições dos iniciadores de tráfego; (b) respostas dos alvos [ZEF03]. 24
Figura 15 – Avaliação da escalabilidade das arquiteturas de comunicação através da análise da latência [ADR03]. 25
Figura 16 – Avaliação do ponto de saturação das arquiteturas de comunicação [ADR03] 26
Figura 17 – Fluxo de projeto QNoC. 27
Figura 18 – Carga relativa nos enlaces da malha. As setas apontam para a maior carga (acima à direita) e as duas menores cargas (abaixo à esquerda)[BOL04a] 29
Figura 19 – Distribuição de latências para o melhor caso (850Gbps de largura de banda alocada e 30.4% de utilização da rede) [BOL04a]. 30
Figura 20 – Carga aplicada vs. latência média obtida [BOL04a] 30
Figura 21 – Fluxo de projeto Æthereal. 31
Figura 22 – (a) Especificação das aplicações; (b) Especificações dos núcleos [GOO05]. 32
Figura 23 – Trecho de arquivo de topologia da rede [GOO05]. 33
Figura 24 – Arquivo de conexão [GOO05] 33
Figura 25 – Exemplo de arquivo de verificação [GOO05]. 33
Figura 26 - Exemplo de arquivo com resultados de simulação [GOO05]. 34
Figura 27 – Arquitetura de emulação [GEN05a]. 34
Figura 28 – Fluxo de projeto da emulação. 35
Figura 29 – Comparação tráfego em rajada com o uniforme [GEN05a]. 36
Figura 30 – Avaliação do tamanho da rajada [GEN05a]: (a) taxa de congestionamento; (b) latência média. 37
Figura 31 – Estrutura de um pacote. 41
Figura 32 – Exemplo de uso da ferramenta para geração de tráfego espacial (a) por todos os nodos da rede e (b) de um nodo específico. 42
Figura 33 – Variação de taxas de injeção: (a) pcksize fixo e idle variável; (b) pcksize variável e idle fixo; (c) pcksize variável e outint fixo; (d) pcksize fixo e outint variável; (e) bsize variável e outint fixo. 43
Figura 34 – Trecho de pseudocódigo para envio de pacotes. 46
Figura 35 – Pseudo-código para geração de uma rajada de pacotes. 46
Figura 36 - Pseudo-código para geração de várias rajadas de pacotes. 47
Figura 37 – Geração de tráfego com especificação de idle/pcksize – carga de 50%. 47
Figura 38 – Tráfego com intervalo de saída fixo – carga de 50%. 48
Figura 39 - Tráfego em rajada – carga de 50%. 48
Figura 40 – Tráfego em rajada – carga de 55%. 49
Figura 41 - Trecho de código para preenchimento da Tabela 6. 50
Figura 42 – Curva gerada pela construção da Tabela 7. 52
Figura 43 – Exemplo de geração com taxas de injeção escolhidas de maneira aleatória, utilizando a distribuição Pareto ON-OFF. 52

Figura 44 – Gráfico com taxas aleatoriamente geradas. *53*
Figura 45 - Pontos de coleta para avaliação de desempenho, exemplificada para uma NoC topologia malha 3x3. ..... *55*
Figura 46 – Estrutura de um arquivo contendo informações do tráfego de um canal que interliga um roteador com seu núcleo. *56*
Figura 47 – Campos do arquivo exemplificado pela Figura 46 considerados para avaliação externa. *56*
Figura 48 - Exemplo de arquivo para avaliação interna. Cada registro corresponde a um par (flit, momento em que foi transmitido). *57*
Figura 49 – Campos do arquivo exemplificado pela Figura 48 considerados para avaliação interna. *57*
Figura 50 – Gráfico CNF para análise de latência. *59*
Figura 51 – pseudocódigo para geração do gráfico de distribuição de latências. *60*
Figura 52 – (a) Exemplo de gráfico de distribuição de latências; (b) Tabela que gerou o gráfico em (a). *61*
Figura 53 - Exemplo de gráfico de superfície ilustrando o valor de avcpf nos enlaces de uma NoC 4x4. ..... *62*
Figura 54 - Mapa textual mostrando o valor do cpf nos canais de transmissão de dados para o exemplo ilustrado na Figura 53. Valores nos canais indicam cpf máximo (superior), médio e mínimo (inferior). *63*
Figura 55 – Exemplo de gráfico CNF para avaliação do tráfego aceito. *64*
Figura 56 – Pseudocódigo para geração do gráfico de distribuição de tráfego aceito. *66*
Figura 57 – (a) Exemplo de gráfico de distribuição de tráfego aceito; *66*
Figura 58 – Exemplo de comparação entre: (a) abw e (b) thr. *68*
Figura 59 – abw para uma NoC 4x4 (a) nos enlaces; (b) nos canais. *69*
Figura 60 - thr para uma NoC 4x4 (a) nos enlaces; (b) nos canais. *69*
Figura 61 – Gráficos CNF. *76*
Figura 62 – Distribuição de latências para o Estudo de Caso 1 ..... *76*
Figura 63 - Largura de banda ocupada por enlace para os dois algoritmos de roteamento (carga de 20%), sem utilização de canais virtuais. *77*
Figura 64 - Tempo médio para transmissão de flits (carga de 20%). *78*
Figura 65 – (a) Núcleos que geram dados segundo uma distribuição normal e seus destinos; (b) distribuição de probabilidade que descreve as taxas de injeção dos núcleos em (a). *79*
Figura 66 – Distribuições de latência de fluxos modelados segundo a distribuição normal da Figura 65(b) concorrendo com tráfego constante com taxas de (a) 5% - Cenário 1 e (b) 10% - Cenário 2. ..... *80*
Figura 67 - Distribuições de tráfego aceito de fluxos modelados segundo a distribuição normal da Figura 65(b) concorrendo com tráfego constante com taxas de (a) 5% e (b) 10%. *81*
Figura 68 – Flutuação de carga: (a) antes da saturação da rede (b) após a saturação da rede. *82*
Figura 69 – Caminho percorrido pelo fluxo com origem no roteador 18 e destino no roteador 45. ..... *82*
Figura 70 - Distribuição de taxas de encaminhamento de pacotes originados no nodo 18 tendo como destino o nodo 60. Núcleos que geram tráfego constante injetam pacotes na taxa relativa de 5%. *83*
Figura 71 - Distribuição de taxas de encaminhamento de pacotes originados no nodo 18 tendo como destino o nodo 60. Núcleos que geram tráfego constante injetam pacotes na taxa relativa de 10%. *84*
Figura 72 - Linhas pontilhadas apresentam regiões onde foram redimensionados os buffers dos roteadores. Alterações consideram medições de parâmetros de desempenho do estudo de caso 1: (a) sem redimensionamento (b) roteadores das bordas; e (c) roteadores da bisecção XY. *85*
Figura 73 – Gráficos CNF para as três configurações de rede utilizadas no Estudo de caso 3. ..... *86*
Figura 74 – Distribuições de latências comparando os casos onde há redimensionamento nas bordas e na bisecção XY da rede. *87*
Figura 75 - Valores de CPF médio obtidos nos enlaces (carga de 20%): (a) sem alterações nos buffers; (b) redimensionamento dos buffers da periferia; (c) redimensionamento dos buffers da bisecção XY. *88*
Figura 76 – Padrões de tráfego utilizados: (a) Cenario1 – número de hops entre todos os pares origem-destino é igual a 1; (b) Cenario2 – fluxos originados pelos processadores P1, P3, P5 e P6 possuem número de hops maior que 1. ..... *89*
Figura 77 – Dimensões da matriz de comparação utilizada no Estudo de Caso 4. ..... *90*
Figura 78 – Momentos capturados para verificação da carga oferecida e avaliação de desempenho. *90*
Figura 79 – Curvas de distribuições de carga oferecida e tráfego aceito similarmente geradas pelos fluxos do estudo de Caso 4. ..... *92*
Figura 80 – Distribuições de latência para o fluxo gerado por P1: (a) em Cenario1; (b) em Cenario2. ..... *93*

Figura 81 – Recursão realizada pelo algoritmo de Smith-Waterman. *103*
Figura 82 – Fluxo de dados do algoritmo de Smith-Waterman. *104*
Figura 83 – Cálculo de H para a célula (2,2). *105*
Figura 84 – Arquitetura utilizada para comunicação dos núcleos. *107*
Figura 85 – Descrição em linguagem assembly do procedimento Send. *107*
Figura 86 - Descrição em linguagem assembly do procedimento Receive. *108*

Lista de Tabelas

Tabela 1 – Distribuições de probabilidade utilizadas em modelagem de tráfego. *16*
Tabela 2 – Dados transmitidos nas arquiteturas PI-Bus e SPIN. *25*
Tabela 3 – Modelagem de tráfego – QNoC com freqüência de 1GHz [BOL04a]. *28*
Tabela 4 – Latência média dos pacotes em função da largura de banda alocada. QoS desejado é marcado em itálico. Não-atendimento a requisitos de QoS é marcado em negrito. *29*
Tabela 5 – Resumo dos trabalhos pesquisados. *40*
Tabela 6 – Modelo de tabela para atribuição de quantidades de gerações de dados com determinada taxa de transmissão. *50*
Tabela 7 – Tabela com taxas de geração preestabelecidas. *51*
Tabela 8 – Exemplo de preenchimento dos valores de latência média. *60*
Tabela 9 – Exemplo de preenchimento dos valores de tráfego aceito. *65*
Tabela 10 - Trecho de tabela com valores gerados e medidos para verificação de atendimento a requisitos de QoS. NoC 8x8, roteamento XY, 2 canais virtuais. Núcleos geram pacotes na taxa de 10% utilizando o padrão complemento. *71*
Tabela 11 – Valores de latência e tráfego aceito obtidos no Estudo de Caso 1. ..... *75*
Tabela 12 – Especificações de taxas de injeção. *79*
Tabela 13 – Valores da curva normal gerada. *79*
Tabela 14 – Valores de latência para os fluxos com taxas de injeção variável. *80*
Tabela 15 - Valores de tráfego aceito para os fluxos com taxas de injeção variável. *81*
Tabela 16 – Valores de latência média e tráfego aceito para os casos em que a rede não é alterada e em que os buffers da rede sofrem redimensionamento. *86*
Tabela 17 – Resultados de desempenho do Estudo de Caso 4. ..... *92*
Tabela 18 – Matriz exemplo. *104*
Tabela 19 – Matriz com escores iniciais. *104*
Tabela 20 – Exemplo de matriz totalmente preenchida. *105*
Tabela 21 - Matriz sendo preenchida por 6 processadores. *106*
Tabela 22 – Matriz sendo preenchida por 3 processadores. *106*

Lista de Abreviaturas

ABR	*Available Bit Rate*
ATM	*Asynchronous Transfer Mode*
BE	*Best Effort*
BPS	*Bits Per Second*
CBR	*Constant Bit Rate*
CNF	*Chaos Normal Form*
CRC	*Cyclical Redundancy Checking*
DMA	*Direct Memory Access*
DS	*Differentiated Services*
ETE	*End-To-End*
Flit	*Flow Control Digit*
FPGA	*Field Programmable Gate Array*
GT	*Guaranteed Throughput*
IP	*Intellectual Property*
LFSR	*Linear Feedback Shift Register*
MBS	*Maximum Burst Size*
MCR	*Minimum Cell Rate*
MMP	*Markov Modulated Process*
MPEG	*Moving Picture Experts Group*
MPSoC	*Multi Processor System On Chip*
NoC	*Network On Chip*
Nrt-VBR	*Non-Real-Time Variable Bit Rate*
OCP	*Open Core Protocol*
OE	*Odd-Even*
PCR	*Peak Cell Rate*
QNoC	*Quality Of Service Network On Chip*
QoS	*Quality Of Service*
RAM	*Random Access Memory*
Rt-VBR	*Real-Time Variable Bit Rate*
RTL	*Register Transfer Level*
SCR	*Sustainable Cell Rate*
SoC	*System On Chip*
SPIN	*Scalable, Programmable Interconnect Network*
UBR	*Unspecified Bit Rate*
VBR	*Variable Bit Rate*
VCI	*Virtual Component Interconnect*
VHDL	*Very high-speed integrated circuit Hardware Description Language*
WF	*West-First*

Lista de Símbolos

abw	utilização média de um canal
acceptedtraffic	tráfego aceito
arb_time	tempo que cada roteador leva para processar o cabeçalho de cada pacote
avcpf	ciclos por flit médio
bits_trans	número de bits transmitidos
bsize	número de pacotes por rajada
bw	largura de banda
chr	capacidade máxima do canal para transmissão (em bps)
cpf	ciclos por flit
hops	quantidade mínima de saltos de roteamento existentes entre a origem e o destino de um fluxo
ideal_avg_lat	latência média ideal
idle	período de ociosidade do canal (em ciclos de relógio)
incr	incremento
ipr	taxa de transmissão do núcleo gerador (em bps)
lastpcksize	tamanho do último pacote da rajada
maxipr	taxa máxima de transmissão
minipr	taxa mínima de transmissão
n_bits	largura do canal
n_ciclos_sim	número de ciclos para que todo o tráfego seja entregue aos destinos
ncyclesflit	quantidade de ciclos de relógio necessários para transmissão de 1 flit
npacks	número de pacotes
nrates	quantidade de taxas de transmissão
ocup	período de ocupação do canal (em ciclos de relógio)
offeredload	carga oferecida
outint	intervalo entre saídas de pacotes
pcksize	tamanho do pacote (em número de flits)
pcktmp	momento de criação do pacote por um núcleo gerador
T	período de relógio utilizado
tpfext	momento da recepção do primeiro flit de um pacote pelo núcleo (em ciclos de relógio)
tpfint	momento da transmissão do primeiro flit de um pacote no canal
tplext	momento da recepção do último flit de um pacote pelo núcleo (em ciclos de relógio)
tplint	momento da transmissão do último flit de um pacote no canal
tsfint	momento da transmissão do primeiro flit do primeiro pacote transmitido no canal
tsfint	momento da transmissão do último flit do último pacote transmitido no canal
vazao	vazão do tráfego em deteminado canal
vazao_rel	vazão relativa
thr	taxa efetiva de transmissão de dados
thrbps	taxa efetiva de transmissão de dados (em bps)

1 INTRODUÇÃO

Um Sistema-em-Chip (SoC, do inglês *System-on-a-Chip* [BER01]) corresponde a um sistema computacional completo implementado em um único circuito integrado. Normalmente um SoC possui um ou mais processadores de propósito geral, lógica digital (programável ou não), circuitos analógicos, além de bancos de memória dinâmica e estática [SCH97]. Em comparação com projetos baseados em múltiplos circuitos integrados em uma placa de circuito impresso, SoCs apresentam como vantagens maior desempenho, menor consumo de potência, menor volume e peso. A gerência da complexidade inerente ao projeto de um SoC pode ser realizada através da divisão computação-comunicação [KEU00]:

- Computação – a funcionalidade dos núcleos de Propriedade Intelectual[1] (IP, do inglês Intellectual Property) que compõem um SoC;
- Comunicação – movimentação de informações entre os núcleos.

Os pontos que esta divisão procura atacar estão relacionados com as fortes pressões de *time-to-market* exercidas pelo mercado e o aumento da complexidade dos SoCs: (*i*) reusabilidade, (*ii*) validação, (*iii*) proteção à propriedade intelectual, e (*iv*) integração. A separação computação-comunicação permite *reutilizar* módulos pré-projetados e pré-validados, o que pode reduzir consideravelmente o tempo de projeto. A *validação* pode ser simplificada, pois os componentes que compõem um SoC foram previamente validados, restando ao projetista a tarefa, não menos complexa, de validação do sistema como um todo. Considerando que os núcleos contém alto valor agregado, é importante que os desenvolvedores destes *protejam a propriedade intelectual* dos mesmos. Como os projetistas de SoCs necessitam apenas a interface externa dos núcleos, estes podem ter sua descrição interna protegida (por exemplo, descrição do processador *MicroBlaze* [XIL05]). Por fim, a separação computação-comunicação impulsiona o desenvolvimento de interfaces padronizadas, como OCP (*Open Core Protocol* [OCP05]), para simplificar o processo de *integração* de núcleos.

No cenário de projeto de SoCs, acima apresentado, o presente trabalho de Dissertação está relacionado à *validação* de estruturas de *comunicação*.

Em relação especificamente à comunicação, geralmente a interconexão entre os núcleos que compõem os SoCs é realizada através de fios dedicados e/ou barramentos. A utilização destas estruturas combinado com SoCs cada vez maiores e mais complexos podem acarretar na queda de desempenho global destes sistemas, segundo diversos autores [BEN02][DAL01][GUE00]. As comunicações realizadas através de fios dedicados apresentam melhor desempenho, por proporcionar alto grau de paralelismo ao sistema, uma vez que transações entre diferentes iniciadores e alvos de tráfego ocorrem por meio de canais exclusivos. No entanto, tal solução apresenta pouca escalabilidade, reusabilidade e flexibilidade por necessitar de um número excessivo de fios para o caso de se interligar um SoC com uma grande quantidade de núcleos.

[1] Referenciados apenas como *núcleos* no presente trabalho.

A utilização de barramentos oferece ao sistema um conjunto de fios a ser compartilhado pelos núcleos. Tal solução apresenta maior escalabilidade e reusabilidade em comparação com a utilização de fios dedicados. Entretanto, barramentos possuem um baixo grau de paralelismo, pelo fato de que apenas uma transação pode ser realizada em um dado instante de tempo. Outro problema diz respeito às altas capacitâncias e resistências parasitas inerentes à utilização de fios longos, o que pode trazer queda no desempenho global do sistema. A utilização de múltiplos barramentos interligados por pontes ou organizados de forma hierárquica apenas reduz os problemas citados, mas não os elimina.

Os problemas apresentados pelas estruturas de comunicação citadas acima motivaram a abertura de um novo campo de pesquisa relacionado às transações que ocorrem entre núcleos. A idéia central desse novo paradigma é trazer para o projeto de sistemas embarcados conceitos oriundos da área de redes de computadores, telecomunicações e sistemas distribuídos. O desafio é estabelecer uma estrutura de comunicação capaz de atender a requisitos de desempenho e consumo de energia para as diferentes aplicações que executam sobre SoCs. Tal paradigma para comunicação em SoCs é denominado Rede-intra-chip (NoC, do inglês *Network-on-chip*[1]) [BEN02] [DAL01] [GOO02][GUE00][HEM00][SGR01].

1.1 NoCs – fluxo de projeto e parâmetros estruturais

Uma NoC é uma estrutura de comunicação onde os núcleos componentes de um SoC são interligados através de *roteadores*. Como já mencionado, o projeto de uma NoC utiliza conceitos oriundos da área de redes de computadores, telecomunicações e sistemas distribuídos. Entretanto, o projetista de uma NoC também deve considerar requisitos de consumo de energia e área, não considerados no projeto de redes de dados tradicionais [BEN01].

O estudo de NoCs propõe-se a encontrar alternativas para minimizar os problemas encontrados em arquiteturas de comunicação baseados em fios dedicados e barramento expostos no início deste Capítulo. As principais características das NoCs são:

- *Paralelismo*: devido à multiplicidade de caminhos possíveis em uma NoC, o paralelismo de comunicação entre pares distintos de núcleos é possível.
- *Possibilitar a estruturação e o gerenciamento de fios em tecnologias sub-micrônicas* [BEN02][DAL01][GOO02]: utilização de fios mais curtos, ponto-a-ponto, com menor capacitância parasita;
- *Compartilhamento de fios* [BOL04a][DAL01][GOO02]: possibilitando sua utilização de maneira mais eficiente;
- *Confiabilidade e eficiência na gerência do consumo de energia* [BEN01][BOL04a];
- *Escalabilidade da largura de banda* [GUE00], o acréscimo de um elemento na rede leva ao aumento do número de canais de comunicação, melhorando de uma maneira

[1] O termo NoC foi introduzido em 2000, no evento NorChip, em artigo publicado por Hemani et al. [HEM00].

global seu desempenho;

- *Reusabilidade*: possibilidade de se aproveitar a mesma estrutura de comunicação em aplicações distintas;
- *Decisões de roteamento distribuídas* [BEN02][GUE00]: o que pode trazer um maior balanceamento na utilização dos recursos da rede.

Porém, a utilização de NoCs traz como principais desvantagens [GUE00]:

- *Considerável consumo de área de silício*, em relação a barramentos;
- *Utilização de wrappers*: núcleos projetados para arquiteturas baseadas em barramento necessitam de adaptadores de protocolo para comunicação, dificultando o reuso de núcleos;
- *Latência (tempo para transmissão de pacotes)*: causada por contenções na rede;
- *Complexidade de projeto*;
- *Mudança de paradigma*: projetistas de SoCs devem se adaptar a novos conceitos de comunicação;

Normalmente o projeto de uma NoC envolve as etapas *construção da rede* e *execução e teste*, que constituem o fluxo ilustrado na Figura 1.

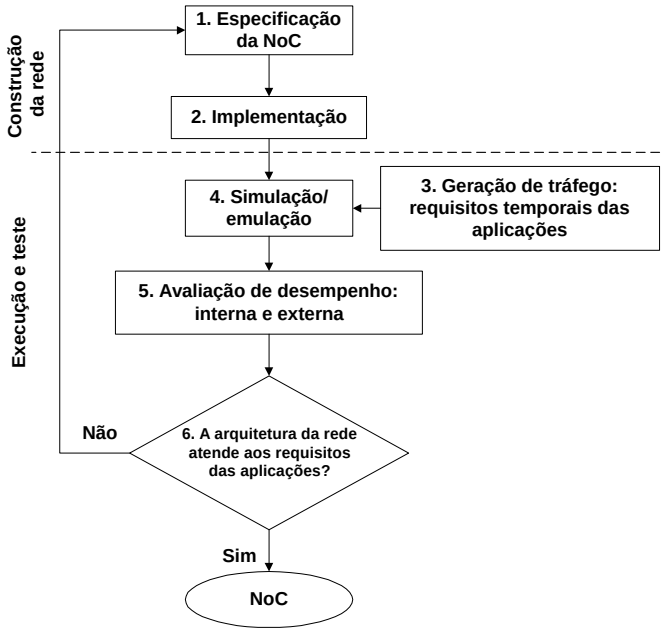


Figura 1 - Fluxo de projeto de NoCs.

Construção da rede

Na etapa *construção da rede*, os parâmetros estruturais da NoC são definidos e a rede é implementada. Inicialmente esses parâmetros são definidos de acordo com o conhecimento que o projetista tem a respeito da movimentação de dados da aplicação-alvo. Tal conhecimento pode ser melhorado através das medições de desempenho obtidas na etapa *execução e teste*, podendo-se realizar a realimentação da **ação 6** até a **ação 1**. Assim, o projetista procura a configuração da NoC

que melhor atenda à aplicação (ou conjunto de aplicações) que executarão sobre a mesma.

A *topologia* corresponde à estrutura de interconexão entre os diversos *nodos* da rede. Tais nodos podem ser de *processamento* (núcleos - Figura 2(a)) ou *roteamento* (roteadores - Figura 2(b)). Quanto à topologia, uma rede pode ser classificada em uma das seguintes classes: (*i*) *direta*, onde para cada roteador há um núcleo associado (Figura 3 (a) e (c)); (*ii*) *indireta*, onde somente alguns roteadores possuem ligação com núcleos (Figura 3(b)).

(a) (b)

Figura 2 – Nodos de (a) processamento e (b) de roteamento.

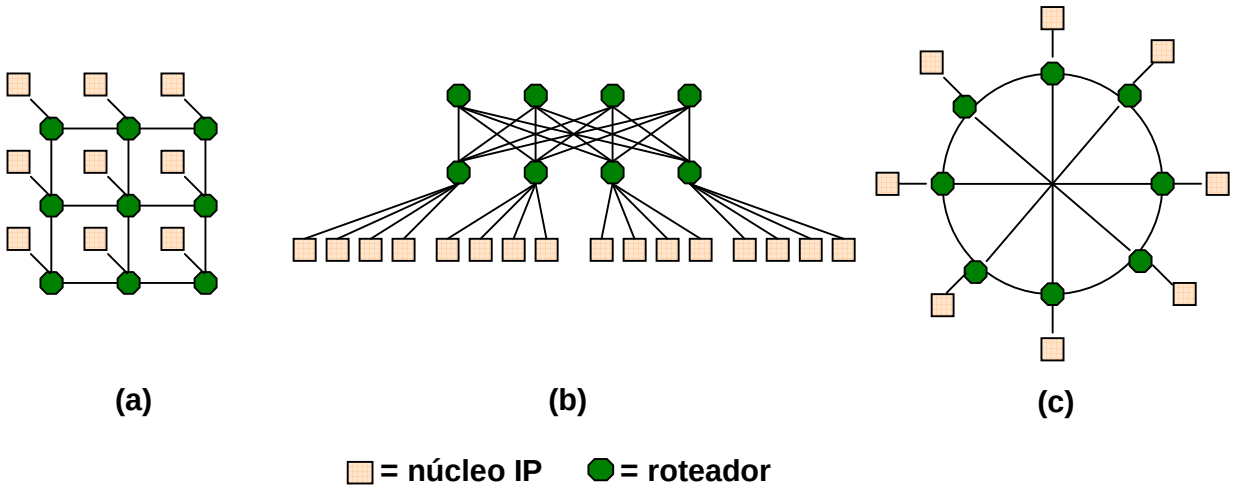


Figura 3 – Exemplos de topologias regulares: (a) malha 2D; (b) árvore gorda e (c) anel cordal.

Os *roteadores* transferem informações entre núcleos, conectando um número de canais de entrada a um número de canais de saída. A estrutura de um roteador geralmente é constituída de um ou mais módulos de *controle de chaveamento*, *roteamento interno*, *buffers* para armazenamento temporário de informações e *portas* de comunicação com outros roteadores ou núcleos. Os *algoritmos de roteamento* são executados pelos roteadores com o objetivo de determinar por qual porta de saída será encaminhado um dado recebido em uma determinada porta de entrada. Um algoritmo de roteamento pode ser *determinístico* (não-adaptativo), se o algoritmo de roteamento sempre escolher o mesmo caminho entre um determinado par origem-destino; ou *adaptativo*, se o algoritmo de roteamento fizer uso de alguma informação a respeito do tráfego da rede para escolher o caminho de um determinado pacote. Um exemplo de algoritmo de roteamento determinístico é o *XY* (também conhecido como *dimension-ordered*). Exemplos de algoritmo de roteamento

adaptativo são *WF* (*West-First*), *Dyad* [HU04] e *Contention-look-ahead* [YE04].

Um *pacote* representa uma informação para comunicação em um formato padrão para as mensagens da rede, constituindo-se de um cabeçalho (*header*), dados úteis (*payload*) e terminador (*trailer*). Baseado nas informações do *header*, o roteador define o caminho a ser seguido pelo restante do pacote. O *trailer* é utilizado para verificação de erros e sinalização de fim do pacote.

O *modo de chaveamento* especifica como os pacotes se movimentam através dos roteadores. No *chaveamento de circuitos* uma *conexão* do núcleo origem até o núcleo destino deve ser inicialmente estabelecida para posterior envio dos dados. Quando o chaveamento for de *pacotes*, são alocados apenas os canais necessários para transmissão de pacotes, permitindo assim que diferentes conexões compartilhem os mesmos recursos da rede. Os métodos de chaveamento de pacotes utilizados são: (*i*) *store-and-forward*, onde o pacote é completamente armazenado no *buffer* do roteador para posteriormente ser repassado; (*ii*) *virtual-cut-through*, onde um roteador pode enviar o pacote assim que o roteador destino assegurar sua completa recepção; (*iii*) *wormhole*, que é uma variação do *virtual-cut-through*, que reduz a necessidade de memória para *buffers*, sendo os pacotes transmitidos em unidades menores denominadas *flits* (***fl****ow control dig****its***), que avançam pela rede em um modo *pipeline*. O *flit* é a unidade básica para alocação de largura de banda na transmissão de pacotes, sendo esta alocação realizada pelos métodos de controle de fluxo.

As estratégias de *controle de fluxo* especificam as negociações que devem ocorrer para a concretização de cada transferência de dados que ocorre entre os roteadores (alocação de largura de banda). Na estratégia *handshake*, o emissor informa a intenção de enviar um dado ao receptor através de um sinal de disponibilização (*request*) e o receptor confirma a disponibilidade de espaço em *buffer* para receber esse dado através de uma linha de aceitação (*acknowledge*). Na estratégia baseada em créditos (*credit-based*), uma transmissão é realizada sempre que o receptor está sinalizando ter espaço suficiente em seu *buffer* para armazenamento dos *flits* a serem recebidos. Em uma rede sem congestionamento, espera-se que sejam necessários dois ciclos para se transmitir um *flit*, se o controle de fluxo adotado for o *handshake*. Para o controle de fluxo baseado em créditos, apenas um ciclo de relógio é necessário para transmissão de um flit.

Com a definição dos parâmetros acima citados, o projetista implementa a rede para a segunda etapa, onde é verificado se a estrutura de comunicação projetada atende aos requisitos de desempenho da aplicação-alvo. Nos experimentos realizados durante a elaboração do presente trabalho fez-se uso de redes intra-chip *diretas*, com topologia *malha*, chaveamento *por pacotes* com o método *wormhole* e controle de fluxo *baseado em créditos*. Entretanto, tais restrições não se aplicam ao desenvolvimento deste trabalho como um todo, podendo as técnicas de geração de tráfego e avaliação de desempenho que serão apresentadas serem aplicadas a qualquer especificação de arquitetura de NoC.

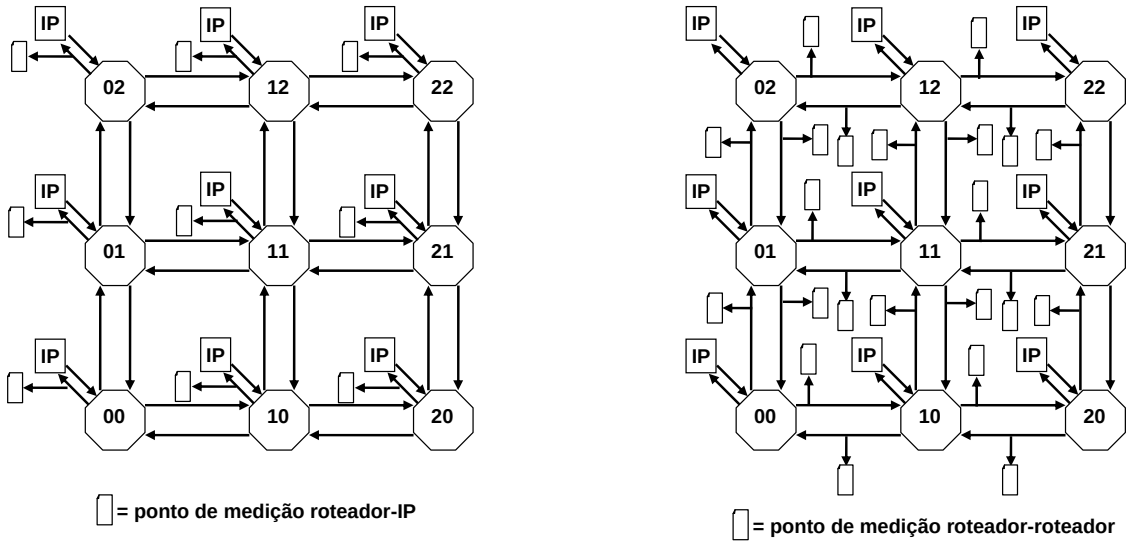
Execução e teste

Na etapa *execução e teste* a rede é simulada, verificando-se se ela atende ou não aos

requisitos de desempenho das aplicações que nela executará. Nesse caso é realizada a simulação da arquitetura de rede escolhida associada com os padrões de transmissão de dados utilizados em cada núcleo gerador de tráfego.

Durante a *geração de tráfego* (**ação 3**, Figura 1), são caracterizadas as aplicações que executarão sobre a rede. Para cada iniciador de comunicação são definidos os seguintes parâmetros: (*i*) *distribuição espacial*, relação posicional entre cada iniciador e destinatário de tráfego; (*ii*) *taxa de injeção*, que é a freqüência em que o núcleo tenta inserir os dados gerados. Para dimensionar tais parâmetros, normalmente são utilizados *modelos de tráfego*, que possuem informações probabilísticas de geração de dados similares a aplicações reais. Outra alternativa é a utilização de *traces* reais de tráfego. A utilização de modelos não traz uma caracterização exata de uma fonte real, mas é adequada para simulações, uma vez que é possível gerar uma quantidade relativamente pequena de dados com propriedades probabilísticas semelhantes a uma dada aplicação (por exemplo, vídeo). A utilização de *traces* normalmente não é adotada em simulação, pelo fato da grande quantidade de dados envolvida que deve ser transmitida. Por esse motivo, experimentos envolvendo *traces* normalmente são realizados através de emulação.

A *avaliação de desempenho* (**ação 4**, Figura 1) utiliza as informações da etapa da geração de tráfego como referência para a verificação de atendimento ou não dos requisitos de desempenho de um dado conjunto de aplicações. O projetista pode, desta forma, analisar quantitativa e qualitativamente os eventos de comunicação que ocorreram na rede. Duas maneiras de avaliar desempenho em NoCs são utilizadas: (*i*) *avaliação externa* (Figura 4(a)), onde a NoC é considerada como sendo um núcleo de comunicação, sendo o tráfego para análise coletado nas interfaces externas (que se comunicam com os núcleos de processamento) – abordagem mais comumente utilizada pelos autores pesquisados; (*ii*) *avaliação interna* (Figura 4(b)), onde o tráfego é coletado em cada canal da NoC, possibilitando a detecção de pontos de contenção da rede, bem como regiões onde há maior fluxo de dados.



(a) – Avaliação externa (b) – Avaliação interna

Figura 4 – Pontos de coleta para avaliação de desempenho, exemplificada para uma NoC topologia malha 3x3.

1.2 Objetivos

A presente Dissertação tem por objetivo propor métodos que farão parte do conjunto de etapas *execução e teste* para NoCs. A constituição dos objetivos do trabalho de mestrado é dividida em duas partes: objetivos estratégicos e objetivos específicos.

Os objetivos estratégicos compreendem:

- Domínio da tecnologia de redes intra-chip;
- Domínio de técnicas de geração de tráfego e avaliação de redes de comunicação de dados.

Os objetivos específicos compreeendem:

- Aplicação de técnicas de geração tráfego para redes de comunicação a redes intra-chip;
- Desenvolvimento de ferramentas de apoio à geração e avaliação de tráfego para redes intra-chip;
- Aplicação das técnicas e ferramentas desenvolvidas para a NoC HERMES [MOR04].

1.3 Organização do documento

Os Capítulos 2 e 3 incluem conceitos básicos e revisão bibliográfica necessários à compreensão deste documento. O Capítulo 2 apresenta conceitos básicos sobre geração de tráfego e avaliação de desempenho para redes de comunicação em geral. No Capítulo 3 são revisados trabalhos relacionados à geração de tráfego e avaliação de desempenho em NoCs. Ao final deste Capítulo compara-se o estado-da-arte ao trabalho que será apresentado nos Capítulos subseqüentes, relacionando-se as contribuições do presente trabalho.

Os Capítulos 4, 5, e 6 descrevem as contribuições deste trabalho. O Capítulo 4 apresenta um método para geração de tráfego espacial com taxas de injeção fixas e com distribuição de probabilidade. O Capítulo 5 descreve o conjunto de parâmetros para avaliação de desempenho, interno e externo, em NoCs. Os estudos de caso utilizados para validar os conceitos sobre geração de tráfego e avaliação de desempenho são objeto do Capítulo 6.

Conclusões e direções para trabalhos futuros são apresentados no Capítulo 7.

2 GERAÇÃO DE TRÁFEGO E AVALIAÇÃO DE DESEMPENHO – CONCEITOS BÁSICOS

Este Capítulo apresenta conceitos básicos relacionados à geração de tráfego e avaliação de desempenho. São abordados conceitos de geração de tráfego, incluindo distribuição espacial e taxa de injeção de dados, modelagem de tráfego e geração de tráfego para redes com suporte a QoS (Qualidade de Serviço, do inglês *Quality of Service*). Posteriormente são revisados parâmetros utilizados para avaliação de desempenho.

Cabe salientar que tais conceitos são normalmente utilizados na área de redes de comunicação tradicionais, especialmente ATM (*Asynchronous Transfer Mode*), cujo projeto é voltado para aplicações com requisitos rígidos de desempenho. Os métodos para utilização dos conceitos vistos no presente Capítulo e a inserção de outras variáveis para aplicação em NoCs são objeto do Capítulo 4.

2.1 Geração de tráfego

A geração de tráfego tem por objetivo definir uma estrutura de transmissão de dados originados por iniciadores para seus destinos, além de definir os iniciadores e destinos. É através da geração de tráfego que são caracterizadas as aplicações que venham fazer uso de uma determinada rede. Essa caracterização oferece ao projetista uma referência para a avaliação de desempenho, visto que nesta etapa é verificado se a rede atende aos requisitos das aplicações que usam seus serviços.

Esta Seção mostra primeiramente padrões utilizados para geração espacial de tráfego. Posteriormente são abordados conceitos sobre injeção de pacotes. Complementam a Seção aspectos de modelagem de tráfego, classes de serviço e categorias de serviço.

2.1.1 Padrões para distribuição espacial de tráfego

Os padrões de tráfego especificam a relação entre iniciadores e destinos, ou seja, que ponto da rede se comunica com que outro ponto. Devido às similaridades existentes entre arquiteturas paralelas e SoCs multiprocessados (MPSoCs, do inglês *Multi Processor System on Chip*), padrões normalmente observados em aplicações paralelas podem ser usados para definir a distribuição espacial de pacotes em NoCs. Por exemplo, a transformada rápida de Fourier (FFT) e aplicações de ordenamento utilizam o padrão *perfect shuffle* (Figura 6), enquanto que operações de transposição de matrizes podem ser caracterizadas pelo padrão *matrix transpose* (Figura 8) [DAL04].

Dois conceitos são considerados em padrões de tráfego [DUA03]: *localidade espacial* e *localidade temporal*. Uma aplicação apresenta *localidade espacial* quando a distância média entre

nodos é menor do que a observada no padrão de tráfego uniforme (onde todos os nodos têm a mesma probabilidade de serem destinos). Como resultado, cada mensagem consome menos recursos da rede o que leva a redução na contenção de tráfego. Uma aplicação apresenta *localidade temporal* quando ocorre uma afinidade de comunicação entre um subconjunto de nodos. Como conseqüência, a probabilidade de enviar mensagens para nodos que foram recentemente escolhidos como destinatários para outras mensagens é maior do que para outros nodos. É importante salientar que nodos que possuem afinidade de comunicação entre si não precisam estar próximos na rede, ou seja, não é necessário que uma aplicação possua localidade espacial para que tenha também localidade temporal.

Outro aspecto importante a enfatizar é a escolha do algoritmo de rotamento. Tal decisão pode introduzir diferenças significativas de desempenho, pelo fato de haver a possibilidade de congestionamento em pontos críticos da rede. A combinação correta de um algoritmo de rotamento com um dado padrão de tráfego pode levar a um uso mais balanceado da rede.

Inicialmente são apresentadas as distribuições *uniforme e não-uniforme*, onde não há um destino específico para cada pacote gerado por núcleo, exibindo, dessa forma, um baixo grau de localidade espacial. Posteriormente são apresentadas as distribuições *bit-reversal*, *perfect shuffle*, *butterfly*, *matrix transpose* e *complemento*, que possuem um destino específico para cada núcleo gerador de tráfego, exibindo localidade temporal.

Uniforme e não uniforme

No padrão de tráfego *uniforme*, todos os nodos têm a mesma probabilidade de serem destinos. No estudo de redes de comunicação de dados, a distribuição uniforme é a mais freqüentemente utilizada. Alguns dos trabalhos pesquisados para elaboração desta dissertação que utilizam a distribuição uniforme são [ADR03][BOL04a][VEL04][GEN05a][SAN05].

No padrão de tráfego *não-uniforme*, a probabilidade de um nodo enviar pacotes para um de seus nodos vizinhos é o dobro em relação ao envio de pacotes para os nodos restantes. A natureza de tal distribuição produz uma maior localidade espacial do tráfego em relação à distribuição uniforme, o que a torna mais adequada para caracterização de aplicações reais. [BOL04a] e [SAN05] realizam experimentos com o padrão de tráfego não-uniforme.

Bit-reversal

No padrão de tráfego *bit-reversal*, o nodo que possui coordenadas no formato binário a_{n-1}, a_{n-2}, ..., a_1, a_0, comunica-se com o nodo a_0, a_1, ..., a_{n-2}, a_{n-1}. A Figura 5 mostra um exemplo do padrão de tráfego *bit-reversal* aplicado sobre uma NoC com topologia 4x4.

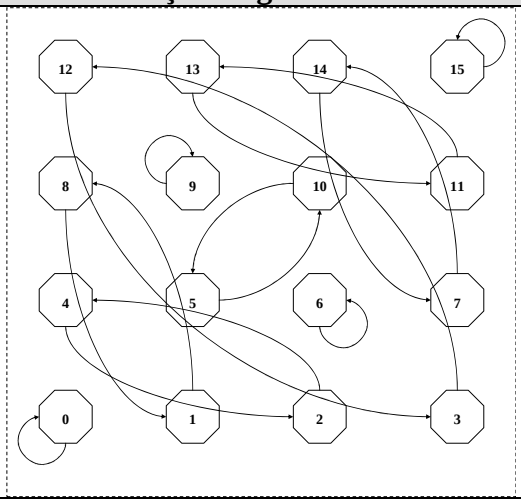
Origem	Destino	Relação origem-destino
$(0)_{10} = (0000)_2$	$(0)_{10} = (0000)_2$	
$(1)_{10} = (0001)_2$	$(8)_{10} = (1000)_2$	
$(2)_{10} = (0010)_2$	$(4)_{10} = (0100)_2$	
$(3)_{10} = (0011)_2$	$(12)_{10} = (1100)_2$	
$(4)_{10} = (0100)_2$	$(2)_{10} = (0010)_2$	
$(5)_{10} = (0101)_2$	$(10)_{10} = (1010)_2$	
$(6)_{10} = (0110)_2$	$(6)_{10} = (0110)_2$	
$(7)_{10} = (0111)_2$	$(14)_{10} = (1110)_2$	
$(8)_{10} = (1000)_2$	$(1)_{10} = (0001)_2$	
$(9)_{10} = (1001)_2$	$(9)_{10} = (1001)_2$	
$(10)_{10} = (1010)_2$	$(5)_{10} = (0101)_2$	
$(11)_{10} = (1011)_2$	$(13)_{10} = (1101)_2$	
$(12)_{10} = (1100)_2$	$(3)_{10} = (0011)_2$	
$(13)_{10} = (1101)_2$	$(11)_{10} = (1011)_2$	
$(14)_{10} = (1110)_2$	$(7)_{10} = (0111)_2$	
$(15)_{10} = (1111)_2$	$(15)_{10} = (1111)_2$	

Figura 5 - Padrão de tráfego *bit-reversal*.

Perfect Shuffle

No padrão de tráfego *perfect shuffle*, o nodo que possui coordenadas no formato binário a_{n-1}, a_{n-2}, ..., a_1, a_0, comunica-se com o nodo a_{n-2}, a_{n-3}, ..., a_0, a_{n-1} (rotação de 1 bit para a esquerda). A Figura 6 mostra um exemplo do padrão de tráfego *perfect shuffle* aplicado sobre uma NoC com topologia 4x4.

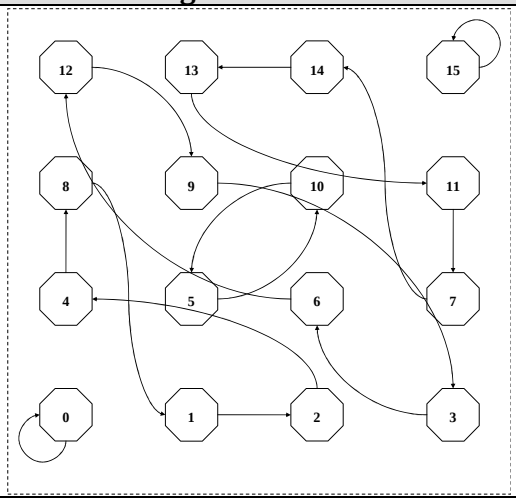
Origem	Destino	Diagrama de fluxo
$(0)_{10} = (0000)_2$	$(0)_{10} = (0000)_2$	
$(1)_{10} = (0001)_2$	$(2)_{10} = (0010)_2$	
$(2)_{10} = (0010)_2$	$(4)_{10} = (0100)_2$	
$(3)_{10} = (0011)_2$	$(6)_{10} = (0110)_2$	
$(4)_{10} = (0100)_2$	$(8)_{10} = (1000)_2$	
$(5)_{10} = (0101)_2$	$(10)_{10} = (1010)_2$	
$(6)_{10} = (0110)_2$	$(12)_{10} = (1100)_2$	
$(7)_{10} = (0111)_2$	$(14)_{10} = (1110)_2$	
$(8)_{10} = (1000)_2$	$(1)_{10} = (0001)_2$	
$(9)_{10} = (1001)_2$	$(3)_{10} = (0011)_2$	
$(10)_{10} = (1010)_2$	$(5)_{10} = (0101)_2$	
$(11)_{10} = (1011)_2$	$(7)_{10} = (0111)_2$	
$(12)_{10} = (1100)_2$	$(9)_{10} = (1001)_2$	
$(13)_{10} = (1101)_2$	$(11)_{10} = (1011)_2$	
$(14)_{10} = (1110)_2$	$(13)_{10} = (1101)_2$	
$(15)_{10} = (1111)_2$	$(15)_{10} = (1111)_2$	

Figura 6 - Padrão de tráfego *perfect shufle*.

Butterfly

No padrão de tráfego *butterfly*, o nodo que possui coordenadas no formato binário a_{n-1}, a_{n-2}, ..., a_1, a_0, comunica-se com o nodo a_0, a_{n-2}, ..., a_1, a_{n-1} (trocar o bit mais significativo com o menos significativo). A Figura 7 mostra um exemplo do padrão de tráfego butterfly aplicado sobre uma NoC com topologia 4x4.

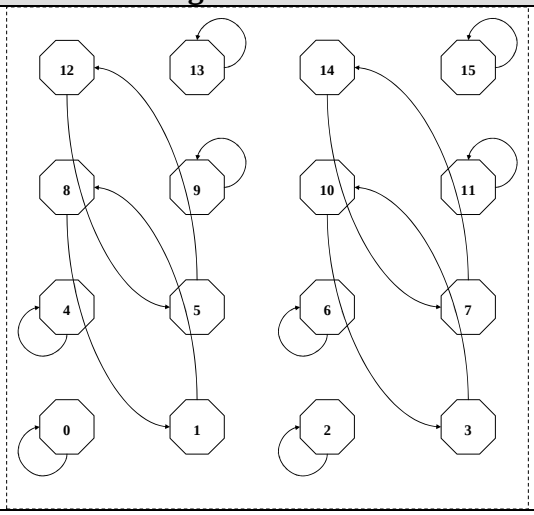
Origem	Destino	Diagrama de fluxo
$(0)_{10} = (0000)_2$ $(1)_{10} = (0001)_2$ $(2)_{10} = (0010)_2$ $(3)_{10} = (0011)_2$ $(4)_{10} = (0100)_2$ $(5)_{10} = (0101)_2$ $(6)_{10} = (0110)_2$ $(7)_{10} = (0111)_2$ $(8)_{10} = (1000)_2$ $(9)_{10} = (1001)_2$ $(10)_{10} = (1010)_2$ $(11)_{10} = (1011)_2$ $(12)_{10} = (1100)_2$ $(13)_{10} = (1101)_2$ $(14)_{10} = (1110)_2$ $(15)_{10} = (1111)_2$	$(0)_{10} = (0000)_2$ $(8)_{10} = (1000)_2$ $(2)_{10} = (0010)_2$ $(10)_{10} = (1010)_2$ $(4)_{10} = (0100)_2$ $(12)_{10} = (1100)_2$ $(6)_{10} = (0110)_2$ $(14)_{10} = (1110)_2$ $(1)_{10} = (0001)_2$ $(9)_{10} = (1001)_2$ $(3)_{10} = (0011)_2$ $(11)_{10} = (1011)_2$ $(5)_{10} = (0101)_2$ $(13)_{10} = (1101)_2$ $(7)_{10} = (0111)_2$ $(15)_{10} = (1111)_2$	

Figura 7 - Padrão de tráfego *butterfly*.

Matrix Transpose

No padrão de tráfego *matrix transpose*, o nodo que possui coordenadas no formato binário $a_{n-1}, a_{n-2}, ..., a_1, a_0$ comunica-se com o nodo $a_{\frac{n}{2}-1}, ..., a_0, a_{n-1}, ..., a_{\frac{n}{2}}$ (rotação de *n/2* bits para a esquerda, onde *n* é o número de bits que identificam o nodo). A Figura 8 mostra um exemplo do padrão de tráfego matrix transpose aplicado sobre uma NoC com topologia 4x4.

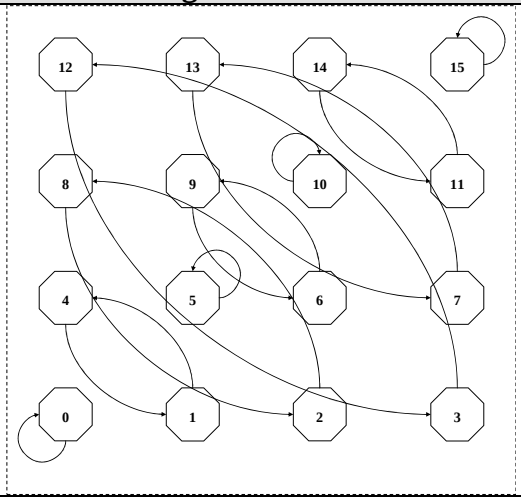
Origem	Destino	Diagrama de fluxo
$(0)_{10} = (0000)_2$ $(1)_{10} = (0001)_2$ $(2)_{10} = (0010)_2$ $(3)_{10} = (0011)_2$ $(4)_{10} = (0100)_2$ $(5)_{10} = (0101)_2$ $(6)_{10} = (0110)_2$ $(7)_{10} = (0111)_2$ $(8)_{10} = (1000)_2$ $(9)_{10} = (1001)_2$ $(10)_{10} = (1010)_2$ $(11)_{10} = (1011)_2$ $(12)_{10} = (1100)_2$ $(13)_{10} = (1101)_2$ $(14)_{10} = (1110)_2$ $(15)_{10} = (1111)_2$	$(0)_{10} = (0000)_2$ $(4)_{10} = (0100)_2$ $(8)_{10} = (1000)_2$ $(12)_{10} = (1100)_2$ $(1)_{10} = (0001)_2$ $(5)_{10} = (0101)_2$ $(9)_{10} = (1001)_2$ $(13)_{10} = (1101)_2$ $(2)_{10} = (0010)_2$ $(6)_{10} = (0110)_2$ $(10)_{10} = (1010)_2$ $(14)_{10} = (1110)_2$ $(3)_{10} = (0011)_2$ $(7)_{10} = (0111)_2$ $(11)_{10} = (1011)_2$ $(15)_{10} = (1111)_2$	

Figura 8 - Padrão de tráfego *matrix transpose*.

Complemento

No padrão de tráfego *complemento*, o nodo que possui coordenadas no formato binário $a_{n-1}, a_{n-2}, ..., a_1, a_0$ comunica-se com o nodo $\overline{a_{n-1}}, \overline{a_{n-2}}, ..., \overline{a_1}, \overline{a_0}$ (inversão de todos os bits). A Figura 9 mostra um exemplo do padrão de tráfego *complemento* aplicado sobre uma NoC com topologia 4x4.

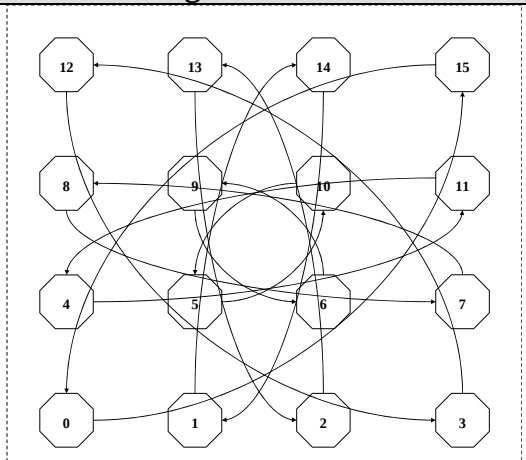
Origem	Destino	Diagrama de fluxo
$(0)_{10} = (0000)_2$ $(1)_{10} = (0001)_2$ $(2)_{10} = (0010)_2$ $(3)_{10} = (0011)_2$ $(4)_{10} = (0100)_2$ $(5)_{10} = (0101)_2$ $(6)_{10} = (0110)_2$ $(7)_{10} = (0111)_2$ $(8)_{10} = (1000)_2$ $(9)_{10} = (1001)_2$ $(10)_{10} = (1010)_2$ $(11)_{10} = (1011)_2$ $(12)_{10} = (1100)_2$ $(13)_{10} = (1101)_2$ $(14)_{10} = (1110)_2$ $(15)_{10} = (1111)_2$	$(15)_{10} = (1111)_2$ $(14)_{10} = (1110)_2$ $(13)_{10} = (1101)_2$ $(12)_{10} = (1100)_2$ $(11)_{10} = (1011)_2$ $(10)_{10} = (1010)_2$ $(9)_{10} = (1001)_2$ $(8)_{10} = (1000)_2$ $(7)_{10} = (0111)_2$ $(6)_{10} = (0101)_2$ $(5)_{10} = (0100)_2$ $(4)_{10} = (0011)_2$ $(3)_{10} = (0010)_2$ $(2)_{10} = (0111)_2$ $(1)_{10} = (0001)_2$ $(0)_{10} = (0000)_2$	

Figura 9 - Padrão de tráfego *complemento*.

A Figura 10 ilustra o padrão de tráfego complemento em uma NoC 4x4, utilizando o algoritmo de roteamento XY. As setas da Figura indicam os caminhos dos rotadores fonte até os destinos de tráfego. Dois iniciadores distintos utilizam simultaneamente os canais das *bisecções* X e Y. Devido ao congestionamento nas bisecções da rede, o padrão de tráfego complemento pode ser utilizado para avaliar os ganhos de desempenho quando da utilização de canais virtuais, considerando que neste caso é realizado o compartilhamento de uso do canal físico por fluxos de pacotes distintos. A utilização de diferentes algoritmos de roteamento acarreta em alterações no uso dos canais, para o mesmo padrão de tráfego.

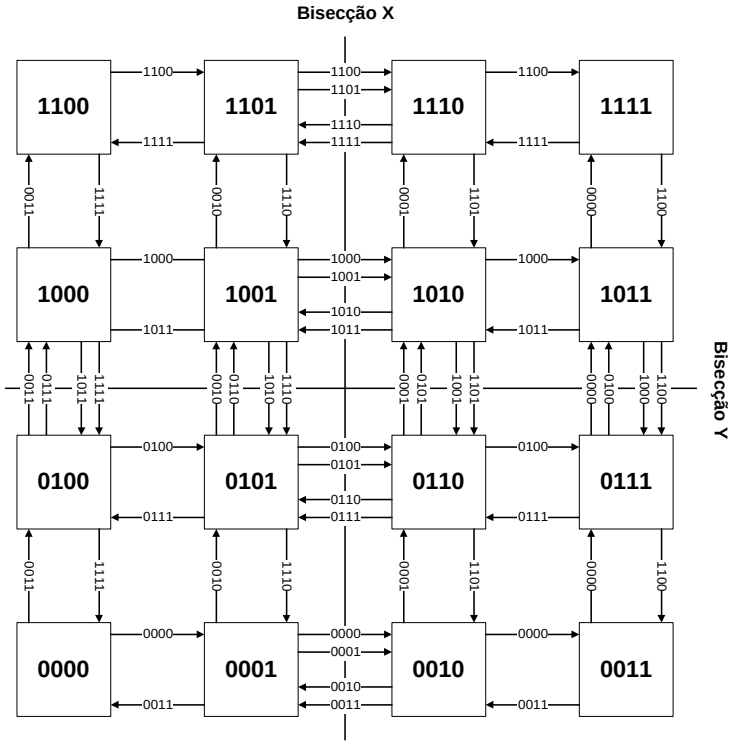


Figura 10 - Padrão de tráfego complemento, em uma malha 4x4, utilizando o algoritmo de roteamento XY.

2.1.2 Carga oferecida

A carga oferecida corresponde ao percentual de ocupação do canal, a ser definida pelas comunicações geradas por um núcleo, ou mesmo de um conjunto de núcleos. Tal parâmetro relaciona a taxa de injeção de dados de determinado núcleo com a capacidade total do canal que conecta o mesmo núcleo a rede.

Para estabelecer a carga oferecida por um determinado núcleo, o projetista pode escolher uma dentre as seguintes alternativas: (*i*) fixar o tamanho para o pacote e variar o intervalo entre os mesmos; (*ii*) variar o tamanho dos pacotes e fixar o intervalo entre o fim de um pacote e o início de outro; (*iii*) variar o intervalo decorrido entre o envio do início de um pacote e o envio do início do próximo pacote (intervalos de saída). A Figura 11 ilustra os parâmetros que podem ser variados quando do dimensionamento da carga oferecida.

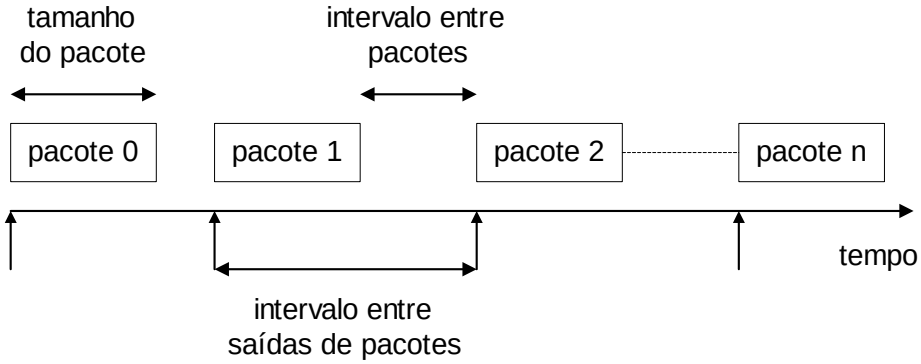


Figura 11 – Parâmetros para o dimensionamento da carga oferecida.

Aplicações MPEG (*Moving Picture Experts Group*) normalmente possuem tamanhos de *frames* variados, com um intervalo de transmissão entre *frames* fixo. Dependendo do modelo de tráfego que se quer estabelecer, um *frame* pode estar contido em um único pacote ou em um conjunto de pacotes que devem ser gerados em rajada. Aplicações envolvendo sinalização (como por exemplo, redes de sensores) normalmente possuem tamanhos de pacotes pequenos. Outro exemplo está relacionado com aplicações que executam sobre multicomputadores, que geram mensagens longas, enquanto que sistemas multiprocessados com coerência de *cache* geram mensagens curtas [DUA03].

2.1.3 Modelagem de tráfego

A modelagem de tráfego tem por objetivo inserir propriedades probabilísticas de aplicações reais à geração de tráfego. Neste caso, é possível variar os parâmetros mostrados na Figura 11, o tamanho de rajadas de pacotes ou então a distribuição de probabilidade dos valores de carga oferecida do conjunto de pacotes a ser gerado (número de pacotes com determinado valor de carga oferecida). Desta forma, o projetista da rede consegue atribuir um certo grau de realismo às suas simulações. A utilização da modelagem de tráfego possui a desvantagem de não oferecer uma caracterização exata de uma fonte de tráfego real. Entretanto, é possível simular um número bem

menor de pacotes em relação a uma simulação baseada em *traces* (amostras de tráfego real). Em modelagem de tráfego são utilizadas distribuições de probabilidade, sendo algumas delas mostradas na Tabela 1.

No modelo de tráfego *constante*, os pacotes são gerados a uma taxa fixa. Exemplos de aplicações com tráfego constante estão relacionadas a sinais telefônicos digitalizados não-compactados (amostras de 8 bits transmitidos a 64Kbps) e *streams* de áudio e vídeo não-compactados.

O modelo *ON-OFF* caracteriza-se por possuir períodos de atividade e inatividade [CHE99]. Durante os períodos de atividade, a fonte de tráfego gera pacotes de tamanho fixo em intervalos regulares, enquanto que em períodos de inatividade não há geração de pacotes. Neste caso é variado o tamanho da *rajada* de pacotes. Segundo [PAN05], a modelagem ON-OFF com a distribuição de probabilidade Pareto pode ser utilizada para caracterização de tráfego de aplicações de vídeo MPEG-2 e aplicações de rede.

O modelo *Markov ON-OFF* é um dos mais populares modelos utilizado para caracterização de fontes de voz, *streams* de vídeo e Internet [DAL04]. Neste modelo, conhecido como *Markov Modulated Process* (MMP), a ocorrência de geração de dados a uma taxa *r* é especificada pelo estado corrente da fonte de tráfego em uma cadeia de Markov (Figura 12). Os parâmetros α e β indicam a probabilidades de mudança de estado da fonte. A função de probabilidade que descreve as mudanças de estado é a exponencial com média $1/\alpha$ (mundança de OFF para ON) e $1/\beta$ (mudança de ON para OFF).

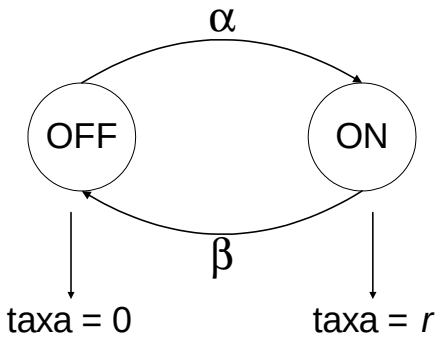


Figura 12 – Cadeia de Markov de 2 estados.

Os *processos de Poisson* foram originalmente utilizados para modelagem de eventos que ocorrem em sistemas de telefonia, sendo um dos mais antigos modelos de tráfego. Tais processos utilizam a distribuição Exponencial de probabilidade para descrever o *intervalo entre chegadas* de pacotes.

A distribuição *normal* é utilizada na modelagem de aplicações VBR (*Variable Bit Rate*), como áudio e vídeo compactados [YUM00]. A curva normal caracteriza-se por possuir forma de sino, onde sua ordenada máxima se situa na *média*. Outro parâmetro que caracteriza essa distribuição é um número que define o espalhamento relativo da curva normal em torno da média, conhecido como *desvio padrão*.

Tabela 1 – Distribuições de probabilidade utilizadas em modelagem de tráfego.

Distribuição	Função de probabilidade	Parâmetros
Exponencial	$f(x) = \frac{1}{\beta} e^{\frac{-x}{\beta}}$	β = média da distribuição
Normal	$f(x) \equiv \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$	μ = média σ = desvio padrão
Pareto ON-OFF	$t_{ON} = (1-r)^{\frac{-1}{\alpha_{ON}}}$ $t_{OFF} = (1-r)^{\frac{-1}{\alpha_{OFF}}}$	t = tempo de permanência no estado ON/OFF; r = número aleatoriamente escolhido entre 0 e 1; α = parâmetro de formatação ON/OFF.

2.1.4 Categorias de serviço

A tecnologia de redes de comunicação ATM foi projetada para oferecer suporte a uma gama variada de tipos de tráfego, com ênfase particular em aplicações multimídia, como voz e vídeo, mas com flexibilidade para atender com eficiência aplicações que não possuam requisitos rígidos de desempenho [PRY95]. Redes ATM são baseadas em *conexão*, ou seja, antes da transmissão de dados, um circuito deve ser estabelecido, reservando recursos da rede ao longo do caminho conectando a fonte com o destino do tráfego. Enquanto que as *conexões* são baseadas em *circuitos*, a *transferência de dados* e o *chaveamento* são baseados em *pacotes*. Outra característica importante é que pacotes ATM (denominados *células*) possuem tamanho fixo: 53 bytes. Tal característica objetiva reduzir a complexidade do projeto de roteadores ATM [DAL04].

A exemplo de uma rede ATM, uma NoC também possui tem por objetivo oferecer alto desempenho às aplicações que nela executam. No entanto, algumas diferenças básicas podem ser observadas entre NoCs e redes ATM. Uma delas é o modo de chaveamento, que além de poder ser *de circuitos* ela poder também ser *de pacotes* (Seção 1.1). Outra característica das NoCs é o tamanho de seus pacotes poder ser variável. Conforme já discutido na Seção 1.1, o presente trabalho pressupõe o uso do chaveamento de pacotes. Os conceitos de geração de tráfego que serão apresentados no Capítulo 4 consideram também a variação do tamanho dos pacotes. Nesta Seção são mostradas *categorias de serviço*, um conceito oriundo da tecnologia ATM, que têm por objetivo especificar fontes de tráfego de acordo com uma série de requisitos de desempenho. A geração de tráfego utilizando categorias de serviço é destinada a redes que oferecem suporte à QoS. Alguns dos trabalhos utilizados na elaboração desta dissertação tratam de projeto de NoCs com suporte a QoS [YUM00] [RIJ03] [BOL04a] [VEL04] [GOO05][RAD05][SAN05].

A divisão de aplicações em categorias envolve a diferenciação das mesmas quanto à variabilidade de suas taxas de transmissão, latência e variação de latência dos fluxos de dados. Tais parâmetros são especificados por *descritores de tráfego*, através dos quais a fonte de tráfego solicita *serviços* a rede de comunicação. A rede por sua vez atribui níveis de QoS ao tráfego oriundo da fonte solicitante, de modo a procurar atender aos seus requisitos de desempenho. Os tipos de

serviços que uma rede pode disponibilizar são mostrados na Seção 2.1.5. O compromisso básico que deve haver entre a rede e a aplicação é que, uma vez a conexão estabelecida, a QoS negociada é assegurada para todos os pacotes que respeitam os limites de taxa de injeção estabelecidas nos parâmetros especificados pelos descritores de cada categoria de serviço. São apresentadas nesta Seção as categorias de serviço CBR, VBR, ABR e UBR.

CBR

A categoria de serviço CBR (*Contant Bit Rate*) é usada para conexões que exigem uma quantidade estática de largura de banda, que permanece continuamente disponível durante o tempo de vida da conexão [HÄN98]. Um serviço CBR aloca largura de banda com base no descritor de tráfego PCR (*Peak Cell Rate*), que indica o limite máximo para a taxa de injeção de dados. Neste caso, a fonte injeta pacotes na taxa de pico ou abaixo do PCR em qualquer momento e durante o tempo que desejar (ou não emitir nada) [CAS03].

As aplicações que utilizam essa categoria de serviço normalmente têm requisitos de latência rigorosos, como no caso de aplicações de tempo real, áudio e vídeo não compactados [YUM00]. Outro exemplo de aplicação são conexões de voz com taxa de transmissão de 64kbits/s [HÄN98].

VBR

A categoria VBR (*Variable Bit Rate*) é utilizada por fontes que geram dados a taxas de injeção que variam com o tempo. Esta categoria utiliza os descritores de tráfego PCR, SCR (*Sustainable Cell Rate*) e MBS (*Maximum Burst Size*). O parâmetro SCR representa a taxa de transmissão de pacotes média esperada ou obrigatória atingida ao longo de um intervalo pré-determinado [TAN97]. O parâmetro MBS especifica o número máximo de pacotes que podem ser transmitidos pelo gerador de tráfego na taxa de pico, ou seja, o tamanho máxima de rajada. A categoria VBR se divide em duas sub-categorias, rt-VBR e nrt-VBR.

As aplicações que utilizam a categoria de serviço rt-VBR (*real time VBR*) não possuem a mesma garantia de QoS dada em serviços CBR. No entanto, é permitido a tais aplicações uma melhor utilização do canal, pelo fato dos dados não serem enviados continuamente, o que torna possível somente a alocação necessária de recursos (como por exemplo, filas) nos roteadores que estiverem recebendo os dados [CAS03]. As aplicações que utilizam essa categoria de serviço são as de tempo real (como voz e vídeo compactados) e exigem controle rígido de latência média e variação de latência.

A sub-categoria nrt-VBR (*non-real time VBR*) é recomendada para uso em aplicações que não sejam de tempo real, mas variam a sua taxa de injeção de pacotes. Um exemplo de aplicação que utiliza esta categoria é a de correio eletrônico multimídia [TAN97].

ABR

A categoria de serviço ABR (*Available Bit Rate*) foi projetada para tráfegos cuja variação de largura de banda é praticamente desconhecida [TAN97] e que pode sofrer ajustes, de acordo com a demanda de tráfego [DAL04]. A utilização dessa categoria de serviço evita a necessidade do estabelecimento de um compromisso em longo prazo com uma largura de banda fixa.

A categoria ABR é especificada pelos descritores MCR (*Minimum Cell Rate*) e PCR, este último já descrito anteriormente. O descritor MCR especifica a menor taxa de transmissão de pacotes. Uma fonte de tráfego ABR deve utilizar largura de banda com valor situado entre MCR e PCR, podendo variar dinamicamente durante a vida útil da conexão. É possível, por exemplo, especificar que a capacidade entre dois pontos dever ser de 5Mbps, podendo haver, no entanto, picos de 10Mbps. A partir de então, a rede garantirá 5Mbps o tempo inteiro e fará o possível para fornecer 10Mbps, embora não possa dar essa certeza. Esta categoria de serviço é utilizada por aplicações que não possuem requisitos de tráfegos de tempo-real [YUM00].

UBR

A categoria de serviço UBR (*Unspecified Bit Rate*) não solicita serviços de rede com garantias, sendo adequada na transmissão de pacotes de Internet. Os pacotes UBR são aceitos e encaminhados, caso a capacidade da rede não for ultrapassada. Caso haja congestionamento, os pacote UBR são descartados.

2.1.5 Serviços de comunicação

Em redes de comunicação de dados, a divisão do tráfego em *tipos de requisição de serviço* pode proporcionar uma alocação de recursos da rede mais eficiente. Neste caso *a rede* define quais usuários, aplicações, dispositivos, fluxos e conexões possuem maior prioridade no atendimento de requisições a serviços, ou seja, *quem* obtém os recursos e *quanto* destes recursos são utilizados. Para realizar essa classificação, a rede faz uso dos parâmetros especificados pelos *descritores de tráfego*, vistos na Seção 2.1.4. Pacotes com determinada característica (por exemplo, carga oferecida máxima (similar ao PCR), tamanho máximo de rajada (similar ao MBS)) são analisados e classificados em um nível de prioridade que deve ser considerado pelo roteador no momento em que ele decidir qual pacote encaminhar, ou ainda, quanta largura de banda tais pacotes terão à disposição.

Os serviços de comunicação *best-effort*, *differented services* e *guaranteed throughput* são aqui mostrados. Tais serviços foram definidos para especificação de serviços oferecidos por redes ATM.

Best-effort

Os pacotes pertencentes à classe de tráfego *best-effort* (BE) não possuem garantias de tempo de entrega. A rede neste caso realiza seu melhor esforço para encaminhar o pacote para o seu

destino [DAL04]. Uma rede que oferece o serviço *best-effort* é caracterizada por filas FIFO, as quais não diferenciam os fluxos das aplicações entre si. No contexto de NoCs, os pacotes são injetados pelo núcleo na fila de entrada do seu roteador, desde que haja espaço nesta fila. Dentro da rede, a mesma estratégia é utilizada no encaminhamento de cada flit do pacote do canal de saída de um roteador para um canal de entrada de um roteador vizinho.

Os serviços oferecidos pela rede em resposta a requisições *best-effort* caracterizam-se por serem imprevisíveis e não-confiáveis. As categorias de serviço atendidas por serviços *best-effort* são ABR e UBR.

Differented Services

Na classe de tráfego *differented services* (DS) alguns fluxos possuem prioridade em relação a outros, tendo direito, por exemplo, a um tratamento mais rápido e/ou maior largura de banda disponível. Tal priorização se define normalmente através de dados estatísticos, não oferecendo às aplicações pertencentes a essa classe garantias rígidas de desempenho. A categoria de serviço atendida pelos serviços diferenciados é a nrt-VBR.

Guaranteed Throughput

Na classe de tráfego *guaranteed throughput* (GT) ocorre a reserva absoluta de recursos da rede para um tráfego específico, ao contrário do que acontece em redes *best-effort*.

Para obtenção dos recursos, o tráfego injetado pelas fontes deve estar de acordo com um conjunto de restrições [DAL04] especificadas por descritores de tráfego de categorias de serviço. É estabelecido desta forma um *contrato de serviço* entre a fonte e o destino de tráfego. As categorias de serviço atendidas pelos serviços *guaranteed throughput* são CBR e rt-VBR.

2.2 Avaliação de desempenho

Como exposto na Seção 1.1, uma NoC é formada por núcleos autônomos interligados através de roteadores. Duas características que compõem uma NoC são os *serviços* que ela oferece e um *sistema de comunicação*. [RIJ03] descreve como alguns dos serviços essenciais a *integridade dos dados* e *garantia de vazão e latência*. O sistema de comunicação deve garantir a entrega dos dados gerados em uma determinada fonte para um dado destino. A avaliação de desempenho tem como objetivo verificar se a rede está oferecendo os serviços de maneira a garantir a correta entrega das mensagens com as restrições de vazão e latência das aplicações.

Nesta Seção são mostrados parâmetros de desempenho utilizados em redes de comunicação e que são considerados no projeto de NoCs.

2.2.1 Vazão

O parâmetro *vazão* é à taxa na qual pacotes são encaminhados pela rede [DAL04], ou seja, a quantidade de informação encaminhada por unidade de tempo. Este parâmetro é medido através da contagem do número de bits que chegam no destino em um intervalo de tempo para cada fluxo (par origem-destino).

A *largura de banda* (*bw*) de um canal é a sua capacidade máxima para transmissão de dados. A equação abaixo apresenta o cálculo para a largura de banda.

$$bw = \frac{n\_bits}{T} \qquad \text{Equação 1}$$

onde:

n_bits: largura do canal;

T: período de relógio utilizado.

A vazão de tráfego em determinado canal (*vazao*) é a taxa na qual os dados são transmitidos, sendo expressa na seguinte equação

$$vazao = \frac{bits\_trans}{n\_ciclos\_sim * T} \qquad \text{Equação 2}$$

onde:

bits_trans: número de *bits* transmitidos;

n_ciclos_sim: número de ciclos para que todo o tráfego seja entregue aos destinos;

A vazão relativa (*vazão_rel*) corresponde ao percentual de ocupação do canal de comunicação, ou seja, a fração utilizada da capacidade máxima do canal. Este parâmetro também é conhecido como *tráfego aceito*. Um valor de vazão relativa igual a 1 significa que o canal de comunicação está sendo utilizado em 100% de sua capacidade. A vazão relativa é expressa pela seguinte fórmula

$$vazao\_rel = \frac{vazao}{vazao\_\max} \qquad \text{Equação 3}$$

2.2.2 Latência

A latência é um parâmetro relacionado com a quantidade de tempo que o pacote leva para sair de um ponto a outro da rede. Normalmente quantifica-se a latência através do número de ciclos de relógio gastos para o pacote percorrer um caminho. Dependendo a partir de que momento se

quer medir a latência e a quantidade de recursos da rede utilizados, podemos ter diferentes medidas de latência.

A *latência de rede* é o tempo decorrido a partir do momento em que a transmissão de um pacote é iniciada (a partir do momento em que o primeiro flit é injetado na rede), até o momento em que seu último flit é recebido no nodo destino [DUA03]. Neste caso é considerada apenas a latência relacionada aos roteadores que constituem o caminho percorrido pelo fluxo considerado.

No contexto deste trabalho, consideramos a *latência total* como sendo o tempo decorrido desde a *criação* de um dado pacote até o momento em que seu último flit atinge o destino, considerando eventuais permanências de flits em *buffers* de roteadores. Tal interpretação é válida por considerar as contenções que o pacote pode sofrer devido às condições adversas de tráfego que podem surgir ao longo do caminho percorrido. Assim, o fato de determinado pacote entrar na rede muito tempo após ele ser criado indica congestionamento na rede.

Aplicações do tipo *real-time* possuem requisitos rígidos de latência (afinal, devem ser atendidas em um curto espaço de tempo), enquanto aplicações multimídia devem apresentar pouca *variação* nos valores de latência (pelo fato de tais aplicações não tolerarem distorções de imagem ou som durante sua execução).

2.3 Conclusões

Neste Capítulo foram apresentados conceitos básicos sobre geração de tráfego e avaliação de desempenho em redes de comunicação de dados.

Através da *geração de tráfego* caracterizam-se as aplicações que executarão sobre a rede. Tal caracterização envolve a definição de origens e destinos de tráfego (padrões de *tráfego espacial*) e as *taxas de injeção* de dados. Com a *modelagem de tráfego* atribui-se aos dados gerados características probabilísticas pertencentes à aplicações reais. A divisão do tráfego em *categorias de serviço* estabelece as requisições de serviços a serem atendidos por redes com suporte a QoS. Tais redes devem reconhecer tais requisições e atribuir *serviços de comunicação* específicos, de acordo com as propriedades das fontes de tráfego solicitantes.

A *avaliação de desempenho* permite a verificação do atendimento ou não aos requisitos de desempenho das aplicações que executam sobre a rede. Foram mostrados os parâmetros vazão (taxa de transmissão de pacotes) e latência (tempo de propagação de pacotes na rede).

Os parâmetros mostrados neste Capítulo servirão como base para a geração de tráfego e avaliação de desempenho para NoCs, apresentados nos Capíulos 4 e 5, respectivamente.

3 ESTADO DA ARTE

Este Capítulo apresenta os projetos de NoCs utilizados como referência no desenvolvimento deste trabalho. Nas seções 3.1 e 3.2 são abordados superficialmente aspectos arquiteturais, e a geração de tráfego e avaliação de desempenho mostrados com base nos experimentos realizados. Nas seções 3.3 e 3.4, o enfoque será dado no fluxo de projeto que envolve a construção da rede e a geração de tráfego e avaliação de desempenho. Na seção 3.5 são mostrados outros trabalhos pesquisados relacionados à geração de tráfego e avaliação de desempenho. Finalmente será mostrado na seção 3.6 o posicionamento da dissertação em relação ao estado da arte.

3.1 SPIN

[GUE00], [ADR03] e [ZEF03] apresentam uma das primeiras propostas para redes de comunicação intra-chip. Na NoC SPIN (*Scalable, Programmable, Integrated Network*) os roteadores são interligados segundo a topologia *fat-tree* (Figura 3(b)). Cada roteador conectado aos nodos folha possui quatro nodos terminais, sendo que a rede cresce na razão (n log_2 n)/8, onde *n* é o número de nodos terminais. O chaveamento adotado é o de pacotes, utilizando o modo *wormhole*. O controle de fluxo utilizado é o baseado em créditos. Cada pacote é uma seqüência de flits de 32 bits, sendo a primeira palavra o *header*. O 1º byte do *header* indica o destino (possibilitando à rede ser escalada em até 256 terminais) e os demais bits são utilizados para marcação de pacotes o opções de roteamento. O *payload* carrega os dados úteis, podendo ter tamanhos variados. Finalmente, o *trailer* contém o código de verificação de erros (CRC – *Cyclical Redundancy Checking*).

O roteamento de um pacote na rede SPIN varia de acordo com a direção do deslocamento do mesmo. Quando o pacote “sobe” (afasta-se dos terminais-folha), o roteamento é adaptativo, sendo a escolha da porta de saída em direção ao nível superior da rede realizada de acordo com a disponibilidade da porta e de um mecanismo de escolha randômica, para o caso de haver a mesma disponibilidade em duas ou mais portas de saída. Quando o pacote “desce” (aproxima-se de seu destino), o roteamento é determinístico, sendo a porta de saída a ser utilizada identificada no endereço do destinatário (especificada no pacote pelo seu emissor).

A avaliação de desempenho da rede SPIN é feita através da comparação desta com uma arquitetura de comunicação baseada em barramento. A arquitetura de barramento escolhida foi a PI-Bus [SIE94]. A arquitetura genérica de comunicação é mostrada na Figura 13. Todos os núcleos do sistema são compatíveis com o padrão de interface para interconexão de núcleos em sistemas integrados VCI (*Virtual Component Interconnect*) [VSI00]. *Wrappers* (adaptadores de protocolo) realizam a conversão entre o protocolo VCI e o protocolo da arquitetura de comunicação utilizada.

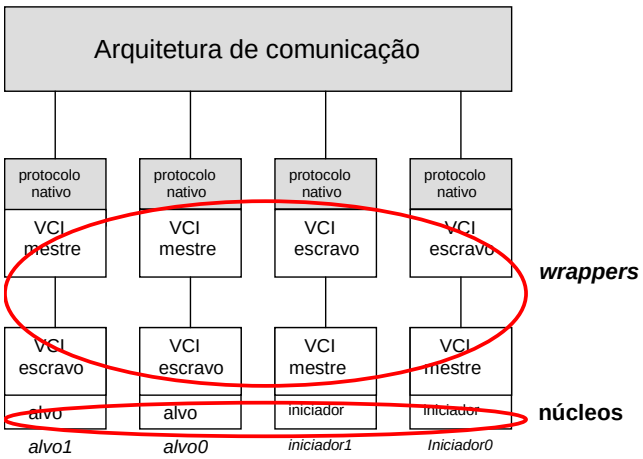


Figura 13 – Arquitetura genérica de comunicação.

A simulação da rede foi realizada no nível de ciclo de relógio utilizando-se o simulador CASS [PET97]. Dois aspectos foram considerados na avaliação de desempenho: a escalabilidade e o ponto de saturação da arquitetura de comunicação.

Para avaliar a *escalabilidade* foram realizados experimentos com 4, 8, 16 e 32 núcleos, sendo mantido fixo o tamanho dos pacotes. Em cada experimento uma metade dos núcleos é formada por iniciadores (instâncias de um modelo de gerador de tráfego - GT), enquanto que a outra metade é composta pelos alvos (instâncias de um modelo de memória RAM – *Random Access Memory*). O padrão espacial de tráfego é mostrado na Figura 14.

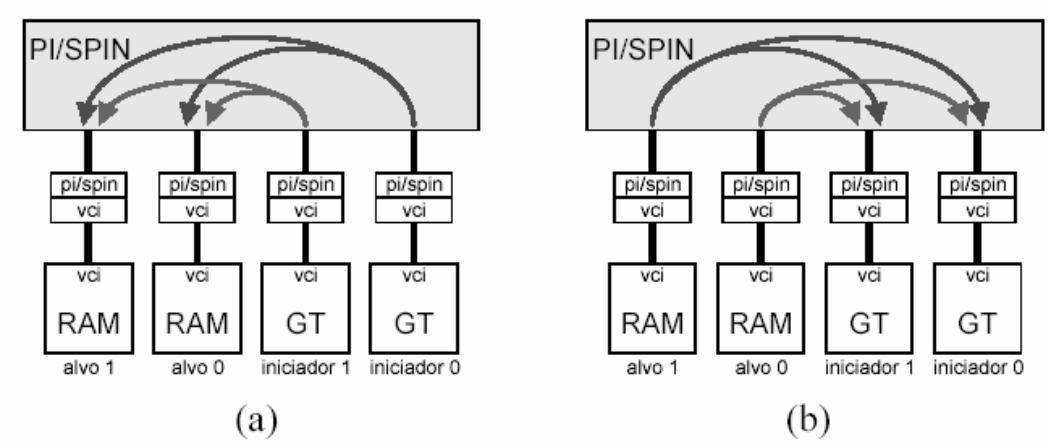


Figura 14 – (a) Requisições dos iniciadores de tráfego; (b) respostas dos alvos [ZEF03].

Na Tabela 2 é mostrado as quantidade de pacotes para requisição e de resposta, bem como a quantidade de flits transmitidos em cada arquitetura de comunicação, com diferentes quantidade de núcleos. Observa-se que na arquitetura SPIN são transmitidos mais flits do que na arquitetura PI-Bus. Isso ocorre devido à estrutura do pacote SPIN, que, ao contrário do pacote que trafega na PI-Bus, possui flits de tamanho limitado a 32 bits, e as palavras de endereço e dado devem ser multiplexadas sendo ainda precedidas por um cabeçalho de roteamento.

A Figura 15 mostra o número de ciclos gastos (*number of cycles*) pelas arquiteturas PI-Bus e SPIN para encaminhar requisições de iniciadores e respostas de alvos modelados segundo o

padrão de tráfego mostrado na Figura 14. Nota-se que o desempenho da arquitetura SPIN vem a ser melhor quando o número de núcleos (*number of cores*) do sistema é maior do que 12 (destaque Figura 15). Os autores do trabalho destacam que o tamanho de pacote utilizado representa o pior caso para a rede. Se forem utilizados pacotes maiores (modelando, por exemplo, transferências de blocos de *cache*) o desempenho será ainda melhor, devido ao fato de haver diminuição da sobrecarga de roteamento, ocasionada pelo processamento do *header* de cada pacote.

Tabela 2 – Dados transmitidos nas arquiteturas PI-Bus e SPIN.

Número de núcleos	Pacotes de requisição VCI	Pacotes de resposta VCI	Flits transferidos no PI-Bus	Flits transferidos na SPIN
4	4	4	8	24
8	46	16	32	96
16	64	64	128	384
32	256	256	512	1536

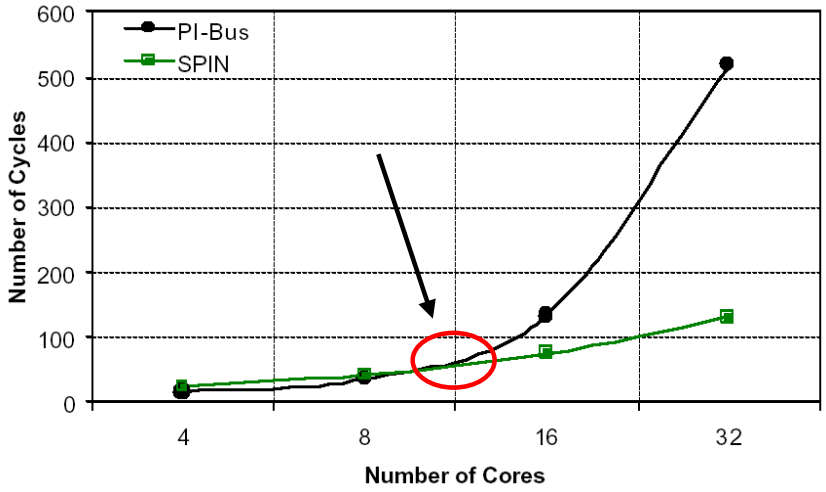


Figura 15 – Avaliação da escalabilidade das arquiteturas de comunicação através da análise da latência [ADR03].

A segunda avaliação refere-se à análise do *ponto de saturação* da rede. Este ponto corresponde à carga máxima que se pode aplicar à rede, com valores de latência situados em um limite aceitável. Se a carga oferecida for maior que o ponto de saturação, a rede não será capaz de aceitar a carga aplicada e serão obtidos valores extremamente elevados de latência. Nos experimentos realizados, o tamanho do sistema foi fixado em 32 núcleos (16 iniciadores de tráfego e 16 alvos), sendo a carga oferecida estabelecida de acordo com a variação dos *tamanhos* das mensagens. A distribuição de tráfego espacial utilizada foi a uniforme (destinos escolhidos aleatoriamente) e a cada instância de modelo de memória RAM foi atribuída uma faixa de endereços. Foram realizadas em torno de 100.000 transações do tipo requisição/resposta.

A Figura 16 mostra os pontos de saturação encontrados para as duas arquiteturas de comunicação utilizadas. Observa-se que, apesar da latência quando da utilização do barramento PI-Bus ser relativamente baixa (abaixo de 50 ciclos), o ponto de saturação obtido também foi baixo (4%). Para a arquitetura SPIN, a latência média esteve sempre em torno de 40 ciclos até atingir o ponto de saturação, que tem o valor de 28%.

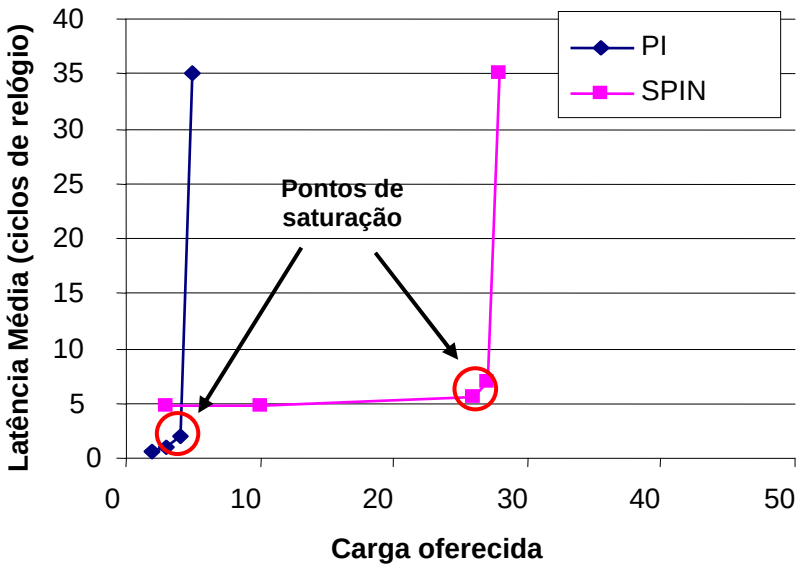


Figura 16 – Avaliação do ponto de saturação das arquiteturas de comunicação [ADR03].

3.2 QNoC

Os trabalhos [BOL04a],[BOL04b] e [BOL04c] propõem um fluxo de projeto e uma arquitetura de rede intra-chip (denominada QNoC), que satisfaça aos requisitos de *QoS* das aplicações (vazão e latência), sem deixar de atender a requisitos de *custo* da rede (consumo de área e energia). Os requisitos de QoS são verificados através de cálculos analíticos e simulações. A personalização da rede é realizada com a alteração de parâmetros arquiteturais da rede original (remoção de enlaces ociosos, dimensionamento de enlaces e variação na freqüência de transmissão), sem comprometer o desempenho do sistema.

A Figura 17 mostra o fluxo de projeto estabelecido para construção da QNoC. Observa-se que tal fluxo é similar ao fluxo de projeto de NoCs ilustrado pela Figura 1. O fluxo de projeto QNoC é dividido nas seguintes etapas:

- *definição dos módulos e sua interconexão*: onde os módulos do sistema são definidos e assume-se que estejam conectados em uma infra-estrutura ideal, com largura de banda infinita e latência programável;
- *caracterizações do tráfego entre módulos*: que ocorre através da análise da interconexão dos módulos e sua especificação de tráfego;
- *validação das caracterizações de tráfego*: onde uma ferramenta em alto nível mede o tráfego entre os módulos e o particiona em *níveis de serviço*. Nesta etapa também são obtidos os requisitos de QoS para cada classe de tráfego;
- *posicionamento de módulos*: para minimizar a densidade de tráfego espacial do sistema;

- *mapeamento da NoC na topologia escolhida*, nesta etapa os núcleos são posicionados na rede e realiza-se análise interna para verificar a utilização dos canais;
- *otimização da rede*: para balancear sua utilização, minimizar custo e atingir QoS. Nesta etapa é possível remover enlaces ociosos, dimensionar enlaces e variar a freqüência de transmissão, com base em medições de carga relativa nos enlaces (avaliação interna);
- *estimativa de custo*. Verifica-se nesta etapa se a otimização da rede conseguiu minimizar custo e atingir QoS.

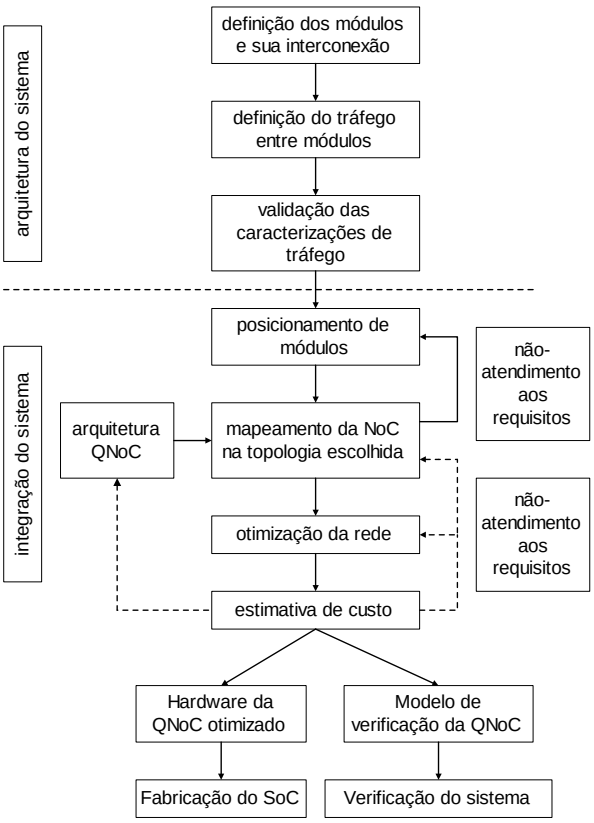


Figura 17 – Fluxo de projeto QNoC.

No experimento realizado, a topologia de rede utilizada é uma *mesh* 4x4. O algoritmo de roteamento utilizado é o XY simétrico. Neste algoritmo, se a coordenada X do destino for maior que a coordenada X da origem, é utilizado o roteamento XY, caso contrário é utilizado o roteamento YX. A vantagem deste método é a utilização do mesmo caminho de roteamento para as comunicações bi-direcionais que ocorrem entre cada par origem-destino. O controle de fluxo utilizado é o baseado em créditos. O tamanho de flit adotado é de 16 bits, sendo que os *buffers* dos roteadores situam-se na entrada dos mesmos e tem capacidade para armazenar 2 flits.

Para transmissão de dados é definida uma *escala de prioridades*, sendo que cada pacote deve estar associado a um *nível de serviço*. Quatro níveis de serviço são definidos: (*i*) *sinalização* (maior prioridade): sinais de controle e interrupção, sendo utilizados pacotes pequenos (2 flits), que devem ser encaminhados na menor latência possível; (*ii*) *tempo real*: processamento de *streams* em aplicações de áudio e vídeo, que exigem alta largura de banda e baixa latência; (*iii*) *leitura/escrita*:

acessos a pequenas quantidades de memória e registradores; (*iv*) *transferência de blocos* (menor prioridade): mensagens longas e grandes blocos de dados como preenchimento de setores de memória *cache* e transferências DMA (*Direct Memory Access*).

A carga oferecida de cada nível de serviço, bem como os requisitos de latência para cada nível de serviço são mostradas na Tabela 3. As distribuições espaciais de tráfego utilizadas foram a *uniforme* e a *não-uniforme*. A carga total oferecida (somatório das taxas de injeção de todos os 16 núcleos) é de 92.16Gbps (16*(2.56+2.56+0.32+0.32)).

Tabela 3 – Modelagem de tráfego – QNoC com freqüência de 1GHz [BOL04a].

Nível de Serviço	Interpretação do tráfego	Tamanho Médio do pacote (*em flits*)	Intervalo de chegada médio (em *ns*)	Carga Total	Requisitos de Latência ETE (fim-a-fim) máximo (para 99% dos pacotes)
Sinalização	A cada 100 ciclos cada módulo envia 1 sinal de interrupção para um módulo aleatoriamente escolhido	2	100	(16 bits x 2 flits)/100 *ns* = **0.32Gbps**	20 *ns*
Real-Time	15 conexões periódicas a partir de cada módulo (para os outros 15) de 320 canais de voz (PCM 64 Kbps)	40	2000	(16 bits x 40 flits)/2000 *ns* = **0.32Gbps**	125μs
Leitura/ Escrita	Transações de leitura e escrita realizadas entre módulos aleatórios a cada 25 ciclos	4	25	(16 bits x 4 flits)/25 *ns* = **2.56Gbps**	150*ns*
Transferência de Blocos	Transações realizadas entre módulos aleatoriamente escolhidos, envolvendo grandes blocos, a cada 12.5μs	2000	12500	(16 bits x 4000 flits)/125000 *ns* = **2.56Gbps**	50μs

A Figura 18 ilustra a carga de tráfego relativa (*Relative load*) de todos os canais da malha, utilizando o padrão de tráfego uniforme (avaliação interna). As linhas (*row*) e colunas (*column*) representam os índices dos roteadores, (por exemplo, o ponto (0,0) corresponde ao roteador 00) e as barras verticais representam a carga relativa entre os canais. Os valores de carga nos canais (0,0)→(1,0) e (2,0)→(3,0) possuem a menor carga relativa no sistema, (com 1 unidade) servindo como medida de referência de carga relativa para outros canais. Verifica-se também que a maior carga relativa pertence ao enlace (1,3)→(2,3).

De acordo com a distribuição de carga relativa mostrada na Figura 18, é realizada a otimização da rede. Tal otimização é realizada através do re-dimensionamento de enlaces, que pode ser feita variando-se parâmetros como número de fios ou freqüência. Caso existam enlaces sub-utilizados, os mesmos podem ser removidos, tornando a rede uma malha irregular. A Tabela 4 ilustra para cada largura de banda alocada[1], os valores obtidos para a utilização média dos canais da rede, bem como os valores de latência média em cada classe de serviço. Para avaliação de

[1] Somatório das larguras de banda em cada enlace da rede.

desempenho deve-se tomar como referência a Tabela 3, que descreve os requisitos de cada nível de serviço. Nas duas primeiras situações (2560 e 1280 Gbps) a rede apresenta desempenho melhor que o requerido, mas apresenta baixos valores de utilização dos enlaces, sendo desta forma sub-utilizada. Para a largura de banda alocada de 850 Gbps obteve-se a melhor relação custo/desempenho (destaque em *itálico*). Reduzindo ainda mais o custo da rede (redução da largura de banda para 512Gbps) os requisitos de QoS para as pacotes pertencentes às classes de sinalização e transferência de blocos não são satisfeitos.

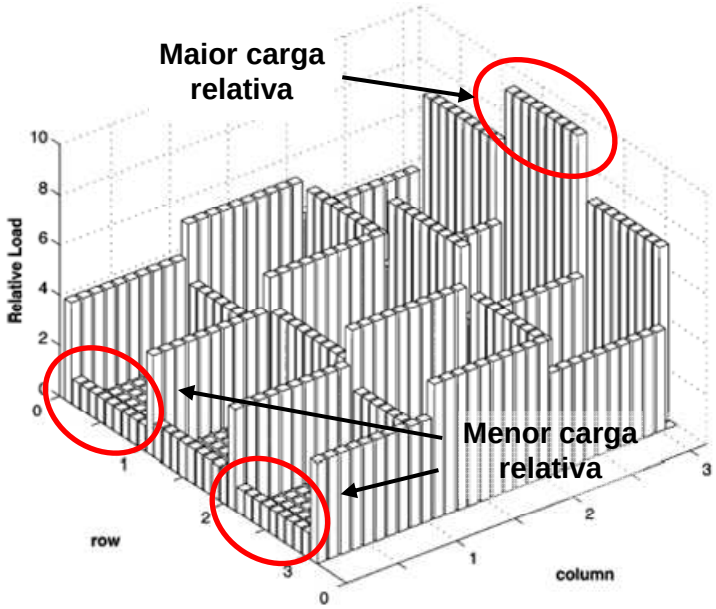


Figura 18 – Carga relativa nos enlaces da malha. As setas apontam para a maior carga (acima à direita) e as duas menores cargas (abaixo à esquerda)[BOL04a].

Tabela 4 – Latência média dos pacotes em função da largura de banda alocada. QoS desejado é marcado em *itálico*. Não-atendimento a requisitos de QoS é marcado em negrito.

Largura de Banda Alocada (Gbps)	Utilização Média do Canal (%)	Latência média (*ns*)			
		Sinalização (para 99.9% dos pacotes)	Real-Time (para 99.9% dos pacotes)	RD/WR (para 99.9% dos pacotes)	Transferência de blocos (para 99% dos pacotes)
2560	10.3	6	80	20	4000
1280	20	11	150	50	12000
850	*30.4*	*20*	*250*	*80*	*50000*
512	**44**	**35**	**450**	**1000**	**300000**

Para avaliar o desempenho também foram gerados gráficos de distribuição de latências, onde no eixo *x* são impressos os valores de latência (*ETE cycles*) e no *y* a percentagem de pacotes que atingiu determinada latência (*% of packets with given ETE*). A Figura 19 mostra a distribuição de latências para o melhor caso (obtido com a largura de banda alocada de 850Gbps). As elipses destacam as quantidades de pacotes que obtiveram os requisitos de QoS atendidos, para cada nível de serviço.

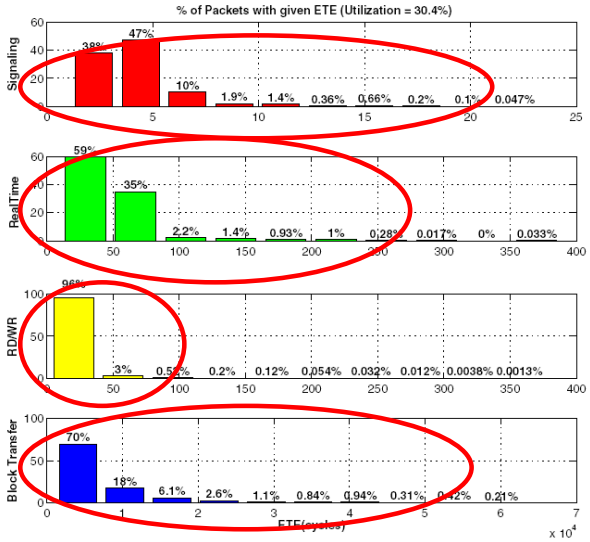


Figura 19 – Distribuição de latências para o melhor caso (850Gbps de largura de banda alocada e 30.4% de utilização da rede) [BOL04a].

Finalmente, observou-se o comportamento da latência média na rede (*mean ETE delay [cycles]*) em função da carga de tráfego oferecida (*Total traffic load [Gbps]*) (Figura 20). Foram escolhidos uma configuração de rede e largura de banda fixos e aplicadas várias cargas de tráfego, expandindo (diminuição de carga) e reduzindo (aumento de carga) o intervalo de chegada dos pacotes para cada nível de serviço. Pode-se observar que enquanto a carga de tráfego está crescendo, a latência média dos tráfegos relacionados à transferência de blocos e leitura/escrita cresce de maneira rápida, mas a latência média de tráfegos sensíveis ao atraso (*real-time* e sinalização) permanece praticamente constante.

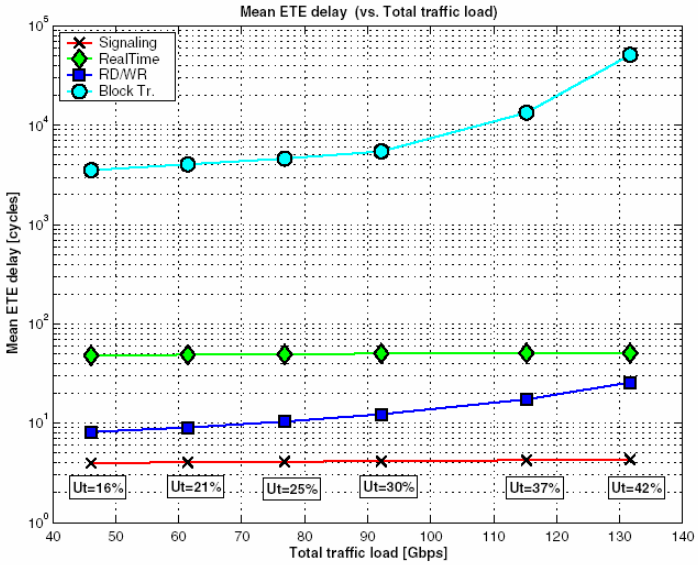


Figura 20 – Carga aplicada *vs.* latência média obtida [BOL04a].

Nos experimentos utilizando tráfego não-uniforme, os requisitos de QoS em termos de largura de banda e latência média e os resultados de desempenho obtidos foram semelhantes aos apresentados pelo cenário de tráfego uniforme, sendo apresentados com mais detalhes em [BOL04a].

3.3 Æthereal

Os trabalhos [GOO02][GOO05][RAD05] propõem a NoC Æthereal. Os serviços que esta rede oferece são baseados em *conexão*, oferecendo desta forma garantias de comunicação aos núcleos. Os principais objetivos do projeto Æthereal são: (*i*) *desacoplamento da computação e comunicação*: como mencionado no início do Capítulo 1, este método de projeto é fundamental para gerenciamento da construção dos SoCs atuais; (*ii*) *oferecer compatibilidade com os projetos de barramento*: através de um modelo de comunicação baseado em transações, sendo que cada núcleo deverá ser *mestre* e/ou *escravo*; (*iii*) *oferecer suporte a aplicações de tempo-real*: onde são dadas *garantias* de vazão e latência; (*iv*) *possuir implementação de baixo custo em termos de área*.

A construção de uma NoC Æthereal baseia-se em um fluxo de projeto que possui como etapas principais a *geração*, *configuração* e a *verificação/simulação* (Figura 21). Segundo os autores, tal divisão traz como benefícios: a simplificação para se adotar heurísticas, aumentando o controle do usuário sobre o processo; redução da complexidade para otimização da rede; 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 enlaces com menor carga), diferentemente do que acontece na QNoC [BOL04a], onde a construção da rede é feita, seguida da análise interna de desempenho, para posteriormente otimizar a rede.

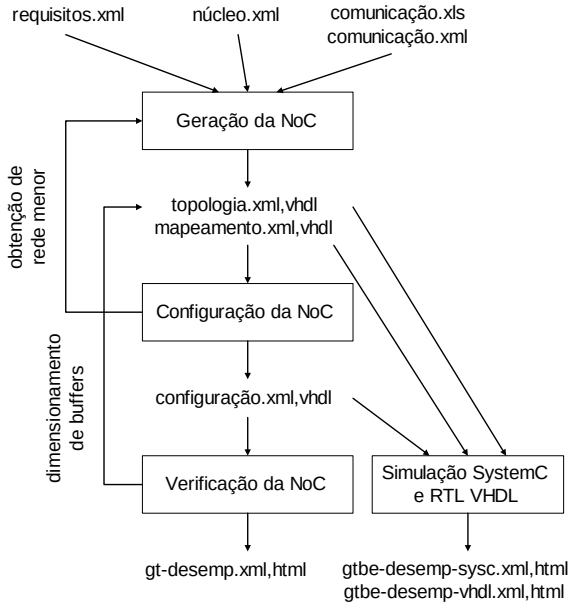


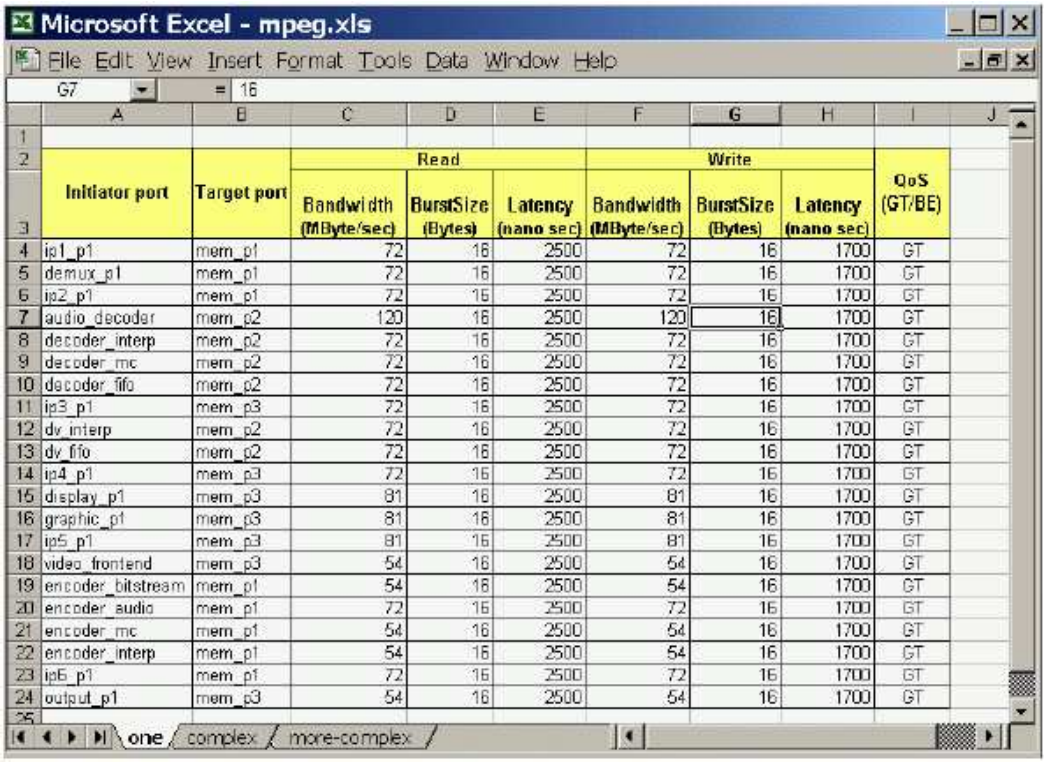
Figura 21 – Fluxo de projeto Æthereal.

A etapa de *geração da NoC* 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 (carga oferecida), 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 *best-effort* ou *guaranteed throughput*). Um exemplo de especificação da aplicação (arquivo de *comunicação*) é mostrado na Figura 22(a);

- *especificação dos núcleos conectados à NoC* (*núcleo.xml*). Cada porta deve especificar o protocolo de comunicação com a rede e o tamanho da largura da palavra de dados. Um exemplo de especificação de núcleos é mostrado na Figura 22(b).



Initiator port	Target port	Read			Write			QoS (GT/BE)
		Bandwidth (MByte/sec)	BurstSize (Bytes)	Latency (nano sec)	Bandwidth (MByte/sec)	BurstSize (Bytes)	Latency (nano sec)	
ip1_p1	mem_p1	72	16	2500	72	16	1700	GT
demux_p1	mem_p1	72	16	2500	72	16	1700	GT
ip2_p1	mem_p1	72	16	2500	72	16	1700	GT
audio_decoder	mem_p2	120	16	2500	120	16	1700	GT
decoder_interp	mem_p2	72	16	2500	72	16	1700	GT
decoder_mc	mem_p2	72	16	2500	72	16	1700	GT
decoder_fifo	mem_p2	72	16	2500	72	16	1700	GT
ip3_p1	mem_p3	72	16	2500	72	16	1700	GT
dv_interp	mem_p2	72	16	2500	72	16	1700	GT
dv_fifo	mem_p2	72	16	2500	72	16	1700	GT
ip4_p1	mem_p3	72	16	2500	72	16	1700	GT
display_p1	mem_p3	81	16	2500	81	16	1700	GT
graphic_p1	mem_p3	81	16	2500	81	16	1700	GT
ip5_p1	mem_p3	81	16	2500	81	16	1700	GT
video_frontend	mem_p3	54	16	2500	54	16	1700	GT
encoder_bitstream	mem_p1	54	16	2500	54	16	1700	GT
encoder_audio	mem_p1	72	16	2500	72	16	1700	GT
encoder_mc	mem_p1	54	16	2500	54	16	1700	GT
encoder_interp	mem_p1	54	16	2500	54	16	1700	GT
ip6_p1	mem_p1	72	16	2500	72	16	1700	GT
output_p1	mem_p3	54	16	2500	54	16	1700	GT

(a)

```
<architecture id="MPEG">
  <IP id="display">
    <initiator id="p1"     protocol="MMBD" word="32"/>
  </IP>
  <IP id="decoder">
    <initiator id="interp" protocol="MMBD" word="32"/>
    <initiator id="mc"     protocol="MMBD" word="32"/>
    <initiator id="fifo"   protocol="MMBD" word="32"/>
  </IP>
```

(b)

Figura 22 – (a) Especificação das aplicações; (b) Especificações dos núcleos [GOO05].

São gerados arquivos XML descrevendo a *topologia* e o *mapeamento* dos módulos, servindo como entrada para a etapa de configuração. A descrição da *topologia* contém a especificação dos parâmetros da *rede* (como tamanho do flit e de *slots* de tempo para alocação de canais), de cada *roteador* (como número de portas e tamanho de buffer) e para cada instância de interface externa (número de portas, conexões por porta e tamanhos de *buffer* por conexão). O arquivo de *mapeamento* contém a descrição da conexão dos núcleos com as interfaces externas da NoC. Um trecho de arquivo gerado de topologia é mostrado na Figura 23.

```
<AENetwork id="MPEG" flitClk="6" slots="128">
  <AERouter id="R0000" iq="8">
    <AEPort id="NI"    link="L_0000" />
    <AEPort id="South" link="L_0000_0100" />
    <AEPort id="West"  link="L_0000_0001" />
  </AERouter>
  <AENI id="NI0101">
    <AEPort  id="Router"       link="L_0101" />
    <SlaveP  id="CONFIG"       conn="1" iq="4"  oq="4"/>
    <MasterP id="display.p1"   conn="1" iq="40" oq="21"/>
    <MasterP id="decoder.mc"   conn="1" iq="40" oq="21"/>
    <MasterP id="decoder.fifo" conn="1" iq="40" oq="21"/>
    ...
  </AENI>
```

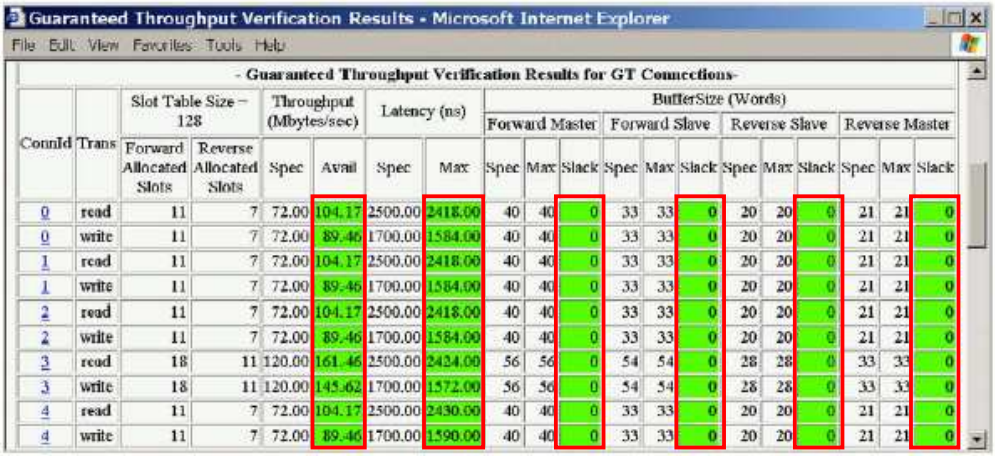
Figura 23 – Trecho de arquivo de topologia da rede [GOO05].

Na etapa de *configuração* da NoC é gerado um arquivo que contém informações para síntese do *hardware*, contendo os valores dos registradores das interfaces de rede e, 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 24.

```
<Connection master="decoder.mc" cidm="0"
            slave="mem.p2" cids="2">
  <Request type="GT" path="3 1 0" credits="33"
           slots="22 23 24 25 26 27 28 29 30 31 32"/>
  <Response type="GT" slots="7 8 9 10 11 12 13"
            path="2 1 0" credits="21"/>
</Connection>
```

Figura 24 – Arquivo de conexão [GOO05].

A etapa de *verificação* da NoC (que no contexto desta dissertação corresponde à avaliação de desempenho) recebe como entrada o arquivo de configuração e são realizados cálculos analíticos para o cálculo da vazão mínima obtida, da latência máxima, e da quantidade máxima de *slots* de buffer utilizada. Os resultados obtidos são comparados aos requisitos das aplicações (descritos nos arquivos de comunicação) e mostrados em uma tabela indicando o atendimento ou não a estes requisitos. Um exemplo de arquivo de verificação é mostrado na Figura 25, onde os destaques indicam atendimento aos requisitos da aplicação.


Guaranteed Throughput Verification Results - Microsoft Internet Explorer

File Edit View Favorites Tools Help

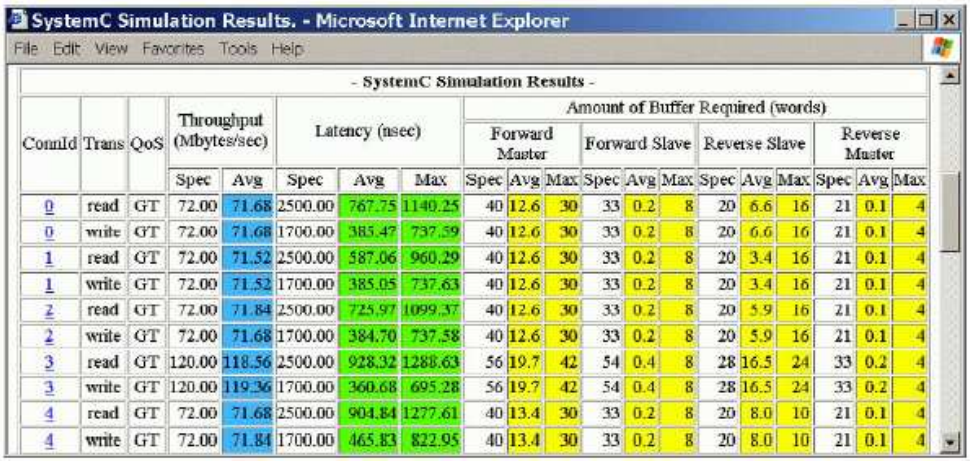
- Guaranteed Throughput Verification Results for GT Connections-

ConnId	Trans	Slot Table Size = 128		Throughput (Mbytes/sec)		Latency (ns)		BufferSize (Words)											
								Forward Master			Forward Slave			Reverse Slave			Reverse Master		
		Forward Allocated Slots	Reverse Allocated Slots	Spec	Avail	Spec	Max	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack
0	read	11	7	72.00	104.17	2500.00	2418.00	40	40	0	33	33	0	20	20	0	21	21	0
0	write	11	7	72.00	89.46	1700.00	1584.00	40	40	0	33	33	0	20	20	0	21	21	0
1	read	11	7	72.00	104.17	2500.00	2418.00	40	40	0	33	33	0	20	20	0	21	21	0
1	write	11	7	72.00	89.46	1700.00	1584.00	40	40	0	33	33	0	20	20	0	21	21	0
2	read	11	7	72.00	104.17	2500.00	2418.00	40	40	0	33	33	0	20	20	0	21	21	0
2	write	11	7	72.00	89.46	1700.00	1584.00	40	40	0	33	33	0	20	20	0	21	21	0
3	read	18	11	120.00	161.46	2500.00	2424.00	56	56	0	54	54	0	28	28	0	33	33	0
3	write	18	11	120.00	145.62	1700.00	1572.00	56	56	0	54	54	0	28	28	0	33	33	0
4	read	11	7	72.00	104.17	2500.00	2430.00	40	40	0	33	33	0	20	20	0	21	21	0
4	write	11	7	72.00	89.46	1700.00	1590.00	40	40	0	33	33	0	20	20	0	21	21	0

Figura 25 – Exemplo de arquivo de verificação [GOO05].

A etapa de verificação oferece suporte apenas às comunicações *guaranteed throughput*, e computa 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 *best-effort* quanto *guaranteeed throughput*. 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. A tabela gerada (Figura 26) é semelhante à gerada no processo de verificação, tendo como adicional resultados do medições de valores médios para vazão, latência e alocação de *buffers*.



SystemC Simulation Results. - Microsoft Internet Explorer

File Edit View Favorites Tools Help

- SystemC Simulation Results -

ConnId	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	767.75	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	385.47	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	587.06	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	385.05	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	737.58	40	12.6	30	33	0.2	8	20	5.9	16	21	0.1	4
3	read	GT	120.00	118.56	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 26 - Exemplo de arquivo com resultados de simulação [GOO05].

3.4 Genko et al.

Os trabalhos [GEN05a][GEN05b] apresentam um ambiente de emulação *hardware-software* onde é possível instanciar e comparar vários tipos de NoCs em nível físico. Esta plataforma de emulação (destaque em pontilhado da Figura 27) possui como principais componentes um *gerador de tráfego* (*Traffic Generator* - TG), um *receptor de tráfego* (*Traffic Receptor* - TRs) e uma *rede de interconexão* (*Network of switches*).

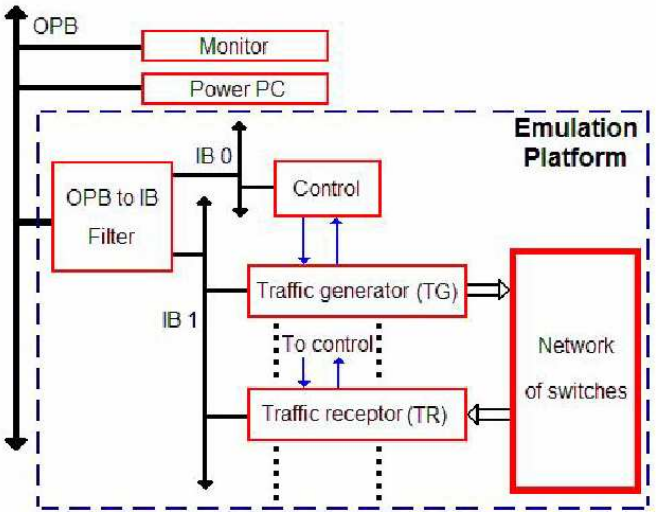


Figura 27 – Arquitetura de emulação [GEN05a].

O fluxo para emulação é mostrado na Figura 28. Na *compilação da plataforma* são especificados os parâmetros da rede, sendo configurado o arquivo em *verilog* que descreve a plataforma. Na *inicialização da plataforma* é realizada a geração de tráfego, que pode ser estocástica ou baseada em *traces* de aplicações reais (tal geração é detalhada adiante). A próxima etapa é a de *emulação em FPGA*, onde são executadas as comunicações e as estatísticas de tráfego

são geradas. Finalmente, no *relatório final* as estatísticas obtidas são enviadas ao usuário descrevendo o tráfego que ocorreu na rede. As informações contidas no relatório de tráfego correspondem à latência média obtida, quantidade de pacotes enviados/recebidos em cada TG/TR, duração de tempo para cada rajada de flits, tempo total de emulação e histograma de flits encaminhados com a granularidade definida pelo usuário.

Figura 28 – Fluxo de projeto da emulação.

Dois tipos de geradores de tráfego podem ser utilizados. No *gerador de tráfego estocástico* o usuário pode especificar a taxa de injeção de dados, bem como as características dos pacotes, utilizando as distribuições de probabilidade. Dois tipos de tráfego estocático podem ser gerados. No que os autores denominam tráfego *uniforme* o tamanho de cada pacote, o intervalo entre geração de pacotes e o destino dos pacotes são aleatoriamente escolhidos. No tráfego de *modo-rajada*, uma cadeia de Markov de dois estados (Seção 2.1.3) é utilizada. No estado *rajada*, pacotes são enviados com intervalo de geração especificado pelo usuário. No estado *estável*, o gerador de tráfego é interrompido durante um intervalo de tempo, também especificado pelo usuário. A distribuição de probabilidade que modela as mudanças de estado é especificada pelo usuário no início da emulação. O momento da troca entre estados é escolhido aleatoriamente por um registrador LFSR (*Linear Finite Shift Register*). O tráfego pode também ser gerado utilizando *traces de aplicações reais*. Neste caso, o gerador de tráfego utiliza uma amostra de um tráfego real obtido a partir da execução de alguma aplicação. Cada *trace* de tráfego consiste de três campos: *tamanho* do pacote, *destino* e *momento* em que o pacote deve ser injetado na rede. Antes de cada emulação, o usuário deve armazenar os *traces* em memória.

Assim como na geração de tráfego, dois tipos de *receptores de tráfego* podem ser utilizados. No primeiro, o usuário especifica a granularidade da análise da emulação, ou seja, a duração (em ciclos de relógio) da coleta de dados. Durante este tempo, são contados os flits

transmitidos, podendo esta contagem ser mostrada em um histograma gerado pelo receptor. O segundo tipo de receptor de tráfego gera um *trace* para cada pacote recebido, no mesmo formato utilizado pelo gerador baseado em *traces*. Neste caso o *trace* recebido é armazenado em memória para posterior análise.

Para validação foram realizados três experimentos. No primeiro, foi utilizada uma rede malha 3x2, sendo um TG e um TR conectados em cada roteador. Dois tipos de taxas de injeção foram utilizados. Para o tráfego denominado pelos autores como *uniforme* (*uniform*), o tamanho dos pacotes foi mantido fixo (em 5 flits) com intervalo de injeção escolhido aleatoriamente entre 4 e 8 ciclos de relógio. Para o tráfego em *rajada*, pacotes de mesmo tamanho são gerados consecutivamente. O gráfico da Figura 29 mostra que o tempo total para encaminhamento de pacotes (*Run time (M Clk)*) pertencentes ao tráfego em rajada é maior que para o tráfego uniforme. Tal resultado se justifica pelo fato de haver maior probabilidade de colisões quando se utiliza o tráfego em rajadas.

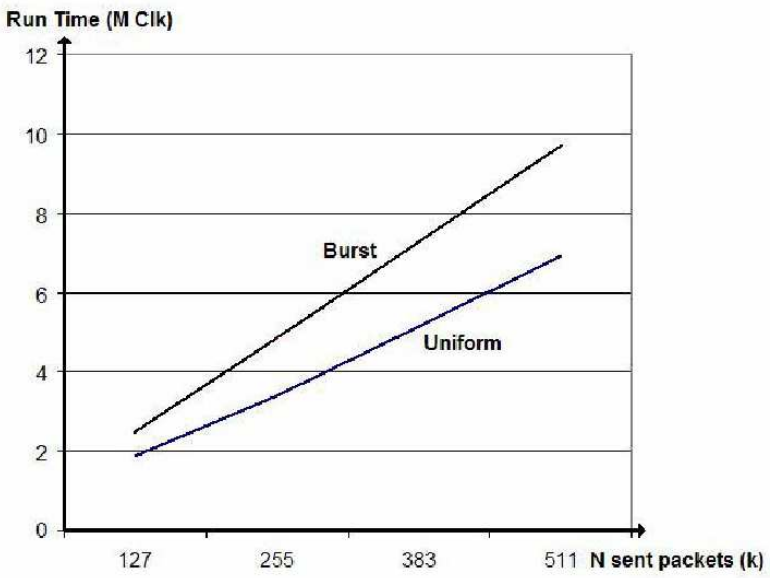
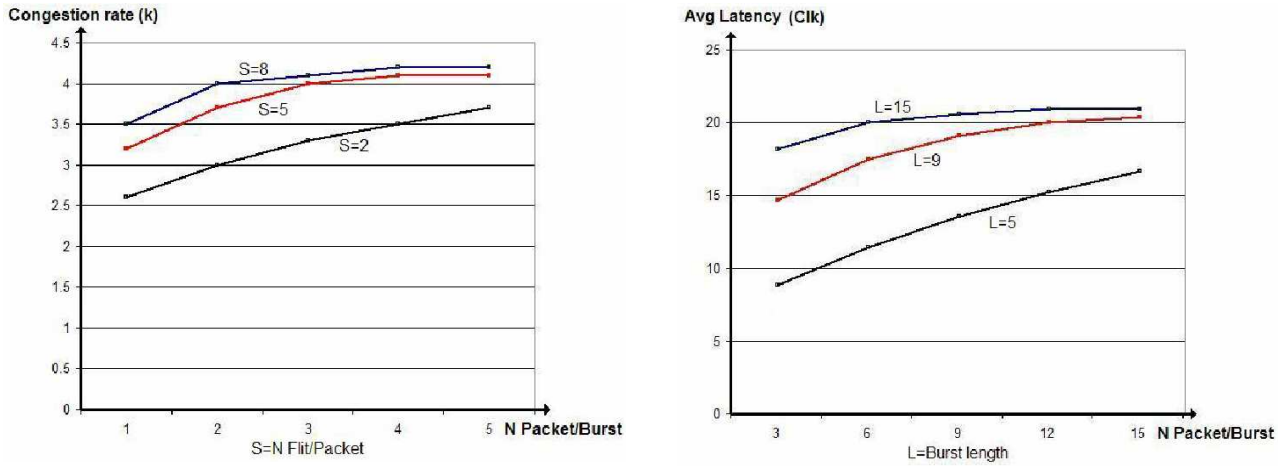


Figura 29 – Comparação tráfego em rajada com o uniforme [GEN05a].

No segundo experimento, foi utilizada uma malha 2x2 com um TG e um TR para cada roteador. Foram gerados em cada TG rajadas de pacotes com modelagem obtida a partir de *traces* de aplicações reais. A avaliação de desempenho considerou a taxa de congestionamento (*Congestion rate (k)*) com relação a diferentes tamanhos de pacotes e tamanhos de rajada (Figura 30(a)). Os resultados mostraram que a taxa de congestionamento não cresce linearmente com o tamanho de pacotes por rajada. Isso porque quanto mais pacotes são enviados em modo rajada, maior o período de inatividade entre duas rajadas. Desta forma, a probabilidade de colisão cresce menos do que linearmente. Finalmente, foram avaliados resultados de latência média (*Avg Latency (Clk)*) para diferentes tamanho de rajadas (*N Packet/Burst*). Através do gráfico ilustrado pela Figura 30(b) verifica-se o ponto de saturação para cada tamanho de rajada.



(a) (b)

Figura 30 – Avaliação do tamanho da rajada [GEN05a]: (a) taxa de congestionamento; (b) latência média.

3.5 Outros trabalhos relacionados

3.5.1 Hu e Marculescu

Hu e Marculescu [HU04] propuseram um algoritmo de roteamento denominado Dyad, o qual combina a baixa latência obtida com roteamento determinístico com a alta vazão obtida com roteamentos adaptativos. Este algoritmo monitora seus vizinhos para verificar congestionamento. Caso haja congestionamento, o algoritmo adaptativo *odd-even* é utilizado, caso contrário, utiliza-se o algoritmo determinístico *oe-fixed*. O algoritmo de roteamento proposto foi comparado com três outros (XY, OE-fixo e *odd-even*). A avaliação de desempenho foi realizada de maneira externa, considerando a latência média obtida pelos pacotes com relação às cargas oferecidas.

Dois cenários de tráfego foram adotados. No primeiro, os núcleos foram conectados em uma malha 6x6 gerando pacotes de 5 flits, com intervalo de geração variando de acordo com uma distribuição exponencial. Os padrões de tráfego utilizados foram o uniforme e o complemento. Os resultados obtidos neste cenário mostraram que no padrão de tráfego uniforme o desempenho utilizando o algoritmo XY foi melhor, enquanto que no tráfego complemento, o algoritmo Dyad apresentou melhor desempenho. Tal resultado ocorreu pelo fato de algoritmos adaptativos apresentarem melhor desempenho com padrões de tráfego que apresentam maior localidade, pois consultam seus vizinhos para obter informações de congestionamento.

No outro cenário simulado, nove núcleos dispostos em uma malha 4x4 foram aleatoriamente escolhidos para gerar tráfego multimídia (dados obtidos a partir de traces reais), e os núcleos restantes gerando tráfego a uma taxa constante. Apesar do padrão de tráfego espacial utilizado ter sido o uniforme, o melhor desempenho foi obtido utilizando-se o algoritmo Dyad. Segundo os autores, tal resultado se justifica pelo fato do tráfego multimídia gerar rajadas de pacotes, o que provoca maior congestionamento, sendo melhor gerenciado por algoritmos adaptativos.

3.5.2 Santi et al.

Santi et al. [SAN05] realizaram experimentos para verificar sob quais condições de tráfego mecanismos de QoS podem ser oferecidos sem acrescentar a complexidade de um mecanismo de QoS explícito. Foram realizados experimentos com uma NoC sem mecanismos de suporte a QoS (*best-effort*) e dividindo o tráfego das aplicações em *guaranteed thrughput* (10% dos pacotes) e *best-effort* (90% dos pacotes). A topologia de rede escolhida foi uma torus 8x8 com chaveamento de pacotes utilizando o modo wormhole. O algoritmo de roteamento escolhido foi o XY. Foram associados 4 canais virtuais a cada canal físico. O método de controle de fluxo adotado foi o baseado em créditos.

Dois padrões de tráfego foram utilizados: uniforme e não-uniforme. Três processos de injeção de injeção de pacotes foram utilizados: (*i*) *Bernoulli*, onde um pacote pode ser inserido na rede a qualquer momento com uma determinada taxa de injeção; (*ii*) *Marcov ON-OFF* (Seção 2.1.3); (*iii*) *Pareto ON-OFF* (Seção 2.1.3), modelando rajadas de pacotes.

A avaliação de desempenho foi realizada de maneira externa com a geração de três tipos de gráfico: carga oferecida por latência média; distribuição de latências; carga oferecida por percentagem de pacotes que atingiram QoS. Com base na análise de desempenho, os autores concluíram que a inserção de mecanismos de QoS se justifica quando o objetivo é melhorar o desempenho das comunicações de aplicações cujo tráfego esperado ocorre em rajada (modelado nos experimentos pelo processo Pareto ON-OFF).

3.5.3 Yum et al.

Yum et al. [YUM00] investigaram a possibilidade de se oferecer suporte de QoS em roteadores que utilizam o modo de chaveamento de pacote *wormhole* para aplicações multimídia (categorias de serviço CBR e VBR) e *best-effort* (categoria de serviço ABR). Para escalonamento de recursos para diferentes categorias de serviço foi utilizado o esquema de alocação de largura de banda *virtual clock* [ZHA91a], onde para cada pacote é atribuído um momento de chegada (tomando como base um relógio global) em cada canal pertencente ao seu caminho. Foi utilizado nos experimentos o roteador *MediaWorm*, cujas chaves estiveram dispostas em uma malha 8x8. O tamanho de flit adotado foi de 32 bits.

Na geração de tráfego VBR (caracterizando uma aplicação MPEG-2), o tamanho de cada frame foi selecionado a partir de uma distribuição normal (Seção 2.1.3) com média 16,66 KBytes, desvio padrão de 3,33 Kbytes e intervalo entre *frames* de 33 ms, o que resulta em uma taxa média de injeção de 4 Mbps (3,2 Mbps mínimo e 4,8 Mbps máximo). Para o tráfego CBR com tamanho fixo de frame de 16,66 Kbytes, com intervalo entre *frames* de 33 ms, o que resulta em uma taxa de injeção de 4 Mbps. Para transmissão de dados na rede, adotou-se para as duas categorias de serviço a divisão de cada frame em vários pacotes de tamanho fixo (exceto o último pacote do frame, para o caso da categoria VBR). Para o tráfego ABR (*best-effort*) fixou-se o tamanho do pacote em 20 flits. A distribuição espacial de tráfego utilizada nos três casos foi a uniforme.

A avaliação de desempenho foi realizada de maneira externa, considerando a média e o desvio padrão dos intervalos entre *frames* para pacotes VBR e CBR, enquanto que para tráfego *best-effort* foi considerada a latência média. Os autores concluíram que a utilização do mecanismo *virtual clock* para gerenciamento de carga fez com que não houvesse influência do tráfego ABR no desempenho dos tráfegos VBR/CBR. Um importante resultado obtido foi relacionado ao mínimo de variação de latência obtido para as categorias VBR/CBR com taxas de injeção até 70%.

3.6 Posicionamento da dissertação em relação ao estado-da-arte

A Tabela 5 mostra um resumo das principais características dos trabalhos pesquisados. Foram considerados o objetivo de cada trabalho, a prática de experimentos com simulação e/ou emulação, a geração de tráfego tanto espacial quanto com taxas de injeção, a utilização de mecanismos de QoS e o método para avaliação de desempenho.

O *objetivo* da maioria dos trabalhos foi oferecer uma arquitetura de comunicação com suporte a QoS, onde aplicações com características em comum foram agrupadas em níveis de serviço. Bolotin et al. [BOL04a] dividiram o tráfego em quatro níveis, enquanto que outros autores ([GOO05][SAN05][YUM00]) ofereceram suporte às aplicações através dos níveis de serviço *guaranteed throughput* e *best-effort*. Os outros objetivos foram comparar arquiteturas de interconexão ([ADR03]), proposição de um ambiente de emulação ([GEN05a]) e avaliação de um algoritmo de roteamento proposto ([HU04]).

A maior parte dos trabalhos realizou experimentos de geração de tráfego e avaliação de desempenho através de simulação. Apenas [GOO05] e [GEN05a] prototiparam seu projetos de NoC. A grande vantagem observada em tais experimentos é a grande quantidade de pacotes que podem ser gerados nos experimentos.

Na geração de tráfego espacial, o padrão uniforme só não foi utilizado por [GOO05]. Para caracterização de tráfego de acordo com taxas de injeção, apenas [GOO05] utilizou somente a taxa de injeção constante. [BOL04a] realizou experimentos tanto utilizando taxas fixas quanto variáveis. Os trabalhos restantes trabalharam somente com taxas variáveis.

Finalmente, observa-se que a maioria dos experimentos realizou análise de desempenho de maneira externa, considerando a NoC como sendo uma caixa preta, sendo os resultados obtidos a partir das interface externas. Apenas [BOL04a] verifica a utilização média dos canais, realizando otimizações na rede a partir desta métrica.

Tabela 5 – Resumo dos trabalhos pesquisados.

Autor/Ano	Objetivo	Simulação/ Emulação	Geração de tráfego espacial	Taxa de injeção	Utiliza mecanismos de QoS?	Avaliação interna/externa
Adriahantenaina (SPIN)/2003 [ADR03]	Comparação de arquiteturas baseadas em rede com barramento	Sim/Não	Uniforme	Variável	Não	Não/Sim
Bolotin (QNoC)/2004 [BOL04a]	Proposta de um fluxo de projeto de NoCs com suporte a QoS	Sim/Não	Uniforme e não-uniforme	Constante e variável (4 níveis de serviço)	Sim	Sim/Sim
Goossens (Æthereal)/2005 [GOO05]	Proposta de uma interface de rede para oferecer suporte a GT e BE	Sim/Sim	De acordo com a aplicação	Constante	Sim	Não/Sim
Genko/(2005) [GEN05a]	Ambiente de emulação para exploração de soluções de comunicação baseadas em NoCs	Não/Sim	Uniforme	Variável	Não	Não/Sim
Hu/(2005) [HU04]	Avaliação do algoritmo de roteamento Dyad	Sim/Não	Uniforme e complemento	Variável	Não	Não/Sim
Santi/(2005) [SAN05]	Avaliação do impacto da inserção de mecanismos de QoS para diferentes padrões de comunicação	Sim/Não	Uniforme e não-uniforme	Variável	Sim	Não/Sim
Yum/(2005) [YUM00]	Avaliação do suporte a QoS oferecido pelo roteador *MediaWorm* para atender tráfego multimídia concorrendo com ABR.	Sim/Não	Uniforme	Variável	Sim	Não/Sim

No contexto deste trabalho, será utilizada a NoC HERMES, como suporte aos métodos de geração de tráfego e avaliação de desempenho. Esta estrutura de comunicação não oferece suporte de QoS às aplicações que nela executam. Os pacotes são tratados da mesma maneira, sendo encaminhados na medida do possível (nível de serviço *best-effort*). O padrão de tráfego a ser utilizado será o complemento, por ser adequado para exploração do que acontece na bisecção XY. Serão simulados e emulados cenários de tráfego com taxas de injeção fixas e variáveis, utilizando nos experimentos a distribuição normal. Essa distribuição será utilizada por permitir melhor visualização do fenômeno *flutuação de carga*, que são desvios de taxas que ocorrem no encaminhamento. Finalmente, a avaliação de desempenho será realizada de maneira interna e externa, considerando métricas que influenciam na contenção e liberação de pacotes na rede.

4 GERAÇÃO DE TRÁFEGO PARA NOCS

Este Capítulo apresenta a primeira contribuição do trabalho, representada pelo método para geração de tráfego, e sua respectiva automação, integrando-a ao ambiente de geração da rede HERMES denominado MAIA [OST05]. Inicialmente, é abordado o conceito de *pacote*, que é a estrutura que deve ter a informação a ser transmitida na rede. Posteriormente, é mostrado o método para geração de padrões espaciais de tráfego (introduzidos na Seção 2.1.1), onde são definidos os iniciadores e alvos das comunicações. Finalmente, é mostrado o método para geração de tráfego com variação na taxa de injeção de dados.

4.1 Pacote

Como mencionado na Seção 1.1, o *pacote* representa uma informação para comunicação em um formato padrão para as mensagens da rede, constituindo-se normalmente de um cabeçalho (*header*), dados úteis (*payload*) e terminador (*trailer*) (Figura 31). Embora o método de geração aqui mostrado seja genérico, é fundamental o conhecimento da estrutura do pacote pelo gerador de tráfego. Tecnologias como ATM, Ethernet e IEEE 802.11 adotam basicamente a mesma estrutura apresentada na Figura 31, podendo no entanto utilizar os campos de maneiras distintas.

Figura 31 – Estrutura de um pacote.

Na NoC HERMES, o *header* especifica o destino do pacote no primeiro flit e o seu tamanho no segundo flit. O *payload* contém os dados úteis transmitidos pelo núcleo conectado ao roteador. Na geração de tráfego, pode-se inserir informações adicionais, relativas à origem do pacote, ao seu momento de criação, número de seqüência (identificação única do pacote na rede), o momento em que o pacote entra na rede e os dados úteis. O número de seqüência é uma informação importante a ser adicionada ao *payload* quando forem utilizados algoritmos de roteamento adaptativos, isto porque os pacotes neste caso podem chegar desordenados em seus destinos, sendo necessário uma referência para seu re-ordenamento. Para o caso dos pacotes de um fluxo seguirem o mesmo caminho (normalmente estabelecido por um roteamento determinístico), o número de seqüência torna-se desnecessário. O *payload* pode ainda armazenar informações relativas à descrição da categoria de serviço ao qual o pacote pertence, quando a rede oferece mecanismos de QoS. Os pacotes da NoC HERMES não possuem *trailer*.

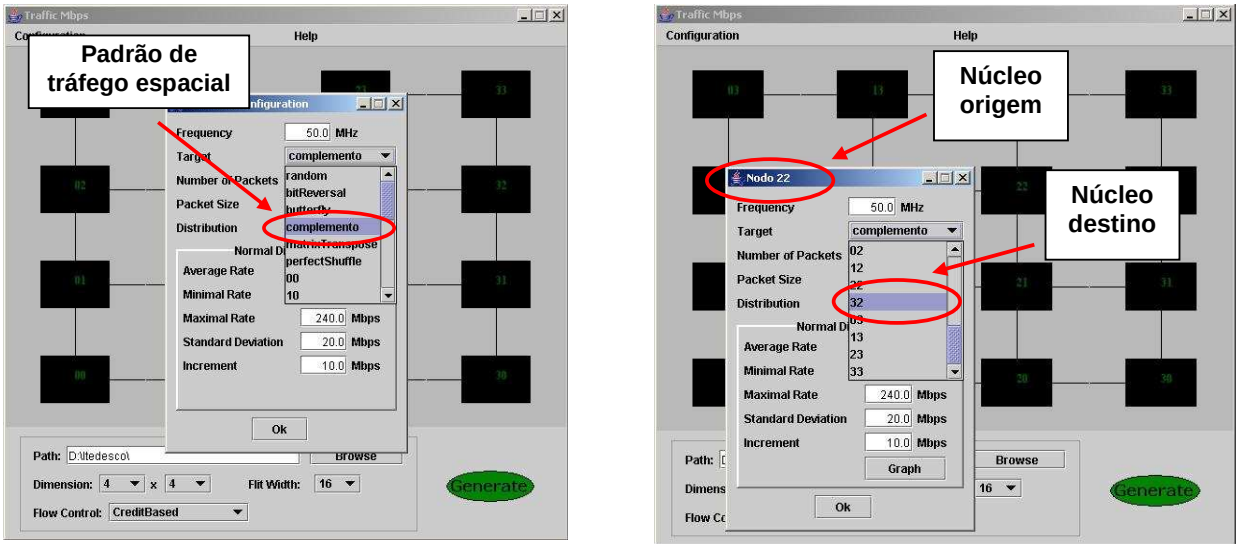
Além de fornecer informações para fins de roteamento, a parametrização das origens e destinos dos pacotes que trafegarão na rede definem a estrutura do *padrão espacial de tráfego*. A parametrização do tamanho do pacote (ou da rajada de pacotes) combinado com o momento de sua inserção na rede especifica a sua *taxa de injeção*. Dessa forma, é possível ao projetista caracterizar as aplicações que executam sobre a rede de acordo com seus requisitos espaciais e temporais de

tráfego. Essa especificação visa oferecer uma referência para o receptor do pacote verificar se o mesmo está chegando com a taxa especificada no momento de sua criação e com a latência característica da aplicação que o gerou (avaliação *externa* de desempenho - Figura 4(a)). A geração de tráfego com padrões espaciais é detalhada na Seção 4.2 e a geração de tráfego na Seção 4.3.

4.2 Geração de padrões para distribuição espacial de tráfego

Como exposto na Seção 2.1.1, os padrões espaciais de tráfego definem a relação entre os iniciadores e destinos de tráfego, e quais são estes iniciadores e destinos. Esse tipo de geração modela comunicações normalmente observadas em aplicações que executam sobre arquiteturas paralelas e também em SoCs multiprocessados, sendo sua utilização adequada para simular a distribuição espacial de pacotes em NoCs. Foram mostrados na Seção 2.1.1 os padrões de tráfego *uniforme*, *não-uniforme*, *bit-reversal*, *perfect shuffle*, *butterfly*, *matrix transpose* e *complemento*.

Para gerar tráfego de maneira espacial, primeiramente é necessário suporte para que o projetista do tráfego possa visualizar a disposição dos nodos da rede, ou seja, como é a sua topologia. A partir desta referência pode-se optar pela geração do tráfego espacial como um todo (qual o padrão a ser seguido por *todos* os nodos da rede) ou então especificar o destino do tráfego gerado por um nodo (ou conjunto de nodos) *específico*. A Figura 32(a) mostra um exemplo de geração de tráfego onde é especificado que todos os núcleos devem seguir o padrão complemento. Outro exemplo é mostrado pela Figura 32(b), onde o nodo com coordenadas **22** gera dados para o nodo **32**.



(a) (b)

Figura 32 – Exemplo de uso da ferramenta para geração de tráfego espacial (a) por todos os nodos da rede e (b) de um nodo específico.

Através do suporte exemplificado na Figura 32, é possível caracterizar o tráfego quanto à sua localidade espacial (onde nodos podem gerar dados para outros mais próximos com maior probabilidade) e temporal (onde nodos podem estabelecer afinidade de comunicação com sub-conjuntos de nodos).

4.3 Geração de tráfego variando taxas de injeção

Com a definição das origens e destinos de tráfego, é necessário definir qual a carga oferecida pelo núcleo iniciador de comunicação para seu respectivo destino (ou conjunto de destinos). A carga oferecida corresponde à fração da largura de banda máxima do canal utilizada para transmissão de dados, ou seja, a relação da taxa de injeção de dados com a capacidade de transmissão do canal.

Quatro parâmetros são considerados na geração de tráfego com taxas de injeção: (*i*) *tamanho do pacote* (*pcksize*), ou seja, sua quantidade de flits; (*ii*) *tamanho da rajada* (*bsize*), que é a quantidade de pacotes que são enviados consecutivamente; (*iii*) *período de ociosidade* (*idle*), que é o intervalo de tempo decorrido entre o envio do *último* flit de um pacote e o início da transmissão do próximo pacote, ou da próxima rajada de pacotes; (*iv*) *intervalos entre saídas* (*outint*) que é o intervalo de tempo decorrido entre o envio do *primeiro* flit de um pacote e o início da transmissão do próximo pacote, ou da próxima rajada de pacotes.

A combinação dos parâmetros acima citados é o que caracteriza determinada aplicação que executará sobre a NoC. Cinco combinações de variação podem ser adotadas: (*i*) manter o tamanho dos pacotes fixo e variar o tempo entre o término da geração de um pacote e o início da geração do próximo pacote (Figura 33(a)); (*ii*) variar o tamanho dos pacotes e manter fixo o intervalo entre o término da geração de um pacote e o início da transmissão do próximo (Figura 33(b)); (*iii*) variar o tamanho do pacote e manter fixo o intervalo entre saídas de pacotes (Figura 33(c)); (*iv*) manter fixo o tamanho dos pacotes e variar o intervalo de saídas dos mesmos (Figura 33(d)) (*v*) variar o tamanho de cada rajada e manter fixo o intervalo entre saídas de rajadas (Figura 33(e)).

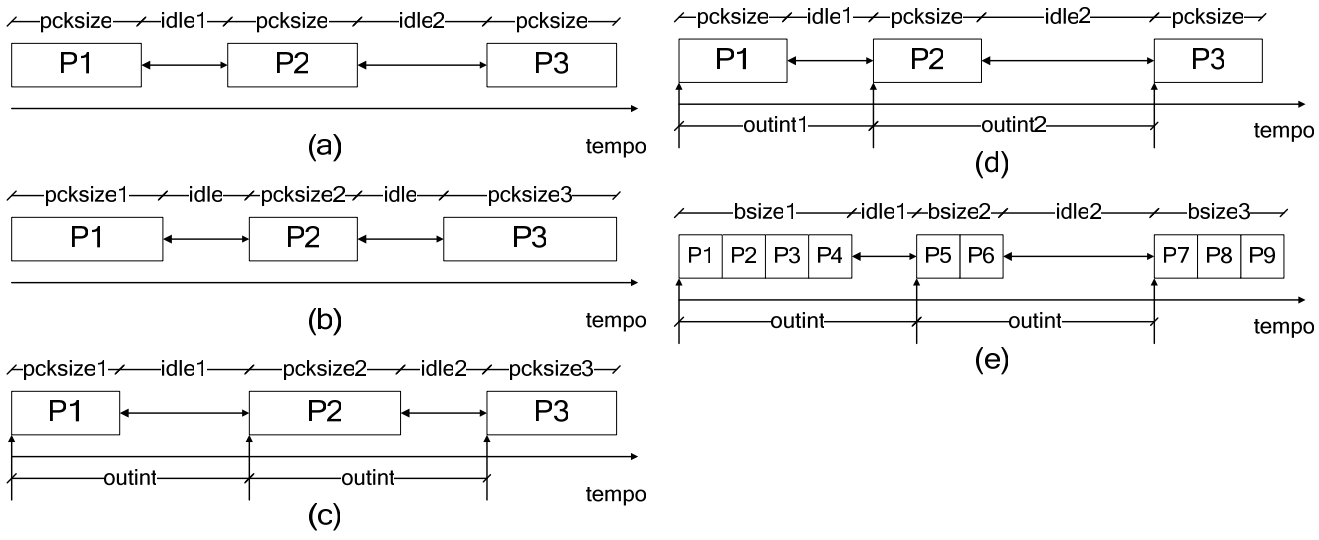


Figura 33 – Variação de taxas de injeção: (a) *pcksize* fixo e *idle* variável; (b) *pcksize* variável e *idle* fixo; (c) *pcksize* variável e *outint* fixo; (d) *pcksize* fixo e *outint* variável; (e) *bsize* variável e *outint* fixo.

4.3.1 Parametrização do tráfego

Para especificação da carga oferecida nos casos ilustrados pela Figura 33 em (a) e (b), é necessária a utilização de variáveis indicando a ocupação do canal e o tempo de ociosidade do mesmo (*idle*). Considerando a ocupação de um canal como sendo

$$ocup = pcksize * ncyclesflit \qquad \text{Equação 4}$$

Onde:
ocup: período de ocupação do canal (em ciclos de relógio);
pcksize: tamanho do pacote (em número de flits);
ncyclesflit: quantidade de ciclos de relógio necessários para transmissão de 1flit.

O parâmetro *ncyclesflit* representa o tempo gasto pelo roteador para transmitir um flit quando não há bloqueio, sendo parametrizado em função da estratégia de controle de fluxo adotada quando do projeto da rede. Na rede HERMES, se o controle de fluxo for baseado em créditos, *ncyclesflit* é 1. Entretanto, se o controle de fluxo for *handshake*, *ncyclesflit* é 2.

Sabendo-se que a carga oferecida corresponde à relação entre a taxa de transmissão do núcleo com a capacidade do canal (*ipr/chr*), temos

$$\frac{ipr}{chr} = \frac{ocup}{ocup + idle} \qquad \text{Equação 5}$$

Onde:
ipr: taxa de transmissão do núcleo gerador (em bps);
chr: capacidade máxima do canal para transmissão (em bps);
idle: período de ociosidade do canal (em ciclos de relógio).

A Figura 33(a) mostra o caso onde é fixo o tamanho do pacote e variado o tempo *idle*. O cálculo para o tempo *idle* envolve o isolamento desta variável na Equação 5, o que leva à Equação 6.

$$idle = ocup * \left(\frac{chr}{ipr} - 1 \right) = pcksize * ncyclesflit * \left(\frac{chr}{ipr} - 1 \right) \qquad \text{Equação 6}$$

Se quiséssemos variar o tamanho do pacote (caso (b)), isolaríamos *pcksize*, conforme a Equação 7.

$$pcksize = \frac{idle}{\left(\frac{chr}{ipr} - 1 \right) * ncyclesflit} \qquad \text{Equação 7}$$

Para o caso ilustrado pela Figura 33(c), temos um intervalo fixo entre saídas de pacotes, combinado com a variação do tamanho dos pacotes. A Equação 8 modela a carga oferecida nesta situação:

$$\frac{ipr}{chr} = \frac{ocup}{outint} = \frac{pcksize * ncyclesflit}{outint} \qquad \text{Equação 8}$$

Onde:
outint: intervalo entre saídas de pacotes (em ciclos de relógio);

É necessário então isolar a variável *pcksize* (que é a que sofre variação), conforme a Equação 9.

$$pcksize = \frac{outint}{ncyclesflit} * \frac{ipr}{chr} \qquad \text{Equação 9}$$

O caso mostrado pela Figura 33(d) mostra a geração de pacotes com tamanho fixo e intervalo de saídas variável. Esta situação é também modelada pela Equação 8, sendo no entanto necessário isolar a variável outint (que é a que sofre variação), levando a

$$outint = pcksize * ncyclesflit * \frac{chr}{ipr} \qquad \text{Equação 10}$$

Finalmente, a Figura 33(e) mostra a geração de rajadas de pacotes. Neste caso, o tamanho de pacotes é fixo, devendo ser fornecido pelo projetista do tráfego da rede. Varia-se então o número de pacotes por rajada (*bsize*). A Equação 11 modela a carga oferecida nesta situação

$$\frac{ipr}{chr} = \frac{bsize * pcksize * ncyclesflit}{outint} \qquad \text{Equação 11}$$

Isolando o parâmetro a ser variado (*bsize*) obtém-se

$$bsize = \frac{(ipr * outint)}{(chr * pcksize * ncyclesflit)} \qquad \text{Equação 12}$$

Para o caso da obtenção de um valor fracionário para *bsize*, é necessário saber do tamanho do último pacote da rajada (que não é múltiplo do tamanho de pacotes escolhido). O parâmetro *lastpcksize* é obtido calculando-se o resto da divisão ilustrada pela Equação 12, ou seja, quantos *flits* faltam para completar a rajada com a taxa de injeção especificada.

$$lastpcksize = (ipr * outint)\%(chr * pcksize * ncyclesflit) \qquad \text{Equação 13}$$

4.3.2 Momento de criação do pacote

Um parâmetro a ser considerado adicionalmente em relação aos ilustrados na Seção 4.3.1 para caracterização de tráfego é o momento em que o pacote é criado pelo núcleo gerador (*pcktmp*). Este momento serve como referência de tempo para o receptor de pacote ter condições de avaliar se o mesmo obedece às restrições temporais especificadas pela aplicação que o gerou.

É necessário ao gerador de tráfego calcular o momento da criação do pacote (que também é o momento ideal para sua inserção na rede) e monitorar um relógio global, que possui a quantidade de ciclos decorridos desde o início das comunicações entre os núcleos. Se o valor deste

relógio for igual ou maior que *pcktmp*, o pacote deve se injetado na rede pelo núcleo que o criou. A hipótese de o pacote ser inserido quando o relógio global for maior que *pcktmp* indica problemas de congestionamento na rede, retardando a sua transmissão. A Figura 34 mostra um pseudocódigo ilustrando a utilização de um relógio global `globalcounter` para invocar o envio de pacotes (rotina `sendpck`).

```
for (nseq=0; nseq<npacks ;nseq++)//para todos os pacotes
  if (packet[nseq].pcktmp >= globalcounter)//verifica se é o momento de enviar o pacote
    for (; sendpck(nseq)!=true; )//só sai do laço quando o pacote nseq for enviado
            ;
```

Figura 34 – Trecho de pseudocódigo para envio de pacotes.

O cálculo de *pcktmp* para os casos ilustrados pela Figura 33((a) e (b)) é realizado de acordo com a Equação 14, sendo variados o período de ociosidade (*idle* – caso (a)) e o tamanho do pacote (*pcksize* – caso (b)).

$$pcktmp = prvtmp + (pcksize * ncyclesflit) + idle \qquad \text{Equação 14}$$

Os casos ilustrados pela Figura 33 (c) e (d) são ainda mais simples, pois apenas levam em consideração os intervalos entre saídas de pacotes (*outint*). Desta forma, o valor de *pcktmp* para cada pacote é obtido de acordo com a Equação 15:

$$pcktmp = prvtmp + outint \qquad \text{Equação 15}$$

Para a geração do tráfego em rajadas, dois tipos de momentos de criação são considerados: o momento de criação do primeiro pacote da rajada (*btmp*) e o momento de criação dos pacotes restantes da rajada. O momento da criação do primeiro pacote da rajada serve como referência para saber qual o *timestamp* do início da próxima rajada. O *timestamp* dos pacotes restantes da rajada serve como referência para geração consecutiva destes. A função abaixo (*burst generator* – *bgen* gerador de rajadas) recebe como parâmetros *bsize* (tamanho da rajada - Equação 12), *btmp* (momento de criação do primeiro pacote da rajada), *pcksize* (tamanho do pacote, especificado pelo usuário) e *lastpcksize* (tamanho do último pacote da rajada - Equação 13).

```
bgen(int bsize,btmp,pcksize,lastpcksize){
  for (int i=0; i>bsize; i++){
       if (i==0){ //primeiro pacote da rajada
            pcktmp=btmp;
            packgen(pcktmp,pcksize,source,target);
            pcktmp = pcktmp + (pcksize*ncyclesflit);
       }
       else if (i==bsize-1){ //último pacote da rajada
            packgen(pcktmp,lastpcksize,source,target);
       }
       else{ //pacotes do meio da rajada
            packgen(pcktmp,pcksize,source,target);
            pcktmp = pcktmp + (pcksize*ncyclesflit);
       }
  }
}
```

Figura 35 – Pseudo-código para geração de uma rajada de pacotes.

O pseudo-código ilustrado pela Figura 36 corresponde à geração de *nbursts* rajadas:

```
for (int j=0; j>nbursts; j++){
     bgen(bsize,btmp,pcksize,lastpcksize);
     btmp=btmp+outint;
}
```

Figura 36 - Pseudo-código para geração de várias rajadas de pacotes.

4.3.3 Geração de tráfego a uma taxa específica

Supor que o projetista da rede queira especificar para o núcleo gerador de tráfego a carga oferecida de 50%. Supor também que um flit seja igual a um bit e que o período de relógio seja de 100ms. Se for utilizado controle de fluxo baseado em créditos (que deve gastar um ciclo para transmitir um flit – *ncyclesflit=1*) obtém-se para o canal a capacidade de transmissão (*chr*) de:

$$chr = \frac{1\,\text{bit}}{\text{ciclo}} = \frac{1\,\text{bit}}{100\text{ms}} = 10\text{bps}$$

Para a carga oferecida de 50% temos então a taxa de 5bps. Para o caso ilustrado pela Figura 33(a), é necessário especificar o tamanho de pacotes, para obtenção do período ocioso (*idle*). Supondo então que o tamanho do pacote seja de 10 flits. O valor de *idle* será de acordo com a Equação 6:

$$idle = pcksize * ncyclesflit * \left(\frac{chr}{ipr} - 1\right) = 10 * 1 * \left(\frac{10}{5} - 1\right) = 10\,\text{ciclos}$$

Supor agora que, ao invés da aplicação especificar o tamanho do pacote, esta especificasse o período ocioso (*idle*) em, por exemplo, 10 ciclos (modelagem segundo a situação ilustrada na Figura 33(b)). Para a carga de 50% seria então obtido para o tamanho de cada pacote o valor de:

$$pcksize = \frac{idle}{\left(\frac{chr}{ipr} - 1\right) * ncyclesflit} = \frac{10}{\left(\frac{10}{5} - 1\right) * 1} = 10\,\text{flits}$$

A geração de três pacotes (P1, P2 e P3) utilizando qualquer uma das duas primeiras abordagens se daria de acordo com o mostrado na Figura 37, já com os valores de *pcktmp* (calculados de acordo com a Equação 14).

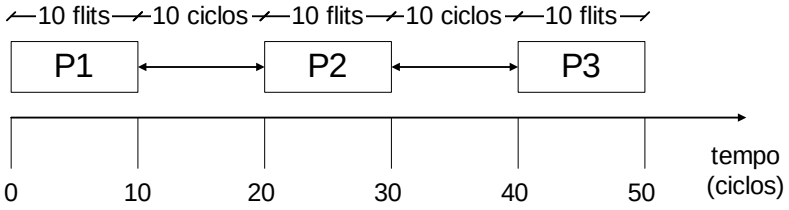


Figura 37 – Geração de tráfego com especificação de idle/pcksize – carga de 50%.

Para o caso ilustrado pela Figura 33(c), supor que o intervalo entre saídas de pacotes fosse de 10 ciclos. Neste caso, para uma carga oferecida de 50%, o tamanho de cada pacote deverá ser, de acordo com a Equação 9

$$pcksize = \frac{outint}{ncyclesflit} * \frac{ipr}{chr} = \frac{10}{1} * \frac{5}{10} = 5 \text{ flits}$$

A geração de três pacotes com este método é mostrada na Figura 38

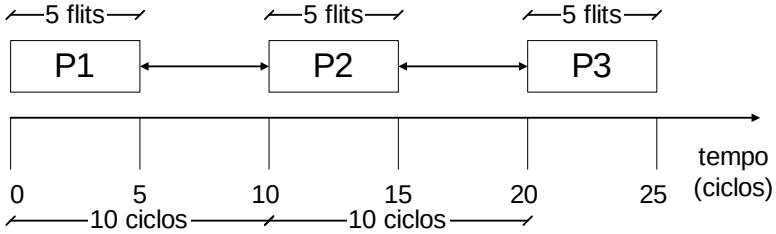


Figura 38 – Tráfego com intervalo de saída fixo – carga de 50%.

O exemplo para geração envolvendo intervalos de saídas variáveis (Figura 33(d)) é similar ao da Figura 38, com a diferença da variável a ser isolada. Considerando a carga oferecida de 50% e pacotes de 5 flits, temos para *outint* o valor de

$$outint = pcksize * ncyclesflit * \frac{chr}{ipr} = 5 * 1 * \frac{10}{5} = 10 \text{ ciclos}$$

A geração de três pacotes sendo gerados desta maneira também pode ser ilustrada pela Figura 38.

Finalmente, para o caso onde a geração envolve rajadas de pacotes (Figura 33(e)) é necessário especificar o tamanho da rajada para a carga de 50%. Supondo que o tamanho do pacote seja de 10 flits e o intervalo entre inícios de rajadas é de 100 ciclos, *bsize* é calculado em

$$bsize = \frac{(ipr * outint)}{(chr * pcksize * ncyclesflit)} = \frac{(5*100)}{10*10*1} = 5 \text{ pacotes}$$

A geração de tráfego ocorre como mostra a Figura 39, incluindo o cálculo de cada *pcktmp* (de acordo com o pseudo-algoritmo da Figura 35).

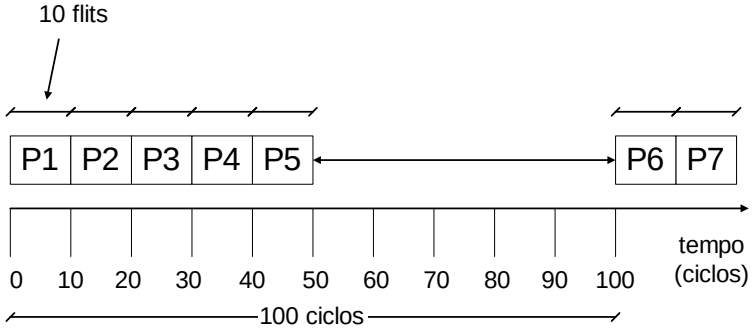


Figura 39 - Tráfego em rajada – carga de 50%.

Se quiséssemos especificar uma carga de 55% (ipr=5.5bps), teríamos um valor de *bsize* de

$$bsize = \frac{(ipr * arrint)}{(chr * pcksize * ncyclesflit)} = \frac{(5.5*100)}{10*10*1} = 5.5 \text{ pacotes}$$

ou seja, um valor fracionário. O núcleo gerador de tráfego deve então gerar 5 pacotes de 10 flits e um último pacote da primeira rajada (P6) com tamanho *lastpcksize* de

$$lastpcksize = (ipr * arrint)\%(chr * pcksize * ncyclesflit) = (5.5*100)\%(10*10*1) = 5 \text{ flits}$$

A Figura 40 mostra a geração de tráfego com carga oferecida de 55%.

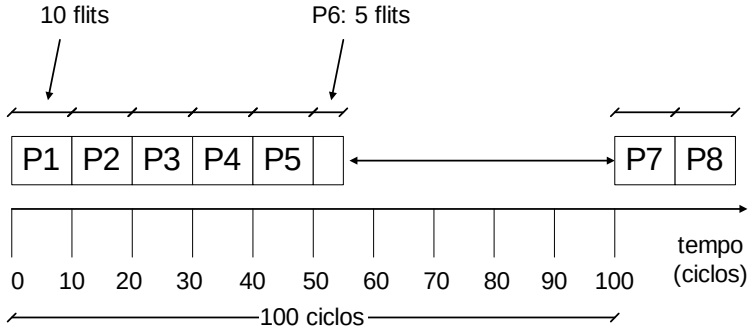


Figura 40 – Tráfego em rajada – carga de 55%.

4.3.4 Modelagem de tráfego

A modelagem de tráfego atribui características probabilísticas de aplicações reais ao fluxo gerado, onde são variadas as taxas de injeção de dados. Dois tipos de geração podem ser adotados. No primeiro, o usuário define todas as taxas de injeção que serão utilizadas. No segundo caso, uma variável aleatória é utilizada para definir as taxas de injeção nas quais serão gerados os dados.

4.3.4.1 Modelagem de tráfego com taxas de injeção pré-estabelecidas

Neste tipo de geração o projetista deve especificar ao gerador de tráfego: (*i*) quais os parâmetros deverão sofrer variação (uma dentre as opções mostradas na Figura 33); (*ii*) os valores das taxas de injeção; (*iii*) quantas gerações deverão ser realizadas para cada taxa de injeção definida, ou seja, quantos pacotes ou quantas rajadas deverão ser geradas.

Inicialmente devem estar definidas as taxas de injeção dos núcleos da rede, onde para cada núcleo é necessário estabelecer basicamente três parâmetros: taxa *mínima* (*minipr*), taxa *máxima* (*maxipr*), e *incremento* (*incr*). A variável *incremento* auxilia na definição de quantas taxas de injeção deverão ocorrer entre a taxa mínima e a máxima, e que taxas são essas. Através da Equação 16 é definida a quantidade de taxas de injeção que vão ser estabelecidas no sistema:

$$nrates = \frac{maxipr - minipr}{incr} \qquad \text{Equação 16}$$

Onde:
nrates: quantidade de taxas de transmissão;
maxipr: taxa máxima de transmissão;
minipr: taxa mínima de transmissão;
incr: incremento.

Dessa forma, temos *nrates* taxas de transmissão, variando de *minipr* a *maxipr* com passo de variação *incr*. A próxima etapa é definir para cada taxa de transmissão estabelecida, o número de gerações de dados. Deve ser preenchida uma tabela com as entradas índice (*index*), taxa de transmissão do núcleo (*ipr*) e quantidade de gerações de pacotes ou de rajadas de pacotes (*ngenerations*) - Tabela 6.

Tabela 6 – Modelo de tabela para atribuição de quantidades de gerações de dados com determinada taxa de transmissão.

index	***ipr***	***ngenerations***
0	*minipr*	
...	...	
nrates	*maxipr*	

A atribuição de valores ao campo *ngenerations* ocorrerá de acordo com a distribuição de probabilidades escolhida. Cada célula preenchida corresponde a uma percentagem do total de gerações especificado pelo projetista do tráfego (*totalrequired*). A Figura 41 mostra o pseudocódigo para atribuição de valores a *ngenerations* para cada índice de uma tabela com os mesmos atributos da Tabela 6. O laço repete tantas vezes quanto forem o número de taxas obtidas em *nrates*.

```
for (int index=0; j=minipr ; i<=nrates ; index++, j+=incr){

    table[index].ipr = j;
    table[index].ngenerations = (int)totalrequired*fprob(j,other_params);
    inserted += table[index].ngenerations;

}
table[maior(table)].ngenerations += totalpacks-inserted;
```

Figura 41 - Trecho de código para preenchimento da Tabela 6.

Seja `fprob(j,other_params)` uma função de distribuição de probabilidades que recebe como parâmetros a própria taxa de injeção `j` e parâmetros adicionais para sua construção (como média e desvio padrão). Esta função auxilia na obtenção de *ngenerations*, que é o campo que descreve quantas vezes deverão ser gerados dados a uma taxa específica (estabelecida pelo campo *ipr*). No contexto deste trabalho, a função de probabilidade *normal* é a utilizada para gerar tráfego com taxas pré-definidas pelo projetista.

A variável *inserted* é importante para os casos onde o valor atribuído a *ngenerations* é menor do que 1. Esta variável estabelece um controle de quantos pacotes foram inseridos, devendo ao final ser subtraída de *totalrequired* para se saber quantas gerações ainda faltam realizar para completar o número de gerações requerido pelo projetista do tráfego. O valor de tal subtração corresponde ao número vezes que serão adicionalmente gerados pacotes, na célula onde estiver localizado o maior número de gerações. Com isso, a maioria dos dados serão gerados de acordo com a taxa média estabelecida quando da especificação do tráfego.

Exemplo de geração

Considerando que o projetista especifique que determinado núcleo deva gerar 1000 pacotes, com taxa mínima (*minipr*) de 80Mbps e taxa máxima de 320Mbps (*maxipr*). O valor da variável incremento (*incr*) é de 10Mbps. Temos então para *nrates*

$$nrates = \frac{maxipr - minipr}{incr} = \frac{320 - 80}{10} = 24$$

Incrementando cada taxa a ser utilizada iniciando de 80Mbps até 320Mbps temos o preenchimento do campo *ipr* (Tabela 7(a)). Considerando para o preenchimento de cada valor de

ngenerations a utilização da função de probabilidade *normal* com média de 190Mbps e desvio padrão de 30Mbps, obtém-se o resultado mostrado na Tabela 7(b). Por problemas de arredondamento decorrente de valores de *ngeneration* menores do que um (*index* 0, 1, 21, 22 e 23), o total de *ngenerations* chegou a 986, faltando 14 para os 1000 requisitados. Nesse caso, essas 14 gerações devem ser realizadas na taxa média (*ipr*=190Mbps), obedecendo a propriedade da curva normal, que deve possuir a maioria de seus valores na média. Desta forma, 146 gerações de pacotes são realizadas na taxa de 190Mbps. A curva gerada para essa distribuição é ilustrada na Figura 42

Tabela 7 – Tabela com taxas de geração preestabelecidas.

index	***ipr***	***ngenerations***
0	80	
1	90	
2	100	
3	110	
4	120	
5	130	
6	140	
7	150	
8	160	
9	170	
10	180	
11	190	
12	200	
13	210	
14	220	
15	230	
16	240	
17	250	
18	260	
19	280	
20	290	
21	300	
22	310	
23	320	

(a)

→

index	***ipr***	***ngenerations***
0	80	0
1	90	0
2	100	1
3	110	3
4	120	8
5	130	17
6	140	33
7	150	54
8	160	80
9	170	106
10	180	125
11	190	132→146
12	200	125
13	210	106
14	220	80
15	230	54
16	240	33
17	250	17
18	260	8
19	280	3
20	290	1
21	300	0
22	310	0
23	320	0

(b)

A geração de dados deverá ocorrer com a escolha aleatória de qualquer uma das células da Tabela 7, sendo decrementado *ngenerations* para cada célula escolhida, até completar a geração dos 1000 pacotes.

É importante salientar que o exemplo acima pode ser aplicado em qualquer um dos casos ilustrados na Figura 33, sendo necessária a substituição dos valores de *ipr* nas equações que modelam cada um dos cenários ali mostrados.

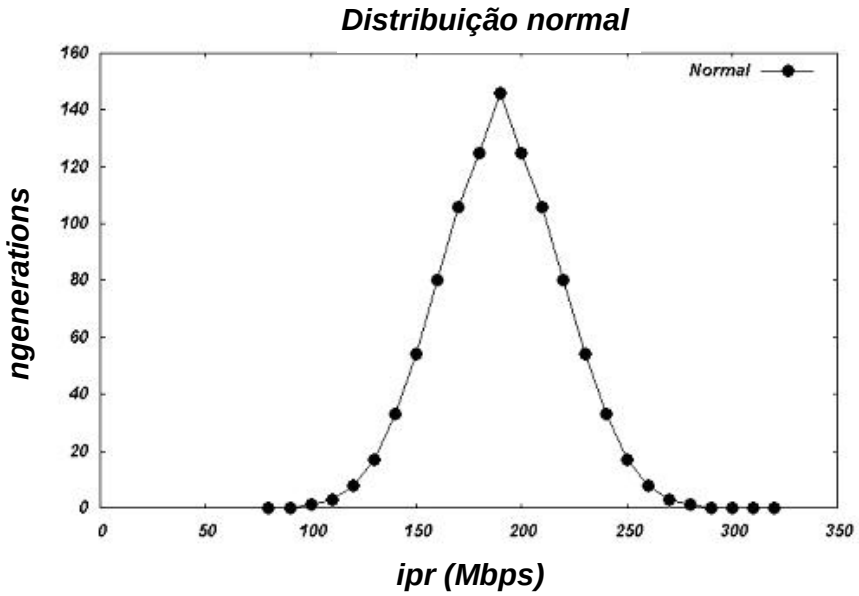


Figura 42 – Curva gerada pela construção da Tabela 7.

4.3.4.2 Modelagem de tráfego com taxas de injeção estabelecidas de maneira aleatória

A segunda hipótese para geração com taxas de injeção é deixar a cargo do gerador de tráfego a definição de quais taxas de injeção de dados serão utilizadas. Este tipo de geração é mais simples que o mostrado na Seção 4.3.4.1, porque o projetista da rede entra apenas com o número de gerações que serão realizadas e o gerador de tráfego retorna as taxas de injeção de dados. No outro caso o usuário fornecia as taxas limite e o passo de incremento para obtenção das taxas restantes. A função de probabilidade calculava o *número de gerações* para cada taxa.

O trecho de código mostrado pela Figura 43 ilustra um tipo de geração de taxas de injeção fornecidas pelo gerador de tráfego. Neste caso, a função de probabilidade retorna *a taxa de injeção* para cada geração de dados. A distribuição de probabilidades Pareto ON-OFF (Tabela 1) é utilizada por Pande et al. [PAN05] para modelagem de tráfego MPEG utilizando este método.

```
pareto_on_off(float alfaon, float alfaoff, long int npacks, float ipr[npacks], float chr){
int r;
float ton, toff;
   for (int i=0; i<ngenerations ;i++){

         r=random();
         ton=pow((1-r),(-1/alfaon));
         toff=pow((1-r),(-1/alfaoff));
         ipr[i]=(ton/(ton+toff))*chr;

   }
}
```

Figura 43 – Exemplo de geração com taxas de injeção escolhidas de maneira aleatória, utilizando a distribuição Pareto ON-OFF.

Como especificado na Tabela 1, atribui-se à variável `r` um valor entre 0 e 1. Um exemplo de curva gerado pelo algoritmo mostrado na Figura 43 é mostrado na Figura 44. São gerados 1000 pacotes, sendo α_{ON}=1.9 e α_{OFF}=1.25. A largura de banda disponível em cada canal é de 100Mbps. Para cada geração (*ngeneration* – eixo x) há um valor de taxa de injeção (*ipr* – eixo y).

De acordo com o tipo de variação escolhida (uma dentre as opções escolhidas da Figura 33) o valor de *ipr* pode ser substituído na Equação 6 (caso (a)), Equação 7 (caso (b)), Equação 9 (caso (c)), Equação 10 (caso (d)) ou na Equação 12 (caso (e)).

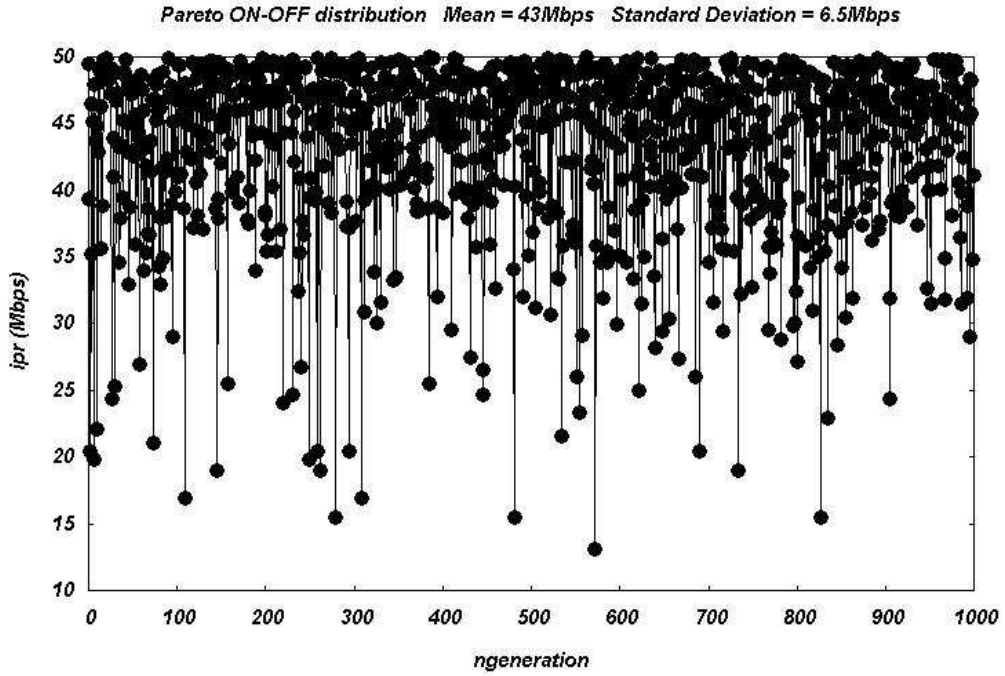


Figura 44 – Gráfico com taxas aleatoriamente geradas.

4.4 Conclusões

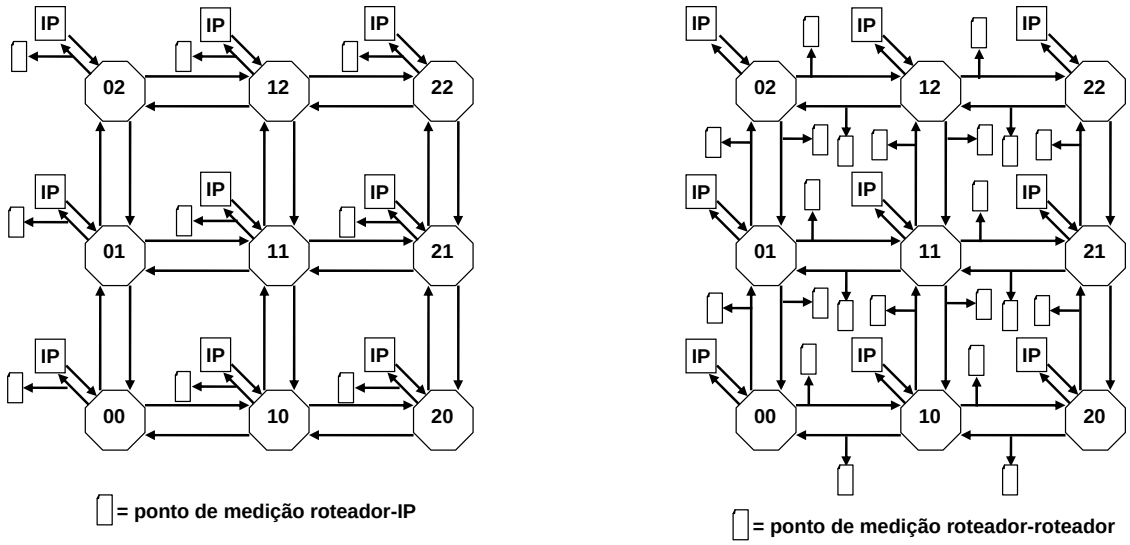
Este Capítulo apresentou a primeira contribuição deste trabalho, um método para geração tráfego com diferentes relações origem-destino e com diferentes taxas de injeção de dados. Através da geração de tráfego o projetista caracteriza as aplicações que executarão sobre a rede. Tal caracterização deve levar em consideração propriedades probabilísticas de aplicações reais. Dependendo da aplicação, são variadas as relações entre as origens e destinos de tráfego (padrão de tráfego espacial), os tamanhos dos pacotes, os intervalos entre gerações de pacotes (ou entre pacotes) e tamanhos de rajadas de pacotes. A geração de tráfego oferece, desta forma, uma referência para avaliação de desempenho quando os dados chegarem aos seus destinos. Isto porque as propriedades agregadas aos dados gerados devem ser mantidas ao longo de seu caminho na rede.

Observa-se a generalidade do método devido aos parâmetros apresentados levarem em consideração um conceito fundamental em redes de comunicação de dados em geral (como os protocolos Ethernet e ATM): o *pacote*. Para a geração de tráfego espacial, é necessário inserir no *header* do pacote o *endereço do nodo destino* do tráfego. Quanto às taxas de injeção são configurados o *tamanho* dos pacotes, o *intervalo entre gerações* de pacotes e o *tamanho das rajadas* de pacotes. Adicionalmente, um parâmetro foi apresentado: o *momento de inserção* de um dado pacote na rede (*pcktmp*). Com este parâmetro, é oferecido ao tráfego gerado propriedades temporais, que, combinados com a quantidade de dados a serem transmitidos, estabelece a taxa de injeção de um dado núcleo.

Resumidamente, a geração de tráfego é dividida em quatro etapas: (*i*) definição do momento de geração de cada pacote; (*ii*) definição dos destinos de cada pacotes; (*iii*) inserção do momento de geração do pacote no *payload*, permitindo o cálculo da latência e avaliações estatísticas de desempenho; (*iv*) parametrização do tamanho do pacote. O próximo Capítulo detalha o método proposto para avaliação de desempenho de NoCs, onde é verificado se a rede consegue atender aos requisitos temporais das aplicações, especificadas na geração de tráfego.

5 AVALIAÇÃO DE DESEMPENHO PARA NOCS

Neste Capítulo apresenta-se a segunda contribuição deste trabalho, o método de avaliação de desempenho utilizado para verificar se os requisitos especificados na geração de tráfego estão sendo atendidos. A avaliação de desempenho é vista neste trabalho sob duas óticas: (*i*) *externa* Figura 45(a), onde a NoC é vista como uma caixa preta, sendo os dados para análise coletados nas interfaces externas da rede; (*ii*) *interna* Figura 45(b), onde os dados para análise são coletados nos enlaces que interligam os roteadores. Através destas duas formas de avaliar desempenho, o projetista pode verificar o tráfego desde a sua entrada na rede, detectando os efeitos causados pela concorrência de recursos pelas diversas aplicações que executam sobre a mesma.



(a) – Avaliação externa (b) – Avaliação interna

Figura 45 - Pontos de coleta para avaliação de desempenho, exemplificada para uma NoC topologia malha 3x3.

O suporte para avaliação externa de desempenho é oferecido por arquivos onde são impressos os dados coletados nos canais de recepção dos núcleos. Um exemplo de arquivo é mostrado na Figura 46. Este tipo de arquivo constitui-se das partes *pacote* (com informações do pacote em si) e *informações adicionais*. Cada linha do arquivo representa um pacote. No arquivo-exemplo da Figura 46, cinco pacotes foram recebidos pelo núcleo 10 (coordenada XY). Todos eles tiveram tamanho de 16 flits (1 flit de destino, 1 flit de tamanho e mais 14 flits de *payload*). Seus números de seqüência são respectivamente 1, 2, 3, 4 e 5. O arquivo também mostra que os pacotes entraram na rede no momento em que forma criados, o que leva à conclusão que os mesmos não sofreram bloqueio em seu ingresso. A parte de informações adicionais inclui os momentos de chegada do primeiro e último flits de cada pacote. A impressão de tais valores deve ser proporcionada pela leitura de um relógio global, que armazena a quantidade de ciclos de relógio decorridos desde o início das comunicações entre os núcleos. Este relógio é o mesmo tomado como referência para a geração de tráfego (Seção 4.3.2).

Na parte *pacote* são mostradas informações sobre a origem do pacote, o seu tamanho,

quando foi criado, quando entrou na rede, seu número de seqüência (que é único em toda a rede) e o restante do *payload*. O restante do *payload* corresponde ao *payload* original do pacote, sem o acréscimo das informações adicionais, necessárias para o cálculo de métricas de desempenho. As informações na parte *pacote* são expressas em hexadecimal. Na parte de *informações adicionais* são guardadas informações do momento em que o primeiro flit do pacote chegou no núcleo (para o cálculo de métricas de ocupação de canal) e do momento em que o último flit chegou (para cálculo da latência), sendo essas informações expressas na base decimal.

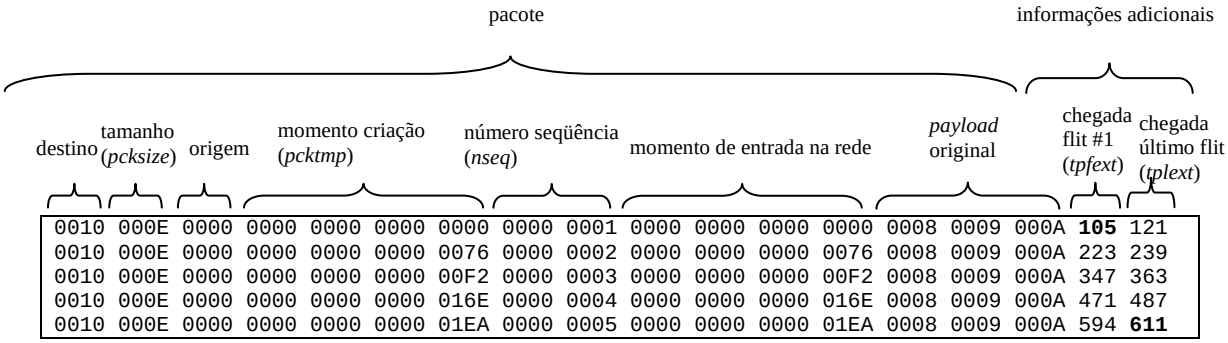


Figura 46 – Estrutura de um arquivo contendo informações do tráfego de um canal que interliga um roteador com seu núcleo.

Os principais registros considerados para avaliação externa de desempenho são os de tamanho do pacote (*pcksize*), os momentos de chegada do primeiro (*tpfext*) e último flit (*tplext*) de cada pacote e os momentos de chegada do primeiro flit do primeiro pacote (*tsfext*) e do último flit do último pacote (*tslext*). A Figura 47 ilustra estes registros. Por exemplo, o primeiro pacote mostrado no arquivo da Figura 46 possui *tpfext*=105, *tplext*=121 e *pcksize*=16. O arquivo ainda mostra que *tsfext*=105 e *tslext*=611.

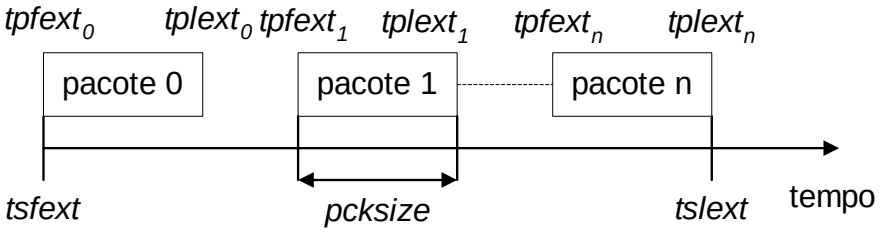


Figura 47 – Campos do arquivo exemplificado pela Figura 46 considerados para avaliação externa.

Onde:

tpfext: momento da recepção do primeiro flit de um pacote pelo núcleo;
tplext: momento da recepção do último flit de um pacote pelo núcleo;
packsize: tamanho do pacote recebido pelo núcleo;

O suporte para avaliação interna é oferecido por arquivos que contém dados com informações do tráfego que ocorre em cada canal que interliga os roteadores. Para cada flit transmitido escreve-se uma dupla no formato ‘(flit, momento de transmissão)’. Da mesma forma que na construção de arquivos para análise externa, os momentos de transmissão dos flits são obtidos através da leitura de um relógio global. No exemplo mostrado na Figura 48 três pacotes são transmitidos (cada duas linhas representa um pacote).

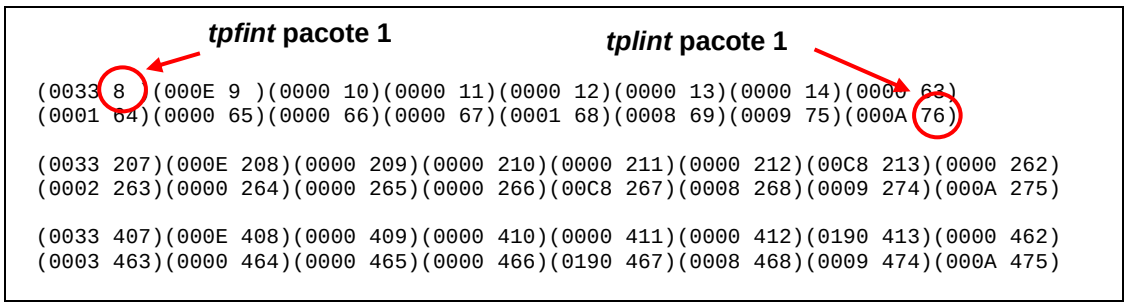


Figura 48 - Exemplo de arquivo para avaliação interna. Cada registro corresponde a um par (flit, momento em que foi transmitido).

Os principais registros utilizados para avaliação interna de desempenho são os mostrados na Figura 49. Tais dados são aos momentos em que o primeiro (*tpfint*) e o último flit (*tpfint*) são transmitidos, o tamanho dos pacotes (*pcksize*) e os momentos inicial (*tsfint*) e final (*tslint*) entre os quais o canal transmitiu dados. Por exemplo, o primeiro pacote mostrado no arquivo da Figura 48 possui *tpfint*=8, *tplint*=76 e *pcksize*=16. O arquivo mostra ainda *tsfint*=8 e *tslint*=475.

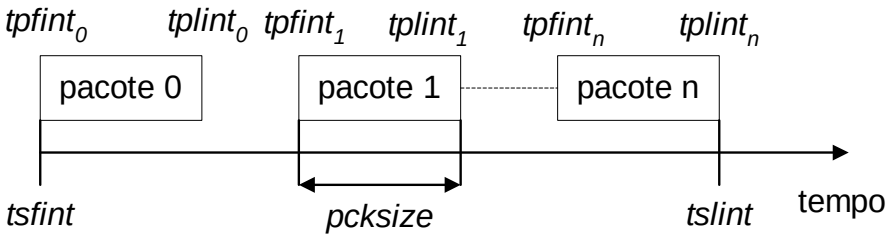


Figura 49 – Campos do arquivo exemplificado pela Figura 48 considerados para avaliação interna.

Onde:

tpfint: momento da transmissão do primeiro flit de um pacote no canal;
tplint: momento da transmissão do último flit de um pacote no canal;
pcksize: tamanho do pacote transmitido no canal;
tsfint: momento da transmissão do primeiro flit do primeiro pacote transmitido no canal;
tslint: momento da transmissão do último flit do último pacote transmitido no canal;

São mostradas nas próximas Seções as métricas que farão uso dos valores lidos nos arquivos de medição de tráfego e que determinam o valor dos parâmetros adotados para avaliação de desempenho. Inicialmente são mostradas medidas para avaliação do tempo de transmissão dos dados (latência e *cpf*). Posteriormente são mostradas métricas para avaliação da taxa de encaminhamento de dados (tráfego aceito e taxa de utilização de canais).

Dois parâmetros são considerados na avaliação das distribuições de valores medidos para os parâmetros de avaliação de desempenho adotados. A *média* (Equação 17) é o número obtido somando-se os valores medidos e dividindo-se a soma obtida pelo número de medições. O *desvio padrão* (Equação 18) indica a dispersão dos valores medidos em relação à média. Um pequeno desvio padrão indica, desta forma, valores da distribuição próximos ao valor de sua média.

$$media = \frac{\sum_{i=1}^{nvalores} x_i}{nvalores}$$

Equação 17

Onde:
nvalores: quantidade de valores da distribuição;
x: valor do parâmetro de desempenho medido.

$$desviopadrao = \sqrt{\frac{\sum_{i=1}^{nvalores}(x_i - media)}{nvalores}} \qquad \text{Equação 18}$$

5.1 Latência

Como mostrado na Seção 2.2.2, a latência é um parâmetro relacionado com a quantidade de tempo que um dado pacote leva para sair de um ponto e chegar a outro na rede. Normalmente quantifica-se a latência através do número de ciclos gastos para um pacote percorrer um caminho. Dependendo a partir de que momento se deseja medir a latência e quais recursos são considerados durante o tráfego dos pacotes, podemos ter diferentes medidas de latência, mostradas na Seção 2.2.2.

No contexto deste trabalho, a latência é considerada como sendo o tempo decorrido entre a criação do pacote (*pcktmp* – Seção 4.3.2) e a chegada do último flit do mesmo em seu destino, sendo desta forma um parâmetro de avaliação externa de desempenho. Esta medida de latência considera as permanências dos pacotes em *buffers* de roteadores e o tempo de arbitragem e de roteamento. A vantagem em se utilizar esta medida de latência é a possibilidade da análise da contenção de pacotes na rede. Por exemplo, a entrada de um pacote na rede muito tempo após ele ser criado indica que houve contenção por parte da rede a este pacote, devido ao congestionamento provocado por pacotes pertencentes a outros fluxos.

A análise da latência tem como objetivo a verificação do atendimento ou não aos requisitos temporais da aplicação que gera o tráfego. Aplicações de tempo real, por exemplo, possuem requisitos rígidos de latência, pelo fato da necessidade de seus dados terem de ser entregues no menor tempo possível. Aplicações de vídeo geram tráfego com intervalos fixos entre a geração de *frames*, o que exige que a rede encaminhe os pacotes com o mínimo de variação de latência.

5.1.1 Obtenção da latência

Para obter a latência é necessária a leitura do *payload* do pacote, mais especificamente no campo relacionado ao momento da criação do mesmo (*pcktmp* - Seção 4.3.2). Outro parâmetro lido é o momento em que o último flit do pacote chega ao destino (*tplext* - Figura 46). O cálculo da latência (*lat*) para um pacote *i* é realizado segundo a Equação 19:

$$lat_i = tplext_i - pcktmp_i \qquad \text{Equação 19}$$

Por exemplo, o primeiro pacote do arquivo da Figura 46 possui valor

$$lat_0 = tplext_0 - pcktmp_0 = 121 - 0 = 121 \text{ ciclos}$$

5.1.2 Avaliação da latência

A avaliação da latência é realizada neste trabalho através da utilização dois gráficos: *Chaos Normal Form* (CNF) [DUA03] e *distribuição de latências*.

Nos gráficos CNF são impressos no eixo *x* os valores de carga oferecida estabelecidos pelos geradores de tráfego e no eixo *y* os valores de latência média. Através deste tipo de gráfico o projetista consegue verificar o *ponto de saturação* da rede. O ponto de saturação corresponde ao valor de carga oferecida a partir do qual inicia-se um significativo crescimento da latência. A Figura 50 mostra um exemplo onde o ponto de saturação ocorre a partir de uma carga oferecida de 15% da capacidade total da rede.

A construção do tal gráfico baseia-se na leitura de arquivos de tráfego onde são computados o momento de criação, o tamanho (*pcktmp*) e a latência de cada pacote (obtida através da Equação 19). A partir da leitura dos arquivos, preenche-se uma tabela com os campos *carga oferecida*, *somatório de latências* e o *número de pacotes* encontrados.

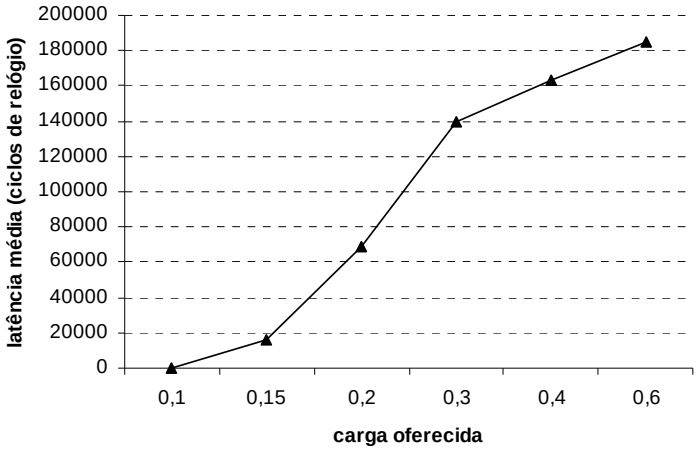


Figura 50 – Gráfico CNF para análise de latência.

Os registros do campo *carga oferecida* deve ser definidos através da verificação de todos os pacotes da rede. Cada novo valor de carga oferecida encontrado é adicionado como um novo registro nesta tabela. O valor de carga oferecida de cada pacote considera seu tamanho, o momento de sua criação e o momento de criação de seu subseqüente. A Equação 20 mostra como se computa a carga oferecida *offeredload* para cada pacote i.

$$offeredload_i = \frac{pcksize_i * ncyclesflit}{tpfext_{i+1} - tpfext_i} \qquad \text{Equação 20}$$

Com a definição de todos os registros de carga oferecida, o próximo passo é calcular a latência média para cada registro. Para cada pacote computa-se a sua latência e acrescenta-se este

valor ao da latência acumulada correspondente à carga oferecida pelo pacote. Deve-se também incrementar o valor do número de pacotes encontrados. A latência média para cada carga oferecida é obtida dividindo-se o valor do acúmulo de latências pelo total de pacotes encontrados. A Tabela 8 mostra os valores para a geração do gráfico ilustrado pela Figura 50. O valor de latência média em itálico indica o ponto de saturação da rede.

Tabela 8 – Exemplo de preenchimento dos valores de latência média.

Carga oferecida	Latência acumulada	Número de pacotes	Latência média
0,1	17.664.000	64.000	276
0,15	*1.041.408.000*	*64.000*	*16.272*
0,2	4.382.336.000	64.000	68.474
0,3	8.962.112.000	64.000	140.033
0,4	10.444.864.000	64.000	163.201
0,6	11.854.784.000	64.000	185.231

O gráfico de *distribuição de latências* objetiva verificar o tempo médio de transmissão dos pacotes de um dado fluxo, ou até mesmo de todos os pacotes da rede, e a variação deste tempo. No eixo *x* são impressas as latências e no eixo *y* são impressos o número de pacote com determinada latência. Um baixo valor da variação de latências leva o projetista a concluir que os pacotes chegam aos seus destinos de maneira uniforme no tempo. Em redes com suporte a QoS, a classe de tráfego *Guaranteed Throughput* deve oferecer aos pacotes pertencentes a esta classe garantias de valores reduzidos de latência média e de variação de latência.

Para impressão dos valores dos intervalos de latência pertencentes ao eixo *x* (vetor `latency`), é necessário o conhecimento das latências mínima (`min_latency`) e máxima (`max_latency`) que ocorreram no tráfego. Com estes dois limites e um passo de incremento (`increment`) são definidos os valores intermediários. A obtenção dos valores do eixo *y* ocorre verificando a latência de cada pacote (`latency-read`), e incrementando o valor do eixo *y* no intervalo de latências ao qual latency read pertence. A Figura 51 mostra um pseudocódigo para geração de um gráfico de distribuição de latências.

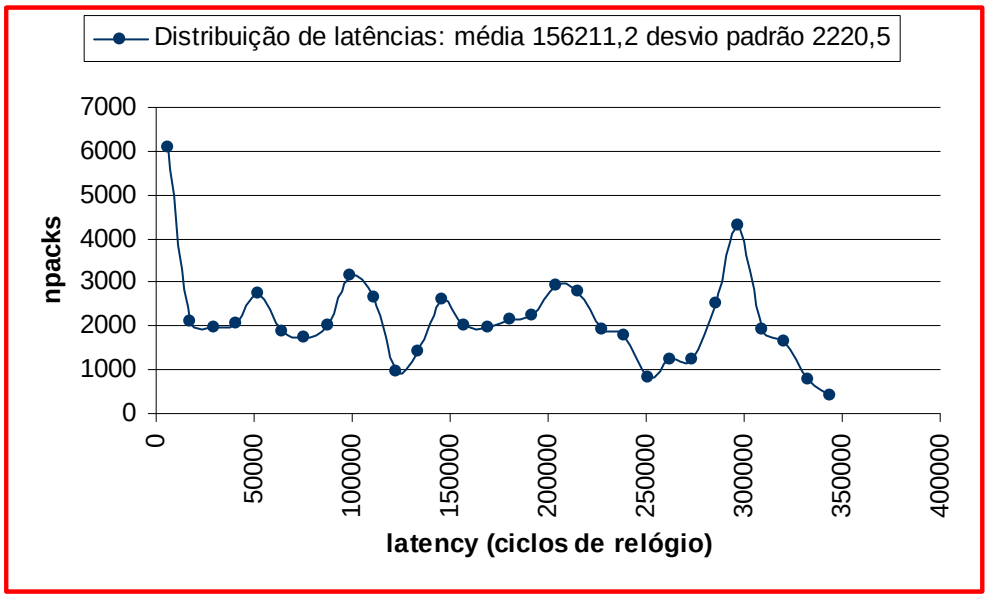
```
        //valores do eixo x
        latency[0]=min_latency;
        latency[npoints-1]=max_latency;
        for(i=1;i<npoints-1;i++)
            latency[i]=latency[i-1]+increment;

        //valores do eixo y
        for(i=0;i<npackets;i++)
        if ((latency_read=>latency[i])&&(latency_read<latency[i+1]))
            npacks[i]++;
```

Figura 51 – pseudocódigo para geração do gráfico de distribuição de latências.

A Figura 52 mostra um exemplo de gráfico de distribuição de latências gerado (em (a)) e a tabela com os valores correspondentes (em (b)). A tabela mostra que o campo *latency* possui valor mínimo de 5894 ciclos e máximo de 343802 ciclos. Observa-se também que a maior parte dos

pacotes (*npacks*=6069) teve latência no intervalo de 5894 a 17545 ciclos.

Observa-se o alto grau de espalhamento dos dados do gráfico (desvio padrão é de 2220,5 ciclos de relógio). O gráfico ilustra resultados de latência obtidos em um experimento onde foram injetados 64.000 pacotes em uma NoC 8x8 na taxa de 30% da capacidade total da rede (acima do ponto de saturação). O tamanho dos pacotes é de 50 flits.



(a)

i	latency	npacks
0	5894	6069
1	17546	2084
2	29198	1985
3	40850	2041
4	52502	2732
5	64154	1877
6	75806	1747
7	87458	1993
8	99110	3144
9	110762	2651
10	122414	983
11	134066	1429
12	145718	2602
13	157370	2033
14	169022	1959
15	180674	2134
16	192326	2242
17	203978	2942
18	215630	2768
19	227282	1899
20	238934	1797
21	250586	821
22	262238	1251
23	273890	1232
24	285542	2519
25	297194	4318
26	308846	1939
27	320498	1644
28	332150	760
29	343802	405

(b)

Figura 52 – (a) Exemplo de gráfico de distribuição de latências; (b) Tabela que gerou o gráfico em (a).

5.2 Tempo médio para transmissão de flits

A segunda métrica para avaliação do tempo de transmissão de dados é o tempo médio (em ciclos de relógio) para transmissão de flits (*avcpf – average cycles per flit* – ciclos por flit médio). O *avcpf* é uma métrica para avaliação interna de desempenho, sendo os dados para análise coletados nos canais de transmissão que interligam os roteadores entre si.

5.2.1 Obtenção do avcpf

A obtenção de *avcpf* consiste em inicialmente no cálculo do *cpf* (ciclos por flit) para cada pacote, que é o tempo gasto para sua transmissão dividido pelo seu tamanho. A Equação 21 mostra

o cálculo de cpf para um pacote i.

$$cpf_i = \frac{tplint_i - tpfint_i}{pcksize_i} \qquad \text{Equação 21}$$

Posteriormente deve-se realizar o somatório de todos os *cpf*s calculados e dividir pelo número de pacotes transmitidos. A Equação 22 mostra o cálculo de *avcpf*.

$$avcpf_{channelXY} = \frac{\sum_{i=1}^{npck} cpf_i}{npck} \qquad \text{Equação 22}$$

5.2.2 Avaliação de avcpf

A métrica *avcpf* oferece uma medida quantitativa do congestionamento dos canais dos roteadores. Na NoC HERMES sem congestionamento, idealmente *avcpf* possui valor 1 (quando a estratégia de controle de fluxo adotada for baseada em créditos) e 2 (quando o controle fluxo for *handshake*). Valores elevados de *avcpf* indicam a alocação de canais sem efetiva transmissão de dados.

Duas maneiras de avaliar *avcpf* (e demais métricas para avaliação interna de desempenho) são utilizadas neste trabalho: (*i*) *gráfico de superfície* e (*ii*) *mapa textual de tráfego*. A visualização de valores para avaliação interna deve ser feita considerando a rede como um todo, devendo ser possível comparar todos os canais/enlaces[1] da rede ao mesmo tempo em diversas regiões.

A Figura 53 mostra um exemplo de gráfico de superfície onde é avaliado o *avcpf* nos enlaces de uma NoC 4x4. Os destaques indicam que os maiores valores de *avcpf* estão em torno de 2,3 ciclos. Estes valores são apresentados nos enlaces formados pelos pares de roteadores (000,001), (002,003), (012, 013) e (014,015).

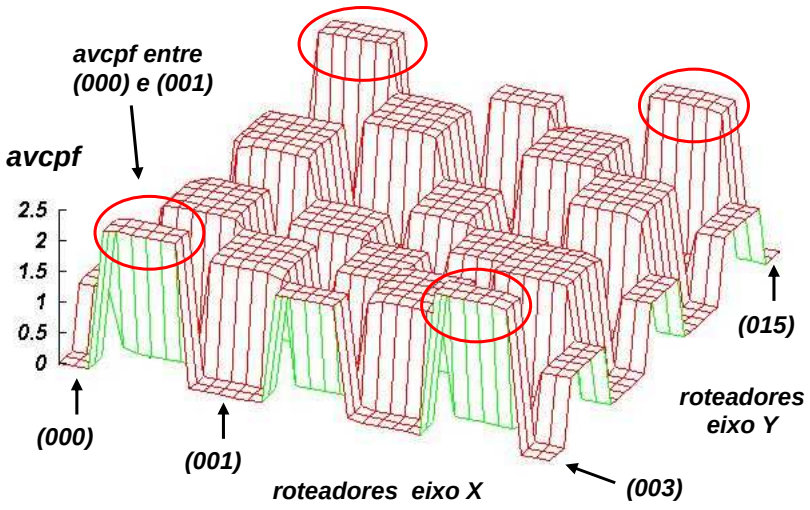


Figura 53 - Exemplo de gráfico de superfície ilustrando o valor de *avcpf* nos enlaces de uma NoC 4x4.

[1] Canais bidirecionais

A Figura 54 mostra a visualização de *avcpf* nos canais de *transmissão* da mesma NoC apresentada na Figura 53. Em cada canal são mostrados os valores máximo (superior), médio e mínimo (inferior) de *cpf*. São destacados os canais que formam os enlaces com maior *avcpf*. No enlace conectando o par de roteadores (000,001) (destaque no canto inferior esquerdo) o valor de *avpcf* é de:

$$avcpf_{(000,001)} = \frac{3{,}6 + 1{,}05}{2} = 2{,}325 \text{ ciclos}$$

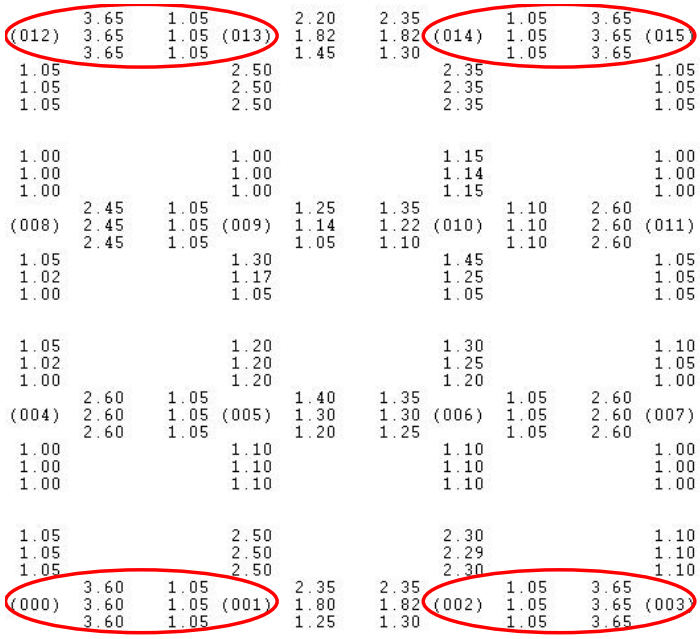


Figura 54 - Mapa textual mostrando o valor do *cpf* nos canais de transmissão de dados para o exemplo ilustrado na Figura 53. Valores nos canais indicam *cpf* máximo (superior), médio e mínimo (inferior).

5.3 Tráfego aceito

O tráfego aceito é uma medida através da qual é verificado se os pacotes movimentam-se na rede mantendo as características estatísticas do modelo utilizado quando de sua geração. Pode-se avaliar o tráfego aceito de maneira tanto externa quanto interna. De maneira externa, observa-se o ponto de saturação da rede (através dos gráficos CNF) e a distribuição do tráfego aceito. Internamente é verificado, em cada canal, a distribuição do tráfego aceito.

5.3.1 Obtenção do tráfego aceito

Para avaliação externa de desempenho, o tráfego aceito (*acceptedtraffic*) é obtido através da leitura de arquivos onde são armazenados os pacotes coletados nos canais que interligam os roteadores com os núcleos, mais especificamente nos campos *tpfext* e *pcksize*. A Figura 46 mostra a estrutura de um arquivo utilizado para avaliação externa. O cálculo de *acceptedtraffic* de um pacote *i* para análise externa é realizado através da Equação 23.

$$acceptedtraffic_i = \frac{pcksize_i}{tpfext_{i+1} - tpfext_i}$$ Equação 23

A obtenção de *acceptedtraffic* para análise interna é similar à obtenção de *acceptedtraffic* para análise externa. Neste caso, captura-se o momento de envio do primeiro flit de cada pacote nos arquivos de dados coletados nos canais de transmissão que interligam os roteadores. O cálculo de *acceptedtraffic* para um pacote *i* para análise interna é realizado através da Equação 25.

$$acceptedtraffic_i = \frac{pcksize_i}{tpfint_{i+1} - tpfint_i}$$ Equação 24

5.3.2 Avaliação do tráfego aceito

A avaliação do tráfego aceito é realizada através de gráficos de *distribuição de tráfego aceito* (para avaliação interna e externa de desempenho) e de gráficos CNF (para avaliação externa de desempenho).

Nos *gráficos CNF* são impressos no eixo *x* todas as cargas oferecidas ao sistema (Equação 20) e no eixo *y* o tráfego aceito para cada carga oferecida. Um exemplo de gráfico de tráfego aceito é mostrado na Figura 55. Da mesma forma que o gráfico CNF para latência média (Seção 5.1.2), o objetivo é encontrar o ponto de saturação da rede, sendo neste caso o ponto a partir do qual o aumento de carga oferecida não produz qualquer aumento no tráfego aceito. A Figura 55 mostra um exemplo onde o ponto de saturação ocorre para uma carga oferecida de 40% da capacidade da rede.

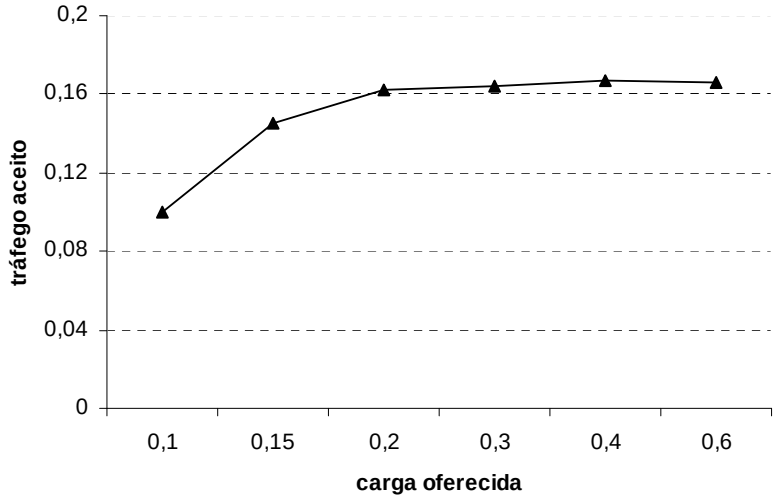


Figura 55 – Exemplo de gráfico CNF para avaliação do tráfego aceito.

É necessário gerar uma tabela similar à que se deve gerar para um gráfico CNF de latência média (Tabela 8). No entanto substitui-se o campo *latência acumulada* por *tráfego aceito acumulado* e *latência média* por *tráfego aceito*. Com a computação de todas as cargas oferecidas que ocorreram no sistema, é momento de calcular o tráfego aceito para cada carga oferecida. Para cada pacote computa-se o seu tráfego aceito e acrescenta-se este valor ao acumulado no campo *tráfego aceito acumulado* no registro relativo à sua carga oferecida. Deve-se também incrementar o

número de pacotes encontrados com aquela carga oferecida. O tráfego aceito para cada carga oferecida é então obtido através da divisão do valor de *tráfego aceito acumulado* pelo total de pacotes encontrados. A Tabela 9 mostra os valores para a geração do gráfico ilustrado pela Figura 55. O valor do tráfego aceito destacado em itálico indica o ponto de saturação. Embora a carga oferecida no último registro seja de 0,45 (45% da capacidade total da rede), a rede não consegue encaminhar dados a mais do que 40%.

Tabela 9 – Exemplo de preenchimento dos valores de tráfego aceito.

Carga oferecida	Tráfego aceito acumulado	Número de pacotes	Tráfego aceito
0,1	6.405,76	64.000	0,10009
0,15	9.287,68	64.000	0,14512
0,2	10.378,24	64.000	0,16216
0,3	10.517,12	64.000	0,16433
0,4	10.659,20	64.000	0,16655
0,6	10.622,08	64.000	0,16597

A *distribuição de tráfego aceito* de um determinado fluxo (par origem-destino específico) permite a verificação ao longo do caminho percorrido pelo fluxo da manutenção ou não das taxas de injeção de dados estabelecidas na geração de tráfego. É indesejável, neste caso, a ocorrência de desvios em relação ao tráfego de entrada. O gráfico de distribuição de tráfego aceito é, desta forma, uma medida de tráfego interno (onde são verificadas as taxas de encaminhamento de pacotes em cada canal percorrido pelo fluxo) e externa (onde é verificado se o núcleo destino recebe os dados nas taxas estabelecidas na origem).

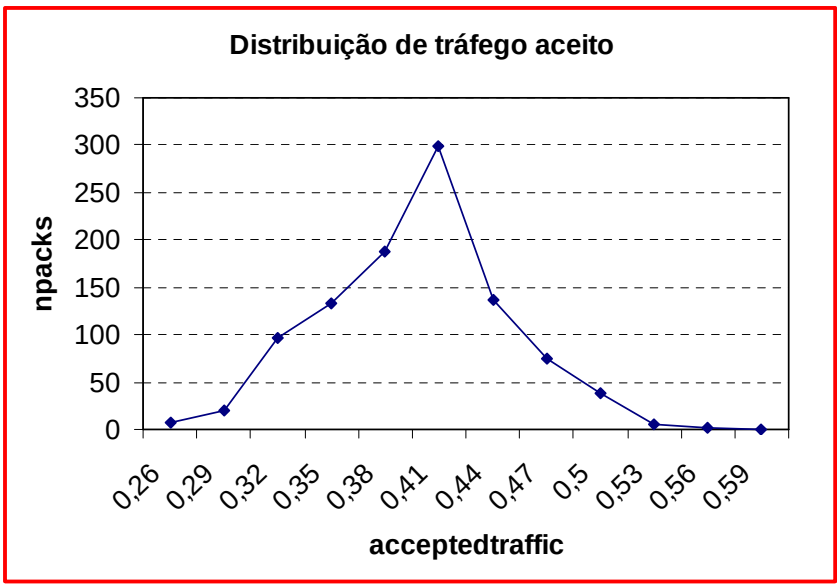
No gráfico de distribuição de tráfego aceito são impressos no eixo *x* os valores de tráfego aceito que ocorreram durante as comunicações entre os núcleos e no eixo *y* o número de pacotes que tiveram determinado tráfego aceito. A construção deste gráfico é similar ao da construção do gráfico de distribuição de latências (Seção 5.1.2). São considerados os parâmetros *tpfext* (nos arquivos utilizados para avaliação externa - Figura 46) e *tpfint* (nos arquivos utilizados para avaliação interna - Figura 48).

Os valores do eixo *x* são obtidos encontrando-se os valores de tráfego aceito mínimo (*min_acceptedtraffic*) e máximo (*max_acceptedtraffic*) e com a utilização de passo de incremento (*increment*) para obtenção dos valores intermediários. Estes valores são indexados em uma tabela similar ao da distribuição de latências, substituindo-se neste caso o campo *latency*, por *acceptedtraffic*. Para cada pacote é calculado então o seu tráfego aceito (*acceptedtraffic_read*) através da Equação 23, para avaliação externa, e da Equação 24, para avaliação interna, e verifica-se a qual intervalo de tráfegos aceitos impressos no eixo *x* o tráfego aceito do pacote em questão pertence. No intervalo encontrado, incrementa-se *npacks* (número de pacotes - eixo *y*) em uma unidade (algoritmo da Figura 56).

```
//valores do eixo x
acceptedtraffic[0]=min_acceptedtraffic;
acceptedtraffic[npoints-1]=max_acceptedtraffic;
for(i=1;i<npoints-1;i++)
   acceptedtraffic[i]=acceptedtraffic[i-1]+increment;

//valores do eixo y
for(i=0;i<npackets;i++)
   if((acceptedtraffic_read=>acceptedtraffic[i])&&(acceptedtraffic_read<acceptedtraffic[i+1]))
      npacks[i]++;
```

Figura 56 – Pseudocódigo para geração do gráfico de distribuição de tráfego aceito.

A Figura 57 mostra um exemplo de gráfico de distribuição de tráfego aceito para um determinado canal da rede. Observa-se que a maior parte dos pacotes possui o tráfego aceito de 41%. O tráfego aceito mínimo é de 26% e o máximo de 56% da capacidade total da rede. Foram observados, nesta medição, 998 pacotes.



(a)

i	acceptedtraffic	npacks
0	0,26	7
1	0,29	20
2	0,32	97
3	0,35	133
4	0,38	187
5	0,41	299
6	0,44	136
7	0,47	74
8	0,5	38
9	0,53	6
10	0,56	1

(b)

Figura 57 – (a) Exemplo de gráfico de distribuição de tráfego aceito; (b) Tabela que gerou o gráfico em (a).

5.4 Utilização média de canais

A utilização média de canais corresponde ao percentual de largura de banda utilizado pelos pacotes para transmissão de dados. A avaliação da utilização média de canais toma como referência duas métricas: a largura de banda ocupada média utilizada e a taxa efetiva de transmissão de dados. Avalia-se esta utilização de maneira interna, ou seja, considerando todos os canais de transmissão da rede. Se a rede está corretamente dimensionada, é esperada uma utilização uniforme de todos os canais. Otimizações na rede, como inserção de canais virtuais e dimensionamento de *buffers*, podem ser necessárias se a utilização da rede não está balanceada.

5.4.1 Obtenção da utilização média dos canais

A primeira métrica considerada neste trabalho para avaliação da utilização média dos canais é o *abw* (*average bandwidth*). O *abw* é a largura de banda média utilizada para transmissão de dados nos canais. Para a obtenção de *abw* primeiramente calcula-se o somatório dos tempos de transmissão de cada pacote. O tempo de transmissão de cada pacote no canal é obtido através do cálculo da diferença de tempo entre a transmissão do primeiro flit (*tpfint*) e a transmissão do último flit do pacote (*tplint*). Divide-se então este somatório pelo tempo total em que o canal transmitiu dados, que é a diferença entre o momento em que foi transmitido o primeiro flit do primeiro pacote e o momento em que foi transmitido o último flit do último pacote. A Equação 25 mostra o cálculo de *abw*. A Figura 49 ilustra a utilização da Equação 25.

$$abw_{channelXY} = \frac{\sum_{i=1}^{npck} (tplint - tpfint)_i}{tslint - tsfint} \qquad \text{Equação 25}$$

A Equação 25 modela a utilização média do canal para redes com mecanismo de chaveamento *store-and-forward* e *virtual-cut-through*, mas pode superestimar a utilização média do canal em redes que utilizam o mecanismo *wormhole*. Ou seja, é possível a obtenção de altos valores de *abw*, no entanto, com uma baixa taxa efetiva de transmissão de dados. Isto pode acontecer por que no mecanismo *wormhole*, uma vez que um dado canal é alocado para transmissão de um pacote, ele permanece indisponível para transmissão de outros pacotes até o final da transmissão do pacote para o qual ele foi alocado.

A taxa efetiva de transmissão de dados (*thr*) é o número efetivo de bits transmitidos em uma dada quantidade de tempo, em número de ciclos de relógio. A Equação 26 computa *thr* para cada canal da NoC, dividindo o número total de bits transmitidos pelo número de ciclos de relógio gastos para transmissão de pacotes.

$$thr_{channelXY} = \frac{\sum_{i=1}^{npck} pcksize_i * flitsize}{nsimc} \qquad \text{Equação 26}$$

Onde:

flitsize: largura do flit, em número de bits;

nsimc: número total de ciclos de relógio gastos para envio de todos os pacotes que passaram pelo canal;

Pode-se ainda extender a Equação 26 para computar o valor de *thr* em bps (bits por segundo). Para isso, no entanto, é necessário o conhecimento *a priori* do período de relógio utilizado. A Equação 27 mostra como calcular *thr* utilizando o período de relógio

$$thrbps_{channelXY} = \frac{\sum_{i=1}^{npck} pcksize_i * flitsize}{nsimc * T} \qquad \text{Equação 27}$$

Onde:

T: período de relógio utilizado;

A Figura 58 ilustra a comparação entre *abw* e *thr*. Considera-se para análise o intervalo de tempo de 16 unidades. Considera-se também que os pacotes transmitidos possuem tamanho de 4 flits. Nas situações em (a) e (b), *abw* é 0,5, ou seja, o canal é ocupado durante 50% do tempo. Entretanto, em (a) são transmitidos 8 flits no intervalo de 16 unidades, enquanto que em (b) são transmitidos 4 flits no mesmo período de tempo. Conclui-se então que, apesar de (a) e (b) apresentarem transmissões de pacotes com mesmo *abw*, (a) transmite efetivamente o dobro de bits por unidade de tempo em relação a (b).

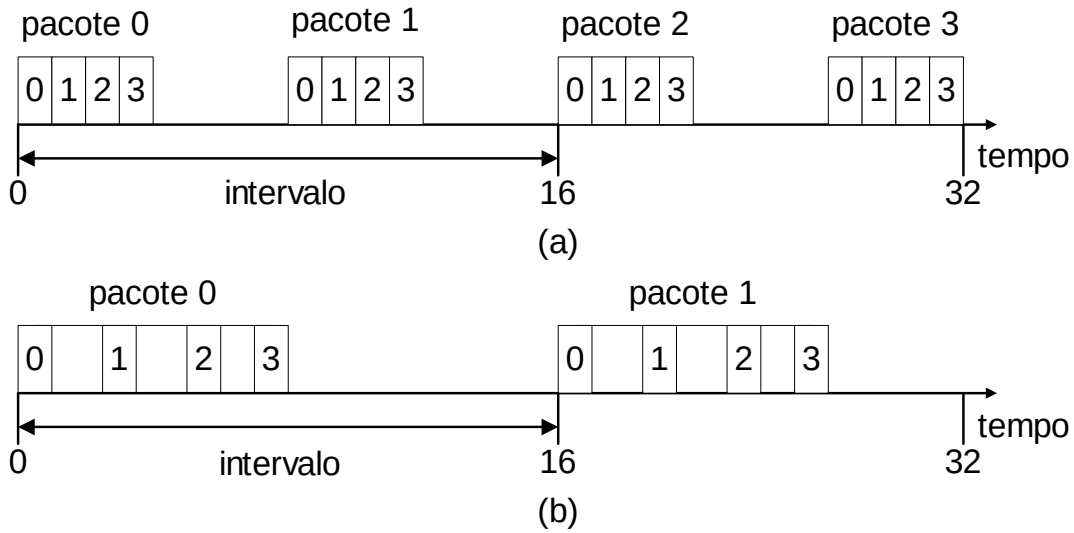
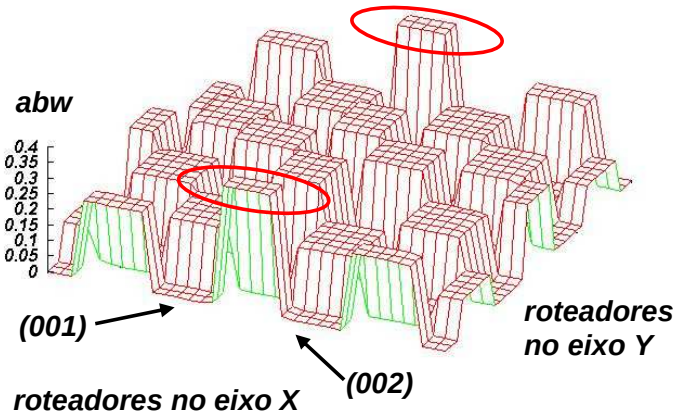


Figura 58 – Exemplo de comparação entre: (a) *abw* e (b) *thr*.

5.4.2 Avaliação da utilização média dos canais

A exemplo da métrica *avcpf*, as métricas *abw* e *thr* também são métricas de avaliação interna da rede. Dessa forma, a visualização dos valores das métricas de desempenho ocorre também por meio de gráficos de superfície e mapas textuais de tráfego. A Figura 59 em (a)/(b) mostra *abw* nos enlaces/canais de uma NoC 4x4. Observa-se que os maiores valores de *abw* (0,38) estão localizados nos enlaces formados pelos pares de roteadores (001,002) e (013,014).

A Figura 60 mostra *thr* para a mesma NoC cujo *abw* foi mostrado na Figura 59. Os enlaces da bisecção XY são os que exibem os maiores valores de *thr* (destaque em (a) e (b)). Através dos valores de *abw* e *thr* mostrados, verifica-se que os pacotes que trafegam nos enlaces formados pelos pares de roteadores (001,002) e (013,014) ocupam uma maior largura de banda para poder transmitir dados na mesma taxa efetiva que os pacotes que trafegam nos demais enlaces da bisecção, devido aos bloqueios que os pacotes são sujeitos e também à utilização do padrão de tráfego complemento.

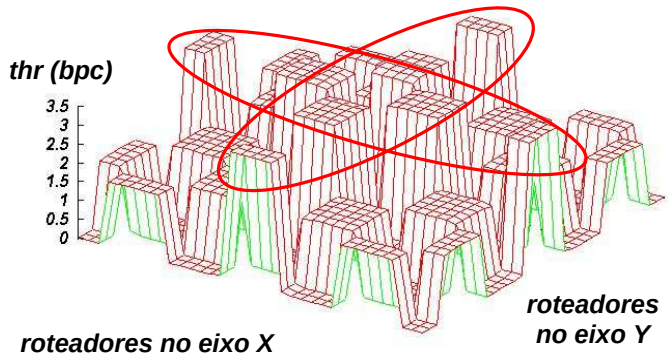


(a)

(012)0.38 0.11	0.11(013)0.38 0.27	0.38(014)0.11 0.25	0.38(015) 0.11
0.10 (008)0.26 0.22	0.10 0.11(009)0.24 0.25	0.12 0.26(010)0.12 0.26	0.10 0.28(011) 0.22
0.22 (004)0.28 0.10	0.25 0.11(005)0.27 0.12	0.26 0.27(006)0.11 0.12	0.22 0.28(007) 0.10
0.11 (000)0.38	0.27 0.11(001)0.38	0.24 0.38(002)0.11	0.12 0.38(003)

(b)

Figura 59 – *abw* para uma NoC 4x4 (a) nos enlaces; (b) nos canais.



(a)

(012)1.70 1.75	1.75(013)3.40 1.73	3.40(014)1.75 1.73	1.70(015) 1.75
1.75 (008)1.73 3.46	1.75 1.75(009)3.45 3.44	1.75 3.44(010)1.75 3.45	1.75 1.72(011) 3.45
3.46 (004)1.72 1.75	3.44 1.75(005)3.44 1.75	3.45 3.44(006)1.75 1.75	3.46 1.72(007) 1.75
1.75 (000)1.71	1.73 1.75(001)3.40	1.73 3.40(002)1.75	1.75 1.70(003)

(b)

Figura 60 - *thr* para uma NoC 4x4 (a) nos enlaces; (b) nos canais.

5.5 Verificação do atendimento de requisitos de carga oferecida e latência ideal

Esta Seção apresenta um método para visualização de medições de parâmetros de avaliação externa de desempenho, de maneira que seja possível compará-los com os parâmetros estabelecidos na geração de tráfego. Trata-se de uma tabela que mostra medições de parâmetros de desempenho para cada fluxo gerado, permitindo avaliar se o tráfego medido obedece aos requisitos especificados na geração. Este tipo de visualização é adotado em [GOO05].

A geração deste tipo de tabela baseia-se na leitura dos arquivos de tráfego que conectam os roteadores com os núcleos (ver exemplo Figura 46 – página 56). Os passos para geração de uma tabela são os seguintes:

- Identificação dos fluxos gerados: são percorridos todos os relatórios, onde são pesquisados todos os pares origem-destino gerados. Para cada novo par encontrado é criado um novo índice na tabela;
- Para cada fluxo é calculada a carga oferecida média e o desvio padrão desta carga;
- Para cada fluxo é calculada a latência média ideal (*ideal_avg_lat* - Equação 28). Dois parâmetros utilizados para o cálculo desta métrica destacam-se: (*i*) *hops*, a quantidade mínima de saltos de roteamento existentes a origem e o destino do fluxo considerado e (*ii*) *arb_time*, o tempo que cada roteador gasta para processar o cabeçalho de cada pacote. Para a NoC HERMES, por exemplo, *arb_time* possui o valor de 7 ciclos de relógio.

$$ideal\_avg\_lat = \frac{\sum_{i=1}^{npacks} pcksize_i + (hops * arb\_time)}{npacks} \qquad \text{Equação 28}$$

Onde:
npacks: quantidade de pacotes do fluxo considerado;
$pcksize_i$: tamanho de um pacote *i* pertencente ao fluxo considerado;
hops: quantidade mínima de saltos de roteamento existentes entre a origem e o destino do fluxo;
arb_time: tempo que cada roteador gasta para processar o cabeçalho de cada pacote.

- Para cada fluxo mede-se o tráfego aceito médio (Equação 24 mostra o tráfego aceito para cada pacote) e o desvio padrão da distribuição de tráfego aceito;
- Para cada fluxo mede-se a latência média e o desvio padrão da latência (Equação 19 calcula a latência de cada pacote).

A Tabela 10 apresenta um trecho de resultados obtidos para uma NoC 8x8, onde nela é gerado um tráfego onde cada núcleo envia 1000 pacotes para destinos estabelecidos de acordo com o padrão complemento. O parâmetro *tolerância* (localizado ao lado direito da latência ideal) especifica a porcentagem permitida para valores excedentes de latência ideal, sendo especificado no momento em que se deseja gerar a tabela.

Observa-se no exemplo que, embora as medições de tráfego aceito estejam de acordo com

o especificado na geração dos fluxos, o mesmo não pode se dizer da latência média obtida, que em nenhum dos fluxos analisados esteve próximo da latência média ideal. Tal comportamento justifica-se pela alta concorrência estabelecida entre os fluxos gerados, visto que foi especificado para cada núcleo a geração de 1000 pacotes utilizando o padrão complemento, o que acarreta congestionamento nas bisecções da rede.

Tabela 10 - Trecho de tabela com valores gerados e medidos para verificação de atendimento a requisitos de QoS. NoC 8x8, roteamento XY, 2 canais virtuais. Núcleos geram pacotes na taxa de 10% utilizando o padrão complemento.

origem	destino	gerado				medido			
		carga oferecida		latência		tráfego aceito		latência	
		media	desvio	ideal	tolerância	media	desvio	média	desvio
0	63	10.0	0.31	155.0	10.0	10.00	0.31	404.00	0.03
1	62	10.0	0.31	141.0	10.0	10.00	0.31	396.00	0.03
2	61	10.0	0.31	127.0	10.0	10.00	0.31	277.00	0.03
3	60	10.0	0.31	113.0	10.0	10.00	0.31	268.00	0.03
4	59	10.0	0.31	113.0	10.0	10.00	0.31	259.00	0.03
5	58	10.0	0.31	127.0	10.0	10.00	0.31	272.00	0.03
6	57	10.0	0.31	141.0	10.0	10.00	0.31	377.00	0.03
7	56	10.0	0.31	155.0	10.0	10.00	0.31	416.00	0.03
8	55	10.0	0.31	141.0	10.0	10.00	0.31	384.00	0.03
9	54	10.0	0.31	127.0	10.0	10.00	0.31	381.00	0.03
10	53	10.0	0.31	113.0	10.0	10.00	0.31	263.00	0.03

5.6 Conclusões

Neste Capítulo foram apresentados os parâmetros que servirão como referência para avaliação de desempenho para a NoC HERMES. Foram mostradas métricas relacionadas às características temporais das aplicações (latência e tempo médio para transmissão de flits) e com a taxa de encaminhamento de dados (tráfego aceito e taxa de utilização de canais). As métricas mostradas puderam ainda ser caracterizadas quanto ao ponto de vista adotado, podendo ser: (*i*) externas, com a verificação do tráfego que ocorre nos canais que interligam roteadores e núcleos; (*ii*) internas, com a verificação do tráfego que ocorre nos canais que interligam os roteadores entre si. Através das métricas apresentadas é possível ao projetista verificar o desempenho das aplicações que executam sobre uma determinada arquitetura de rede sob diferentes cenários de tráfego, observando o comportamento tanto de fluxos específicos (avaliação externa) como de diferentes regiões da rede (avaliação interna).

6 VALIDAÇÃO DOS MÉTODOS PARA GERAÇÃO DE TRÁFEGO E AVALIAÇÃO DE DESEMPENHO

Neste Capítulo são apresentados os experimentos realizados para validar os conceitos de geração de tráfego e avaliação de desempenho abordados nos Capítulos 4 e 5. Para cada estudo de caso são mostradas as especificações de geração de tráfego e os parâmetros de desempenho tomados como referência para análise.

A NoC utilizada nos experimentos é a HERMES. Essa rede possui chaveamento baseado em pacotes, utiliza a topologia malha, e tem como elementos básicos roteadores com lógica de controle centralizada e cinco portas bidirecionais: leste, oeste, norte, sul e local. Cada porta local é conectada a um núcleo e as outras portas são conectadas aos roteadores vizinhos. Cada porta possui *buffers* de entrada para armazenamento temporário de dados. O modo de chaveamento utilizado é o *wormhole*. Para controle de fluxo pode ser utilizada a estratégia *handshake* ou baseada em créditos. Nos experimentos realizados o controle de fluxo adotado é o baseado em créditos. O tamanho do flit é parametrizável, sendo o número máximo de flits em um pacote fixado em $2^{(\text{tamanho do flit, em bits})}$. O primeiro e o segundo flits de cada pacote indicam o cabeçalho (*header*), sendo respectivamente o endereço do núcleo destino e o tamanho do pacote, em número de flits. O restante do pacote é constituído pelos dados úteis (*payload*). O tempo de roteamento/arbitragem em cada roteador da NoC HERMES é de 7 ciclos de relógio.

Neste Capítulo, são discutidos 4 estudos de caso. No primeiro, núcleos enviam dados a taxas fixas utilizando o padrão de tráfego complemento. É realizada avaliação externa e interna de desempenho. Na avaliação externa, o objetivo é verificar o ponto de saturação da rede e o espalhamento dos valores de latência a partir do ponto de saturação. A avaliação interna tem como objetivo verificar a ocupação de enlaces utilizando diferentes algoritmos de roteamento e a contenção de pacotes quando da utilização ou não de canais virtuais.

No segundo estudo de caso, tráfegos gerados de maneira probabilística concorrem com tráfegos gerados com taxas de injeção fixas. Avaliação externa e interna de desempenho são realizadas. A avaliação externa objetiva verificar os valores de latência para diferentes quantidades de saltos de roteamento que formam os fluxos com taxa variável. Também se verifica a distorção de valores de tráfego aceito pela rede em relação ao tráfego gerado pelos diferentes fluxos. A avaliação interna objetiva verificar a fidelidade ou não do tráfego encaminhado nos roteadores intermediários em relação à carga oferecida na origem, para os fluxos com taxa variável.

O terceiro estudo de caso discute a influência do redimensionamento de *buffers* de roteadores no desempenho das comunicações. Avaliação interna e externa de desempenho são realizadas. Na avaliação interna é analisada a contenção nos enlaces para cada política de redimensionamento adotada. Na avaliação externa são observadas as distribuições de latências. Os experimentos realizados mostram a diferença nos resultados obtidos, com o mesmo acréscimo de *slots* de *buffers* em regiões distintas da rede.

Finalmente, no quarto Estudo de Caso, os métodos de geração de tráfego e avaliação de desempenho apresentados são validados em FPGA. A aplicação em questão é um comparador de seqüências de caracteres, sendo muito utilizada, por exemplo, para verificação de similaridade entre segmentos de DNA.

6.1 Estudo de Caso 1

No primeiro Estudo de Caso, núcleos enviam dados a uma taxa constante utilizando padrão de tráfego complemento (Seção 4.2). A topologia utilizada é uma malha 8x8. Os *buffers* de cada roteador possuem profundidade de 8 flits, sendo que cada flit possui largura de 16 bits. Para efeitos de avaliação dois algoritmos de roteamento XY (determinístico) e WF (oeste-primeiro, parcialmente adaptativo) e implementações com e sem canais virtuais são utilizados. Este experimento embasou a escrita do artigo [TED05], publicado no SBCCI'05 (*18th annual Symposium on Integrated Circuits and System Design*).

O objetivo deste Estudo de Caso é verificar: (*i*) o ponto de saturação da rede e o espalhamento dos valores de latência a partir do ponto de saturação – avaliação *externa*; (*ii*) a ocupação de enlaces utilizando diferentes algoritmos de roteamento e a contenção de pacotes quando da utilização ou não de canais virtuais – avaliação *interna*.

6.1.1 Geração de tráfego

Na geração de tráfego espacial, cada núcleo envia 1000 pacotes com 50 flits para destinos determinados segundo o padrão de tráfego complemento, o que resulta em um total de 64.000 pacotes transmitidos em cada simulação. Nesse padrão de tráfego, o nodo que possui coordenadas no formato binário a_{n-1}, a_{n-2}, ..., a_1, a_0 comunica-se com o nodo $\overline{a_{n-1}}$, $\overline{a_{n-2}}$,..., $\overline{a_1}$, $\overline{a_0}$ (inversão de todos os bits). Foram realizadas seis simulações, onde em cada uma as taxas de injeção de pacotes para cada núcleo tiveram valores de 10, 15, 20, 30, 40 e 60% da carga máxima da rede. Considerando que a largura do canal de transmissão de dados é de 16 bits e que a freqüência da rede é de 50MHz (período de ciclo de relógio de *20ns*), tem-se para *chr*:

$$chr = \frac{16\,\text{bits}}{\text{ciclo}} = \frac{16\,\text{bits}}{20\,\text{ns}} = 800\,\text{Mbps}$$

Conseqüentemente, as taxas de injeção de pacotes gerados em cada núcleo (*ipr*) foram de 80, 120, 160, 240, 320, 400, e 480 Mbps. A geração de tráfego com taxas de injeção adotou a alternativa (a) da Figura 33, onde é mantido fixo o tamanho do pacote (neste caso, 50 flits) e varia-se o período ocioso (*idle*) para cada taxa. Apresenta-se abaixo o cálculo do parâmetro *idle* (Equação 6) em ciclos de relógio para a taxa de 80Mbps.

$$idle = pcksize * ncyclesflit * \left(\frac{chr}{ipr} - 1 \right) = 50 * 1 * \left(\frac{800}{80} - 1 \right) = 450$$

Para as demais taxas os resultados obtidos foram de 283 (para 120 Mbps), 200 (para 160 Mbps), 117 (para 240 Mbps), 75 (para 320 Mbps), 50 (para 400 Mbps) e de 33 ciclos (para 480

Mbps). Neste primeiro estudo de caso, foram adotadas taxas de injeção fixas com o objetivo de verificar o comportamento da rede frente a tráfegos de dados com características similares a aplicações que solicitam serviços à rede de acordo com a categoria CBR.

6.1.2 Avaliação de desempenho

Inicialmente a avaliação de desempenho da rede para o cenário de tráfego estabelecido é realizada de maneira *externa*. São considerados dois tipos de gráfico: (*i*) *CNF*, para obtenção do ponto de saturação da rede; (*ii*) *distribuição de latências*, para verificação do grau de variação das latências dos pacotes.

Os *gráficos CNF* apresentam os valores médios de latência e tráfego aceito obtidos para cada carga oferecida à rede. Desta forma, é possível verificar os limites da rede em termos destes parâmetros, de acordo com os padrões especificados na geração de tráfego. A Tabela 11 mostra os valores de latência média e tráfego aceito obtidos nos experimentos.

Tabela 11 – Valores de latência e tráfego aceito obtidos no Estudo de Caso 1.

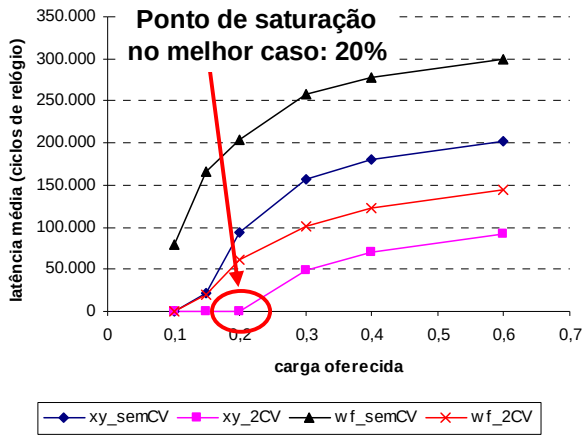
Carga oferecida	Latência média (ciclos)				Tráfego aceito (normalizado)			
	xy_semCV	xy_2CV	wf_semCV	wf_2CV	xy_semCV	xy_2CV	wf_semCV	wf_2CV
0.1	293	261	79.266	320	0,10	0,10	0,09	0,10
0.15	20.854	255	165.954	19.126	0,14	0,15	0,11	0,14
0.2	93.918	875	203.125	62.104	0,15	0,20	0,13	0,17
0.3	157.200	48.977	257.726	101.372	0,16	0,21	0,13	0,18
0.4	180.508	70.856	278.372	122.544	0,16	0,21	0,13	0,19
0.6	201.774	91.956	300.198	143.802	0,16	0,21	0,13	0,19

A Figura 61(a) mostra o quanto as latências médias são maiores para o caso em que é utilizado o algoritmo de roteamento WF sem canais virtuais (*wf_semCV*). Observa-se que para o caso onde é utilizado o algoritmo XY com 2 canais virtuais (*xy_2VC*) a latência é praticamente constante até o ponto de saturação, que ocorre a partir da carga oferecida de 20%. Após este valor a latência apresenta significativo crescimento. Tal fato ocorre devido à dificuldade dos pacotes serem injetados na rede, à medida que vai aumentando a carga oferecida por todos os nodos de origem, ou seja, os pacotes vão sendo injetados na rede em um momento cada vez mais atrasado em relação ao de criação.

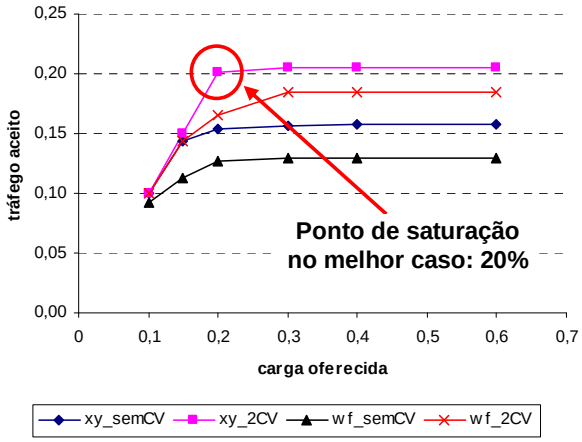
Na Figura 61(b) observa-se que a utilização de canais virtuais também acarreta aumento no tráfego aceito máximo. Também é ilustrando mais uma vez a superioridade do algoritmo de roteamento XY sobre o WF. O maior ponto de saturação da rede (20%) ocorre onde se utiliza o algoritmo de roteamento XY e canais virtuais (*xy_2VC*).

Os gráficos CNF mostraram em quanto a utilização do algoritmo de roteamento XY foi superior ao WF. Os gráficos de *distribuição de latências* apresentados abaixo mostram a superioridade em termos de desempenho alcançado com a utilização de canais virtuais, para o algoritmo de roteamento XY. São mostrados nos gráficos o número de pacotes (*npacks*) com um dado valor de latência (*latency*). O objetivo é avaliar o grau de espalhamento das curvas obtidas,

que é um indicativo do cumprimento ou não dos requisitos temporais do tráfego gerado.



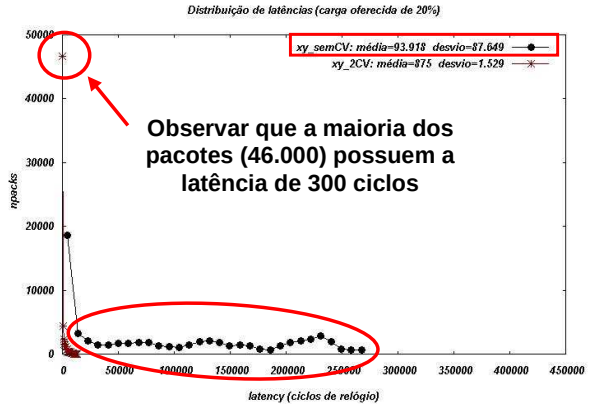
(a) – tráfego aceito



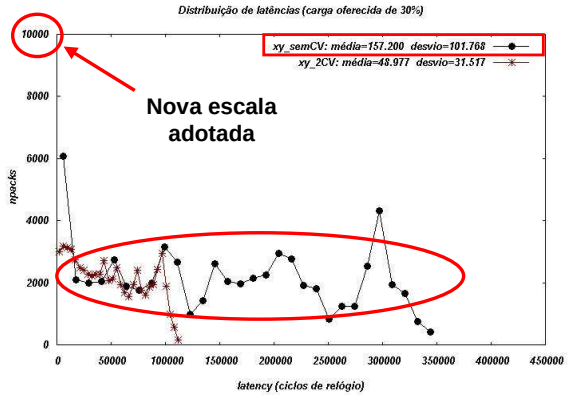
(b) – latência média

Figura 61 – Gráficos CNF.

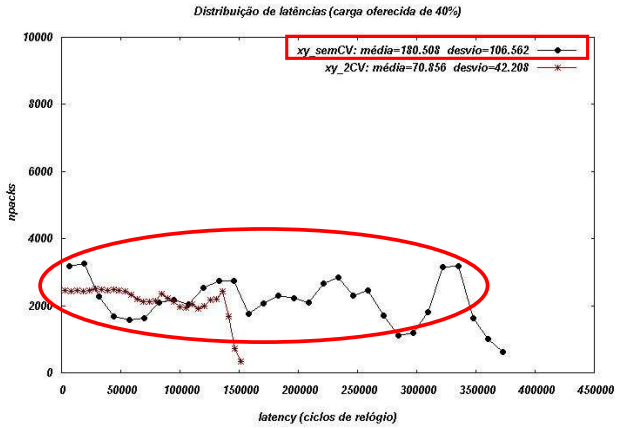
As curvas ilustradas na Figura 62 mostram a distribuição de latências a partir do ponto de saturação quando se usa o algoritmo de roteamento XY, com e sem a utilização de canais virtuais. A plotagem de gráficos a partir do ponto de saturação se justifica pelo fato de tais gráficos apresentarem a maior diferença entre implementações com e sem canais virtuais. Os gráficos de distribuição de latências com cargas oferecidas de 10 e 15% apresentam praticamente os mesmos valores com e sem a utilização de canais virtuais.



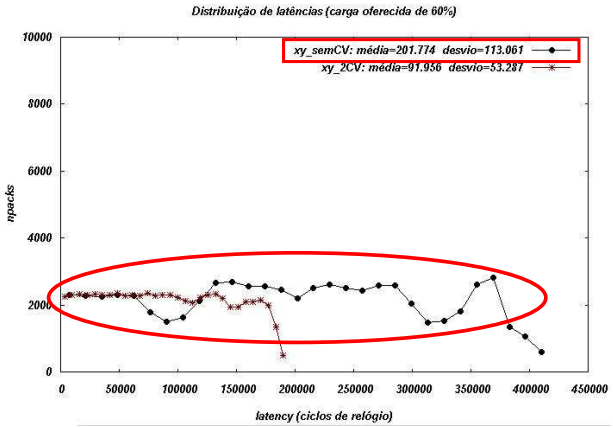
(a) – Carga oferecida: 20%



(b) – Carga oferecida: 30%



(c) – Carga oferecida: 40%



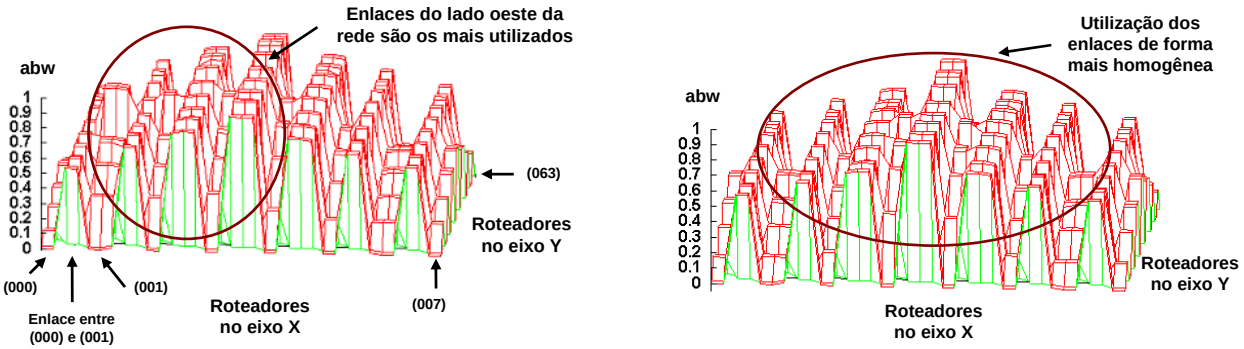
(d) – Carga oferecida: 60%

Figura 62 – Distribuição de latências para o Estudo de Caso 1.

Observa-se nas quatro curvas que a implementação da rede sem canais virtuais apresenta o maior espalhamento em relação à distribuição de latências referente à utilização de canais virtuais (destaques em (a), (b), (c) e (d)). Outro detalhe a observar é o aumento deste espalhamento com o aumento da carga oferecida, em ambas implementações. O menor espalhamento observado na implementação com canais virtuais permite estimar o tempo para transmitir pacotes de maneira mais precisa, um fator fundamental para implementação de QoS em NoCs.

A avaliação interna da rede considera neste experimento as métricas *abw* (largura de banda média utilizada) em cada enlace da NoC e *avcpf* (número médio de ciclos gastos para encaminhar cada flit). Valores elevados de *abw* indicam altas taxas de encaminhamento de pacotes, enquanto que valores elevados de *avcpf* indicam alta contenção.

A Figura 63 mostra *abw* quando são utilizados os algoritmos de roteamento WF (em (a)) e XY (em (b)), sem a utilização de canais virtuais. A carga oferecida pelos núcleos é de 20%, o que provoca a saturação da rede, em ambas as configurações. A Figura 63(a) mostra a maior utilização dos enlaces no lado oeste da rede quando comparada à Figura 63(b). Tal fato é explicado devido à natureza do algoritmo WF, visto que este tenta rotear pacotes primeiramente na direção oeste para posteriormente tentar rotear nas outras direções. Na Figura 63(b) é observada uma distribuição mais homogênea da largura de banda utilizada.

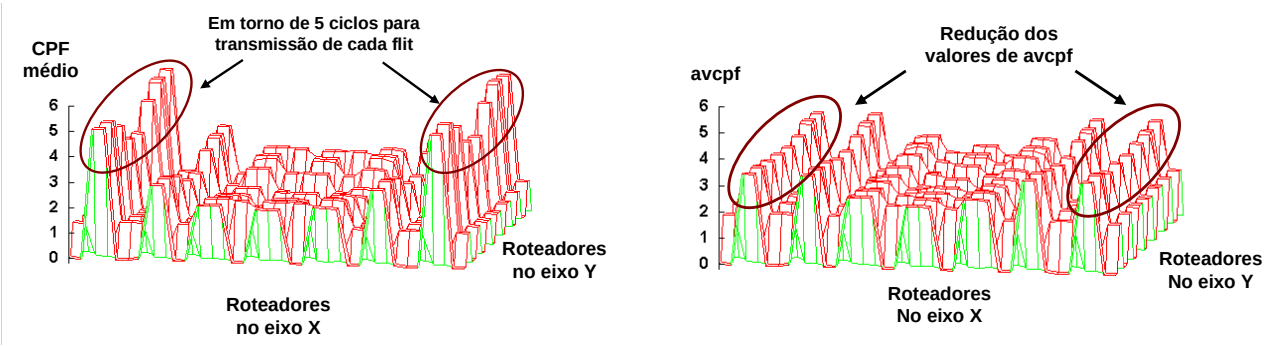


(a) – Roteamento WF **(b) – Roteamento XY**

Figura 63 - Largura de banda ocupada por enlace para os dois algoritmos de roteamento (carga de 20%), sem utilização de canais virtuais.

Outra análise interna realizada leva em consideração o *avcpf* nos enlaces da NoC, onde é possível observar pontos de contenção da rede. Neste caso é verificada a contenção imposta aos flits em decorrência da não utilização de canais virtuais e a diminuição desta contenção quando da adoção de canais virtuais. Valores elevados de *avcpf* indicam em que pontos há maior dificuldade dos pacotes serem encaminhados. A Figura 63(b) mostra que o centro da rede apresenta maior densidade de tráfego (maior utilização da largura de banda - *abw*). Espera-se neste caso que haja maior contenção de pacotes nos roteadores localizados na periferia da rede. A Figura 64(a) mostra que o CPF médio em tais roteadores está em torno de 5 quando não são utilizados canais virtuais, o que significa que o tempo médio de permanência em buffer para cada flit é de 5 ciclos. Com a utilização de canais virtuais (onde o canal físico é multiplexado) ocorre redução da contenção dos flits nos roteadores da periferia da rede (Figura 64(b)) para valores em torno de 3 ciclos. Tal fato

demonstra um benefício direto da utilização de canais virtuais: diminuição da contenção em redes com modo de chaveamento *wormhole*.



(a) – Sem utilização de canais virtuais, roteamento XY.

(b) – Com a utilização de 2 canais virtuais, roteamento XY.

Figura 64 - Tempo médio para transmissão de flits (carga de 20%).

6.2 Estudo de caso 2

O segundo Estudo de Caso envolve a geração de tráfego com taxas constantes concorrendo com tráfegos que possuem taxas de injeção variando segundo a distribuição de probabilidade normal. O objetivo do experimento é verificar o efeito causado por fluxos com taxas de injeção constantes em fluxos gerados com taxas variáveis. Através da avaliação da *distribuição de tráfego aceito* será possível observar o fenômeno *flutuação de carga*, que são desvios que ocorrem nas taxas geradas pelos núcleos de origem, devido ao congestionamento na rede. A distribuição de latência permite analisar a *média* e o *espalhamento dos valores de latência* encontrados de acordo com diferentes distâncias entre pares de núcleos origem-destino, a exemplo do que foi visto no Estudo de Caso 1. A topologia escolhida foi uma malha com dimensão 8x8, sendo associado a cada roteador, 2 canais virtuais.

6.2.1 Geração de tráfego

O padrão de tráfego espacial escolhido foi o complemento. A geração de tráfego com taxas de injeção utilizou as distribuições constante e normal. Em cada núcleo foram gerados 1000 pacotes de 50 flits cada. A variação das taxas de injeção é definida através do período de ociosidade entre pacotes (Figura 33(a)). As especificações das taxas de injeção praticadas são mostradas na Tabela 12. A Tabela 13 mostra os valores das taxas a serem geradas e o número de gerações para cada taxa, para o tráfego modelado pela distribuição normal.

Dois cenários de tráfego foram avaliados. No *Cenário 1* é estabelecida uma concorrência de recursos entre os fluxos gerados com taxas variando segundo a distribuição normal (Figura 65(a) e (b)) e os fluxos gerados com taxas de injeção constante a 40Mbps. No *Cenário 2*, fluxos gerados a 80Mbps concorrem com os fluxos com taxas especificadas pela mesma distribuição normal do *Cenário 1*. O objetivo é verificar a influência do grau de concorrência estabelecido pelas duas taxas constantes no fluxo modelado segundo a distribuição normal.

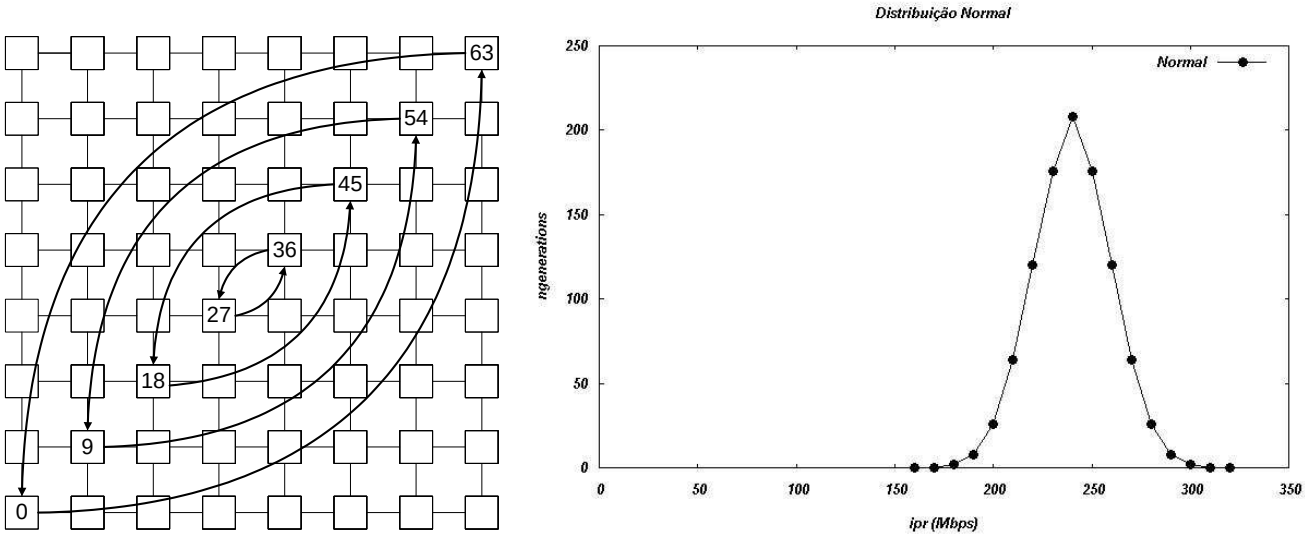
Tabela 12 – Especificações de taxas de injeção.

Tráfego normal
- Taxa mínima: 160 Mbps (20%) - Taxa média: 240 Mbps (30%) - Taxa máxima: 320 Mbps (40%) - Desvio padrão: 20 Mbps (2,5%) - Pares (Figura 65(a)): (0,63), (9,54), (18,45), (27,36).

Tráfego constante
- Taxa *Cenário 1*: 40 Mbps (5%) - Taxa *Cenário 2*: 80 Mbps (10%) - Pares (padrão complemento): (1,62), (2,61), (3,60), (4,59), (5,58), (6,57), (7,56), (8,55), (10,53), (11,52), (12,51), (13,50), (14,49), (15,48), (16,47), (17,46), (19,44), (20,43), (21,42), (22,41), (23,40), (24,39), (25,38), (26,37), (28,35), (29,34), (30,33), (31,32)

Tabela 13 – Valores da curva normal gerada.

index	***ipr (Mbps)***	***ngenerations***
0	160	0
1	170	0
2	180	2
3	190	8
4	200	26
5	210	64
6	220	120
7	230	176
8	240	208
9	250	176
10	260	120
11	270	64
12	280	26
13	290	8
14	300	2
15	310	0
16	320	0



(a) (b)

Figura 65 – (a) Núcleos que geram dados segundo uma distribuição normal e seus destinos; (b) distribuição de probabilidade que descreve as taxas de injeção dos núcleos em (a).

6.2.2 Avaliação de desempenho

A avaliação de desempenho considerou as comunicações envolvendo tráfego probabilístico, verificando-se nas interfaces externas da rede as distribuições de vazão e latência correspondentes aos fluxos gerados. A Tabela 14 relaciona os fluxos medidos com a latência média e o desvio padrão obtidos nos dois cenários experimentados. A coluna *Sem concorrência* mostra as medições de latência quando apenas o tráfego probabilístico movimenta-se na NoC, servindo como referência para avaliação de *Cenário 1* e *Cenário 2*.

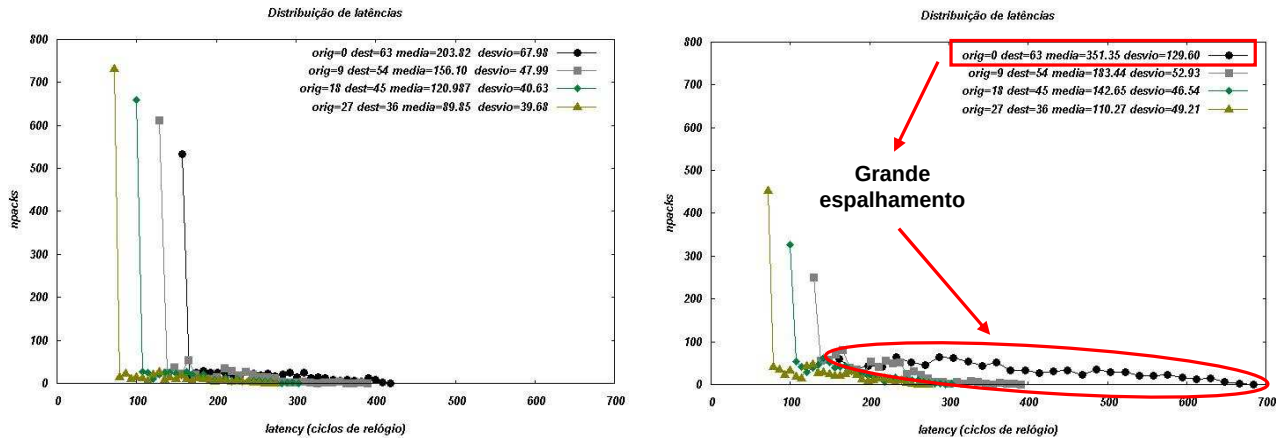
Tabela 14 – Valores de latência para os fluxos com taxas de injeção variável.

origem	destino	número de *hops*	*Sem concorrência*		*Cenário 1*		*Cenário 2*	
			latência média (ciclos)	desvio padrão (ciclos)	latência média (ciclos)	desvio padrão (ciclos)	latência média (ciclos)	desvio padrão (ciclos)
0	63	14	154,19	0,89	203,82	67,98	351,35	129,60
9	54	8	126,19	0,83	156,10	47,99	183,44	52,93
18	45	4	98,17	0,77	120,99	40,63	142,65	46,54
27	36	2	70,15	0,72	89,85	39,68	110,27	49,21
36	27	2	70,18	0,76	91,52	37,93	111,97	44,59
45	18	4	98,2	0,87	120,36	40,91	141,71	47,69
54	9	8	126,12	0,66	158,71	51,64	186,24	54,17
63	0	14	154,12	0,70	199,92	64,89	359,63	97,26

As menores médias e desvios de latências foram apresentados pelos fluxos gerados pelos núcleos 27 e 36. Tal fato era esperado pela pouca distância (em *hops* - número de saltos de roteamento) entre os roteadores mencionados (*hops*=2). Observa-se também que as maiores médias e desvios de latência foram apresentados pelos fluxos gerados pelos roteadores 0 e 63 (*hops*=14). A Figura 66 ilustra a distribuição de latências para os fluxos gerados nos roteadores 0, 9, 18 e 27 quando chegam ao destino. Observa-se a influência que o número de *hops* entre pares origem e destino nos valores de latência, bem como no espalhamento destes valores. A latência esperada entre dois nodos (*ideal_latency*) é proporcional ao tamanho do pacote (50 flits) mais o número de hops multiplicado pelo tempo de roteamento/arbitragem (*arb_time*) em cada roteador (7 ciclos). No par 0-63 temos 14 *hops*, logo, a latência ideal seria:

$$ideal\_latency = pcksize + (hops * arb\_time) = 50 + (14 * 7) = 148 \text{ ciclos}$$

Ao observarmos a Figura 66(a) temos aproximadamente 500 pacotes com esta latência, e os demais com uma latência superior, devido ao congestionamento na rede. À medida que o congestionamento aumenta (*Cenário 2*) observa-se um sensível aumento no espalhamento das latências em função do número de *hops* (destaque na Figura 66(b)).



(a) (b)

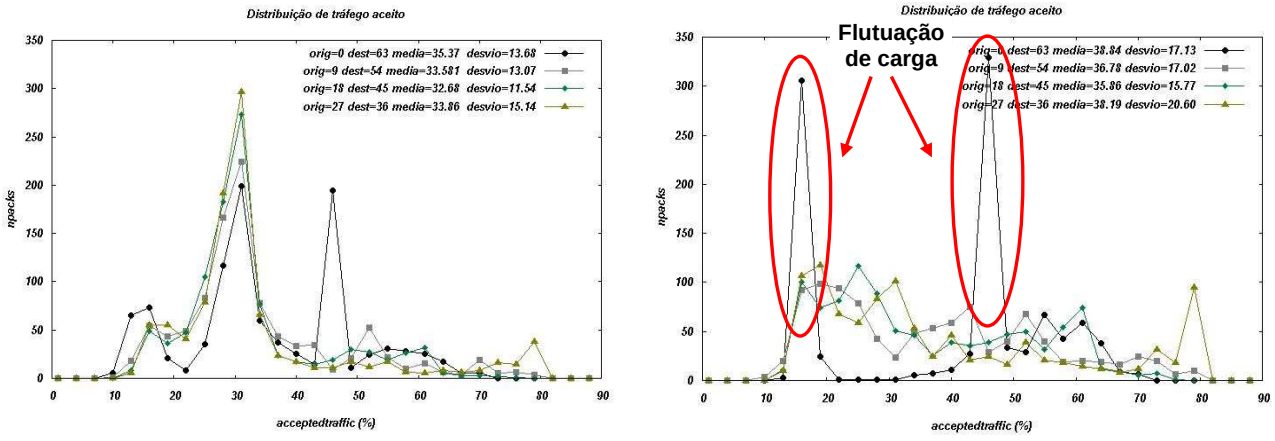
Figura 66 – Distribuições de *latência* de fluxos modelados segundo a distribuição normal da Figura 65(b) concorrendo com tráfego constante com taxas de (a) 5% - *Cenário 1* e (b) 10% - *Cenário 2*.

A Tabela 15 mostra os valores de tráfego aceito para os fluxos com taxa variável. Observa-se na coluna *Sem concorrência* que os valores de tráfego aceito e o desvio padrão para cada fluxo são os mesmos em relação à sua geração.

Tabela 15 - Valores de tráfego aceito para os fluxos com taxas de injeção variável.

origem	destino	número de *hops*	*Sem concorrência*		*Cenário 1*		*Cenário 2*	
			tráfego aceito (%)	desvio padrão (%)	tráfego aceito (%)	desvio padrão (%)	tráfego aceito (%)	desvio padrão (%)
0	63	14	29,6	2,45	35,37	13,68	38,84	17,13
9	54	8	29,6	2,46	33,58	13,07	36,78	17,02
18	45	4	29,6	2,45	32,68	11,54	35,86	15,77
27	36	2	29,6	2,49	33,86	15,14	38,19	20,60
36	27	2	29,6	2,47	32,52	11,57	35,49	15,68
45	18	4	29,6	2,46	33,26	13,14	36,77	17,44
54	9	8	29,6	2,47	33,93	13,45	37,11	17,23
63	0	14	29,6	2,46	35,18	13,79	39,39	17,78

Plota-se, na Figura 67, o tráfego aceito em função do número de pacotes, para os fluxos gerados pelo núcleos 0, 9, 18 e 27. Comparar estas curvas com a Figura 65(b). O objetivo é verificar a fidelidade entre as taxas especificadas e as taxas que efetivamente chegam nos destinos. Quando o tráfego constante concorrente possui taxa relativa de 5% observa-se que o tráfego na chegada não é tão afetado, apesar de já apresentar um acréscimo no desvio padrão (Figura 67(a)). Observa-se para o caso em que o número de *hops* é elevado (par 0-63) dois picos, um à esquerda e um à direita da média, distorcendo a curva. Quando a concorrência é com um tráfego constante a 10%, há uma considerável distorção das taxas de saída em relação à entrada, para todos os fluxos (Figura 67(b)).

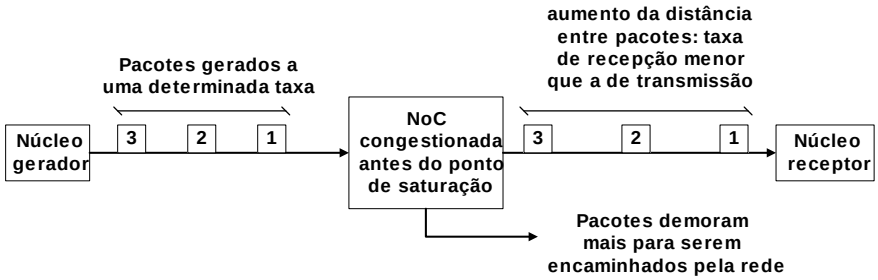


(a) (b)

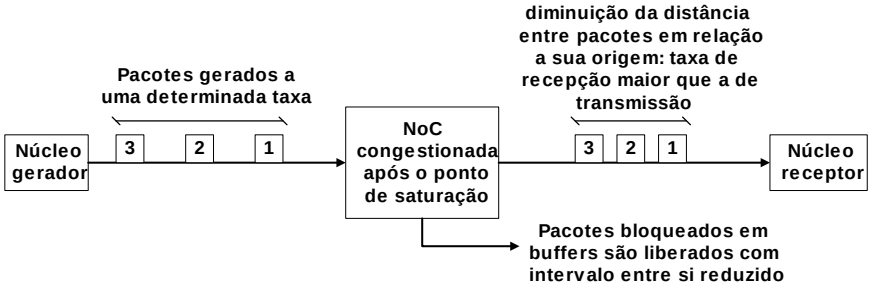
Figura 67 - Distribuições de *tráfego aceito* de fluxos modelados segundo a distribuição normal da Figura 65(b) concorrendo com tráfego constante com taxas de (a) 5% e (b) 10%.

A distorção acima mostrada deve-se ao fenômeno denominado *flutuação de carga* [ZHA91b]. Tal fenômeno é relacionado à variação da taxa de encaminhamento de pacotes decorrente da utilização dos buffers e congestionamento na rede. O destaque à esquerda da Figura 67(b) corresponde a elevados intervalos de entrega entre pacotes consecutivos, diminuindo assim a

vazão. Isto ocorre devido a congestionamentos na rede que ocorrem antes de sua saturação. Da mesma forma, após um pacote ser liberado, outro que estava bloqueado nos buffers pode ser enviado logo em seguida, com um intervalo entre pacotes muito pequeno, ocasionando uma elevada taxa instantânea (ver destaque da direita na Figura 67(b)). Esta elevação na taxa instantânea ocorre após o ponto de saturação. A Figura 68 ilustra o fenômeno da *flutuação de carga* antes do ponto de saturação (em (a)) e após o ponto de saturação (em (b)).



(a)



(b)

Figura 68 – Flutuação de carga: (a) antes da saturação da rede (b) após a saturação da rede.

Na avaliação interna de desempenho de um tráfego gerado de forma probabilística verifica-se a fidelidade com que os dados estão sendo encaminhados nos enlaces que interligam os roteadores. Para isso é necessário saber, para um dado fluxo, se a taxa média de encaminhamento dos pacotes e o desvio padrão se mantêm ao longo do caminho de acordo com as taxas especificadas na geração de tráfego.

Foi considerado para análise o tráfego gerado no nodo 18 com destino ao nodo 45. Cada seqüência de gráficos ilustra as estatísticas obtidas nas portas (Figura 69):

- leste dos roteadores 18, 19 e 20;
- norte dos roteadores 21, 29 e 37.

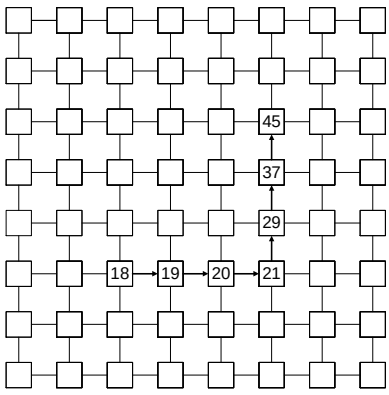
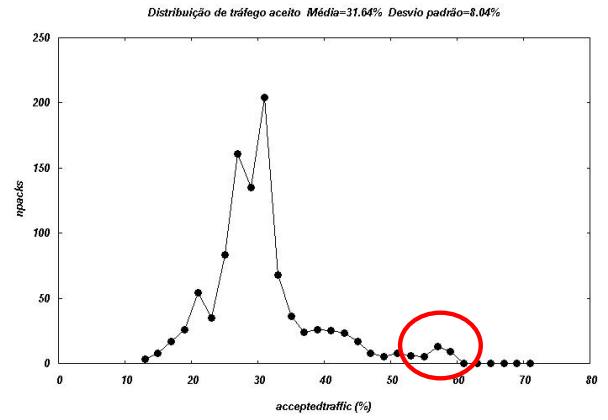
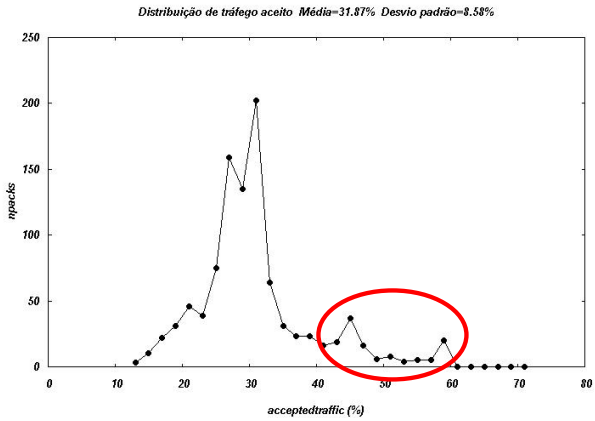


Figura 69 – Caminho percorrido pelo fluxo com origem no roteador 18 e destino no roteador 45.

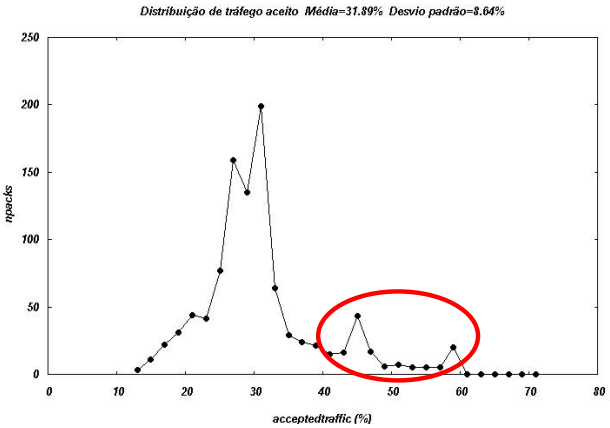
Observa-se, na Figura 70, que a taxa de encaminhamento dos dados ao longo do caminho é semelhante com o especificado na entrada (em torno de 30%). Os destaques na Figura 70 mostram que há alguns pacotes com elevada taxa de transmissão, o que caracteriza uma pequena flutuação de carga.



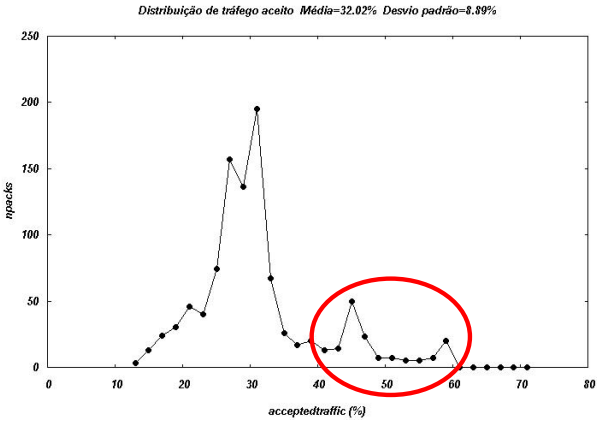
(a) roteador 18 porta leste



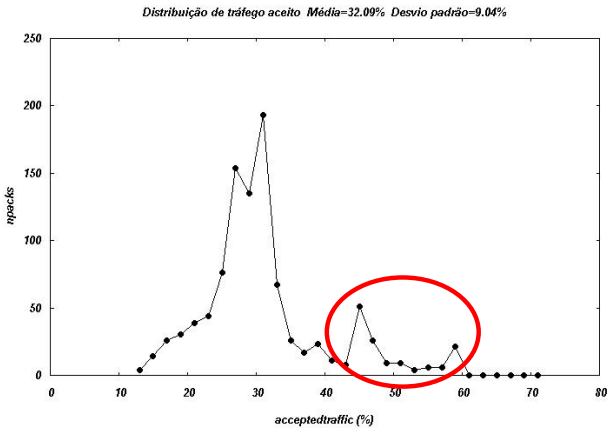
(b) roteador 19 porta leste



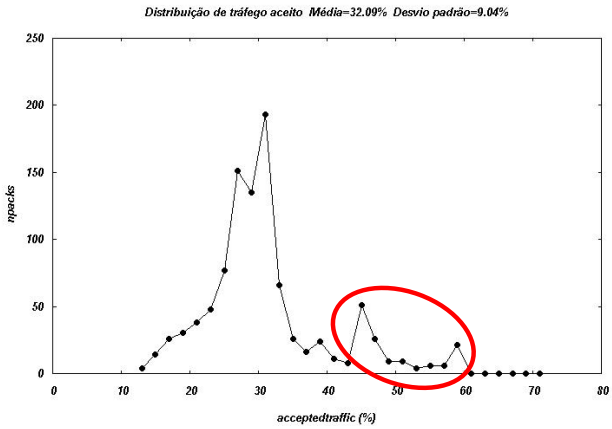
(c) roteador 20 porta leste



(d) roteador 21 porta norte



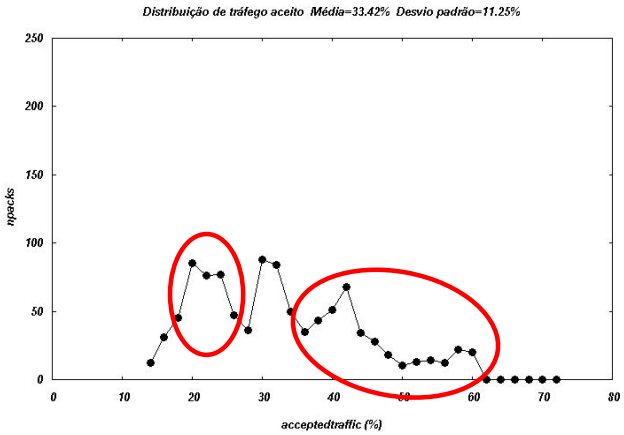
(e) roteador 29 porta norte



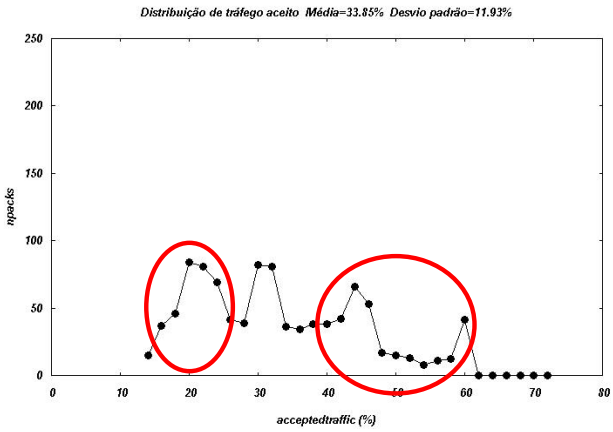
(f) roteador 37 porta norte

Figura 70 - Distribuição de taxas de encaminhamento de pacotes originados no nodo 18 tendo como destino o nodo 60. Núcleos que geram tráfego constante injetam pacotes na taxa relativa de 5%.

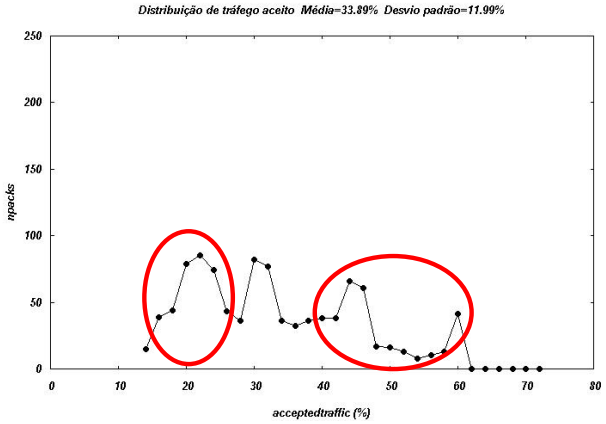
A Figura 71 mostra a disparidade da distribuição de tráfego aceito em relação à carga oferecida, comprovando a influência que o tráfego de 10% exerce sobre o tráfego probabilístico. Mais uma vez verifica-se o fenômeno da flutuação de carga provocada pelo congestionamento que se estabeleceu na rede (destaques na Figura 71).



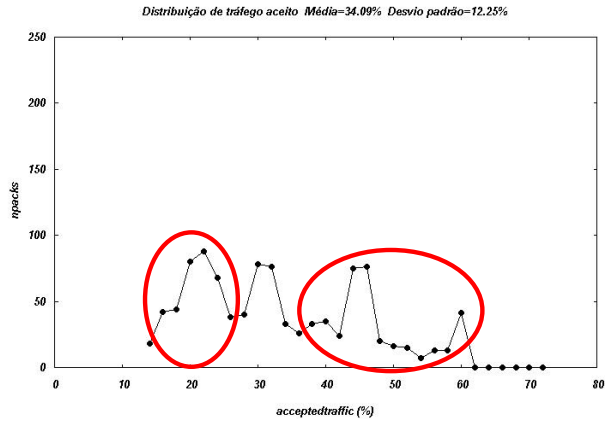
(a) roteador 18 porta leste



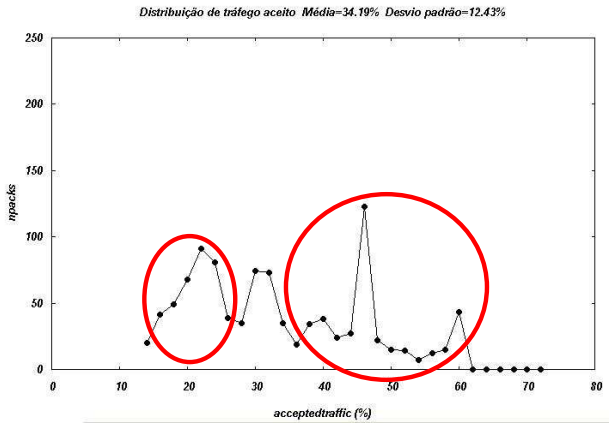
(b) roteador 19 porta leste



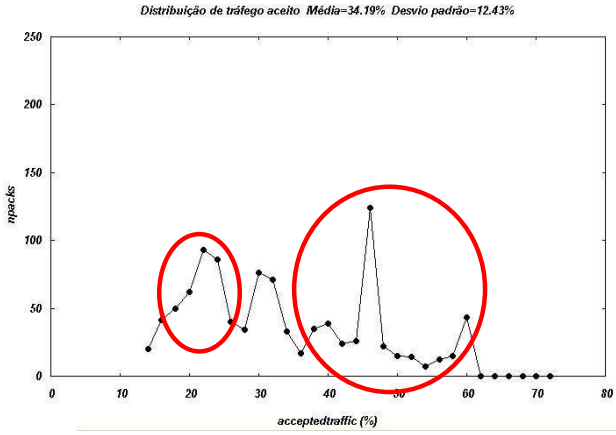
(c) roteador 20 porta leste



(d) roteador 21 porta norte



(e) roteador 29 porta norte



(f) roteador 37 porta norte

Figura 71 - Distribuição de taxas de encaminhamento de pacotes originados no nodo 18 tendo como destino o nodo 60. Núcleos que geram tráfego constante injetam pacotes na taxa relativa de 10%.

Os resultados de avaliação de desempenho apresentados neste Estudo de Caso permitiram a visualização de um fenômeno que pode ocorrer devido a condições de congestionamento na rede. Na flutuação de carga ocorrem desvios nas taxas de encaminhamento de pacotes em relação ao tráfego especificado na entrada. Pacotes gerados com uma determinada taxa podem ser atrasados, aumentando o intervalo entre si, causando diminuição na taxa de encaminhamento. Tais pacotes podem também ser liberados consecutivamente após estarem bloqueados em *buffers*, diminuindo o intervalo entre si, causando aumento em sua taxa de encaminhamento.

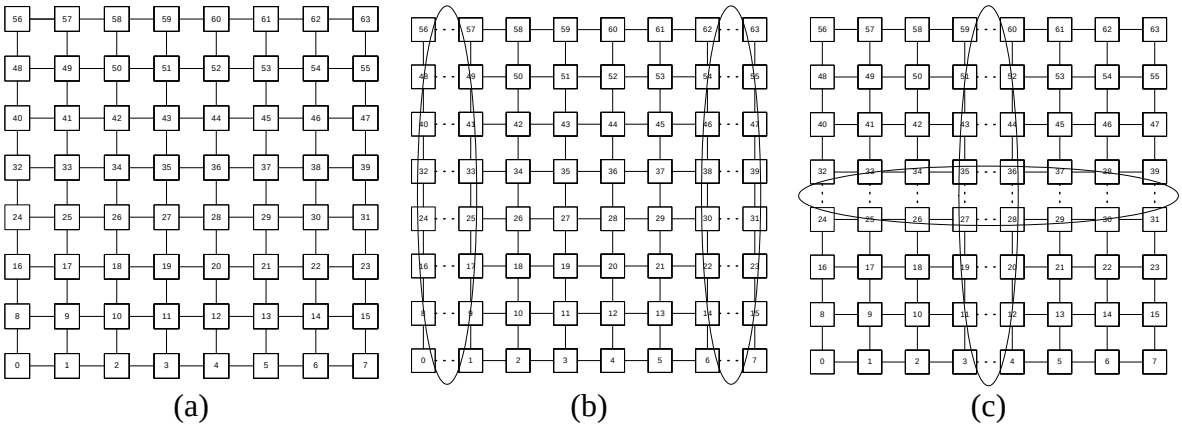
Observou-se que, apesar da rede estar implementada com canais virtuais, a adoção de tal

mecanismo não pôde evitar a flutuação de carga. Conclui-se que apenas a adoção de canais virtuais não traz garantias de QoS para a rede, sendo necessário também mecanismos que realizem o tratamento dos fluxos. Tais mecanismos devem estabelecer: (*i*) *priorização* de encaminhamento de pacotes pertencentes a fluxos com requisitos mais rígidos de QoS; (*ii*) *gerenciamento de tráfego*, onde o acesso a recursos da rede é controlado e os pacotes são condicionados a serem encaminhados na rede dentro de limites pré-estabelecidos de vazão; (*iii*) *escalonamento*, onde é definida a classe de tráfego cujos pacotes devem ser transmitidos e (*iv*) *gerenciamento de filas*, onde é definido como são armazenados os pacotes que esperam ser transmitidos ou encaminhados pela rede. Através da adoção destes mecanismos diferenciam-se aplicações quanto aos seus requisitos de QoS, priorizando pacotes com maior urgência em serem atendidos, em detrimento a pacotes cuja classe não exija da rede um tratamento mais imediato.

6.3 Estudo de caso 3

O terceiro estudo de caso utiliza o mesmo modelo de rede do estudo de caso 1. Apenas o algoritmo de roteamento XY é utilizado e não são usadas implementações com canais virtuais. São observadas as conseqüências no desempenho decorrentes de alterações na estrutura da rede, mais especificamente na profundidade dos *buffers* dos roteadores. Nesse caso, são aumentados para 16 fits a profundidade dos *buffers* de 32 do total de 64 roteadores que compõem a rede, levando em consideração as medições de cpf médio (*avcpf*) e utilização de largura de banda (*abw*) exibidos na avaliação interna do estudo caso 1.

A Figura 72(a) mostra a rede sem alterações estruturais, ou seja, todos os buffers possuem profundidade de 8 fits. A primeira alteração foi realizada nos enlaces onde foram obtidos as maiores quantidade de ciclos para transmissão de cada flit, sendo neste caso os enlaces localizados nas bordas da rede (Figura 72(b)). A segunda alteração estrutural (Figura 72(c)) considera os enlaces onde verificou-se maior taxa de utilização de canais. Neste caso, foram alteradas as profundidades de *buffers* pertencentes aos roteadores localizados na bisecção XY.



(a) (b) (c)

Figura 72 - Linhas pontilhadas apresentam regiões onde foram redimensionados os *buffers* dos roteadores. Alterações consideram medições de parâmetros de desempenho do estudo de caso 1: (a) sem redimensionamento (b) roteadores das bordas; e (c) roteadores da bisecção XY.

6.3.1 Geração de tráfego

Foi utilizada a mesma geração de tráfego do estudo de caso 1, com nodos enviado 1000 pacotes utilizando o padrão de tráfego complemento em cada uma das simulações. Seis simulações foram realizadas, onde as taxas de injeção tiveram valores fixos respectivamente de 80, 120, 160, 240, 320 e 480 Mbps.

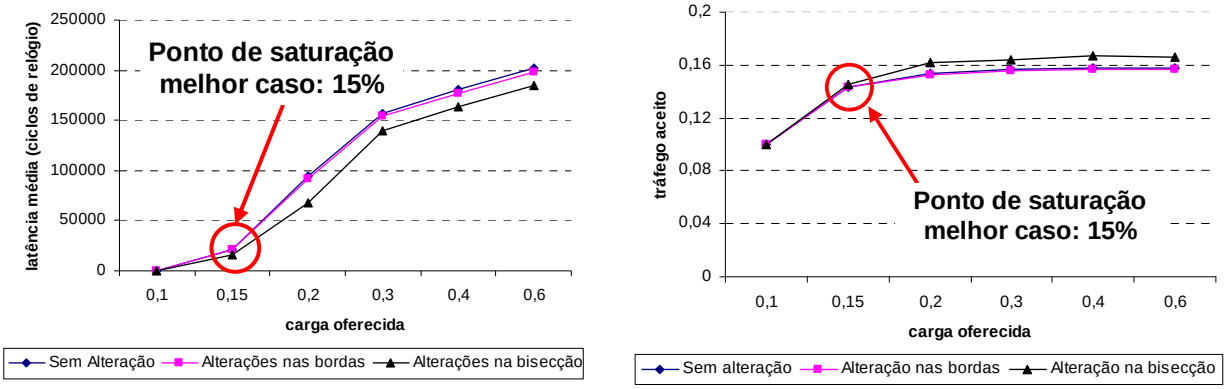
6.3.2 Avaliação de desempenho

Para avaliação externa da rede, procurou-se primeiramente verificar qual o seu ponto de saturação. Foram então construídos gráficos CNF, onde foram comparados os efeitos do redimensionamento dos *buffers* das bordas com o redimensionamento dos *buffers* da bisecção XY. A Tabela 16 mostra os valores de latência e tráfego aceito para as diferentes cargas oferecidas à rede que serviram como referência para construção dos gráficos.

Tabela 16 – Valores de latência média e tráfego aceito para os casos em que a rede não é alterada e em que os buffers da rede sofrem redimensionamento.

Carga oferecida	Latência média (ciclos)			Tráfego aceito (normalizado)		
	Sem alterações	Alterações nas bordas	Alterações na bisecção XY	Sem alterações	Alterações nas bordas	Alterações na bisecção XY
0,1	293	293	276	0,10009	0,10009	0,10009
0,15	20.854	21.174	16.272	0,14355	0,14378	0,14512
0,2	93.918	92.238	68.474	0,15352	0,15302	0,16216
0,3	157.200	154.349	140.033	0,15679	0,15559	0,16433
0,4	180.508	176.917	163.201	0,15754	0,15653	0,16655
0,6	201.774	197.520	185.231	0,15761	0,15701	0,16597

Observa-se que, até o ponto de saturação, não ocorre ganho de desempenho em se redimensionar os *buffers* da periferia da rede, em relação ao não redimensionamento de *buffers*. Um pequeno ganho só se observa a partir da carga oferecida de 15%, que é o ponto de saturação da rede para todos os casos. O melhor desempenho obtém-se com o redimensionamento de *buffers* na bisecção XY, isso porque há maior aceitação de pacotes originados na periferia da rede pelos roteadores pertencentes a bisecção. Desta forma, a latência média rede como um todo é reduzida em relação às configurações onde são re-dimensionados os roteadores das bordas da rede.



(a) (b)

Figura 73 – Gráficos CNF para as três configurações de rede utilizadas no Estudo de caso 3.

Outro gráfico extraído para avaliação externa foi o de distribuição de latências. Os itens (a), (b), (c), (d) e (e) da Figura 74 mostram as distribuições de latência para as cargas oferecidas a partir do ponto de saturação (15%), nas situações onde ocorre redimensionamento dos *buffers*. Para a carga oferecida de 10% não se observam diferenças significativas de valores de latência entre as duas formas de redimensionamento. Os destaques da Figura 74 indicam que, mesmo quando ocorre redimensionamento, a latência média aumenta com a carga oferecida, assim como o espalhamento dos valores da distribuição. Comparando o redimensionamento nas bordas com o realizado na bisecção XY, observa-se que a latência média sempre é menor para o caso em que são redimensionados os *buffers* da bisecção XY. No entanto, o ganho de desempenho para esta configuração é cada vez menor com o aumento da carga oferecida. Quando se redimensiona a bisecção XY, a latência média dos pacotes é 23% menor em relação ao redimensionamento nas bordas para a carga oferecida de 15%, enquanto que para a carga de 60%, a latência média da rede sofre redução de apenas 6%.

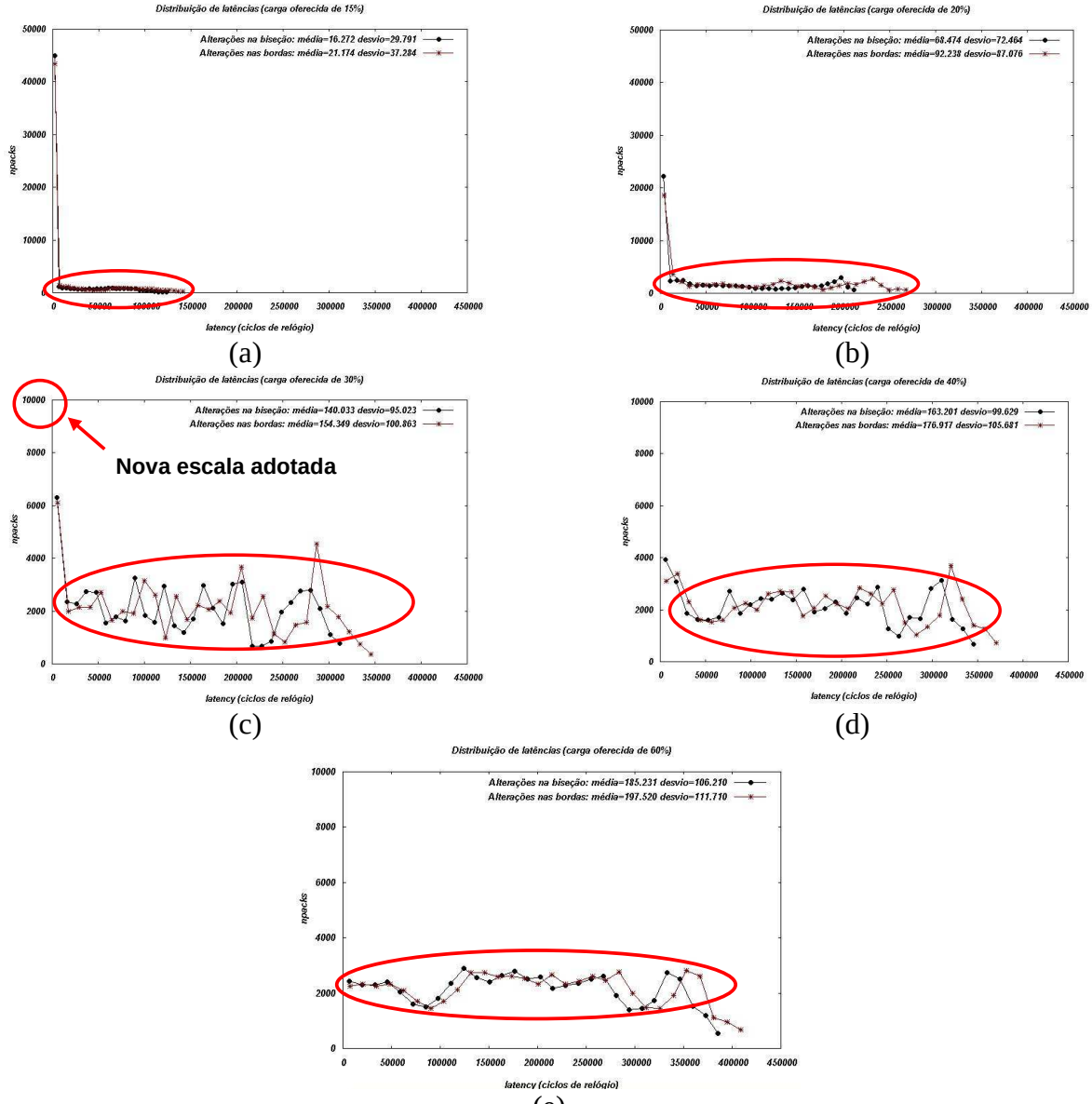
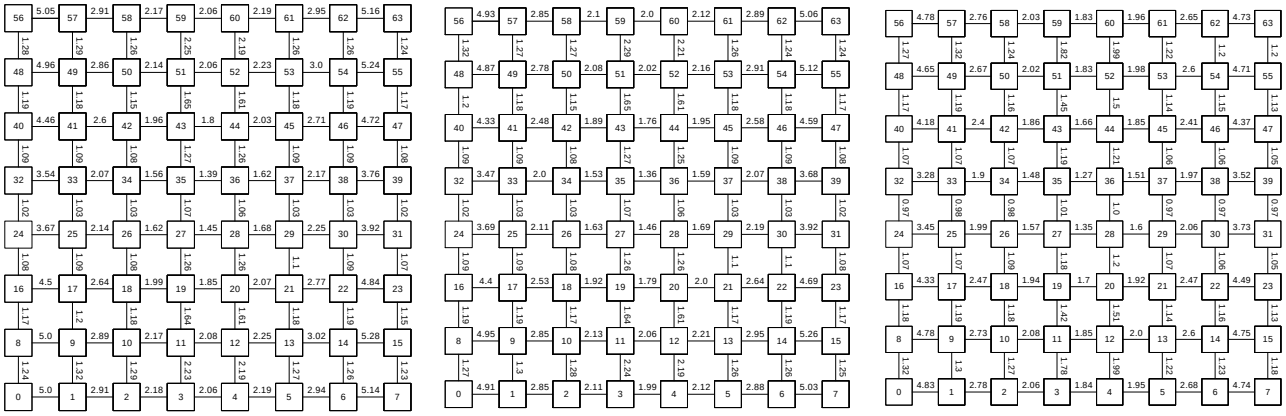


Figura 74 – Distribuições de latências comparando os casos onde há redimensionamento nas bordas e na bisecção XY da rede.

A avaliação interna de desempenho para este Estudo de Caso considera a contenção de flits nos *buffers*. A Figura 75 mostra *avcpf* para cada enlace, para uma carga oferecida de 20%. Escolheu-se esta carga porque ela já está acima do ponto de saturação da rede (que é de 15%). Observa-se que o *avcpf* no caso onde foram redimensionados os *buffers* das bordas apresenta os maiores valores em relação aos cenários onde foram redimensionados os *buffers* dos roteadores da bisecção XY. Tal fato comprova que é necessário considerar quais enlaces possuem as maiores taxas de ocupação (*abw*) para realizar o redimensionamento de *buffers*. Os menores valores de *avcpf* nos enlaces da rede com *buffers* redimensionados na bisecção causam as menores latências apresentadas nos gráficos da Figura 74.



(a) (b) (c)

Figura 75 - Valores de CPF médio obtidos nos enlaces (carga de 20%): (a) sem alterações nos buffers; (b) redimensionamento dos *buffers* da periferia; (c) redimensionamento dos buffers da bisecção XY.

Este experimento tem sua importância no fato de mostrar um parâmetro estrutural que, se ajustado adequadamente, oferece melhorias no desempenho a rede. Os resultados obtidos para o parâmetro de desempenho escolhido (*abw*) podem servir de entrada, por exemplo, para uma ferramenta que avalie compromissos de acréscimo de desempenho com o aumento da área consumida em-chip. Tal ferramenta pode ainda otimizar a quantidade/profundidade de *buffers* por roteador, verificando as regiões da rede onde localizam-se roteadores onde é mais adequado o acréscimo de fits em seus *buffers*. [GOO05] avalia o desempenho da rede avalia o desempenho da rede verificando sobras (*slacks*) na utilização de *buffers* e realiza o ajuste na rede através da eliminação de *slots* ociosos, diminuindo a área utilizada, mantendo os mesmos resultados de vazão e latência obtidos sem otimização.

6.4 Estudo de caso 4

A avaliação de desempenho utilizando simulação apresenta dois problemas: tempo de simulação, e por conseqüência reduzido número de pacotes transferidos durante a simulação. O quarto Estudo de Caso trata de um experimento realizado através de emulação, onde a geração de tráfego, o processamento dos núcleos e a coleta de dados para avaliação é integralmente realizada em FPGA. A placa utilizada nos experimentos foi uma VIRTEX XC2V4000, da Xilinx [XIL05].

A aplicação em questão é um comparador de seqüências de caracteres, sendo esta aplicação normalmente utilizada em bioinformática para verificação de similaridade em seqüências de DNA. O Capítulo 9 contém um anexo que descreve como é o processamento realizado pelos núcleos para comparação de seqüências e quais são as primitivas de comunicação utilizadas. O objetivo deste experimento é executar em FPGA um cenário de tráfego utilizando alguns dos conceitos sobre geração de tráfego e avaliação de desempenho mostrados nos Capítulos 4 e 5. A arquitetura de NoC utilizada é uma malha 3x3, onde são dispostos 8 roteadores conectados a processadores que executam o algoritmo de *Smith-Waterman* [SMI81] e um roteador conectado a um módulo de comunicação serial, onde são enviados dados de tráfego dos pacotes. A profundidade dos buffers é de 4 flits e a largura cada flit é de 8 bits. Não são utilizados canais virtuais.

6.4.1 Padrões de tráfego espacial e quantidade de pacotes gerados

Duas distribuições espaciais de tráfego são utilizadas. Uma ideal (Figura 76(a)), onde os roteadores estão dispostos de modo que a distância entre todos os pares origem-destino seja de 1 hop (situação denominada como *Cenario1*). Na segunda configuração (Figura 76(b)) procurou-se estabelecer maiores distâncias entre determinados pares origem-destino (situação denominada como *Cenario2*). Em ambos os cenários descritos as setas sólidas indicam os fluxos gerados.

Foi realizada a comparação de uma seqüência com 96 caracteres (constituindo a primeira linha da matriz – denominada como *sequencia1*) com outra de 100 caracteres (constituindo a primeira coluna da matriz – denominada como *sequencia2*). Ambas seqüências estão dispostas na matriz segundo mostrado na Figura 77. São gerados em cada processador 1200 pacotes de 6 flits. Estes flits contém a origem, o tamanho, 3 flits de controle e o flit com o resultado da comparação. Cada processador processa 12 colunas da matriz, porque temos 96 colunas distribuídas entre 8 processadores. O processador P8 gera 1100 pacotes porque ele processa a última coluna (que possui o resultado da comparação) e não envia resultados desta coluna para o processador P1.

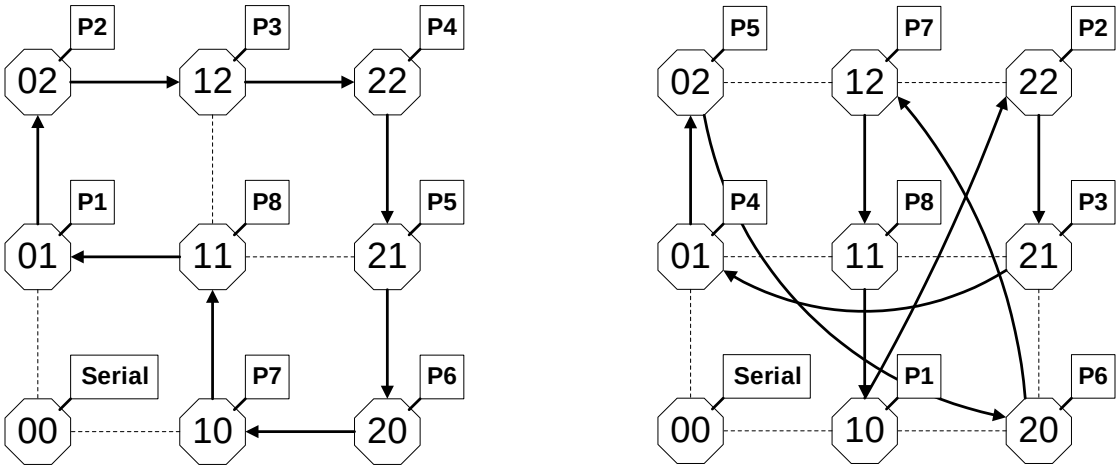


Figura 76 – Padrões de tráfego utilizados: (a) *Cenario1* – número de *hops* entre todos os pares origem-destino é igual a 1; (b) *Cenario2* – fluxos originados pelos processadores P1, P3, P5 e P6 possuem número de *hops* maior que 1.

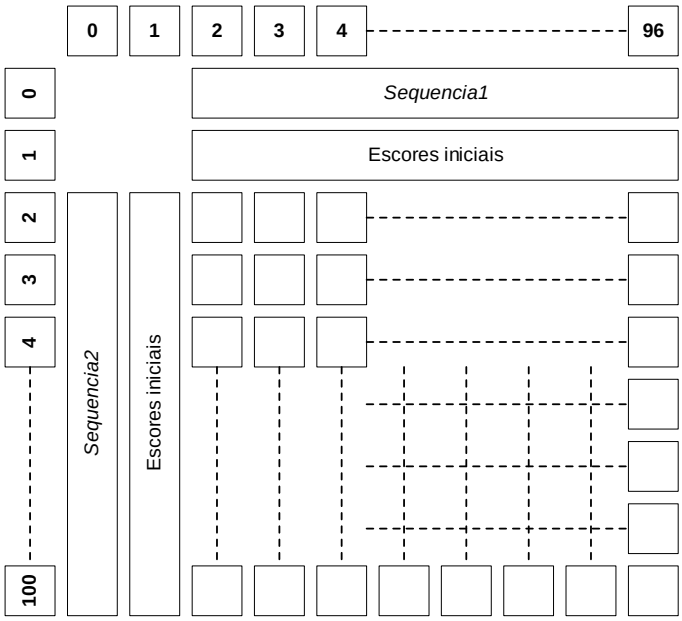


Figura 77 – Dimensões da matriz de comparação utilizada no Estudo de Caso 4.

6.4.2 Coleta de dados de geração de tráfego e para avaliação de desempenho

Os dados coletados para análise de como o tráfego é gerado e verificação de desempenho são mostrados na Figura 78. Os dados obtidos para verificação da *geração de tráfego* são coletados por um *sniffer* no enlace que conecta o gerador de tráfego ao núcleo ao qual ele está associado. Esta coleta é empregada quando se deseja conhecer o comportamento de um tráfego real no momento em que a aplicação que gera este tráfego executa sobre a rede. Desta forma, é possível utilizar este *trace* para auxiliar na caracterização deste tipo de aplicação, quando se deseja simular este tipo de tráfego. O parâmetro a ser medido com os dados desta coleta é a *carga oferecida*. Os dados medidos para *avaliação de desempenho* são coletados tanto no enlace que interliga um núcleo gerador à rede quanto no enlace que interliga a rede a um núcleo receptor. As métricas de avaliação de desempenho adotadas neste experimento são o *tráfego aceito* e *latência da rede*. A latência de rede é o intervalo de tempo entre a inserção do primeiro flit de um dado pacote na rede e a chegada do último flit do mesmo pacote no núcleo destino. A Figura 78 mostra dados coletados para avaliação de desempenho neste Estudo de Caso.

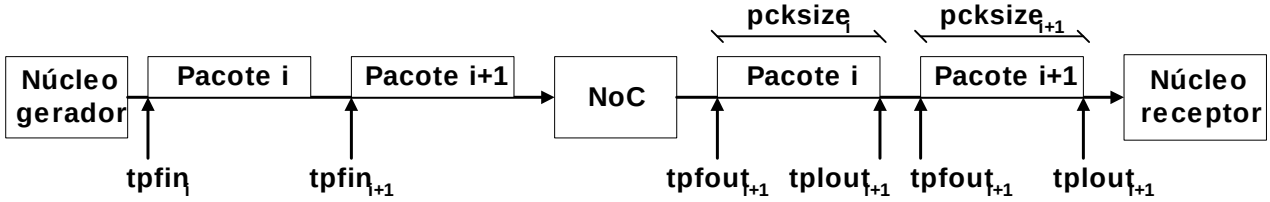


Figura 78 – Momentos capturados para verificação da carga oferecida e avaliação de desempenho.

Onde:

tpfin$_i$: momento em que o primeiro flit de um pacote *i* é injetado na rede;
tpfout$_i$: momento em que o primeiro flit de um pacote *i* chega ao Núcleo receptor;
tplout$_i$: momento em que o último flit de um pacote *i* chega ao Núcleo receptor;

Com relação à geração de tráfego, a carga oferecida por um pacote *i* (*offeredload*$_i$) considera o seu tamanho (*pcksize*$_i$), o momento em seu primeiro flit é disponibilizado para transmissão (*tpfin*$_i$) e o momento em que o primeiro flit de seu subseqüente é disponibilizado para transmissão (*tpfin*$_{i+1}$). A carga oferecida de um pacote *i* é definido pela Equação 29:

$$offeredload_i = \frac{pcksize_i * ncyclesflit}{tpfin_{i+1} - tpfin_i} \qquad \text{Equação 29}$$

O tráfego aceito de um pacote *i* (*acceptedtraffic*$_i$) considera o seu tamanho (*pcksize*$_i$), o momento de saída da rede do primeiro flit (*tpfout*$_i$) e o momento em que o primeiro flit de seu subseqüente sai da rede (*tpfout*$_{i+1}$). O tráfego aceito de um pacote *i* é definido pela Equação 30 (similar à Equação 23):

$$acceptedtraffic_i = \frac{pcksize_i}{tpfout_{i+1} - tpfout_i} \qquad \text{Equação 30}$$

O cálculo da latência de rede é similar ao mostrado na Equação 19. Para calcular a latência de um pacote *i* é necessário o conhecimento do momento em que o pacote está disponível para transmissão (*tpfin*$_i$) e do momento em que o último flit do pacote chega ao destino (*tplout*$_i$). A latência de rede para um pacote *i* (*latency*$_i$) é dado pela Equação 31:

$$latency_i = tplout_i - tpfin_i \qquad \text{Equação 31}$$

6.4.3 Resultados obtidos

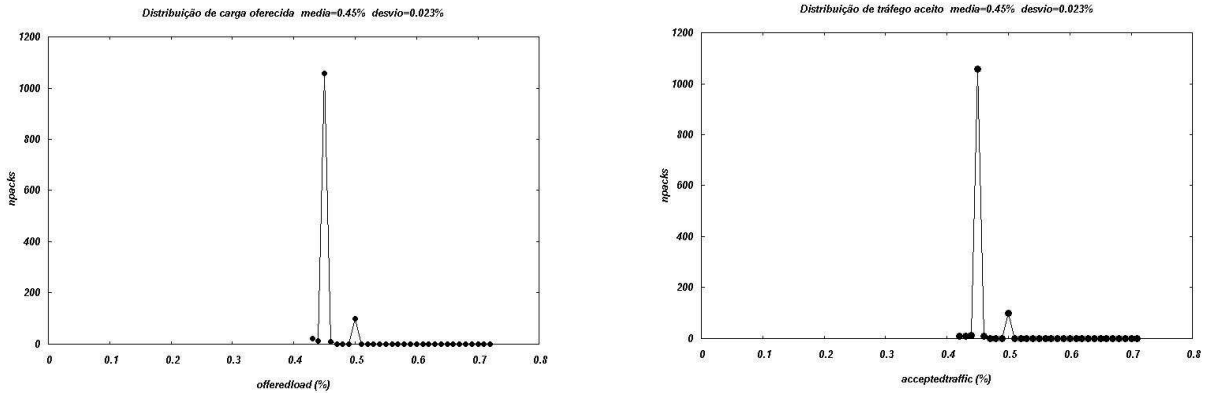
A Tabela 17 resume os resultados obtidos, tanto para as medições da carga oferecida ao sistema, quanto para os parâmetros de avaliação de desempenho adotados. É importante ressaltar que o objetivo desta Seção é mostrar, através de um exemplo simples, a possibilidade de se inserir módulos de captura de informações de carga oferecida por uma aplicação real, bem como de informações a respeito da recepção dos dados pelos núcleos-destino, em um experimento com reduzida duração de tempo, em comparação com a simulação (o tempo total de simulação para ambos os cenários foi 65.7ms).

Através da análise da Tabela 17 é possível perceber dois fatos. O primeiro fato diz respeito à manutenção das taxas de injeção serem mantidas nos destinos. Observa-se que tanto as taxas de injeção quanto as de recepção tiveram praticamente os mesmos valores em termos de média e desvio padrão (média de 0.45% da capacidade total da rede e desvio padrão de 0.023%). Estes baixos valores são justificados por dois fatores: (*i*) *tamanho reduzido do pacote* (6 flits) e (*ii*) *elevado intervalo entre pacotes* (possuindo valores em torno de 1300 ciclos). Tais fatores indicam

que não houve concorrência por recursos em nenhum dos dois cenários estabelecidos. A Figura 79 mostra as curvas de distribuição de carga oferecida (em (a)) e tráfego aceito (em (b)) ocorreram de maneira similar em todos os fluxos, tanto em *Cenario1* quanto em *Cenario2*. A reduzida carga na rede explica-se pelo fato da aplicação gastar a maior parte do tempo realizando *processamento* do algoritmo *Smith-Waterman* e gastando um reduzido tempo na *comunicação* entre os núcleos.

Tabela 17 – Resultados de desempenho do Estudo de Caso 4.

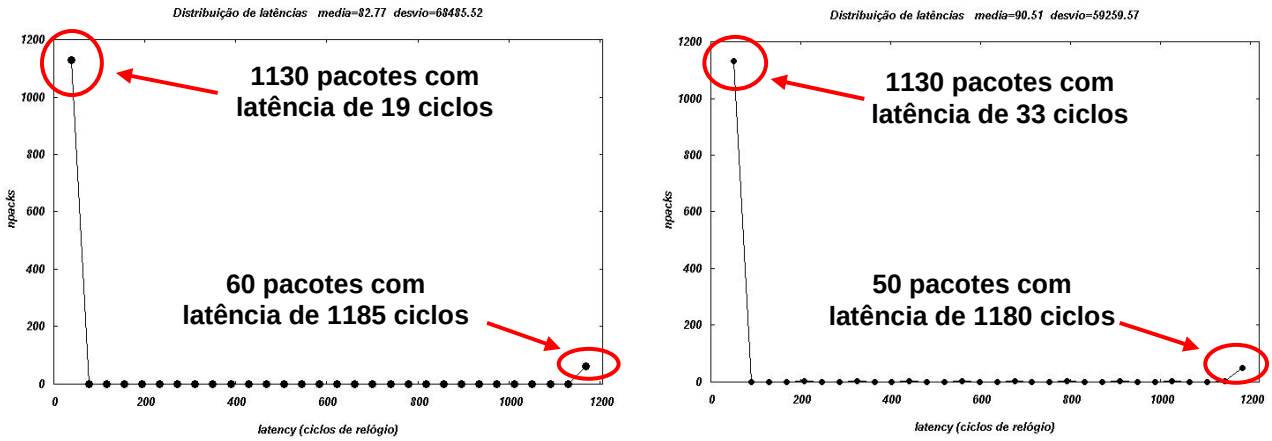
	origem	destino	hops	geração de tráfego		avaliação de desempenho			
				carga oferecida (% da capacidade máxima)		tráfego aceito (% da capacidade máxima)		latência (ciclos de relógio)	
				média	desvio	médio	desvio	média	desvio
Cenario1	P1	P2	1	0.46	0.062	0.46	0.061	83	68485.52
	P2	P3	1	0.45	0.023	0.45	0.023	19	0.0192
	P3	P4	1	0.45	0.023	0.45	0.039	19	0.0192
	P4	P5	1	0.45	0.023	0.45	0.039	19	0.0192
	P5	P6	1	0.45	0.023	0.45	0.023	19	0.0192
	P6	P7	1	0.45	0.023	0.45	0.023	19	0.0192
	P7	P8	1	0.45	0.023	0.45	0.024	19	0.0192
	P8	P1	1	0.45	0.023	0.45	0.023	19	0.0192
Cenario2	P1	P2	3	0.46	0.071	0.46	0.0625	90	59259.57
	P2	P3	1	0.45	0.023	0.45	0.023	19	0.0192
	P3	P4	2	0.45	0.023	0.45	0.023	26	0.0192
	P4	P5	1	0.45	0.023	0.45	0.023	19	0.0192
	P5	P6	4	0.45	0.023	0.45	0.023	40	0.3475
	P6	P7	3	0.45	0.023	0.45	0.023	33	0.1325
	P7	P8	1	0.45	0.024	0.45	0.024	19	0.54
	P8	P1	1	0.45	0.023	0.45	0.023	17	27.08



(a) (b)

Figura 79 – Curvas de distribuições de carga oferecida e tráfego aceito similarmente geradas pelos fluxos do estudo de Caso 4.

Outro fato percebido diz respeito às latências obtidas para o fluxo com origem em P1. Verifica-se-se que a latência média para os pacotes pertencentes a este fluxo possuem valores de média e desvio padrão diferentes do restante dos fluxos, em ambos os cenários. A Figura 80 mostra as duas distribuições de latência para o fluxo gerado por P1, em *Cenario1* (a) e *Cenario2* (b).



(a) (b)

Figura 80 – Distribuições de latência para o fluxo gerado por P1: (a) em *Cenario1*; (b) em *Cenario2*.

A Figura 80(a) mostra que, em torno de 1130 pacotes possuem a latência de 19 ciclos, enquanto que em torno de 60 pacotes tiveram latência de 1185 ciclos. Em (b) observa-se que 1130 pacotes possuem a latência de 33 ciclos, enquanto que em torno de 50 pacotes tiveram latência de 1180 ciclos. Os valores elevados de latência ocorrem no início da execução da aplicação, quando P1 não possui dados a receber de P8. Pelo fato de P1 possuir armazenado os escores iniciais da matriz, ele tem condições de enviar dados de seu processamento imediatamente. O problema é que P2 não consegue processar os dados que recebe na mesma velocidade em que P1 os produz, o que leva os pacotes com resultados da primeira coluna serem aceitos por P2 muito tempo após eles serem disponibilizados, o que gera um alto valor de latência para estes pacotes.

Quando P1 é reutilizado (começa a processar outra coluna), ele depende do processamento de P8. Durante a espera de P8 por P1, o processador P2 consegue computar os dados anteriormente gerados por P1. Quando P1 voltar a gerar dados, P2 poderá consumir os pacotes no mesmo ritmo em que o processador P1 produz, porque agora os processadores demandam a mesma quantidade de tempo para processar uma comparação, pois devem esperar um processamento anterior para realizar o seu próprio processamento.

6.5 Conclusões

Este Capítulo mostrou estudos de caso que serviram para validar os conceitos sobre geração de tráfego e avaliação de desempenho vistos nos capítulos anteriores. Através desta validação foi possível visualizar com maior detalhe os eventos que ocorrem na rede.

No primeiro estudo de caso foram gerados pacotes com taxas fixas utilizando o padrão de tráfego complemento. Verificou-se o efeito causado pela adoção de algoritmos de roteamento com diferentes naturezas. Observou-se que a taxa de utilização dos canais é diretamente afetada pela maneira como os roteadores decidem encaminhar os pacotes. Outra constatação está relacionada à utilização de canais virtuais, através dos quais é possível reduzir a contenção de. A adoção de canais virtuais também permitiu a redução no espalhamento de latências no cenário de tráfego proposto. Outro conceito abordado foi o de ponto de saturação de uma rede, que é uma taxa a partir da qual não se observa acréscimo no tráfego aceito, além de acarretar um rápido crescimento na latência

média dos pacotes.

O segundo estudo de caso envolveu a geração de tráfego com taxas de injeção variáveis, concorrendo com taxas de injeção fixas. Foi discutido o problema da flutuação de carga, que são desvios verificados no tráfego aceito em relação à carga oferecida. Observou-se que não basta somente a adoção de canais virtuais para evitar queda no desempenho. Torna-se necessário também políticas adotadas para tratamento de fluxos como priorização, gerenciamemto de tráfego, escalonamento e gerenciamento de filas.

O terceiro estudo de caso mostrou que o correto dimensionamento de buffers pode levar a um melhor desempenho da rede. A análise de utilização dos canais pode servir como entrada para uma ferramenta de dimensionamento de buffers nas diversas regiões na rede, alcançando com isto acréscimo de desempenho com reduzido gasto em área.

Finalmente, o quarto estudo de caso abordou a geração de tráfego e coleta de dados para avaliação de desempenho realizados através de emulação. Foi mostrado um método para coleta de traces de aplicações executando diretamente em um FPGA, o que pode traz aos experimentos a possibilidade de se trabalhar com dados reais em um reduzido intervalo de tempo.

Com os experimentos realizados, foi mostrado como é possível estabelecer diferentes cenários através do método de geração de tráfego proposto e como testar a rede através dos parâmetros de avaliação de desempenho apresentados.

7 CONCLUSÕES

Neste Capítulo são resumidas as contribuições desta Dissertação na geração de tráfego e avaliação de desempenho em NoCs, as considerações finais e os trabalhos futuros que poderão ser desenvolvidos.

7.1 Contribuições na geração de tráfego

O Estado da Arte mostrou que, em relação à geração de tráfego, a maior parte dos trabalhos utiliza a distribuição espacial uniforme, onde todos os núcleos possuem a mesma probabilidade de serem destinos. Desenvolveu-se no período da elaboração desta dissertação um gerador de tráfego, onde podem ser estabelecidos cenários de distribuição espacial empregados em aplicações paralelas, como *uniforme*, *não-uniforme*, *matrix transpose* e *complemento*, bem como a especificação do destino para um determinado núcleo origem. Nos experimentos foi utilizada a distribuição complemento. A utilização de tal distribuição se justifica por possibilitar a análise do que ocorre nas bisecções de uma NoC.

O Estado da Arte ainda mostrou que a maior parte dos trabalhos adotou taxas variáveis de injeção de pacotes. Foi proposto neste trabalho um método para geração de tráfego com taxas de injeção, onde considera-se o tamanho dos pacotes, o intervalo entre pacotes, o intervalo entre saídas de pacotes e a quantidade de pacotes a serem gerados. Também foi proposto um método para modelagem de tráfego, onde são atribuídas propriedades probabilísticas ao processo de geração, de modo a se aproximar da caracterização de aplicações reais. Tal método utiliza uma referência temporal (*pcktmp* – momento de criação de um pacote), onde o gerador de tráfego monitora um relógio global para saber o momento ideal em que um pacote deve ser enviado. A adoção de uma referência temporal foi inspirada na proposta de [GEN05a].

7.2 Contribuições na avaliação de desempenho

A avaliação de desempenho foi realizada de maneira externa na maioria dos trabalhos revisados. [BOL04a] foi o único a realizar análise interna, com a verificação da carga relativa dos enlaces. As propostas estudadas serviram como referência para as definições de métricas para avaliação externa de desempenho (tráfego aceito e latência) e sua visualização (gráficos CNF e de distribuição de latências) propostas nesta Dissertação.

Foi proposto nesta Dissertação um método para análise interna, o que possibilita a verificação do tráfego nos canais que interligam os roteadores. As métricas propostas para este tipo de análise avaliam a contenção (*avcpf* – número médio de ciclos para transmissão de flits) e o encaminhamento (*abw* – largura de banda média utilizada) de pacotes. A visualização de dados relacionados a estas métricas é oferecida por gráficos de superfície e mapas textuais de tráfego. Também foi considerada para avaliação interna a distribuição de tráfego aceito, onde se verifica em

cada canal percorrido por um fluxo, se as propriedades do tráfego de origem são mantidas ao longo do caminho percorrido pelo fluxo até chegar em seu destino. Um fenômeno que pode ocorrer devido ao congestionamento da rede é a flutuação de carga, onde os pacotes podem demorar a ser liberados (antes da saturação), causando queda na em sua taxa de encaminhamento e também liberação em rajada (a partir do ponto de saturação), aumentando sua taxa de encaminhamento em relação à sua geração.

Para avaliação externa, foram mostrados métodos para geração de gráficos CNF (para avaliação de latência média e tráfego aceito), bem como de distribuição de latências e de tráfego aceito. Os gráficos CNF possibilitam a verificação do ponto de saturação da rede, que é taxa máxima em que a rede encaminha os pacotes para os núcleos de destino. A distribuição de latências permite verificar o espalhamento de valores desta medida para um determinado fluxo, onde se avalia se a transmissão de pacotes obedece às restrições temporais especificadas em sua geração. A distribuição de tráfego aceito auxilia na verificação de flutuação de carga, a exemplo do que ocorre na avaliação interna.

7.3 Conclusões e trabalhos futuros

Esta Dissertação teve como enfoque a geração de tráfego e avaliação de desempenho para a NoC HERMES, onde até então era gerado tráfego sem qualquer caracterização e a avaliação de desempenho era baseada apenas em termos de valores de latência. Os métodos apresentados auxiliarão para a verificação de cumprimento ou não de requisitos de QoS que futuramente serão atribuídos aos pacotes da NoC HERMES, que atualmente oferece serviços apenas de acordo com a categoria *best-effort*.

Entre as atividades previstas relacionam-se:

Na geração de tráfego:

- Desenvolvimento de um método de geração de tráfego auto-similar (propriedade observada em aplicações multimídia);
- Introdução de descritores de tráfego nos pacotes da NoC HERMES, como PCR, SCR e MBS;
- Modelagem de tráfego espacial envolvendo aplicações complexas (como decodificadores MPEG e centrais PABX, utilizadas em telefonia).

Na avaliação de desempenho

- Desenvolvimento de ferramentas para análise de tráfego em tempo real (quando forem realizados experimentos envolvendo emulação). Devido à grande quantidade de pacotes que podem trafegar na rede, é necessário um mecanismo de coleta de amostras para análise;
- Desenvolvimento de métodos para verificação de auto-similaridade, para verificação do cumprimento ou não às restrições de QoS para esse tipo de aplicação;

- Verificação da influência de parâmetros de avaliação interna de desempenho (como *abw* e *cpf*) no consumo de energia e utilização de *buffers* de roteadores e desenvolvimento de algoritmos para otimização de rede que recebem como entrada estes parâmetros;
- Estender os métodos de avaliação de desempenho (interna e externa) apresentados para redes com suporte a QoS, introduzindo métodos para interpretar descritores de tráfego, especificados nos pacotes.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[ADR03] Adriahantenaina, A.; Greiner, A.; Mortiez, L.; Zeferino, C. *SPIN: a Scalable, Packet Switched, On-chip Micro-Network*. **In**: Design Automation and Test in Europe Conference and Exhibition (DATE'03), 2003, pp. 70-73.

[BEN01] Benini, L.; De Micheli, G. *Powering Networks on Chips: Energy-Efficient and Reliable Interconnect Design for SoCs*. **In**: International Symposium on System Synthesis (ISSS'01), 2001, pp. 33 –38.

[BEN02] Benini, L.; De Micheli, G. *Networks on Chips: a New SoC Paradigm*. IEEE Computer, v.35, n.1, 2002, pp. 70-78.

[BER01] Bergamaschi, R.; Bhattacharya, S.; Wagner, R.; Fellenz, C.; Muhlada, M.; White, F.; Daveau J. M.; Lee, W. R. *Automating the Design of SoCs Using Cores*. IEEE Design and Test of Computers, v.18, n.5, 2001, pp. 32–45.

[BOL04a] Bolotin, E.; Cidon, I.; Ginosar, R.; Kolodny, A. *QNoC: QoS Architecture and Design Process for Network on Chip*. Journal of Systems Architecture, v.50, n.2, 2004, pp. 1-24.

[BOL04b]Bolotin, E.; Cidon, I.; Ginosar, R.; Kolodny, A. *Cost Considerations in Network on Chip*. Integration, the VLSI Journal, v.38, n.1, 2004, pp. 19-42.

[BOL04c]Bolotin, E.; Morgenshtein, A.; Cidon, I.; Ginosar, R.; Kolodny, A. *Automatic Hardware-Efficient SoC Integration by QoS Network On Chip*. **In**: Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS'04), 2004, pp. 479 - 482.

[CAS03] Castanheira, L. *Medição e Desempenho de Hardware ATM no Nível de Célula*. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2003, 114p.

[CHE99] Chen, T.; *Characterization of ATM on-off traffic from cell traffic measurements*. **In**: 3rd IEEE International Workshop on Broadband Switching Systems (BSS'99), 1999, pp. 53-57.

[DAL01] Dally, W.; Towles, B. *Route Packets, not Wires: On-Chip Interconnection Networks*. **In:** Design Automation Conference (DAC'01), 2001, pp. 684-689.

[DAL04] Dally, W. J.; Towles, B. *Principles and Practices of Interconnection Networks*. Elsevier, San Francisco. 2004, 550p.

[DUA03] Duato, J.; Yalamanchili, S.; Ni, L. *Interconnection Networks – An Engineering Approach*. Elsevier, San Francisco. 2003, 600p.

[GEN05a] Genko, N.; Atienza, D.; De Micheli, G.; Mendias, J.; Hermida, R.; Catthoor, F. *A Complete Network-On-Chip Emulation Framework*. **In:** Design Automation and Test in Europe (DATE'05), 2005, pp. 246-251.

[GEN05b]Genko, N.; Atienza, D.; De Micheli, G.; Benini, L.; Mendias, J.; Hermida, R.; Catthoor, F. *A Novel Approach for Network on Chip Emulation*. **In:** IEEE International Symposium on Circuits and Systems (ISCAS'05), 2005, pp. 2365-2368.

[GOO02] Goossens, K.; van Meerbergen, J.; Peeters, 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.

[GOO05] Goossens, K.; Dielissen, J.; Gangwal, O.; Pestana, S.; Radulescu, A.; Rijpkema, E. *A Design Flow for Application-Specific Networks On Chip with Guaranteed Performance to Accelerate SOC Design and Verification*. **In:** Design Automation and Test in Europe (DATE'05), 2005, pp. 1182 - 1187.

[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.

[HÄN98] Händel, R.; Huber, M.; Schröder, S. *ATM Networks – Concepts Protocols Applications*. Addison-Wesley. 1998, 327p.

[HEM00]Hemani, A.; Jantsch, A.; Kumar, S.; Postula, A.; Öberg, J.; Millberg, M.; Lindqvist, D. *Network on a Chip: An Architecture for Billion Transistor Era*. **In**: Proceeding of the IEEE NorChip Conference, 2000, pp. 166-173.

[HU04] Hu, J.; Marculescu R. *Dyad - Smart Routing for Networks on Chip.* **In:** Design Automation Conference (DAC'04), 2004, pp. 260-263.

[KEU00] Keutzer, K.; Malik S.; Newton, R.; Rabaey, J.; Sangiovanni-Vincentelli, A. *System-Level Design: Orthogonalization of Concerns and Platform-Based Design*. IEEE Transactions on Computer-Aided Design, v.19, n.12, 2000, pp. 1523-1543.

[KIM92] Kim, J. H.; Chien, A. A. *An Evaluation of the Planar/Adaptive Routing*. **In**: Proceedings of the 4th IEEE Symposium on Parallel and Distributed Processing, 1992, pp. 470-478.

[LAV96] Lavenier, D. *Dedicated Hardware for Biological Sequence Comparison*. The Journal of Universal Computer Science, v. 2, n. 2, 1996, pp. 77-86.

[MOR04]Moraes, F.G.; Calazans, N.L.V.; Mello, A.V.; Möller, L. H.; Ost,L. C. *HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip*. Integration, the VLSI Journal, v.38, n.1, 2004, pp. 69-93.

[OCP05] *Open Core Protocol*. Capturado em: http://www.ocpip.org/home.

[OST05] Ost, L. C., Mello, A. V.; Palma, J. C. S.; Calazans, N. L. V.; Moraes, F. G. *MAIA - A Framework for Networks on Chip Generation and Verification*. **In**: Asia South Pacific Design Automation Conference (ASP-DAC'05), v.1, 2005, pp. 18-20.

[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, v.54 , n.8, 2005, pp. 1025-1040.

[PET97] Pétrot, F.; Hommais, D.; Greiner, A. *A Simulation Environment for Core Based Embedded Systems*. **In**: Proceedings of the 30th Annual Simulation Symposium, 1997, pp. 86-91.

[PRY95] Pricker, M. De. *Asynchronous Transfer Mode: Solution for Broadband ISDN*. Prentice Hall. 3rd Edition 1995, 332p.

[RAD05] Radulescu, A.; Dielissen, J.; Pestana, S.; Gangwal, O.; Rijpkema, E.; Wielage, P.; Goossens, K. *An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory*

Abstraction, and Flexible Network Configuration. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v.24 , n.1, 2005, pp. 4 - 17.

[RIJ03] Rijpkema, E.; Goossens, K.; Radulescu, A.; Dielissen, J.; van Meerbergen, J.; Wielage, P.; Waterlander, E. *Trade Offs in the Design of a Router with both Guaranteed and Best-Effort Services for Networks On Chip*. **In**: Design Automation Test Europe (DATE'03), 2003, pp. 350-355.

[SAN05] Santi, S.; Lin, B.; Kocarev, L.; Maggio, G.; Rovatti, R.; Setti, G. *On the Impact of Traffic Statistics on Quality of Service for Networks on Chip*. **In**: IEEE International Symposium on Circuits and Systems (ISCAS'05), 2005, pp. 2349 - 2352.

[SCH97] Schaller, R. *Moore´s law: past, present and future*. IEEE Spectrum, v.34, n.6, 1997, pp. 53-59.

[SIE94] Siemens. *OMI 324: PI-Bus – Ver.0.3d*. Munich, Siemens AG, 1994.

[SGR01] Sgroi, M.; Sheets, M.; Mihal, A.; Keutzer, K.; Malik, S.; Rabaey, J.; Vincentelli, A. S. *Addressing the System-on-Chip Interconnect Woes Through Communication-Based Design*. **In**: Design Automation Conference (DAC'01), 2001, pp. 667–672.

[SMI81] Smith, T. F.; Waterman, M. S. *Identification of Commom Molecular Subsequences*. Journal of Molecular Biology, v. 147, n. 1, 1981, pp. 195-197.

[TAN97] Tanenbaum, A. *Redes de Computadores*. Editora Campus, 1997, 923p.

[TED05] Tedesco, L. P.; Mello, A. V.; Garibotti, D.; Calazans, N. L. V.; Moraes, F. G. *Traffic Generation and Performance Evaluation for Mesh-based NoCs*. **In**: 18th annual Symposium on Integrated Circuits and System Design (SBCCI'05), 2005, pp. 184-189.

[VSI00] Virtual Socket Interface *Alliance Virtual Component Interface Standard*. OCB 2.1.0, 2000, 71p.

[VEL04] Vellanki, P.; Banerjee, N.; Chatha, K. *Quality-of-service and error control techniques for network-on-chip architectures*. **In**: Proceedings of the 14th ACM Great Lakes Symposium on VLSI (GLSVLSI '04), 2004, pp. 45-50.

[XIL05] Xilinx, Inc. *http://www.xilinx.com*. Capturado em: 2005.

[YE04] Ye, T.; Benini, L.; De Micheli, G. *Packetization and Routing Analysis of On-Chip Multiprocessor Networks*. Journal of Systems Architecture, v.50, n.2-3, 2004, pp. 81-104.

[YUM00]Yum, K.; Vaidya, A.; Das, R.; Sivasubramaniam, A. *Investigating QoS Support for Traffic Mixes with the MediaWorm Router*. **In**: Proceedings of the 6th International Symposium on High-Performance Computer Architecture (HPCA-6), 2000, pp. 97-106.

[ZEF03] Zeferino, C. *Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho*. Tese de Doutorado, PPGC-UFRGS, 2003, 194p.

[ZHA91a]Zhang, L. *VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks*. IEEE Transactions on Computer Systems, v.9, n.2, 1991, pp. 101-124.

[ZHA91b]Zhang, H.; Keshav, S. *Comparison of Rate-based Service Disciplines*. **In**: Proceedings of the Conference on Communications Architecture & Protocols (SIGCOMM '91), 1991, pp. 113-121.

9 ANEXO I: A APLICAÇÃO COMPARADOR DE SEQUÊNCIAS

A aplicação aqui apresentada calcula o grau de semelhança entre duas seqüências de caracteres, sendo muito utilizada em bioinformática para verificação do grau de similaridade entre seqüências de DNA, as quais podem ser facilmente representadas por cadeias de caracteres, devendo cada caractere ser A, T, G ou C. A escolha de tal aplicação se justifica pelo fato de ser possível termos *paralelismo* de processamento e *comunicação* entre processadores, que transferem entre si resultados de similaridade entre caracteres.

O algoritmo que será aqui utilizado para verificação do nível de similaridade entre duas seqüências é o de *Smith-Waterman* [SMI81]. Este algoritmo realiza duas operações básicas: (*i*) substituição de caracteres, ou (*ii*) inserção/deleção de *gaps* (espaços entre caracteres). Através de uma seqüência destas operações, é possível então detectar o grau de similaridade de uma seqüência em relação à outra.

9.1.1 O algoritmo de Smith-Waterman

A comparação de duas seqüências *X* e *Y* através do algoritmo de *Smith-Waterman* é realizada como segue [LAV96]. Considerando $X = (x_1, x_2, ..., x_n)$ e $Y = (y_1, y_2, ..., y_m)$ duas seqüências a serem comparadas em uma matriz de tamanho *n*x*m*. Considerando $p(x, y)$ o custo de se transformar *x* em *y* e *g* o custo de inserção/remoção de *gaps*. O algoritmo de *Smith-Waterman* realiza a seguinte recursão apresentada na Figura 81, onde *H(i,j)* representa a similaridade entre dois caracteres em uma posição (*i* , *j*):

$$H(i,j) = MAX \begin{cases} H(i-1, j-1) + p(i,j) \\ H(i-1, j) + g \\ H(i, j-1) + g \end{cases}$$

Sendo MAX o valor máximo.

Figura 81 – Recursão realizada pelo algoritmo de Smith-Waterman.

A idéia empregada para paralelizar a recursão mostrada na Figura 81 é associar um elemento de processamento com cada valor de *H(i , j)*. Considerando cada processador denotado com $P_{i,j}$, recebendo e encaminhado resultados de comparações de caracteres de acordo com o fluxo mostrado na Figura 82. Supondo que cada processador realiza a computação expressa na Figura 81. O conjunto de dados requisitados por $P_{i,j}$ é representado por setas sólidas. *H(i-1 , j-1)* é produzido por $P_{i-1,j-1}$, *H(i , j-1)* é produzido por $P_{i,j-1}$ e *H(i-1 , j)* é produzido por $P_{i-1,j}$. Com estas informações, o processador $P_{i,j}$ calcula *H(i , j)* e fornece aos processadores $P_{i+1,j+1}$, $P_{i+1,j}$ e $P_{i,j+1}$ o resultado obtido (este dado percorre os caminhos representados pelas setas pontilhadas).

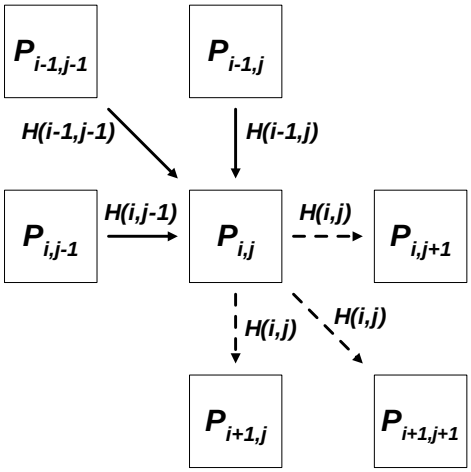


Figura 82 – Fluxo de dados do algoritmo de *Smith-Waterman*.

9.1.2 Exemplo de comparação de duas seqüências de caracteres

Um exemplo de preenchimento de uma matriz 7x7 (Tabela 18) é mostrado. Considerando as seqüências a serem comparadas *X* = CACACA e *Y* = GATACA e adotando-se o seguinte esquema de pontuação:

- *p(i , j) = 1*, se o caractere *i* da seqüência *X* for IGUAL ao *j* caractere da seqüência *Y*;
- *p(i , j) = -1*, se o caractere *i* da seqüência *X* for DIFERENTE ao caractere *j* da seqüência *Y*;
- *g* (penalidade para inserção de espaços entre caracteres) = -2.

Tabela 18 – Matriz exemplo.

	0	1	2	3	4	5	6	7
0			**C**	**A**	**C**	**A**	**C**	**A**
1								
2	**G**							
3	**A**							
4	**T**							
5	**A**							
6	**C**							
7	**A**							

Em seguida são inseridos os escores iniciais como sendo uma progressão aritmética de razão igual ao *g* e primeiro elemento igual a 0 (Tabela 19).

Tabela 19 – Matriz com escores iniciais.

	0	1	2	3	4	5	6	7
0			**C**	**A**	**C**	**A**	**C**	**A**
1		**0**	**-2**	**-4**	**-6**	**-8**	**-10**	**-12**
2	**G**	**-2**						
3	**A**	**-4**						
4	**T**	**-6**						
5	**A**	**-8**						
6	**C**	**-10**						
7	**A**	**-12**						

Chega o momento de calcular o conteúdo das células restantes da matriz. Por exemplo, para calcular o valor da célula da matriz localizada nas coordenadas (2,2):

$$H(2,2) = MAX \begin{cases} H(1,1) + p(2,2) \\ H(1,2) + (-2) \\ H(2,1) + (-2) \end{cases}$$

$$H(2,2) = MAX \begin{cases} 0 + (-1) = -1 \\ -2 + (-2) = -4 \\ -2 + (-2) = -4 \end{cases}$$

$$H(2,2) = -1$$

Figura 83 – Cálculo de *H* para a célula (2,2).

A Tabela 20 apresenta a matriz totalmente preenchida:

Tabela 20 – Exemplo de matriz totalmente preenchida.

	0	1	2	3	4	5	6	7
0			C	A	C	A	C	A
1		0	-2	-4	-6	-8	-10	-12
2	G	-2	-1	-3	-5	-7	-9	-11
3	A	-4	-3	0	-2	-4	-6	-8
4	T	-6	-5	-2	-1	-3	-5	-7
5	A	-8	-7	-4	-3	0	-2	-4
6	C	-10	-7	-6	-3	-2	1	-1
7	A	-12	-9	-6	-5	-2	-1	2

O valor da célula final da matriz (no exemplo a célula final é a com coordenadas (7,7) e seu conteúdo é 2) corresponde ao grau de semelhança entre as duas seqüências. Quanto maior esse valor, maior é a semelhança entre as seqüências. O valor máximo é igual ao tamanho das seqüências (6), se as duas seqüências forem idênticas.

9.1.3 Preenchimento da matriz de comparação em um sistema multiprocessado

O preenchimento da matriz de comparação em um sistema multiprocessado é realizado de modo que cada célula calculada seja de responsabilidade de um processador. Ainda, a cada processador é atribuída uma ou mais colunas da matriz, dependendo do número de processadores utilizados e das dimensões da matriz. Para o exemplo mostrado na Seção 9.1.2, o preenchimento da matriz com 6 processadores é realizado de acordo com o mostrado na Tabela 21. As linhas diagonais indicam em quais pontos da matriz as células estão sendo calculadas em paralelo.

Tabela 21 - Matriz sendo preenchida por 6 processadores.

	0	1	2	3	4	5	6	7
0			**C**	**A**	**C**	**A**	**C**	**A**
1		**0**	**-2**	**-4**	**-6**	**-8**	**-10**	**-12**
2	**G**	**-2**	P1	P2	P3	P4	P5	P6
3	**A**	**-4**	P1	P2	P3	P4	P5	↓
4	**T**	**-6**	P1	P2	P3	P4	↓	↓
5	**A**	**-8**	P1	P2	P3	↓	↓	↓
6	**C**	**-10**	P1	P2	↓	↓	↓	↓
7	**A**	**-12**	P1	↓	↓	↓	↓	↓

Para o caso em que um dos tamanhos de seqüência for maior do que o número de processadores disponíveis é necessário então reutilizar os processadores para realizar o processamento de outras colunas. A Tabela 22 mostra um exemplo onde P1, P2 e P3 (processadores disponíveis) são re-utilizados para calcular o conteúdo das colunas 5, 6 e 7.

Tabela 22 – Matriz sendo preenchida por 3 processadores.

	0	1	2	3	4	5	6	7
0			C	A	C	A	C	A
1		0	-2	-4	-6	-8	-10	-12
2	**G**	-2	P1	P2	P3	P1	P2	P3
3	**A**	-4	P1	P2	P3	P1	P2	P3
4	**T**	-6	P1	P2	↓	P1	P2	↓
5	**A**	-8	P1	↓	↓	P1	↓	↓
6	**C**	-10	↓	↓	↓	↓	↓	↓
7	**A**	-12	↓	↓	↓	↓	↓	↓

Região 1 **Região 2**

Regiões de reutilização de processadores

A cada célula calculada, o valor obtido é enviado ao processador responsável pela coluna à sua direita, pelo fato de ser necessário o valor das células vizinhas para o cálculo das próximas colunas. No caso do exemplo 2, quando a **Região 2** da matriz começa a ser processada, o processador P3 começa a enviar o valor de suas células calculadas para o processador P1.

9.1.4 Implementação utilizando uma NoC

Utilizando a NoC HERMES foi criado um MultiProc (sistema multiprocessado), sobre o qual foi implementado um serviço de troca de mensagens para dar suporte a comunicação entre os processadores. A arquitetura utilizada é apresentada a seguir.

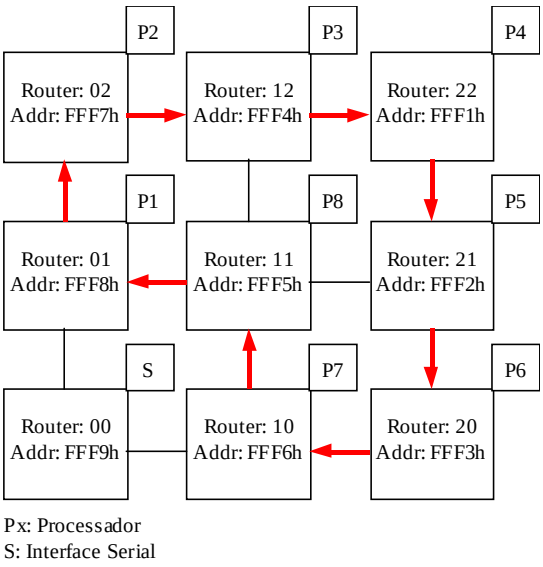


Figura 84 – Arquitetura utilizada para comunicação dos núcleos.

As setas indicam a distribuição espacial das mensagens trocadas pelos processadores para obtenção da latência média ideal das transferências de pacotes. Tal configuração é considerada ideal porque cada Processador *Pi* envia dados exatamente para seu vizinho que calcula os valores da coluna adjacente, ou seja, o número de saltos de roteamento entre processadores que calculam células adjacentes é 1. A Seção 6.4 mostra dados de desempenho com a distribuição espacial apresentada na Figura 84 em comparação com uma distribuição espacial cujos nodos que calculam colunas adjacentes possuem um maior número de saltos de roteamento entre si.

O serviço de troca de mensagens implementado possui duas operações, ambas mapeadas em memória: *Send* e *Receive*. Assim, para enviar uma mensagem para um determinado processador, basta executar a instrução *Store (St)*, utilizando o endereço do roteador no qual o processador destino está conectado. Por exemplo, o envio de um pacote para o processador P4 é realizado pelo código *assembly* mostrado na Figura 85.

```
XOR R0, R0, R0          ; Zera R0.

LDH R1,#FFh             ; Carrega R1 com o endereço do
LDL R1,#F1h             ; router 22, no qual o P4 está conectado.

LDH R2,#12h             ; Carregue em R2 a
LDL R2,#34h             ; mensagem a ser enviada.

ST R2, R1, R0           ; Envia R2 para P4.
```

Figura 85 – Descrição em linguagem *assembly* do procedimento *Send*.

A operação de *Receive* foi mapeada no endereço de memória FFF0h, de maneira que para executá-la basta fazer uma leitura do endereço *FFF0h* utilizando a instrução *Load (Ld)*. O processamento de uma operação *Receive* é realizado de acordo com o algoritmo escrito em *assemby*, mostrado na Figura 86. A operação *Receive*, ao contrário da *Send*, é bloqueante, ou seja, após executá-la, o processador fica bloqueado até que alguma mensagem seja recebida.

```
XOR R0, R0, R0        ; Zera R0.

LDH R1,#FFh           ; Carrega R1 com o endereço
LDL R1,#F0h           ; endereço de entrada.

LD R2, R1, R0         ; R2 armazena a mensagem recebida.
```

Figura 86 - Descrição em linguagem *assembly* do procedimento *Receive*.

Inicialmente todos processadores têm armazenado em suas memórias a matriz inicial, a qual contém somente as seqüências a serem comparadas e os escores iniciais. Durante a execução os processadores enviam as células calculadas e recebem mensagens correspondentes às células das colunas adjacentes às quais estão sendo calculadas.

Ao final da execução, os processadores têm armazenado a matriz parcialmente preenchida, ou seja, os valores da matriz estão distribuídos entre os processadores. Cada processador tem armazenado somente as colunas calculadas por eles e as colunas adjacentes às mesmas, as quais foram enviadas pelos processadores vizinhos.

O grau de semelhança entre as seqüências, que é o valor da última célula da matriz, encontra-se armazenado na memória do processador responsável pelo processamento da última coluna da matriz. Esse valor pode ser buscado da memória utilizando-se a interface serial ou o processador pode enviá-lo à interface serial ao final da execução.