Proposta de Tese de Doutorado

Uma Metodologia de Hardware/Software CoDesign de Protocolos de Comunicação Baseada na Otimização de Desempenho por Algoritmos Genéticos

Aluno: Marcio Nunes de Miranda

Orientadores:
Aloysio de Castro Pinto Pedroza
Antonio Carneiro de Mesquita Filho

Universidade Federal do Rio de Janeiro
Coppe/Elétrica

E-mail:{marcio, aloysio}@gta.ufrj.br mesquita@coe.ufrj.br

UFRJ

AGRADECIMENTOS

Gostaria de expressar meus agradecimentos aos Profs. Otto Carlos Muniz B. Duarte, José Ferreira de Rezende e Jorge L. de Souza Leão, do Grupo de Teleinformática e Automação, pelo apoio dado e constante troca de idéias; ao Prof. Edmundo Souza e Silva, coordenador do laboratório LAND da COPPE/UFRJ, e à toda sua equipe pela infra-estrutura cedida em termos de recursos computacionais, suporte técnico e constante troca de idéias relativas à utilização da ferramenta Tangram-II, particularmente aos alunos Sidney Cunha de Lucena, Flávio Pimentel Duarte e Ana Paula Couto Silva; ao Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da COPPE/UFRJ pelo apoio dado; aos colegas do GTA, de Iniciação Científica, Mestrado e Doutorado.

Agradecimento especial faço aos meus orientadores, Prof. Aloysio de Castro P. Pedroza e Antonio Carneiro de Mesquita Filho, pela oportunidade que me concederam de desenvolver um trabalho interessante e objetivo, pela orientação, pelas várias horas de discussão que tivemos, contribuindo para o meu aprendizado e desenvolvimento como pesquisador e pela paciência que tiveram; aos Profs. Valmir Barbosa, Felipe França e Inês Dutra e aos amigos Pablo Piras e Ariane Larentis, que sempre me apoiaram e me incentivaram nos momentos mais críticos desta jornada.

O meu muito obrigado a todos vocês. Aqueles que, por falha da minha memória, não lembrei, por favor me desculpem.

Este trabalho foi realizado com recursos da UFRJ, FUJB, CNPq, CAPES, COFECUB, FAPERJ e REENGE.

RESUMO

Neste trabalho é apresentada uma metodologia de projeto de protocolos na qual o *hardware* e o *software* são desenvolvidos concorrentemente e de forma integrada. Este processo requer que se faça uma escolha das operações do protocolo que devem ser implementadas em *hardware* e das operações que devem ser implementadas em *software*, ou seja, escolha da melhor *partição hw/sw*. Nos trabalhos existentes na bibliografia, este processo de escolha é manual, através de tentativa e erro, e depende da experiência do projetista. A metodologia proposta utiliza um algoritmo genético para otimizar o desempenho e o custo do protocolo a ser sintetizado. O desempenho é avaliado por uma ferramenta de modelagem e análise de sistemas denominada Tangram-II e o custo do *hardware* por uma ferramenta denominada Synopsys. Na modelagem inicial do protocolo são utilizadas as redes de Petri estocásticas, que podem ser mapeadas diretamente em cadeias de Markov a tempo contínuo finitas, facilitando desta maneira a avaliação do desempenho realizada pelo Tangram-II. Este trabalho propõe-se a automatizar o processo de seleção da melhor partição hw/sw, através de critérios estabelecidos a partir dos resultados da otimização de uma função de erro definida conforme os parâmetros de qualidade de serviço que se deseja para o protocolo e o custo do *hardware*.

ABSTRACT

This work presents a new methodology to the design of protocols in which the hardware and the software are developed together and integrated. This process demands a choice of the protocols operations that must be implemented in hardware and the ones that must be implemented in software, that is, the best partition to choose. In other works in bibliography this process is manual and depends on the designer experience. The proposed methodology uses a genetic algorithm to optimize the performance and the associated cost of the protocol to be synthetized. The performance is evaluated by a modeling and analysis tool called Tangram-II and the cost of hardware by a tool called Synopsys. The first model of the protocol is realized using the Stochastic Petri nets that can be mapped directly onto finite continuous time Markov chains. This facilitates the performance analysis done by Tangram-II. This work proposes to automate the selection process of the best partition hardware/software, through criterions created from the optimization results of an error function defined by the quality of service parameters required for the protocol and by the cost of the hardware desired.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A complexidade dos sistemas concorrentes que estão sendo atualmente projetados tornou muito importante o desenvolvimento de sofisticadas ferramentas e metodologias para a modelagem, o projeto e a análise de desempenho de tais sistemas. O comportamento desses sistemas geralmente é modelado por uma máquina de estados e, atualmente, devido ao recente desenvolvimento de novas tecnologias de *hardware*, modelos com milhares de estados podem ser resolvidos e uma classe bem maior de problemas pode ser solucionada com técnicas numéricas.

Os sistemas de comunicação, em particular, foram bastante beneficiados com essas novas tecnologias, pois protocolos anteriormente implementados totalmente em *software* podem ser implementados em *hardware*, a um custo baixo, aumentando a velocidade de processamento das mensagens e conseguindo atender às exigências das redes de alta velocidade e com alto tráfego.

Na maioria dos casos os modelos de desempenho são modelos estocásticos, cujo comportamento é de natureza probabilística, ou seja, o tempo de serviço em um recurso (ou o tempo entre chegadas de mensagens) é aleatório. Muitos dos métodos e ferramentas utilizados na análise do desempenho de um sistema de comunicação são baseados em diagramas de estados e impõe-se a hipótese de que o tempo de permanência do sistema em cada estado até uma transição é exponencialmente distribuído. Portanto, seria interessante que se pudesse aproveitar esta característica no projeto de protocolos de comunicação, pois além de facilitar a modelagem do protocolo, seria possível a obtenção de medidas que orientassem o projetista a tomar decisões consistentes durante a implementação do protocolo considerado.

Um fator importante para um bom desempenho dos sistemas de comunicação é a utilização integrada de *hardware* e *software* durante as diversas fases do projeto. Algumas operações do sistema, normalmente as menos críticas em termos de velocidade, podem ser implementadas em *software*, enquanto que aquelas que necessitam de uma maior velocidade devem ser implementadas em *hardware*. Devido à crescente demanda por protocolos com altas vazões, as soluções que utilizam *hardware* são cada vez mais investigadas.

Apesar dos tempos de execução em *hardware* serem, normalmente, muito menores do que na implementação em *software*, o custo deste tipo de solução é maior. Portanto, deve-se estabelecer um compromisso entre o desempenho desejado e o custo de implementação no momento de se decidir qual parte de um dado sistema deve ser implementada em *hardware* e qual parte deve ser implementada em *software*. Particularmente no caso de protocolos de comunicação, esta decisão tem impacto direto no desempenho do sistema de comunicação do qual esse protocolo faz parte. Se no desenvolvimento do projeto as partes de *hardware* e *software* são integradas numa única especificação e desenvolvidas conjuntamente, o processo é conhecido como *hardware/software codesign*. No início, o projetista não tem conhecimento de quais partes serão implementadas em *hardware* e quais partes serão implementadas em *software*, portanto a especificação inicial é construída utilizando-se diagramas de blocos. Após a formulação dos requisitos do protocolo, a especificação deve ser refinada e validada. À medida que se avança no detalhamento da especificação, torna-se necessária a aplicação de técnicas que auxiliem o projetista a tomar decisões relativas à partição *hardware/software* das operações do protocolo. Estas operações representam o comportamento do protocolo e podem ser descritas por um diagrama de estados, onde cada transição de estado representa um conjunto de cláusulas e expressões a serem avaliadas.

Deve-se considerar diversos aspectos na partição *hardware/software* do projeto de um protocolo de comunicação [1]. Inicialmente deve-se definir a *granularidade* do sistema, ou seja, as menores partes, indivisíveis, usadas durante a partição, tais como: processos, subrotinas, blocos de instruções, instruções ou operações aritméticas. Quanto maior a granularidade, menor o número de soluções possíveis, o que agiliza o processo mas, por outro lado, diminui o número de partições possíveis. Também devem ser definidos os parâmetros a serem usados para a realização da partição. Alguns parâmetros comumente utilizados incluem custo financeiro, desempenho, taxas de comunicação, área de silício do circuito integrado, confiabilidade, tamanho do programa, quantidade de dados e tamanho da memória. Este trabalho concentra-se nos parâmetros de desempenho e custo.

Os diversos parâmetros utilizados devem ser combinados numa função de custo única, denominada *função objetivo*. Esta função fornece uma medida da qualidade da partição em questão e através de sua avaliação pode-se comparar duas partições e selecionar aquela que satisfaz melhor as especificações.

Atualmente existem diversas abordagens que procuram solucionar o problema da partição *hardware/software*: algoritmos de agrupamento (*clustering*), algoritmos de melhoria incremental (*iterative-improvement*), algoritmos customizados (*custom algorithms*) e algoritmos genéticos. Alguns são mais rápidos, como os de *clustering* enquanto que outros são mais lentos porém encontram soluções melhores, como os algoritmos genéticos, que serão explorados com maiores detalhes.

A decisão de particionar as operações do protocolo entre o *hardware* e o *software* é realizada pelo projetista. Embora existam alguns esforços para automatizar este processo, o projetista ainda dispõe de poucos recursos para auxiliá-lo no refinamento da especificação, no desenvolvimento do projeto e na partição *hardware/software* do protocolo [1][2].

Em Fischer [2] é ressaltada a importância do uso combinado do *hardware* e do *software* para se alcançar um alto desempenho de sistemas distribuídos, como sistemas multimídias. Mais especificamente, é mostrado como a linguagem de especificação formal *Estelle* pode ser utilizada para facilitar o *codesign*. O sistema é inicialmente especificado em Estelle e, posteriormente, dividido em várias especificações menores, sendo as partes de *software* implementadas na linguagem C e as partes de *hardware* especificadas em VHDL, para posterior análise e desenvolvimento. É apresentado um ambiente de desenvolvimento para o suporte às etapas de projeto e implementação.

Em Hidalgo [3] a partição *hardware/software* é realizada usando um algoritmo genético. Nesse trabalho, medidas de custo e desempenho são obtidas através de estimativas e medidas realizadas em trabalho anterior por Scott [4]. Foi construída uma tabela de custos de *hardware* (assumiu-se que o custo do *software* é zero) e desempenho de *hardware* e *software*. Cada bloco da especificação tem um determinado custo de *hardware* e um determinado desempenho de *hardware* e *software*. A partir destes dados a função objetivo é calculada e o algoritmo genético é utilizado para otimização e obtenção da melhor partição *hardware/software*.

O objetivo deste trabalho é apresentar uma nova metodologia que automatize o processo de partição *hardware/software* de um protocolo de comunicação, estabelecendo um critério objetivo para que a mesma seja realizada sem necessidade do projetista utilizar técnicas manuais.

Inicialmente é especificado o desempenho desejado para o protocolo, em termos de uma curva de vazão, ou de banda passante, ou de retardo ou qualquer outra medida de interesse que possa ser obtida a partir da ferramenta de análise de desempenho aqui utilizada. O protocolo é modelado com base nas redes de Petri estocásticas [5][6]. Ao longo do processo, esta descrição é convertida pelo projetista em um novo modelo, apropriado à análise de desempenho do protocolo, desenvolvido no ambiente e na sintaxe da ferramenta de análise de desempenho Tangram-II [7][8][9]. Os valores dos parâmetros do modelo utilizado pela ferramenta Tangram-II são fornecidos por um método de otimização baseado em algoritmos genéticos (AG's) [10][11]. A medida de interesse desejada é utilizada na construção da função objetivo a ser otimizada para se encontrar as taxas de transição da rede de Petri que melhor aproximam a curva especificada. Uma vez determinadas as taxas de transição da rede de Petri que modela o protocolo, é estabelecido um critério para a partição *hardware/software* baseado no fluxo de dados que passa por cada transição. Convém ressaltar que cada transição da rede de Petri equivale a um determinado conjunto de operações que são executadas pelo protocolo.

O capítulo 2 apresenta a teoria básica da síntese de alto nível e do processo de *hardware/software codesign*, suas vantagens e desvantagens, e como eles se aplicam no projeto de protocolos de comunicação. São apresentadas as principais operações envolvidas na síntese: escalonamento e alocação. Também é mostrado como o processo de *hardware/software codesign* pode ser abordado como um problema de otimização. Finalizando o capítulo é apresentada e desenvolvida a nova proposta para a determinação da melhor partição *hardware/software*, baseada nos resultados fornecidos pela ferramenta Tangram-II e pelo algoritmo genético desenvolvido neste trabalho.

O capítulo 3 mostra como medidas de interesse normalmente utilizadas para a avaliação de desempenho de sistemas podem ser utilizadas no projeto de protocolos para auxiliar na definição da melhor partição *hardware/software*. Neste capítulo é apresentada uma descrição suscinta da sintaxe utilizada na ferramenta de análise de desempenho Tangram-II, suas características e aplicações. É descrita a forma como se constrói a função objetivo a ser utilizada na otimização e o critério de partição utilizado neste trabalho.

O capítulo 4 introduz os fundamentos básicos dos algoritmos genéticos, suas vantagens e aplicações, os diversos tipos de algoritmos genéticos existentes na bibliografia e descreve o algoritmo utilizado na metodologia proposta.

No capítulo 5 são apresentados e analisados os resultados obtidos com o programa de algoritmo genético desenvolvido. São apresentados dois protocolos para os quais se conhece curvas de desempenho e sobre os quais é aplicada a metodologia. Os valores para as taxas resultantes da aproximação destas curvas são apresentados em tabelas. São realizadas simulações e gerados gráficos, proporcionando uma maneira fácil de se visualizar a eficiência do algoritmo e a validade do critério de partição proposto.

No capítulo 6 são apresentadas as conclusões do trabalho desenvolvido, as suas vantagens e desvantagens em relação aos métodos conhecidos e as sugestões para a sua continuação, assim como possíveis aplicações em outras áreas.

Capítulo 2

HW/SW Codesign de Protocolos de Comunicação

2.1 Introdução

Tradicionalmente, os protocolos têm sido implementados inteiramente em *software*. Entretanto, para que os requisitos de velocidade das redes atuais sejam atendidos, pode ser necessário implementar parte das especificações de um protocolo em *hardware*, o que implica em promover a partição *hardware/software* de uma determinada especificação [12][13]. Desta forma é necessário realizar inicialmente uma partição funcional da especificação para a definição dos componentes do sistema, ou seja, processos, subrotinas, variáveis e canais de comunicação são alocados a um ou mais processadores padrão, *hardware*s específicos (ASIC's e FPGA's, por exemplo), memórias e barramentos. Pode-se reduzir o tempo de execução de um protocolo totalmente implementado em *software* utilizando-se componentes de *hardware* específicos para determinadas funções. Equivalentemente, pode-se reduzir o custo de um sistema totalmente implementado em *hardware*, movendo-se algumas de suas funções, implementadas em *hardware* específico, para *software*, ou seja, processadores padrão [14].

Assim, durante o projeto de um protocolo de comunicação o projetista define, a partir de um modelo inicial, as tarefas do protocolo que serão implementadas em *hardware* e as que serão implementadas em *software*, procurando uma solução de compromisso entre o seu desempenho e o seu custo, pois apesar dos tempos de execução em *hardware* serem, normalmente, muito menores que na implementação em *software*, o custo deste tipo de solução é maior. Medidas típicas consideradas na síntese do protocolo incluem: os tempos de execução em *hardware* e *software*, o tempo de comunicação entre os módulos, tamanho do código objeto do *software* (incluindo as rotinas de comunicação) e área do circuito integrado.

2.2 Síntese de Alto Nível

A síntese de alto nível pode ser definida como um processo de mapeamento de uma descrição comportamental de um sistema em uma descrição estrutural, análogo ao processo de compilação de linguagens como C ou Pascal em linguagem *assembly*. Para ilustrar este processo utiliza-se o diagrama em **Y** da figura 2.1 [15]. Cada eixo do diagrama representa três diferentes domínios (comportamental, estrutural e físico), enquanto que ao longo dos eixos existem diferentes níveis de abstração dentro cada domínio. O nível de abstração é tanto maior quanto mais se afasta do centro do diagrama. Este diagrama pode ser estendido desenhando-se círculos concêntricos. Cada círculo cruza o eixo **Y** num nível de representação particular dentro de cada domínio.

O domínio comportamental representa **o que** o sistema faz. Neste domínio o projeto é tratado como uma "caixa preta" com um conjunto de entradas, um conjunto de saídas e um conjunto de funções que descrevem o comportamento do sistema ao longo do tempo. No domínio estrutural a funcionalidade prevista no domínio comportamental é descrita por um conjunto de componentes e conexões sob restrições como custo, área e velocidade. Cada componente do domínio estrutural é definido por uma arquitetura que algumas vezes pode ser utilizada como descrição funcional, nos níveis de abstração mais baixos do domínio comportamental. Dependendo do nível de abstração, pode-se ter, no domínio estrutural: transistores, barramentos, flip-flops, ALUs, RAMs, etc. O domínio físico representa células, módulos, circuitos impressos.

Em um mesmo domínio, a mudança de um nível de abstração mais alto para um

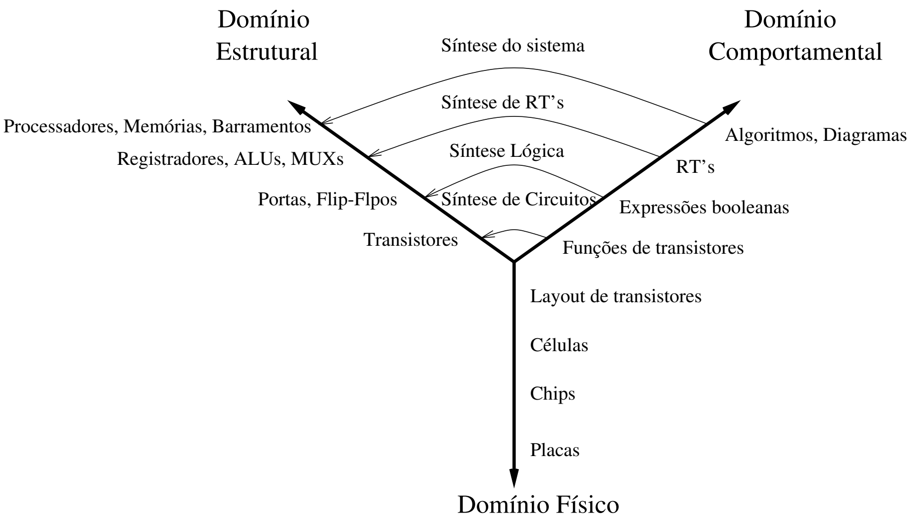

Figura 2.1: Diagrama Y

mais baixo aumenta o grau de detalhamento do projeto, mantendo a consistência da especificação, isto é, as restrições impostas nos níveis superiores de descrição são mantidas nos níveis mais baixos.

Na síntese de alto nível utiliza-se formalismos baseados em diversas áreas, desde o paradigma de linguagem de programação, para a descrição comportamental, até modelos de *layout* para estimativa de custo e desempenho. Frequentemente utiliza-se como método de descrição formal para a síntese de alto nível as máquinas de estado finitas estendidas. Esta descrição é especialmente útil no projeto de protocolos de comunicação, como será visto mais adiante.

A partição *hardware*/*software* é apenas parte do problema de síntese de um sistema de comunicação. A partição produz um conjunto de subsistemas e primitivas de comunicação, onde cada subsistema se comunica com os outros subsistemas [16]. O ponto de partida da síntese de sistemas de comunicação é um modelo composto por um conjunto de processadores dentro de uma rede de comunicação composta por canais. O objetivo é transformar um sistema composto de processadores que se comunicam através de canais e primitivas de alto nível em um conjunto de processadores se comunicando através de barramentos e dividindo o controle da comunicação.

2.2.1 Escalonamento e Alocação

O modelo comportamental especifica as operações a serem executadas pelo *hardware*. Normalmente essas operações são representadas internamente através de um grafo com fluxos de controle e de dados (CDFG), que captura todas as dependências de controle e dados do modelo comportamental. Os algoritmos de escalonamento particionam este grafo em subgrafos, compostos por um subconjunto de operações, onde cada subgrafo é executado num passo de controle. Cada passo de controle pode ser comparado a um estado, num diagrama de uma máquina de estado finita.

Cada operação a ser executada dentro de um passo requer uma unidade funcional. Portanto, o número total de unidades funcionais necessárias para a execução de um passo de controle corresponde ao número total de operações a serem executadas naquele passo. Os algoritmos de escalonamento otimizam o número de passos de controle e o número de operações a serem executadas em cada passo, além de definir qual o próximo conjunto de operações será executado ao final de um determinado passo. Esta otimização tem impacto direto no compromisso entre o custo do projeto e o seu desempenho.

Os algoritmos de alocação consistem na determinação do número e do tipo de unidades funcionais a serem utilizados no projeto, além de efetuar um mapeamento das variáveis e operações escalonadas no CDFG em unidades de armazenamento e interconexão, de maneira a assegurar que o sistema se comportará corretamente com o conjunto de componentes selecionados.

Os algoritmos de escalonamento e alocação estão relacionados. É difícil avaliar a qualidade de um determinado algoritmo de escalonamento sem considerar o algoritmo de alocação que está sendo utilizado. Dois escalonamentos diferentes com o mesmo número de unidades funcionais e passos de controle podem resultar em projetos com desempenhos completamente diferentes após o algoritmo de alocação ser executado.

2.2.2 Aplicação da Síntese de alto nível ao Projeto de Protocolos

Um protocolo de comunicação pode ser caracterizado por uma máquina de estado finita dominada por fluxo de controle. Seu projeto, normalmente, é uma tarefa difícil que tende a gerar muitos erros. Quanto mais complexo o protocolo maior a possibilidade de erros e, por este motivo, a aplicação de métodos formais é essencial. Neste sentido, a síntese de alto nível auxilia na determinação das unidades funcionais, escalonamento de tarefas e alocação do *hardware/software* para a execução das funções do protocolo. Uma das técnicas mais utilizadas para a especificação de protocolos é a modelagem em redes de Petri, pela facilidade de representar a interação entre as diversas partes do protocolo através de *transições*[17][18].

2.3 HW/SW Codesign

2.3.1 Conceitos Básicos

O *HW/SW Codesign* é uma metodologia de projeto concorrente e cooperativa na qual o desenvolvimento do *hardware* e do *software* de um sistema é realizado de forma integrada, explorando tanto as vantagens de uma implementação em *hardware* quanto as vantagens de uma implementação em *software*. Esta metodologia tem por objetivo encontrar uma solução ótima da parte do sistema que deve ser implementada em *hardware* e da parte que deve ser implementada em *software*, a partir de uma especificação funcional. Este processo de partição *hardware/software* se estende pelos diversos níveis do projeto e tem como objetivo encontrar um compromisso entre o desempenho e o custo do sistema. Esta técnica possibilita implementações mais eficientes, melhorando o desempenho, a confiabilidade e o custo-benefício do sistema. Como decisões de implementação em *software* têm influência no projeto de *hardware* (e vice-versa), os problemas decorrentes destas decisões podem ser detectados logo no início do projeto e as respectivas alterações podem ser realizadas imediatamente, ainda num alto nível de abstração [19]. Deve existir um processo contínuo de refinamento do sistema para que seja possível realizar uma validação em cada um dos níveis do projeto.

Quanto maior a complexidade dos sistemas maior a necessidade de uma descrição detalhada. Por isto é extremamente importante o emprego de técnicas de decomposição para facilitar a análise do sistema. A metodologia *HW/SW Codesign* baseia-se fundamentalmente em três pontos: integração dos projetos de *hardware* e *software*, exploração de um compromisso custo versus desempenho entre *software* e *hardware*, através da avaliação das diversas alternativas possíveis de partição HW/SW, o contínuo refinamento da especificação e a constante validação do projeto. A figura 2.2 ilustra este processo. A metodologia de *HW/SW codesign* pode ser vista como uma extensão da síntese de alto nível (HLS), como é ilustrado na figura 2.3. Ou seja, a síntese é uma parte do processo de concepção do sistema. As funções do sistema a serem implementadas são descritas como processos concorrentes que interagem entre si e, na fase da partição, podem ser mapeadas em diferentes componentes físicos.

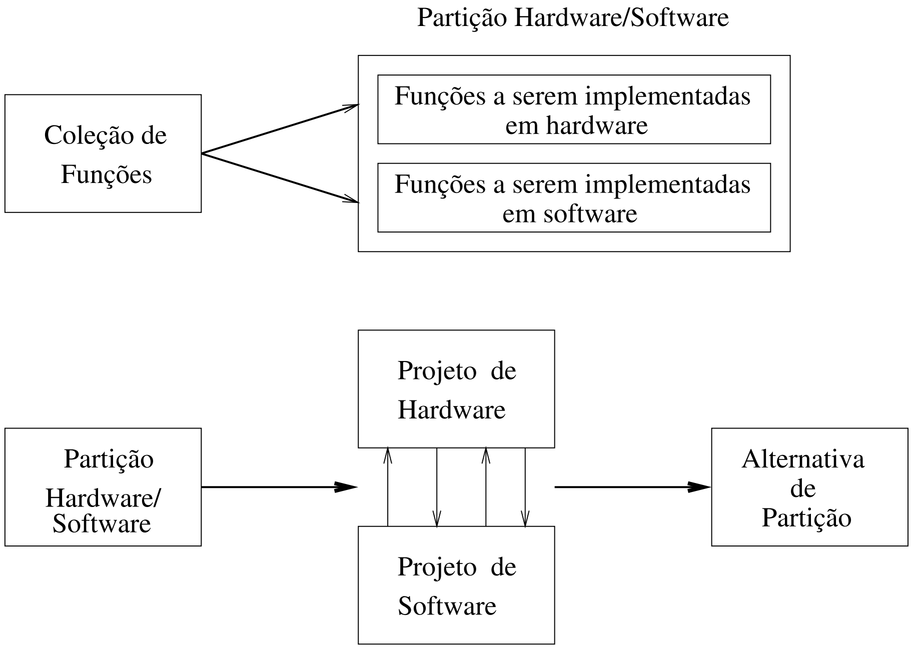

Figura 2.2: Partição HW/SW e Refinamento

Um dos critérios considerados na partição *hardware/software* é a minimização da comunicação entre as funções do sistema. No caso específico de protocolos de comunicação, este é um fator importante para melhorar o desempenho, principalmente em termos de vazão e retardo. O processo de *codesign* consiste em realizar, iterativamente, os seguin-

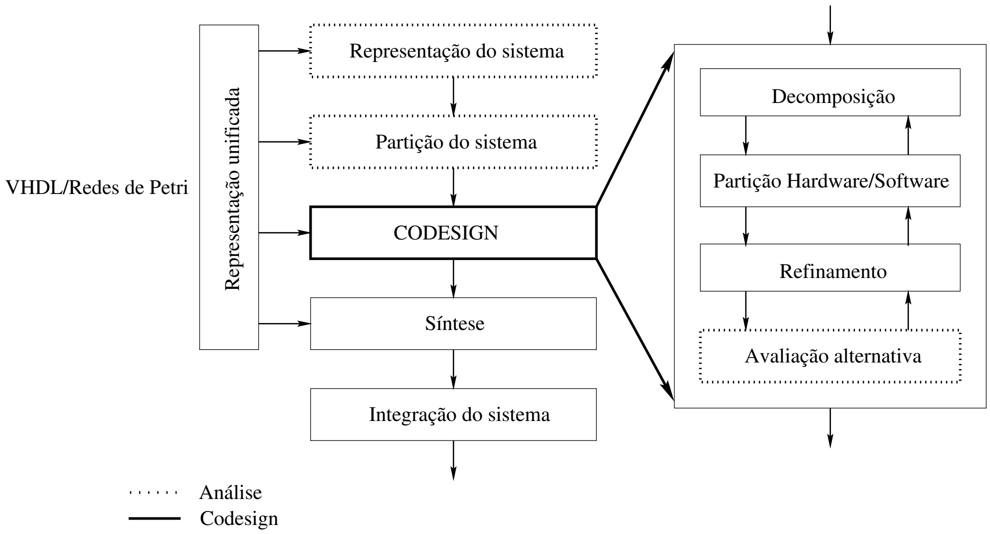

Figura 2.3: Codesign no contexto do projeto do sistema

tes procedimentos: decomposição funcional do sistema, a partição *hardware/software*, o refinamento e a avaliação das alternativas possíveis de partição. Na decomposição, divide-se as funções do sistema em subfunções. A partição determina quais subfunções devem ser implementadas em *hardware* e quais devem ser implementadas em *software*. O refinamento produz novas alternativas de partição *hardware/software*. Na avaliação das alternativas de partição determina-se o desempenho do sistema e qual o custo associado, com o objetivo de selecionar uma determinada implementação *hardware/software*.

No processo de síntese do *software* define-se quais os processos concorrentes e realiza-se um escalonamento, ou seja, define-se quais processadores executam cada um dos processos. Deve-se considerar aspectos como minimização do tempo que o processador espera por eventos externos, restrições de tempo de execução para cada processo, sequência de execução de cada processo e garantia de que todos poderão ser executados pelo menos uma vez. A síntese do *software* consiste em converter uma especificação complexa no nível de sistema em um programa tradicional numa linguagem de alto nível como, por exemplo, a linguagem C. A síntese de *hardware*, descrita detalhadamente na seção anterior, normalmente é realizada a partir de uma linguagem de descrição de *hardware* tal como VHDL [20] e incorpora, além da síntese de alto nível, a síntese de máquinas de estado finitas, síntese da lógica e definição da tecnologia a ser utilizada [1].

Algumas vantagens da utilização do princípio de *codesign* são:

- Manter juntas as partes referentes à especificação do sistema durante todas as etapas do projeto;

- Detecção de erros nas fases iniciais do projeto;

- Possibilitar a verificação e a simulação de todo o sistema em qualquer etapa do projeto;

- O processo iterativo permite um refinamento e uma reavalição da partição em cada etapa do projeto;

- Diminuição nos custos de projeto e teste de sistemas complexos;

2.3.2 HW/SW Codesign de Protocolos

A utilização da metodologia de *codesign* no projeto de protocolos de comunicação tem assumido maior importância devido à crescente demanda por redes com alto desempenho, que suportem um alto tráfego como, por exemplo, aplicações multimídia. As implementações de parte do protocolo em *hardware* têm sido cada vez mais estudadas devido à constante evolução da tecnologia de *hardware* a um baixo custo. Na partição da especificação de um dado protocolo, a parte mais sensível à variação do desempenho deve ser endereçada a um ambiente de implementação em *hardware*, enquanto a parte menos crítica deve ser endereçada a um ambiente de implementação em *software*. O uso de técnicas de descrição formais é importante na fase de especificação do protocolo e as linguagens de descrição de *hardware* tais como o VHDL também podem ser utilizadas nesta fase. Como exemplo de linguagem de descrição de *software* podemos citar a linguagem Estelle.

Neste trabalho são utilizadas as redes de Petri estocásticas para modelar os protocolos, devido à facilidade de visualização e validação da especificação, assim como para a obtenção de medidas de desempenho em ferramentas apropriadas. Cada transição da rede de Petri representa um conjunto de funções a serem executadas pelo protocolo, ou seja, funções a serem executadas por um componente de *hardware* ou por um processador padrão, em *software*, após a realização da partição. Cada *marca* da rede de Petri [17] representa um estado possível do protocolo. Uma *marca* de uma rede de Petri representa uma possível distribuição de *fichas* [17] cada vez que uma transição é disparada. Nos exemplos apresentados no capítulo 5 as redes de Petri são binárias, isto é, cada *lugar* [17] da rede de Petri possui uma ou zero fichas. Isto simplifica a análise do modelo e diminui o número de estados possíveis da cadeia de Markov associada.

2.3.3 A Partição HW/SW como um Problema de Otimização

Um dos maiores desafios dos critérios de partição *hardware/software* existentes é o aumento das alternativas possíveis a serem analisadas. Por este motivo o problema de partição pode ser classificado como um problema de otimização, onde se combina os diversos parâmetros de desempenho e custo do problema numa função única, denominada *função objetivo*. Alguns autores classificam este problema de escolha da melhor partição como NP-completo enquanto que Alomary [21] o classifica como NP-hard. Ambos os casos significam que os algoritmos possuem uma alta complexidade. Muitas metodologias de partição têm sido propostas: Gupta e Micheli [22][23] propõem uma metodologia que desloca paulatinamente as funções do sistema do *hardware* para o *software*, até que se alcance o limite mínimo do desempenho desejado, com o menor custo possível. Ernst [24] realiza o processo inverso, isto é, move as funções mais críticas do *software* para o *hardware* até que o desempenho desejado seja alcançado. Woo [25] divide as funções em grupos de *hardware* e *software* e analisa posteriormente o desempenho da partição, de forma manual.

Verifica-se em todos os trabalhos acima um processo iterativo de tentativa e erro, onde inicialmente faz-se a partição e posteriormente analisa-se o desempenho da mesma. Alomary [21] propõe um método de otimização combinatória, utilizando um algoritmo baseado no método *branch-and-bound* e uma tabela que fornece valores de custo para os componentes de *hardware*, de forma a otimizar o problema da partição.

2.4 Uma Metodologia para automatização da partição HW/SW

Nas metodologias de partição propostas acima não existem métodos formais que automatizem o processo de *codesign*. A melhor partição é escolhida manualmente e depois de serem investigadas muitas alternativas, tornando o processo demorado e custoso. Todas elas dependem muito da experiência do projetista. Mesmo onde já existe uma proposta de automatização [21], o desempenho só é determinado após a escolha da partição.

Em Hidalgo [3] é proposta uma metodologia para a partição *hardware/software* que também utiliza algoritmos genéticos, assim como na metodologia desenvolvida aqui, mas a função objetivo utiliza uma tabela previamente estabelecida para os valores dos desempenhos de *hardware* e de *software*, não calculando os parâmetros de desempenho durante o processo, como é realizado neste trabalho.

A metodologia apresentada neste trabalho, congrega uma série de técnicas em um único processo e propõe uma solução para o problema da automatização da partição *hardware/software*, a saber: redes de Petri [17][18], para a modelagem inicial do protocolo, uma ferramenta de análise de desempenho para uma segunda modelagem e avaliação do desempenho para um determinado conjunto de parâmetros (estabelecidos na primeira modelagem), e um algoritmo genético para a otimização de uma função objetivo que leva em conta o desempenho desejado para o protocolo e o custo associado. Cada partição é definida em função dos valores encontrados pelo algoritmo genético para um conjunto de parâmetros de projeto. A seguir, a metodologia proposta é descrita de forma suscinta e é apresentado um diagrama de blocos da mesma. Nos capítulos que se seguem, cada bloco da metodologia é detalhado.

Uma das técnicas bastante utilizadas para a especificação e projeto de protocolos de comunicação é a modelagem em redes de Petri [17][18]. Na metodologia proposta, os protocolos são modelados por redes de Petri estocásticas (Stochastic Petri Nets - SPN's), que são uma extensão do modelo de rede de Petri tradicional. Nas SPN's o tempo de disparo de uma transição, após a mesma ser habilitada, possui distribuição exponencial com taxa λ_i, onde λ_i é a taxa associada à transição i do modelo. Em Molloy [26] é mostrado que as SPN's são equivalentes a uma cadeia de Markov a parâmetro contínuo finita (CMTC) [27], onde cada *marca* da rede de Petri corresponde a um estado da cadeia e as taxas de disparo da SPN correspondem às taxas de transição entre os estados da cadeia. Por este motivo um protocolo modelado em SPN's [5][6] é facilmente convertido pelo projetista em um novo modelo, apropriado para análise de desempenho. Esse *isomorfismo* com as CMTC's determinou a opção pelas SPN's para a modelagem inicial do protocolo. O novo modelo é descrito na linguagem da ferramenta de análise de desempenho Tangram-II [7][8][9]. Essa ferramenta auxilia o projetista na identificação das operações críticas em termos de tempo (retardo) e outros parâmetros de desempenho tais como vazão e probabilidade de perda de mensagens. Com base nessas medidas de desempenho e em medidas de custo fornecidas por uma ferramenta de síntese de *hardware* denominada Synopsys, o melhor conjunto de parâmetros é determinado pelo algoritmo genético. Os resultados fornecidos orientam a escolha do critério de partição.

O problema consiste em determinar as taxas de transição da SPN que modela o protocolo obedecendo certas restrições de qualidade de serviço (*Quality of service* - **Qos**) impostas pela especificação. Estas taxas são a base para o estabelecimento de um critério de partição.

O modelo de SPN convertido para a linguagem do Tangram-II é um modelo com parâmetros que representam as taxas de disparo de cada transição da rede de Petri. Cada transição da rede de Petri é associada a um *evento* e cada *lugar* representa uma *variável de estado* no modelo do Tangram-II. Os eventos possuem distribuição exponencial com uma taxa a ser determinada pelo algoritmo desenvolvido para a metodologia, que é descrito com detalhes na seção 4.5. Nos exemplos apresentados no capítulo 5 apenas um objeto é utilizado para representar toda a rede de Petri. No capítulo 3 serão fornecidos maiores detalhes sobre a sintaxe da ferramenta Tangram-II. Uma vez construído o modelo do Tangram-II, a cadeia de Markov associada já pode ser gerada, porém ela só pode ser solucionada quando os parâmetros forem substituídos por valores numéricos.

Os valores numéricos dos parâmetros do modelo da ferramenta Tangram-II são fornecidos por um método de otimização baseado em algoritmos genéticos (AG's) [10][11] que gera conjuntos de valores para os parâmetros de forma aleatória. A substituição dos parâmetros pelos valores numéricos gerados pelo AG é realizada automaticamente por um programa denominado *trans_param*, desenvolvido por Figueiredo [8], no qual foram realizadas algumas alterações para que o programa pudesse ler os valores numéricos de um arquivo. Após a substituição dos parâmetros pelos valores numéricos, o Tangram-II resolve a cadeia de Markov resultante do modelo especificado e calcula as probabilidades em estado estacionário de cada um dos seus estados. Estas probabilidades são utilizadas para o cálculo de medidas de interesse relacionadas com o desempenho do protocolo.

Cada protocolo a ser particionado tem uma especificação a ser atendida. Esta especificação normalmente é dada por uma curva, relativa a uma determinada medida de interesse, que deve ser aproximada pelo AG. Isto determina a parte da expressão da função objetivo relativa ao desempenho. A outra parte da função objetivo, ou seja, o custo, será fornecida através da ferramenta de síntese de *hardware* Synopsys. O principal parâmetro de custo a ser utilizado será a área de silício do componente sintetizado, obtida a partir das primeiras taxas calculadas pelo AG. O cálculo do valor da função objetivo é realizado para todos os conjuntos de taxas gerados pelo AG. Cada conjunto é denominado um *indivíduo*, no AG. Inicialmente o algoritmo genético fornece indivíduos de uma *população* gerada aleatoriamente, cujo tamanho é determinado pelo projetista. O projetista também especifica o número de pontos da curva que ele deseja aproximar. O valor da função objetivo, que é a soma dos erros em relação a cada ponto especificado, fornece uma medida da *aptidão* do indivíduo para resolver o problema.

Nesta etapa inicia-se o processo de otimização, calculando-se a aptidão de cada indivíduo da população e verificando-se se algum destes indivíduos representa a solução do problema. Caso a solução não tenha sido encontrada, realiza-se uma seleção dos indivíduos mais aptos e gera-se uma nova população. Realimenta-se o Tangram-II com a nova *geração* de indivíduos, repetindo-se o processo até que se encontre um conjunto de taxas que atenda ao erro máximo especificado pelo projetista. Na figura 2.4 é apresentado o diagrama de blocos da metodologia.

A partir das taxas de transição da rede de Petri calculadas, verifica-se quais as transições do modelo que mais influenciam os parâmetros de Qos estabelecidos na especificação e realiza-se a partição *hardware/software* do protocolo. Até o momento, o critério que se mostrou mais eficiente, como será mostrado nos exemplos do capítulo 5 é o produto $\lambda_i * \pi$, onde λ_i é a taxa de disparo da transição i, determinada pelo AG, e π é a soma das probabilidades do sistema se encontrar em estados onde a transição i está habilitada. Este produto pode ser considerado como a "vazão" de cada transição. Dependendo da transição analisada, pode ser a própria vazão do protocolo, como será

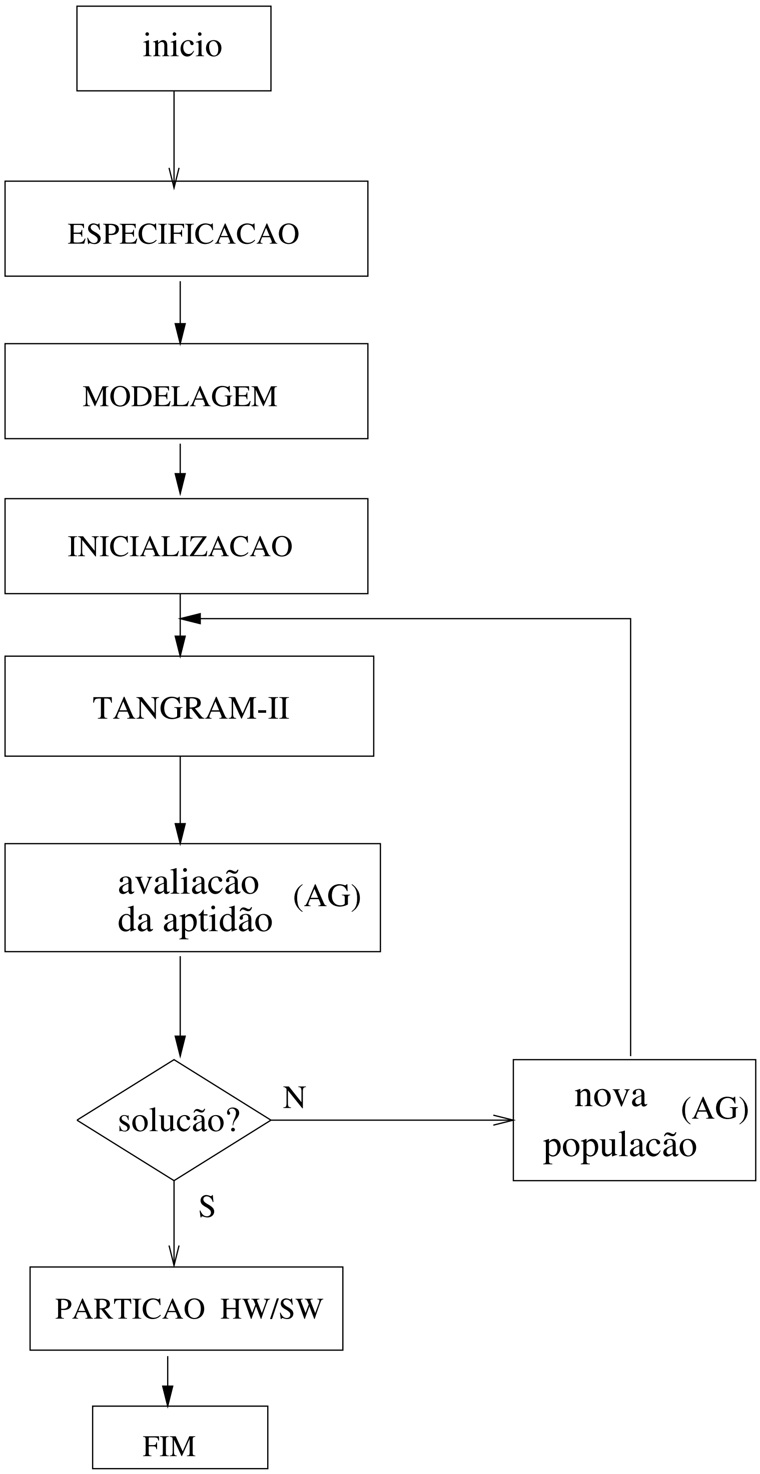

Figura 2.4: Diagrama de blocos da metodologia proposta

visto no exemplo da seção 5.2.1. As tarefas do protocolo relacionadas com as transições de maior "vazão" devem ser implementadas em *hardware*, para que o protocolo atinja um alto desempenho, enquanto que as demais devem ser implementadas em *software*, para que as restrições de custo sejam alcançadas. O critério de partição, que é uma das contribuições importantes deste trabalho, é discutido na seção 3.5.

O resultado da primeira partição, baseado somente no desempenho, é repassado à ferramenta Synopsys, que retorna um valor de custo que é agregado à função objetivo para orientar o processo de seleção do algoritmo genético de forma consistente. A partir dos resultados fornecidos pela metodologia apresentada, o projetista pode se basear em critérios objetivos para realizar a partição *hardware/software* e não apenas em sua experiência.

2.5 Comentários

Neste capítulo foram apresentados os conceitos básicos da síntese de protocolos de comunicação e da metodologia de projeto denominada *hardware/software codesign*. Foi ressaltada a importância de uma correta escolha de uma partição *hardware/software* e, particularmente, sua influência no desempenho dos protocolos. Também foi mostrado como o processo de *HW/SW codesign* pode ser formulado como um problema de otimização. A partir desta formulação foi apresentada uma proposta de uma metodologia de projeto de protocolos para automatizar o processo de partição HW/SW. Ou seja, até a seção 2.3.3 foi desenvolvida a motivação para este trabalho enquanto que na seção 2.4 foi apresentada a proposta de solução em forma de diagrama de blocos. A partir do capítulo 3 é apresentado detalhadamente como as técnicas de análise de desempenho e de otimização utilizando algoritmos genéticos se aplicam à partição HW/SW do protocolo e é realizado um detalhamento da metodologia proposta.

Capítulo 3

Partição HW/SW de Protocolos Baseada na Avaliação de Desempenho

3.1 Introdução

A decisão da melhor partição HW/SW a ser utilizada no projeto de um protocolo de comunicação tem sido realizada pelo próprio projetista, sem a utilização de técnicas objetivas. A maioria dos algoritmos de partição [28] se baseia na premissa de que o projetista deseja implementar um protocolo com o menor número de componentes de HW dedicado possível, de forma a atender um determinado desempenho. Porém o custo de HW específicos tem baixado continuamente. Neste contexto, a maior necessidade do projetista passa a ser dispor de uma metodologia que considere medidas de desempenho do protocolo a ser desenvolvido, de maneira a atender aos requisitos de Qos que as redes atuais exigem.

A seção 3.2 apresenta alguns conceitos básicos sobre a utilização de medidas de desempenho no projeto de um protocolo. Em seguida descreve-se a ferramenta de análise de desempenho utilizada neste trabalho, suas características principais e sintaxe. Na seção 3.4 é apresentada a função utilizada no processo de otimização dos parâmetros do protocolo. A seção 3.5 apresenta o critério de partição utilizado neste trabalho e a seção 3.6 as vantagens do critério utilizado.

3.2 Avaliação de Desempenho de Protocolos

O desempenho de uma rede está fortemente relacionado com o protocolo utilizado e com o tempo de processamento, em cada nó, dos pacotes que transitam pela rede. Esta situação se torna mais crítica em redes de alta velocidade [29]. Portanto, no projeto de um protocolo, devem ser considerados alguns aspectos que influenciam diretamente o seu desempenho e, consequentemente, o desempenho da rede como um todo. Entre alguns fatores importantes, pode-se citar o *timeout*. Quando um pacote é enviado, um temporizador é disparado para que o pacote seja reenviado caso uma confirmação de recebimento (*ACK*) não seja recebida dentro de determinado tempo. Se o *timeout* for ajustado com um valor muito pequeno, serão realizadas várias retransmissões desnecessárias enquanto que se for ajustado com um valor muito alto ocorrem atrasos muito grandes até a detecção da perda e retransmissão do pacote. A ocorrência de *timeouts* normalmente degrada o desempenho da rede em relação à vazão, portanto o protocolo deve ser projetado de forma a evitá-los.

Outros aspectos importantes relacionados ao desempenho de um protocolo incluem o número de pacotes perdidos e como o protocolo se comporta quando a perda é detectada, pois isto determina o número de retransmissões que o emissor terá que realizar. Algumas vezes apenas ocorre uma inversão na ordem de entrega destes pacotes no receptor, não sendo necessário a retransmissão do pacote. Alguns protocolos conseguem distinguir entre estas duas situações [30][31][32].

Uma medida de desempenho importante na especificação de um protocolo é a probabilidade de perda de pacotes. No entanto, esta medida está mais relacionada com a rede em si e com o tamanho dos buffers em cada nó do que com o protocolo usado.

O item que mais influencia o retardo de um protocolo é o *software*, por isso é importante que a CPU utilizada seja veloz. No entanto, como o tratamento de cada pacote gasta um tempo adicional relativo à interrupção gerada na CPU, é importante que se reduza ao máximo o número de transmissões, concentrando em cada uma a maior quantidade de informação possível, diminuindo-se, desta forma, o número de interrupções e, consequentemente, o retardo do protocolo. Pelo mesmo motivo deve-se reduzir ao máximo o número de nós intermediários de uma rede.

Certos protocolos se baseiam no controle do tamanho da janela do emissor e do receptor para determinar a quantidade de pacotes colocados na rede, influenciando diretamente a sua vazão. O RTT (*ROUND TRIP TIME*), ou seja, o tempo decorrido entre o emissor enviar um pacote e receber o seu ACK, é um fator importante quando associado à largura de faixa (BW) disponível. O produto $BW * RTT$ fornece uma medida da quantidade de tráfego (em bits) que deve transitar na rede para que a eficiência máxima seja atingida.

Como já foi mencionado, as redes de alta velocidade são aquelas nas quais o desempenho de um protocolo é mais determinante. Existem muitos modelos utilizados para a análise do desempenho dessas **redes**, markovianos e não-markovianos, entre os quais pode-se citar: modelo de fluidos, MMPP (*Markov-modulated Poisson Process*) [33], processos duplamente estocásticos [33], modelos auto-similares [34][35], entre outros. Na metodologia utilizada neste trabalho utilizou-se um modelo markoviano, uma rede de Petri estocástica, para modelar o **protocolo** a ser particionado.

Este trabalho relaciona as medidas de desempenho descritas acima com um critério de partição HW/SW de um protocolo de tal forma que, ao final da implementação do protocolo, o desempenho da rede atenda a uma determinada especificação em relação às medidas de Qos desejadas para a rede e ao custo de implementação desejado. Esta relação é estabelecida por uma função de erro, a ser descrita na seção 3.4, na qual está embutida a especificação desejada. Esta função é minimizada para a determinação dos parâmetros do protocolo (no caso, as taxas de disparo de cada transição da rede de Petri estocástica). Esses parâmetros permitem o estabelecimento de um critério objetivo para a partição HW/SW, a ser apresentado na seção 3.5. Na próxima seção é realizada uma breve descrição da ferramenta de análise de desempenho utilizada neste trabalho para o cálculo das medidas de interesse relacionadas com um determinado conjunto de parâmetros do protocolo.

3.3 A Ferramenta Tangram-II

3.3.1 Características

O Tangram-II é uma ferramenta utilizada para especificação e análise de desempenho de sistemas, principalmente sistemas de comunicação e computacionais. Foi desenvolvido com objetivos educacionais e de pesquisa no laboratório Land da Coppe/Ufrj. Entre suas características principais estão: possuir uma interface gráfica baseada no paradigma de orientação a objetos e uma variedade de métodos de solução para obtenção de medidas de interesse relacionadas ao modelo.

O sistema a ser modelado é representado por uma coleção de objetos que interagem, enviando e recebendo mensagens. O estado de um objeto é representado por um conjunto de variáveis e o seu comportamento é definido por *eventos* e *mensagens*, assim como pelas *condições* que habilitam os eventos e as *ações* executadas quando um evento é disparado ou uma mensagem é recebida.

Os eventos são gerados espontâneamente por um objeto. Caso as condições associadas àquele evento sejam satisfeitas, a ação relacionada a ele é efetuada. Condições são expressões booleanas avaliadas a partir do estado corrente do objeto e usadas para indicar quando um evento será habilitado ou não. Uma condição deve vir entre parênteses e pode ser composta por variáveis, valores numéricos e operadores aritméticos e lógicos [9].

Ações são usadas para modificar o estado do objeto e para enviar mensagens a outros objetos. Uma ação é executada quando um evento é disparado ou uma mensagem é recebida. Ações são descritas em código C-like e podem usar variáveis e constantes declaradas no objeto. Uma ação também pode conter variáveis locais, declaradas dentro do escopo da ação. As variáveis de estado não podem ter o seu valor alterado durante uma ação. Somente ao final da ação é usada a função set_st(<var_name>, <value>)

para alterar o valor da variável de estado. Várias ações podem ser associadas a um evento ou a uma mensagem, cada uma com uma probabilidade diferente de ocorrência (soma = 1). As probabilidades são especificadas usando-se a palavra-chave PROB seguida de uma expressão aritmética.

Exemplo de múltiplas ações:

```
action= {

/* chegou um pacote correto */

int q;

q = q + 1;

set_st("queue",q);

} : prob = p;

/* chegou um pacote incorreto , não faz nada */

{

;

} : prob = 1 - p;
```

Mensagens são usadas para representar a interação entre os objetos e são enviadas num tempo igual a zero.

Quando um evento ocorre ou uma mensagem é recebida, o conjunto de ações especificadas pelo usuário é executado com uma dada distribuição de probabilidade. Como resultado da execução de uma ação, o estado do objeto pode mudar e mensagens podem ser enviadas para outros objetos do modelo.

O Tangram-II possui diversos métodos de solução da cadeia de Markov gerada a partir da especificação do modelo, tanto para estado estacionário quanto para estado transiente. Existem também métodos de solução para cadeias com eventos determinísticos (modelos não-Markovianos). O Tangram-II também possui um simulador interativo que possibilita a geração de *traces* de variáveis que o projetista deseje monitorar. Esta possibilidade é extremamente útil quando a inclusão de uma variável de estado no modelo provoca a explosão do espaço de estados, tornando a cadeia de Markov associada de difícil solução analítica. Pode-se também realizar a simulação de eventos raros e animação.

Os objetos são especificados utilizando-se uma representação gráfica. A interface gráfica (Tangram Graphical Interface - TGIF) [36] mostra um *template* que auxilia o usuário na construção de cada objeto do modelo. Após a construção do modelo, gera-se a cadeia de Markov associada. O usuário, então, seleciona o método de solução a ser adotado para a solução da cadeia de Markov e obtenção das probabilidades em estado estacionário que serão utilizadas no cálculo das medidas de desempenho.

3.3.2 Construção de Modelos

Define-se um objeto no Tangram-II através da especificação de seis atributos:

- "Declaration attribute"

  Neste atributo são declaradas as variáveis, constantes e os parâmetros utilizados no objeto. A declaração é realizada em três subseções diferentes, uma para cada tipo de declaração (variáveis, constantes e parâmetros).

- "Initialization attribute"

  Neste atributo as constantes são definidas e as variáveis de estado recebem seus valores iniciais. Os valores dos parâmetros são fornecidos posteriormente, quando da solução analítica da cadeia de Markov associada ao modelo.

- "The State variables attribute"

Este atributo é utilizado pelo simulador. As variáveis de estado do modelo devem ser colocadas nesta seção, caso o simulador esteja sendo utilizado.

- "The Events attribute"

  Neste atributo são declarados todos os eventos. Um evento é definido por um nome, uma distribuição de probabilidades e seus parâmetros. Um evento deve ter uma condição e pelo menos uma ação associada a esta condição.

- "The Messages attribute"

  Neste atributo são especificadas todas as mensagens que são recebidas por um objeto. Quando uma mensagem é recebida, é realizada uma ação associada a ela. A recepção de uma mensagem é definida pela constante denominada "Message_port", que indica a porta a ser usada na recepção da mensagem.

- "The Rewards attribute"

  Neste atributo são declaradas as *recompensas* de taxa, associadas com um determinado estado do sistema e as recompensas de impulso, associadas a uma determinada transição de estado. Recompensas são acumuladas ao longo de uma simulação e servem para avaliar medidas de difícil obtenção através das probabilidades de estado estacionário. São utilizadas quando não se deseja criar uma variável de estado, para que o tamanho da cadeia de Markov não cresça indevidamente, criando problemas para a sua solução analítica.

Na figura 3.1 é apresentado um exemplo da sintaxe usada no Tangram-II para a construção de um modelo de fila M/M/1/k, ou seja, um modelo cujo tempo entre chegadas de pacotes possui distribuição exponencial, o tempo de serviço de cada pacote possui distribuição exponencial, existe apenas um servidor e o tamanho da fila é limitado a k pacotes.

Conforme mencionado na seção 2.4, neste trabalho o modelo construído no Tangram-II é um modelo com parâmetros. A cadeia de Markov associada ao modelo é gerada uma única vez, com parâmetros que representam as taxas de transição do protocolo modelado. No entanto, como os métodos de solução do Tangram-II são numéricos, é necessário que valores numéricos sejam passados ao Tangram-II antes que ele resolva a cadeia de Markov. Estes valores numéricos são fornecidos por um algoritmo genético. O algoritmo genético utilizado neste trabalho será apresentado no capítulo 4.

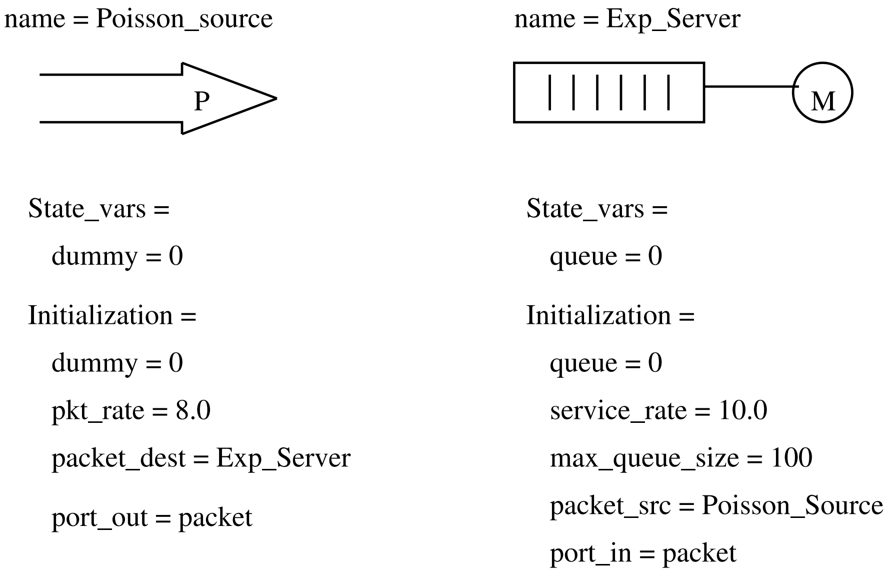

Figura 3.1: Modelo de fila M/M/1/k

Maiores detalhes sobre a sintaxe da ferramenta Tangram-II podem ser obtidos em Silva [9].

3.4 A Função Objetivo

Para a escolha da melhor partição HW/SW a ser utilizada, isto é, para a solução do problema de otimização, devem ser definidos os parâmetros a serem utilizados para a avaliação desta partição. Alguns parâmetros comumente utilizados incluem custo financeiro, desempenho, taxas de comunicação, área de silício, confiabilidade, tamanho do programa, quantidade de dados e tamanho da memória. Este trabalho concentra-se nos parâmetros de desempenho do protocolo e no custo do *hardware*. Este custo baseia-se na área de silício.

É necessário combinar os diversos parâmetros numa função de custo única, denominada *função objetivo*, que fornece uma medida da qualidade da partição. Como os parâmetros da função normalmente "competem entre si", é útil atribuir-se pesos a cada um dos parâmetros que compõem a função objetivo para equalizar as possíveis diferenças de sensibilidade entre eles. A função objetivo utilizada neste trabalho tem a forma:

$$F(x_1, ..., x_n, c_1, ..., c_m) = w_1 * |x_{1d} - x_{1o}| + ... + w_n * |x_{nd} - x_{no}| + \\ w_{n+1} * |c_{1d} - c_{1o}| + ... + w_{n+m} * |c_{md} - c_{mo}| \tag{3.1}$$

Onde:

x_{id} - são os parâmetros de desempenho desejados;

x_{io} - são os parâmetros de desempenho obtidos por cada *indivíduo*;

c_{id} - são os parâmetros de custo do *hardware* desejados;

c_{io} - são os parâmetros de custo obtidos por cada *indivíduo*;

w_i - são os pesos atribuídos a cada parâmetro;

Os parâmetros de desempenho utilizados dependem da especificação e dos requisitos de Qos desejados. A função objetivo fornece uma maneira de se comparar duas partições e selecionar aquela que melhor satisfaz as especificações. Deseja-se minimizar $F(P)$, onde P é uma partição que satisfaz um conjunto de restrições impostas pelo projetista.

3.5 Critério de partição

As medidas de desempenho desejadas são utilizadas para minimizar a função objetivo e calcular as taxas de transição da rede de Petri estocástica que modela o protocolo. Por sua vez, cada transição da SPN está relacionada a um conjunto de operações do protocolo, a ser implementado em *hardware* ou em *software*. O estabelecimento de um critério que relacione as taxas das transições com uma determinada partição HW/SW é uma contribuição fundamental para a automatização do processo de partição.

Neste trabalho foi estabelecido um critério para relacionar as taxas ótimas para o protocolo, calculadas pelo algoritmo genético (AG), com uma partição *hardware/software* que atenda às especificações de desempenho e custo estabelecidas pelo projetista.

Quanto maior a taxa de uma determinada transição maior o número de vezes, por segundo, que um determinado conjunto de operações do protocolo será executado. Este fato poderia levar à conclusão (precipitada) de que essas transições são as que devem ser implementadas em *hardware*. No entanto deve-se considerar, a partir dos resultados obtidos resolvendo-se a cadeia de Markov associada, as probabilidades em estado estacionário de cada estado do sistema. Isto significa que mesmo que uma transição tenha uma alta taxa de disparo, a probabilidade do sistema estar num estado em que a mesma está habilitada pode ser muito baixa.

A partir das considerações realizadas acima foi estabelecido como critério de partição HW/SW das transições, onde cada transição é um conjunto de operações do protocolo, o produto $\lambda_i * \pi$ onde:

λ_i - taxa de disparo da transição i determinada pelo AG.

π - soma das probabilidades em estado estacionário dos estados nos quais a transição i está habilitada.

A cadeia de Markov da figura 3.2, associada ao protocolo *bit alternante* em Molloy [26] ilustra o processo de partição.

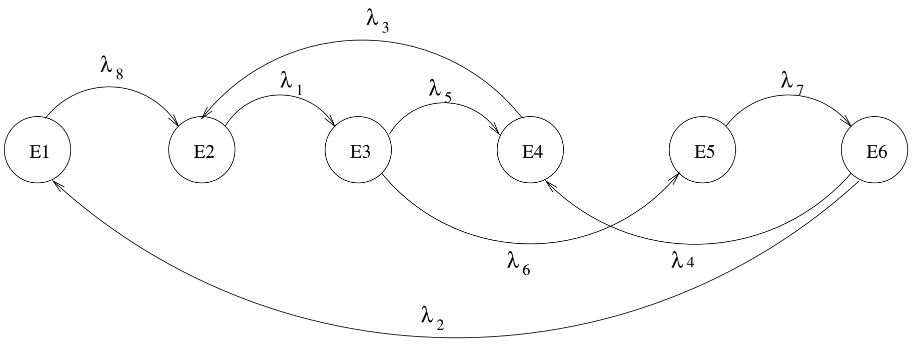

Figura 3.2: Cadeia de Markov associada ao protocolo bit alternante

É como se cada transição tivesse uma "vazão" associada a ela. Desta forma, as operações do protocolo relacionadas com as transições de maior "vazão" seriam implementadas em *hardware* e aquelas associadas às transições de menor "vazão" seriam implementadas em *software*. Este critério tem se mostrado eficiente para os exemplos realizados até o momento, como pode ser comprovado no capítulo 5. Outros protocolos ainda serão desenvolvidos no prosseguimento deste trabalho, possibilitando outros testes referentes à eficiência deste critério.

3.6 Comentários

Neste capítulo foram apresentados conceitos básicos relativos à utilização da análise de desempenho no projeto de um protocolo de comunicação. A metodologia de otimização e o critério de partição propostos fornecem subsídios ao projetista para uma escolha objetiva da melhor partição HW/SW, mantendo a análise em alto nível, ou seja, sem a necessidade de um conhecimento detalhado no nível de circuitos. Os dados de desempenho são fornecidos pelo Tangram-II e os dados provenientes do custo do *hardware* são fornecidos pelo Synopsys. A metodologia também entra imediatamente no ciclo do projeto, pois cada iteração do AG analisa uma solução possível para o problema. No capítulo a seguir são apresentados conceitos básicos sobre algoritmos genéticos assim como o algoritmo utilizado na metodologia proposta.

Capítulo 4

Algoritmos Genéticos

4.1 Conceitos Fundamentais

Os algoritmos genéticos (AG's) vêm tendo larga aceitação devido à simplificação que eles permitem na formulação e solução de problemas de otimização. Esta característica é particularmente útil em problemas de otimização complexos, NP-Completos, como o problema do *caixeiro viajante*, que envolvem um grande número de variáveis e, consequentemente, espaços de solução de dimensões elevadas. Além disso, em muitos casos onde outras estratégias falham na busca de uma solução, os algoritmos genéticos convergem. Outra vantagem da utilização dos AG's em problemas de otimização é que, ao contrário dos métodos tradicionais, os AG's não utilizam derivadas na busca da solução ótima do problema. A estrutura básica do algoritmo genético é mostrada na figura 4.1. Pode-se identificar quatro operações básicas na figura: o **cálculo da aptidão**, realizado com base nos dados fornecidos pelo Tangram-II, a **seleção** dos indivíduos, com maior probabilidade de serem selecionados aqueles que possuem maior aptidão, a **reprodução** e a **mutação**, para que uma nova população seja gerada (**geração**). Estas operações serão descritas com maiores detalhes na seção 4.2.

Algoritmos genéticos simples [11] normalmente trabalham com descrições de entrada formadas por cadeias de bits de tamanho fixo. Outros tipos de AG's podem trabalhar com cadeias de bits de tamanho variável, como por exemplo AG's usados para

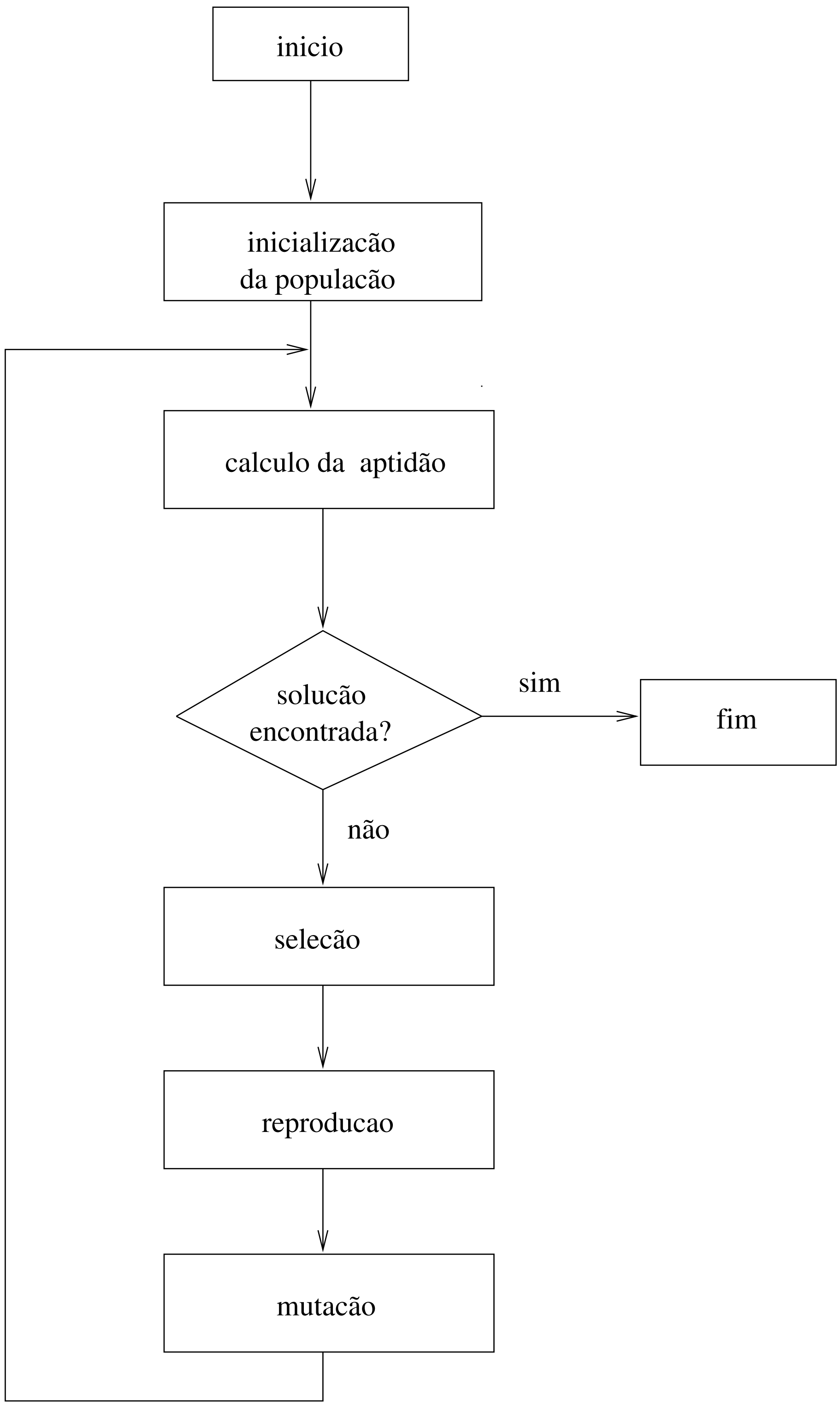

Figura 4.1: Diagrama de blocos de um algoritmo genético simples

Programação Genética [37]. Os AG's possuem um paralelismo implícito decorrente da avaliação independente de cada uma dessas cadeias de bits. Cada cadeia corresponde a um *cromossomo*, onde um ou mais parâmetros são codificados. Cada parâmetro do modelo corresponde a um *gene*. Existem três tipos de representação possíveis para os cromossomos: binária, inteira ou real. A essa representação se dá o nome de *alfabeto* do AG. De acordo com a classe de problema que se deseja resolver pode-se usar qualquer um dos três tipos. Neste trabalho, o protocolo é modelado por uma rede de Petri estocástica e cada cromossomo é formado pelo conjunto de taxas de transição do modelo, onde cada valor é representado por uma variável real. Consequentemente, utilizou-se um vetor de variáveis reais de dimensão m para representar o cromossomo, onde m é o número de parâmetros do modelo. Deve ser notado que cada cromossomo, chamado de indivíduo no AG, corresponde a um ponto no espaço de soluções do problema de otimização. O processo de solução adotado nos algoritmos genéticos consiste em gerar, através de regras específicas, um grande número de indivíduos, *população*, de forma a promover uma varredura tão extensa quanto necessária do espaço de soluções. Neste trabalho, a população com n indivíduos é representada por uma matriz de números reais de dimensão n x m, onde cada linha corresponde a um cromossomo e n é o tamanho da população.

4.2 Operações básicas dos AG's simples

Com referência ao diagrama da figura 4.1, observa-se que cada iteração do algoritmo genético corresponde à aplicação de um conjunto de quatro operações básicas: cálculo de aptidão, seleção, cruzamento e mutação. Ao fim destas operações cria-se uma nova população, chamada de *geração* que, espera-se, representa uma melhor aproximação da solução do problema de otimização que a população anterior.

A população inicial é gerada atribuindo-se aleatoriamente valores aos genes de cada cromossomo. A aptidão bruta de um indivíduo da população é medida por uma função de erro, também chamada de *função objetivo* do problema de otimização. A aptidão bruta é em seguida normalizada (aptidão normalizada), para permitir um melhor controle do processo de seleção. Em geral são usados como critérios de parada do algoritmo a aptidão do melhor indivíduo em conjunto com a limitação do número de gerações.

Outros critérios podem envolver, por exemplo, um erro abaixo de um valor especificado pelo projetista para um determinado parâmetro do problema.

- Inicialização

  Uma população de n indivíduos é gerada aleatoriamente. Cada um dos indivíduos da população representa uma possível solução para o problema, ou seja, um ponto no espaço de soluções.

- Cálculo da aptidão

  Geralmente a aptidão do indivíduo é determinada através do cálculo da função objetivo, que depende das especificações de projeto. Neste trabalho, cada indivíduo é uma entrada para a ferramenta Tangram-II, cuja saída fornece medidas que permitem ao algoritmo genético o cálculo da aptidão do indivíduo relativa ao desempenho. Ainda nesta fase os indivíduos são ordenados conforme a sua aptidão.

- Seleção

  Nesta fase os indivíduos mais aptos da geração atual são selecionados. Esses indivíduos são utilizados para gerar uma nova população por cruzamento. Cada indivíduo tem uma probabilidade de ser selecionado proporcional à sua aptidão. Para visualizar este método considere um círculo dividido em n regiões (tamanho da população), onde a área de cada região é proporcional à aptidão do indivíduo (figura 4.2). Coloca-se sobre este círculo uma "roleta" com n cursores, igualmente espaçados. Após um giro da roleta a posição dos cursores indica os indivíduos selecionados. Este método é denominado *amostragem universal estocástica* [10]. Evidentemente, os indivíduos cujas regiões possuem maior área terão maior probabilidade de serem selecionados várias vezes. Como consequência, a seleção de indivíduos pode conter várias cópias de um mesmo indivíduo enquanto outros podem desaparecer.

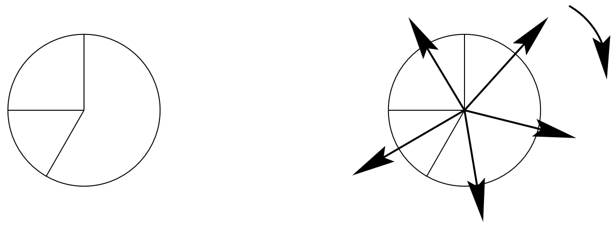

Figura 4.2: Amostragem universal estocástica

- Cruzamento (*Cross-Over*)

  Os indivíduos selecionados na etapa anterior são *cruzados* da seguinte forma: a lista de indivíduos selecionados é embaralhada aleatoriamente criando-se, desta forma, uma segunda lista, chamada lista de *parceiros*. Cada indivíduo selecionado é então cruzado com o indivíduo que ocupa a mesma posição na lista de parceiros. A forma como se realiza este cruzamento é ilustrada na figura 4.3. Os cromossomos de cada par de indivíduos a serem cruzados são particionados em um ponto, chamado ponto de corte, sorteado aleatoriamente. Um novo cromossomo é gerado permutando-se a metade inicial de um cromossomo com a metade final do outro. Deve-se notar que se o cromossomo for representado por uma cadeia de bits, como na figura 4.3, o ponto de corte pode incidir em qualquer posição (bit) no interior de um gene, não importando os limites do gene. No caso de genes representados por números reais, a menor unidade do cromossomo que pode ser permutada é o gene.

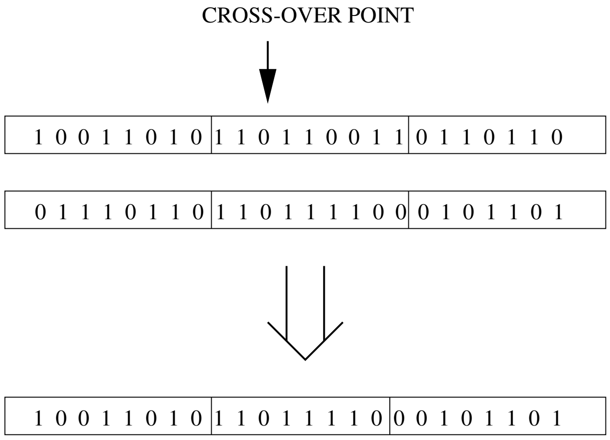

Figura 4.3: Cruzamento de dois indivíduos num algoritmo genético

- Mutação

  A operação de *mutação* é utilizada para garantir uma maior varredura do espaço de estados e evitar que o algoritmo genético convirja muito cedo para mínimos locais. A mutação é efetuada alterando-se o valor de um gene de um indivíduo sorteado aleatoriamente com uma determinada probabilidade, denominada probabilidade de mutação, ou seja, vários indivíduos da nova população podem ter um de seus genes alterado aleatoriamente.

4.3 Parâmetros do Algoritmo Genético

Além da forma como o cromossomo é codificado, existem vários parâmetros do algoritmo genético que podem ser escolhidos para melhorar o seu desempenho, adaptando-o às características particulares de determinadas classes de problemas. Entre eles os mais importantes são: o **tamanho da população**, o **número de gerações**, a **probabilidade de cross-over** e a **probabilidade de mutação**. A influência de cada parâmetro no desempenho do algoritmo depende da classe de problemas que está sendo tratada. Assim, a determinação de um conjunto de valores otimizado para estes parâmetros dependerá da realização de um grande número de experimentos e testes. Na maioria da literatura [11] os valores encontrados estão na faixa de 60 a 65% para a probabilidade de cross-over e entre 0,1 e 5% para a probabilidade de mutação. O tamanho da população e o número de gerações dependem da complexidade do problema de otimização e devem ser determinados experimentalmente. No entanto, deve ser observado que o tamanho da população e o número de gerações definem diretamente o tamanho do espaço de busca a ser coberto.

4.4 Outros tipos de AG's

4.4.1 Algoritmos com Elitismo

- GENITOR

  O Genitor [38] é um algoritmo cujos melhores pontos encontrados são preservados na população. Este procedimento é denominado de *elitismo*. O elitismo provoca uma busca mais agressiva, que na prática é geralmente bastante efetiva. No entanto existe o perigo de uma convergência prematura para mínimos locais. Cada indivíduo selecionado é cruzado com seu parceiro e colocado no lugar do pior indivíduo da população anterior. A aptidão é atribuída de acordo com um *ranking*, ou seja, a aptidão de cada indivíduo assume valores discretos.

- CHC

  Outro AG que coleciona os melhores indivíduos da população atual é o CHC (*Cross generational elitist selection, heterogeneous recombination and Cataclysmic mutation*). Após o cruzamento, realizado aleatoriamente, os N melhores indivíduos são coletados levando-se em consideração a população atual e a população gerada após o cruzamento. Remove-se os indivíduos duplicados. Este método impõe uma busca mais agressiva, assim como no Genitor. Repare que a seleção está implícita no algoritmo, a partir do momento que se escolhe os melhores indivíduos de cada população (anterior e atual). Normalmente, este algoritmo utiliza populações pequenas, com cerca de 50 indivíduos. O ponto de *cross-over* é sempre a metade do cromossomo. Para solucionar o problema de convergência prematura para mínimos locais é utilizada uma alta taxa de mutação, preservando-se, no entanto, o melhor indivíduo da população. A partir da primeira seleção aleatória, utiliza-se o *cross-over* diretamente nas populações subsequentes.

4.4.2 Algoritmos Híbridos

Muitos autores consideram que os AG's nem sempre são a melhor solução para problemas de otimização específicos. Desta forma, os algoritmos híbridos utilizam os AG's como ponto de partida para métodos de otimização tradicionais, como o *Simulated Annealing* [39], método de Powel [40], entre outros. Os AG's introduzem um *overhead* computacional devido à busca baseada em populações. A mistura das técnicas tradicionais com os AG's introduzem uma espécie de aprendizado no AG, pois os cromossomos selecionados resultaram da aplicação da técnica denominada *hill-climbing*, base dos métodos de otimização tradicionais, que utilizam derivadas.

4.5 Algoritmo utilizado na Metodologia Proposta

Nesta seção é apresentado o algoritmo desenvolvido neste trabalho para implementar a metodologia proposta. O algoritmo utilizado neste trabalho é um algoritmo híbrido pois as faixas de variação permitidas para cada taxa de transição do modelo, que são utilizadas para a geração da população inicial, são determinadas de acordo com a sensibilidade de cada taxa em relação ao desempenho desejado do protocolo.

A princípio é estabelecida uma faixa única, pequena, para todos os parâmetros. Após o AG ser executado para 10 gerações, calcula-se as sensibilidades de cada parâmetro e, a partir delas, as novas faixas de variação para cada parâmetro. Utiliza-se métodos tradicionais para o cálculo das sensibilidades, que fazem uso de derivadas. A forma de cálculo das sensibilidades e das novas faixas estão descritas no apêndice **A**.

O algoritmo desenvolvido é um algoritmo genético simples, dentro do qual foram realizadas as alterações necessárias de forma a adaptá-lo para o problema em questão, ou seja, a codificação do cromossomo como um vetor de números reais que representam os parâmetros (taxas de transição do modelo da rede de Petri) a serem determinados. O número de taxas vai depender do modelo construído para o protocolo em questão.

O tamanho da população inicial do AG é estabelecido pelo projetista. Esta população inicial é obtida por uma função que gera números reais aleatoriamente, utilizando um banco de sementes e o relógio do computador. Os valores numéricos correspondentes a cada indivíduo são transferidos pelo AG ao modelo de parâmetros do Tangram-II através de uma função denominada *trans_param*, desenvolvida por Figureiredo [8] e alterada para ler valores numéricos de um arquivo e passá-los ao modelo do Tangram-II. Desta forma é possível executar um dos métodos numéricos de solução da cadeia de Markov existentes no Tangram-II, através do programa **solv**, que é chamado pelo AG. Consequentemente, determina-se as probabilidades em estado estacionário da cadeia de Markov associada a cada indivíduo da população e pode-se obter as respectivas medidas de interesse para cada indivíduo, conforme a qualidade de serviço que se deseja para o protocolo. Nos exemplos apresentados na seção 5.2 foi utilizado o método SOR (*Sucessive Over Relaxation*) porque, como o número de estados da cadeia de Markov é relativamente pequeno, não se detectou diferenças entre os diversos métodos de solução testados no que diz respeito aos resultados. A aptidão bruta é determinada através do valor da função objetivo, dado pela soma de erros entre os valores das medidas obtidas e os valores desejados. Portanto, tendo sido determinada a aptidão bruta de cada indivíduo, calcula-se a sua aptidão normalizada, realiza-se o processo de seleção, cruzamento, mutação e uma nova população é gerada, caso nenhum dos indivíduos atinja a solução estipulada.

Quando o AG obtém os valores de taxas que satisfazem a especificação em termos de desempenho, determina-se a primeira partição HW/SW através do procedimento descrito na seção 3.5 e os valores das taxas a serem implementadas em *hardware* são passados para o Synopsys, que fornece os dados relativos ao custo daquela partição. Uma vez determinado o custo da partição, calcula-se novamente a função objetivo para cada indivíduo, também com os dados relativos ao custo. Repare que, inicialmente, apenas o desempenho é usado para calcular o valor da aptidão do indivíduo porém, nas etapas subsequentes, a aptidão de cada indivíduo já inclui o valor do custo do do *hardware*, fornecido pelo Synopsys (o custo do *software* é considerado igual a zero).

A estrutura do algoritmo genético tem uma complexidade polinomial (n^3), proporcional ao número de pontos que se deseja aproximar, tamanho da população e número de gerações.

4.6 Comentários

Os AG's são apropriados para problemas de otimização complexos, que envolvem muitas variáveis e um espaço de soluções de dimensão elevada, abrangendo um grande número de aplicações. O controle sobre os parâmetros do algoritmo é importante para uma convergência rápida. Para problemas específicos é aconselhável a utilização de algoritmos híbridos, que misturam as técnicas dos AG's com os métodos de otimização tradicionais. Os AG's possuem um custo computacional elevado, devido ao grande número de variáveis envolvidas, às populações elevadas e ao alto número de gerações utilizado para a cobertura do espaço de soluções.

No próximo capítulo o algoritmo genético aqui apresentado será aplicado a dois protocolos bastante conhecidos para a avaliação da metodologia e do critério de partição adotado.

Capítulo 5

Resultados Preliminares

5.1 Introdução

Neste capítulo serão apresentados e analisados os resultados numéricos obtidos através da implementação computacional da metodologia proposta. Com essa finalidade são apresentados gráficos, diagramas e tabelas, sobre os quais são realizadas observações fundamentais para a análise da viabilidade da metodologia e conclusões deste trabalho.

Para a obtenção dos resultados numéricos mostrados a seguir, foi desenvolvido um programa na linguagem de programação C, que implementa a metodologia proposta. Este programa foi executado em um microcomputador Pentium-II 400 Mhz com 128MB de memória RAM, no sistema operacional Linux.

5.2 Exemplos de Modelos

Serão apresentados a seguir alguns exemplos de protocolos modelados utilizando-se a metodologia proposta para análise da sua eficiência e praticidade, além da viabilidade do critério de partição HW/SW utilizado.

5.2.1 Protocolo bit alternante

O protocolo *bit alternante* utilizado em Molloy [26] foi modelado em uma rede de Petri, como é ilustrado na figura 5.1 e, posteriormente, convertido para o modelo do Tangram-II. O modelo equivalente do Tangram-II é ilustrado na figura 5.2. Para se verificar a correção deste último modelo, foram usados para as taxas das distribuições dos eventos os valores numéricos mostrados na tabela 5.1, que são os mesmos valores utilizados em Molloy [26]. A cadeia de Markov gerada foi resolvida para diversos valores de taxa de disparo da transição NEW_MSG (chegada de novas mensagens), que indica a *carga* do protocolo. O produto desta taxa pela probabilidade do sistema estar num estado no qual a transição associada a ela está habilitada fornece a vazão do protocolo. Este processo permite que seja encontrada a curva "vazão versus carga" para o modelo do Tangram-II. Obteve-se a curva da figura 5.3, que é a mesma curva obtida em Molloy [26]. A seguir, implementa-se o algoritmo genético. Nesta etapa, as taxas das transições da rede de Petri são parâmetros, bem como as do modelo do Tangram-II. É importante ressaltar que o algoritmo genético, no momento que gerar os valores aleatórios para cada taxa, deve atender a restrições de projeto. Neste caso, a restrição estipulada foi de que a faixa de variação dos números aleatórios gerados para taxa de disparo da transição Timeout deve ser menor ou igual que a menor faixa de variação entre todas as taxas do protocolo, para que o *timeout* não seja disparado sem necessidade, diminuindo a vazão do protocolo.

Como a curva da figura 5.3 é "bem comportada", 10 pontos são suficientes para aproximá-la e obter os parâmetros desejados. Foram realizados experimentos com uma população de até 5000 indivíduos, 10 gerações e o valor 0,001 para a soma total dos erros dos 10 valores de vazão desejados. Posteriormente, verificou-se que devido à simplicidade da curva a ser aproximada (monotonamente crescente), uma população com 100 indivíduos é suficiente para o algoritmo convergir para os valores de vazão desejados. Os resultados obtidos são apresentados na tabela 5.2.

Repare que o algoritmo genético pode fornecer inúmeras soluções que atendam à função objetivo no que diz respeito ao desempenho. A inclusão da parcela de custo nesta função, que é calculada pela ferramenta Synopsys, reduz o espaço de soluções.

Transição	Taxa (1/seg.)
Send, Send Ack	9,375
Msg Drop, Ack Drop	3,91
Crc ok, Ack ok	74,22
Timeout	1,00

Tabela 5.1: Taxas originais da Rede de Petri do protocolo bit alternante

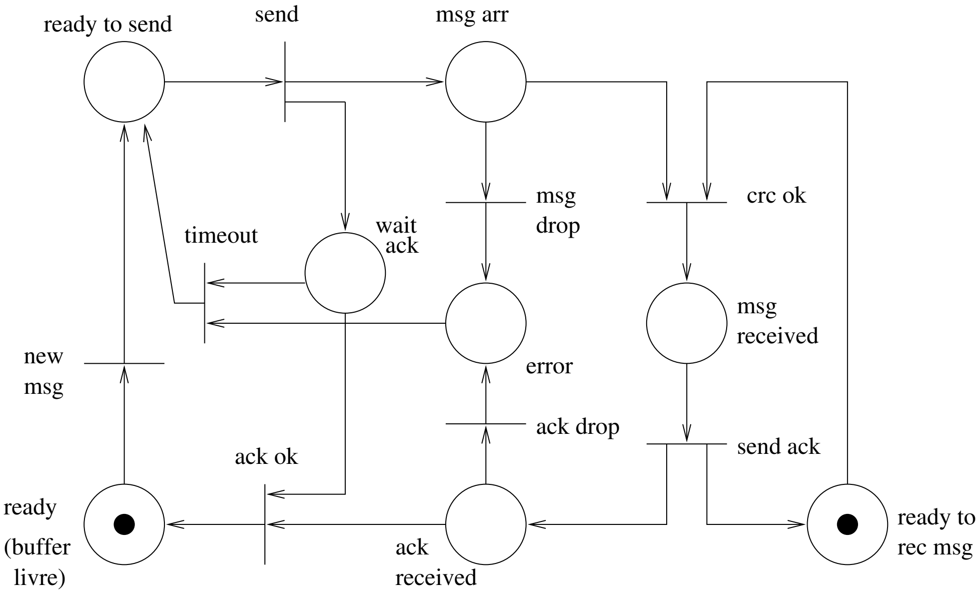

Figura 5.1: Rede de Petri do protocolo bit alternante

Utilizando-se as taxas da tabela 5.2, aplica-se o critério descrito na seção 3.5 e calcula-se o produto $\lambda_i * \pi$ para cada transição do protocolo, onde λ_i é cada uma das taxas de transição da rede de Petri (vide tabela 5.2), calculadas pelo AG, e π é a soma das probabilidades em estado estacionário da cadeia de Markov resolvida pelo Tangram-II e que representa a probabilidade do sistema se encontrar em estados onde a transição i está habilitada. Os resultados obtidos são apresentados a seguir:

```
name=SPN2

    State_vars=
    fp1, fp2
    fp3,fp4,fp5,
    fp6,fp7,fp8

    Initialization=
    fp1=1
    fp2=1
    fp3=0
    fp4=0
    fp5=0
    fp6=0
    fp7=0
    fp8=0
    /*rate1=9.375
    rate2=74.22
    rate3=1.000
    rate4=3.91
    rate5=3.91*/
    /*rate6=74.22*/
    /*rate7=9.375*/
    /*lambda=2.0*/
Rewards=
  impulse_reward=throughput
  event= New_msg,1
  value=1;
Messages=
```

```
Declaration=

Var
 State: fp1,fp2,fp3,fp4,fp5,fp6,fp7, fp8 ;

Param
 Float: lambda1, lambda2, lambda3,lambda4;
 Float: lambda5, lambda6, lambda7, lambda8;

 Events=

   event= Send(EXP,lambda1)
   condition= (fp3 > 0)
   action= {
             int p3,p4,p5;
             p3=fp3;
             p4=fp4;
             p5=fp5;
             p4=p4+1;
             p5=p5+1;
             p3=p3-1;
             set_st("fp3",p3);
             set_st("fp4",p4);
             set_st("fp5",p5);
            };

   event= Ack_ok(EXP,lambda2)
   condition= ((fp5 > 0) && (fp8 >0))
   action= {
             int p1,p8,p5;
             p5=fp5;
             p1=fp1;
             p8=fp8;
             p5=p5-1;
             p1=p1+1;
             p8=p8-1;
             set_st("fp8",p8);
             set_st("fp1",p1);
             set_st("fp5",p5);
            };

  event= Time_out(EXP,lambda3)
   condition= ((fp5 > 0) && (fp7 > 0))
   action= {
             int p3,p7,p5;
             p7=fp7;
             p5=fp5;
             p3=fp3;
             p7=p7-1;
             p5=p5-1;
             p3=p3+1;
             set_st("fp3",p3);
             set_st("fp7",p7);
             set_st("fp5",p5);
            };

 event= Ack_drop(EXP,lambda4)
   condition= (fp8 > 0)
   action= {
             int p7,p8;
             p7=fp7;
             p8=fp8;
             p7=p7+1;
             p8=p8-1;

             set_st("fp7",p7);
             set_st("fp8",p8);
            };

 event= Msg_drop(EXP,lambda5)
   condition= (fp4 > 0)
   action= {
             int p4,p7;
             p4=fp4;
             p7=fp7;
             p4=p4-1;
             p7=p7+1;


             set_st("fp4",p4);
             set_st("fp7",p7);
            };

 event= Crc_ok(EXP,lambda6)
   condition= ((fp4 > 0) && (fp2 > 0))
   action= {
             int p2,p4,p6;
             p2=fp2;
             p4=fp4;
             p6=fp6;
             p4=p4-1;
             p2=p2-1;
             p6=p6+1;
             set_st("fp2",p2);
             set_st("fp4",p4);
             set_st("fp6",p6);
            };

 event= Send_ack(EXP,lambda7)
   condition= (fp6 > 0)
   action= {
             int p2,p6,p8;
             p2=fp2;
             p6=fp6;
             p8=fp8;
             p2=p2+1;
             p8=p8+1;
             p6=p6-1;
             set_st("fp2",p2);
             set_st("fp6",p6);
             set_st("fp8",p8);
            };

 event= New_msg(EXP,lambda8)
   condition= (fp1 > 0)
   action= {
             int p1,p3;
             p1=fp1;
             p3=fp3;
             p1=p1-1;
             p3=p3+1;
             set_st("fp1",p1);
             set_st("fp3",p3);
            };
```

Figura 5.2: Modelo do TANGRAM-II para a Rede de Petri do protocolo bit alternante

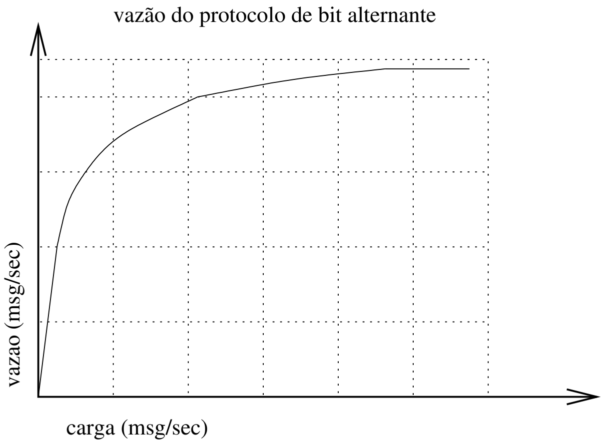

Figura 5.3: Curva vazão versus carga do protocolo bit alternante

Transição	Taxa (1/seg.)
Send (λ_1)	16,20
Send Ack (λ_7)	11,71
Msg Drop (λ_5)	1,27
Ack Drop (λ_4)	1,67
Ack ok (λ_2)	30,95
Crc ok (λ_6)	10,75
Timeout (λ_3)	2,70

Tabela 5.2: Taxas da Rede de Petri do protocolo bit alternante obtidas pelo AG

$$\lambda_1 * \pi_2 = 2,57 \tag{5.1}$$

$$\lambda_7 * \pi_5 = 2,23 \tag{5.2}$$

$$\lambda_2 * \pi_6 = 2,14 \tag{5.3}$$

$$\lambda_6 * \pi_3 = 2,23 \tag{5.4}$$

$$\lambda_4 * \pi_6 = 0,09 \tag{5.5}$$

$$\lambda_5 * \pi_3 = 0,34 \tag{5.6}$$

$$\lambda_3 * \pi_4 = 0,43 \tag{5.7}$$

onde cada π_j representa a probabilidade em estado estacionário de cada um dos estados possíveis do protocolo (uma marca da rede de Petri), dados abaixo:

E_1 - 1 ficha em P1 e outra em P2;
E_2 - 1 ficha em P2 e outra em P3;
E_3 - 1 ficha em P2, outra em P4 e outra em P5;
E_4 - 1 ficha em P5 e outra em P6;
E_5 - 1 ficha em P2, outra em P5 e outra em P7;
E_6 - 1 ficha em P2, outra em P5 e outra em P8;

Observando-se as equações acima, verifica-se que a "vazão" da transição 3 (Timeout) é aproximadamente igual à soma das "vazões" das transições 4 e 5, que correspondem a perdas. Este resultado é coerente, pois pode-se observar na rede de Petri da figura 5.1 que o *timeout* é habilitado sempre que ocorre uma perda. Verifica-se também que a soma das "vazões" das transições 4 e 5 é aproximadamente igual a 5% da soma das "vazões" das transições 1, 2, 6, 7, que são transições de transmissão e recepção de mensagens. Desta forma, a probabilidade de erro especificada em Molloy [26] é atendida. Estes fatos fornecem uma indicação de que o critério de partição utilizado se mostra coerente para este exemplo.

Portanto, as operações do protocolo correspondentes às transições 1, 2, 6 e 7 devem ser implementadas em *hardware* enquanto que aquelas correspondentes às transições 3, 4 e 5 devem ser implementadas em *software*. Este é o primeiro resultado que é passado para o Synopsys, que fornece o custo do *hardware* em termos da área de silício do circuito integrado a ser sintetizado. Com o valor fornecido por esta ferramenta, a função objetivo do algoritmo genético é realimentada e uma nova otimização é realizada. Este processo se repete até que o erro especificado para a função objetivo (desempenho e custo) seja alcançado.

5.2.2 TCP "Congestion Avoidance"

TCP "Congestion Avoidance" com perda determinística

Neste exemplo é analisado o desempenho do TCP quando em fase de controle de congestionamento, o "TCP Congestion Avoidance", utilizando-se o modelo matemático proposto em Mathis [41]. A janela de congestionamento é incrementada de uma parcela constante para cada ACK recebido pelo emissor e dividida pela metade quando é detectado um congestionamento. O modelo é simplificado e supõe que somente o mecanismo "Congestion Avoidance" influencia o desempenho do TCP. Algumas destas suposições são:

- tráfego moderado (com poucas perdas), permitindo que o TCP se recupere das perdas sem a necessidade de *timeout*;
- não existem retransmissões por *timeout*;
- janela do receptor com um tamanho "suficientemente grande", ou seja, a janela de congestionamento é controlada pela janela do emissor;
- múltiplas perdas dentro de um *round trip time* (RTT) indicam sinal de congestionamento;
- o emissor tem sempre dados a enviar;
- o receptor envia um ACK para cada *segmento* de dados (certa quantidade de dados em bytes de tamanho fixo) recebido;
- o RTT é constante porque a banda passante é suficiente para que não ocorra a formação de filas;
- as conexões são longas o suficiente para que o algoritmo de "Congestion Avoidance" alcance estado estacionário;
- perda de pacotes com uma probabilidade constante p, o que significa que aproximadamente $1/p$ pacotes são enviados corretamente, seguidos da ocorrência de uma perda;
- detalhes referentes a recuperação e retransmissão dos pacotes são desprezados, embora a recuperação de perdas seja completada dentro de um RTT.

Com as suposições acima, pode-se aproximar a curva *cwnd versus RTT* por um gráfico com a forma de um *dente de serra*, como é mostrado na figura 5.4, onde *cwnd* é o tamanho da janela de congestionamento do emissor em número de pacotes. O modelo se aplica a quase todas as implementações SACK TCP (TCP com *Selective Acknowledgements*).

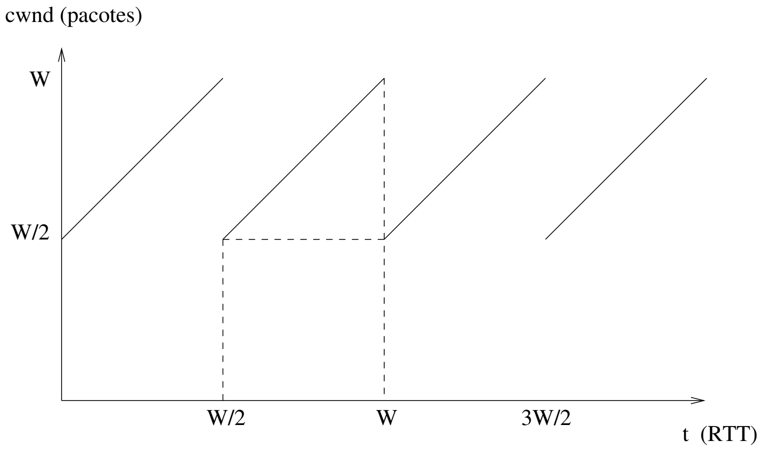

Figura 5.4: cwnd versus RTT com perda periódica

Analisando-se a figura 5.4, encontra-se a equação para o tamanho máximo da janela de congestionamento em função da probabilidade de perda [41], considerando-se a quantidade de pacotes enviada em um ciclo (área sob um ciclo do gráfico) e a informação de que $1/p$ pacotes são enviados em cada ciclo:

$$W = \sqrt{8/(3 * p)} \tag{5.8}$$

onde W é o tamanho máximo da janela de congestionamento, em pacotes. Mas a largura de faixa em bytes, BW, é dada pela quantidade de dados transmitidos por ciclo (em bytes) dividida pelo tempo do ciclo (em segundos), ou seja:

$$BW = (MSS * 3/8 * W^2)/(RTT * W/2) \tag{5.9}$$

onde MSS é o tamanho máximo de um segmento de dados em bytes. Substituindo-se W na equação 5.9:

$$BW * RTT/MSS = C/\sqrt{p} \tag{5.10}$$

onde C é uma constante de proporcionalidade, neste caso igual a $\sqrt{3/2}$. A expressão $BW * RTT/MSS$ fornece o tamanho da janela de congestionamento em número de pacotes. A constante C engloba uma série de termos constantes que dependem do tipo de implementação do TCP, da estratégia de envio de ACK's utilizada e do mecanismo de perda (periódica ou aleatória).

Realizando o mesmo procedimento do exemplo da seção 5.2.1, o mecanismo de controle de congestionamento "TCP Congestion Avoidance" é modelado em uma rede de Petri, como é ilustrado na figura 5.5. Na figura 5.6 é mostrado o modelo do Tangram-II que considera a probabilidade de perda constante (determinística).

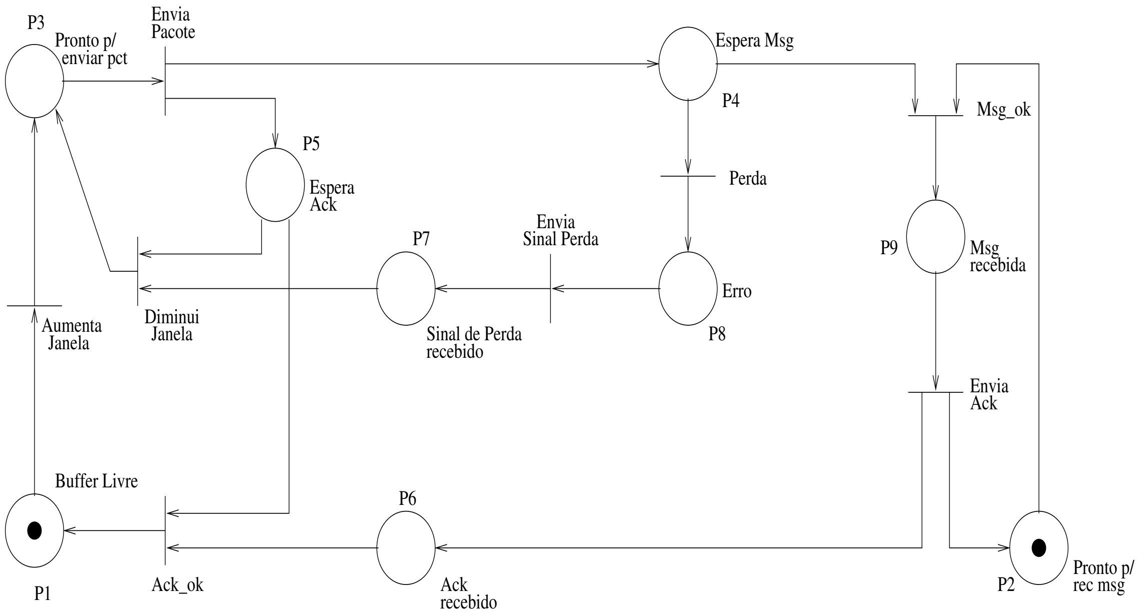

Figura 5.5: Rede de Petri do mecanismo de controle de congestionamento "TCP Congestion Avoidance"

Os valores para as taxas da rede de Petri do modelo determinístico são determinados considerando-se que para cada unidade de tempo (no caso, um RTT), que corresponde à recepção de um ACK, o valor da janela é incrementado de um segmento, a partir de um valor inicial que, em estado estacionário, corresponde à metade da janela máxima W, onde W é calculada a partir da equação 5.8. Como exemplo, considerou-se um valor para a probabilidade de perda constante igual a 0,01, ou seja, aproximadamente a cada 100 pacotes enviados corretamente um é perdido. Desta forma, no modelo do Tangram-II, atribui-se o valor 4 para as taxas dos eventos (transições) Envia_pct (λ_2), Msg_ok (λ_7), Envia_ack (λ_5) e Ack_ok (λ_3) e o valor 1 para o evento Aumenta_jan (λ_1). O termo "DET" indica que todos os eventos são determinísticos. Para garantir que o disparo da transição PERDA seja realizado após a transmissão de exatamente $1/p$ pacotes, criou-se uma variável de estado denominada **janela**, cujo valor máximo, calculado pela equação 5.8, é uma das condições para que o evento **Perda** seja disparado.

Desta forma, a cada RTT $(1/4 + 1/4 + 1/4 + 1/4)$ a janela de congestionamento é aumentada de um segmento. Note que segundo o gráfico da figura 5.4, a diminuição da janela devido a um congestionamento ocorre quase que instantaneamente (1 RTT, segundo a suposição do modelo), por isto são fixados valores altos para as taxas dos eventos **Perda** (λ_8), Sinal_perda (λ_6) e Diminui_jan (λ_4). A partir do modelo da figura 5.6 realiza-se uma simulação para se verificar o comportamento da janela de congestionamento. Verifica-se que o comportamento da janela de congestionamento fornece exatamente um gráfico no formato de uma dente de serra.

TCP "Congestion Avoidance" com perda aleatória

Nesta seção é apresentada uma versão onde a perda de pacotes é aleatória. Desta forma, a taxa da transição PERDA da rede de Petri mostrada na figura 5.5 possui uma distribuição de probabilidades exponencial, segundo a metodologia apresentada. A hipótese de que não se formam filas continua valendo para este caso. Neste caso, o modelo do Tangram-II é um modelo com parâmetros e pode ser visualizado na figura 5.7.

A determinação dos parâmetros do protocolo é realizada pelo AG aproximando a curva $(BW * RTT)/MSS\ versus\ perda$, traçada utilizando-se a equação A.10, com $C = 1$, que pode ser visualizada na figura 5.8. A expressão $BW * RTT/MSS$ é uma **estimativa** do tamanho médio da janela de congestionamento [41]. Deve ser observado que o gráfico está em escala logarítmica.

A cadeia de Markov gerada foi resolvida para diversos valores de probabilidade de perda (transição PERDA), de forma que o algoritmo genético pudesse escolher o conjunto de valores para as taxas de transição que melhor aproxima a curva.

Assim como no exemplo da seção 5.2.1, também foram utilizados 10 pontos para aproximar a curva do modelo e se obter os parâmetros desejados. Foram realizados experimentos com uma população de 100 indivíduos e utilizou-se como critério de parada do algoritmo genético o limite de 10 gerações. A taxa da transição PERDA (λ_8) foi variada (abscissa da curva) e a função objetivo foi calculada baseada na diferença entre o tamanho da janela obtido e o tamanho da janela desejado. Obteve-se um erro mínimo para a função objetivo (considerando apenas o desempenho) igual a aproximadamente 1,67. Este valor indica um erro aproximado de 1,7% para cada um dos pontos escolhidos. Os resultados obtidos são apresentados na tabela 5.3. O pequeno erro no valor das janelas fornecidas pelo AG se deve ao valor da constante C: com perdas aleatórias seu valor é um pouco menor do que 1 [41].

Transição	Taxa (1/seg.)
Aumenta_jan (λ_1)	446,94
Envia_pct (λ_2)	1078,67
Envia_ack (λ_5)	984,02
Diminui_jan (λ_4)	46,17
Ack_ok (λ_3)	803,90
Sinal_perda (λ_6)	116,41
Msg_ok (λ_7)	1

Tabela 5.3: Taxas da Rede de Petri do TCP “Congestion Avoidance” obtidas pelo AG

Observe que a transição MSG_OK(λ_7) foi fixada em 1 porque o modelo original assume que $1/p$ pacotes são enviados corretamente antes da ocorrência de uma perda portanto, em média, $1/p$ pacotes são enviados antes da ocorrência de uma perda. Aplicando-se o mesmo critério de partição utilizado na seção 5.2.1 às taxas da tabela 5.3 obtém-se os seguintes resultados:

$$\lambda_1 * \pi_1 = 1,07 \tag{5.11}$$

$$\lambda_2 * \pi_2 = 1,76 \tag{5.12}$$

$$\lambda_3 * \pi_7 = 1,04 \tag{5.13}$$

$$\lambda_4 * \pi_6 = 0,013 \tag{5.14}$$

$$\lambda_5 * \pi_5 = 1,71 \tag{5.15}$$

$$\lambda_6 * \pi_4 = 0,23 \tag{5.16}$$

$$\lambda_7 * \pi_3 = 0,99 \tag{5.17}$$

onde cada π_j é dado por:

E_1 - 1 ficha em P1 e outra em P2;
E_2 - 1 ficha em P2 e outra em P3;
E_3 - 1 ficha em P2, outra em P4 e outra em P5;
E_4 - 1 ficha em P2, outra em P5 e outra em P8;
E_5 - 1 ficha em P5 e outra em P9;
E_6 - 1 ficha em P2, outra em P5 e outra em P7;
E_7 - 1 ficha em P2, outra em P5 e outra em P6;

Portanto, as operações do protocolo correspondentes às transições 2, 3, 5 e 7 devem ser implementadas em *hardware* enquanto que aquelas correspondentes às transições 1, 4 e 6 devem ser implementadas em *software*. Observando-se a rede de Petri do protocolo, pode-se considerar as transições 2, 3, 5 e 7, que são aquelas que correspondem à transmissão de pacotes e ao recebimento de ACK's, como se fosse o *caminho crítico do protocolo*, isto é, as transições pelas quais o protocolo passa o maior número de vezes quando em estado estacionário. Este resultado é uma boa indicação da eficiência do critério de partição HW/SW. Da mesma forma que no exemplo anterior, este é o primeiro resultado que é passado para o Synopsys, que realimenta a função objetivo do algoritmo genético para que uma nova otimização seja realizada.

Não foi possível encontrar uma solução analítica que considerasse a variação no tamanho da janela no modelo do Tangram-II, pois neste caso o número de estados da cadeia de Markov gerada excede o limite da ferramenta. Foi necessário criar uma *recompensa de impulso* (vide seção 3.3) denominada **tam_jan** para que o tamanho da janela ao longo do tempo pudesse ser analisado. Esta recompensa é incrementada cada vez que o evento **Aumenta_jan** é disparado ou decrementada cada vez que o evento **Diminui_jan** é disparado. Após a determinação das taxas de transição, realizada pelo algoritmo genético, a taxa do evento **Perda**, que tem distribuição exponencial, foi fixada em 0,01 para que se pudesse fazer uma comparação do comportamento do protocolo com perda periódica (caso anterior) com o comportamento do protocolo com perda aleatória. Foi realizada uma simulação onde foi feito um *trace* da recompensa **tam_jan**. O resultado pode ser visualizado na figura 5.9. O modelo do Tangram-II foi construído para que o tamanho da janela volte ao valor original cada vez que ocorre uma perda, ao contrário do modelo original, onde o tamanho da janela cai à metade. Esta alteração foi realizada para se visualizar melhor o comportamento do crescimento do tamanho da janela em relação à variação da probabilidade de perda. Não foi fixado uma valor máximo para a janela como condição para o evento **Perda** ser disparado, como no caso anterior, pois neste caso a probabilidade de perda tem distribuição exponencial.

Observa-se através do gráfico da figura 5.9 e dos valores fornecidos pela simulação realizada no Tangram-II que a média dos valores máximos do tamanho que a janela de congestionamento alcança em cada ciclo é aproximadamente igual ao valor máximo da janela do caso determinístico, para uma mesma probabilidade de perda. Quanto menor a probabilidade de perda, melhor é o resultado fornecido pelo algoritmo genético. É justamente nestes casos (poucas perdas) que o modelo da equação A.10 melhor se ajusta aos casos reais. Este resultado indica que o modelo foi bem construído, os valores das taxas encontradas estão coerentes e o seu comportamento se aproxima mais de um comportamento real do mecanismo "Congestion avoidance" encontrado nas diversas implementações do protocolo TCP. O modelo **ideal** da seção anterior faz muitas suposições que não se verificam na prática. Os resultados das simulações realizadas em Mathis [41] comprovam que o modelo da equação A.10 é um limite superior para a largura de faixa de implementações do protocolo TCP baseadas no mecanismo "Congestion Avoidance".

```
name=TCP2

State_vars=
fp1,fp2
fp3,fp4,fp5,
fp6,fp7,fp8,
fp9,janela;

Initialization=
fp1=0
fp2=1
fp3=0
fp4=0
fp5=0
fp6=0
fp7=0
fp8=0
fp9=0
janela=100
inicio=1
jan_max=200
Messages=

Rewards=

impulse_reward = tam_jan

event = Inicializa_jan,1
value = 100;

event = Aumenta_janela,1
value = 1;

event = Diminui_janela,1
value = -janela/2;
```

```
Declaration=

Var
 State: fp1,fp2,fp3,fp4,fp5,fp6,fp7,fp8,fp9,janela,inicio;

Const
 Integer: jan_max;

/*Param
 Float: lambda1,lambda2,lambda3,lambda4;
 Float: lambda5,lambda6,lambda7,lambda8;*/

  Events=

    event= Inicializa_jan(DET,1000)
    condition= (inicio == 1)
    action= {
             int ini,p1;
             ini=inicio;
             p1=fp1;
             ini=ini-1;
             p1=p1+1;
             set_st("fp1",p1);
             set_st("inicio",ini);
            };

    event= Envia_MSG(DET,4)
    condition= (fp3 > 0)
    action= {
             int p3,p4,p5;
             p3=fp3;
             p4=fp4;
             p5=fp5;
             p4=p4+1;
             p5=p5+1;
             p3=p3-1;
             set_st("fp3",p3);
             set_st("fp4",p4);
             set_st("fp5",p5);
            };

    event= Ack_OK(DET,4)
    condition= ((fp5 > 0) && (fp6 > 0))
    action= {
             int p1,p5,p6;
             p6=fp6;
             p1=fp1;
             p5=fp5;
             p6=p6-1;
             p5=p5-1;
             p1=p1+1;
             set_st("fp1",p1);
             set_st("fp5",p5);
             set_st("fp6",p6);
            };

    event= Diminui_janela(DET,10000)
    condition= ((fp5 > 0) && (fp7 > 0))
    action= {
             int p3,p5,p7,jan;
             jan=janela;
             jan=jan/2;
             p7=fp7;
             p5=fp5;
             p3=fp3;
             p5=p5-1;
             p7=p7-1;
             p3=p3+1;
             set_st("fp3",p3);
             set_st("fp5",p5);
             set_st("fp7",p7);
             set_st("janela",jan);
             };

  event= Msg_perdida(DET,10000)
    condition= ((fp4 > 0) && (janela > jan_max))
    action= {
             int p4,p8;
             p4=fp4;
             p8=fp8;
             p4=p4-1;
             p8=p8+1;
             set_st("fp4",p4);
             set_st("fp8",p8);
            };

  event= Msg_OK(DET,4)
    condition= ((fp4 > 0) && (fp2 > 0) && (janela <= jan_max))
    action= {
             int p2,p4,p9;
             p2=fp2;
             p4=fp4;
             p9=fp9;
             p2=p2-1;
             p4=p4-1;
             p9=p9+1;

             set_st("fp4",p4);
             set_st("fp2",p2);
             set_st("fp9",p9);
            };

  event= Envia_sinal_de_perda(DET,10000)
    condition= (fp8 > 0)
    action= {
             int p8,p7;
             p8=fp8;
             p7=fp7;
             p8=p8-1;
             p7=p7+1;
             set_st("fp8",p8);
             set_st("fp7",p7);
            };

  event= Envia_ack(DET,4)
    condition= (fp9 > 0)
    action= {
             int p2,p6,p9;
```

```
Declaration=

Var
 State: fp1,fp2,fp3,fp4,fp5,fp6,fp7,fp8,fp9;

Param
 Float: lambda1,lambda2,lambda3,lambda4;
 Float: lambda5,lambda6,lambda7,lambda8;
```

```
name=TCP2

State_vars=
fp1,fp2
fp3,fp4,fp5,
fp6,fp7,fp8,
fp9;

Initialization=
fp1=1
fp2=1
fp3=0
fp4=0
fp5=0
fp6=0
fp7=0
fp8=0
fp9=0

Messages=

Rewards=
```

```
Events=

  event= Envia_MSG(EXP,lambda2)
  condition= (fp3 > 0)
  action= {
           int p3,p4,p5;
           p3=fp3;
           p4=fp4;
           p5=fp5;
           p4=p4+1;
           p5=p5+1;
           p3=p3-1;
           set_st("fp3",p3);
           set_st("fp4",p4);
           set_st("fp5",p5);
          };

  event= Ack_OK(EXP,lambda3)
  condition= ((fp5 > 0) && (fp6 > 0))
  action= {
           int p1,p5,p6;
           p6=fp6;
           p1=fp1;
           p5=fp5;
           p6=p6-1;
           p5=p5-1;
           p1=p1+1;
           set_st("fp1",p1);
           set_st("fp5",p5);
           set_st("fp6",p6);
          };

 event= Diminui_janela(EXP,lambda4)
  condition= ((fp5 > 0) && (fp7 > 0))
  action= {
           int p3,p5,p7;
           p7=fp7;
           p5=fp5;
           p3=fp3;
           p5=p5-1;
           p7=p7-1;
           p3=p3+1;
           set_st("fp3",p3);
           set_st("fp5",p5);
           set_st("fp7",p7);
           };

event= Msg_perdida(EXP,lambda8)
  condition= (fp4 > 0)
  action= {
           int p4,p8;
           p4=fp4;
           p8=fp8;
           p4=p4-1;
           p8=p8+1;

           set_st("fp4",p4);
           set_st("fp8",p8);
          };

event= Msg_OK(EXP,lambda7)
  condition= ((fp4 > 0) && (fp2 > 0))
  action= {
           int p2,p4,p9;
           p2=fp2;
           p4=fp4;
           p9=fp9;
           p2=p2-1;
           p4=p4-1;
           p9=p9+1;

           set_st("fp4",p4);
           set_st("fp2",p2);
           set_st("fp9",p9);
          };

event= Envia_sinal_de_perda(EXP,lambda6)
  condition= (fp8 > 0)
  action= {
           int p8,p7;
           p8=fp8;
           p7=fp7;
           p8=p8-1;
```

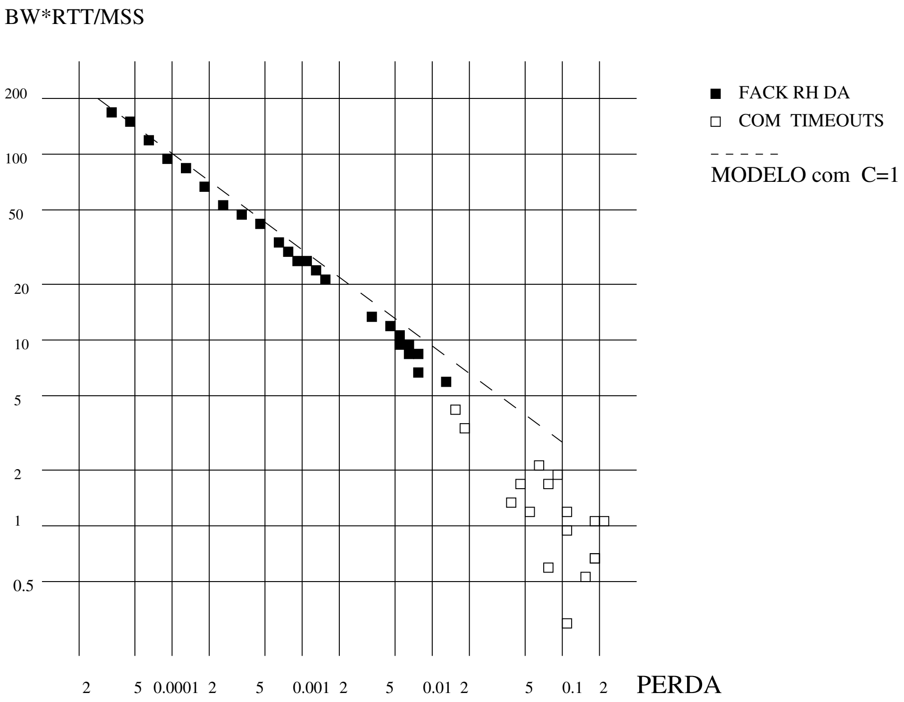

Figura 5.8: Curva Janela versus *perda* para o modelo matemático dado pela equação A.10

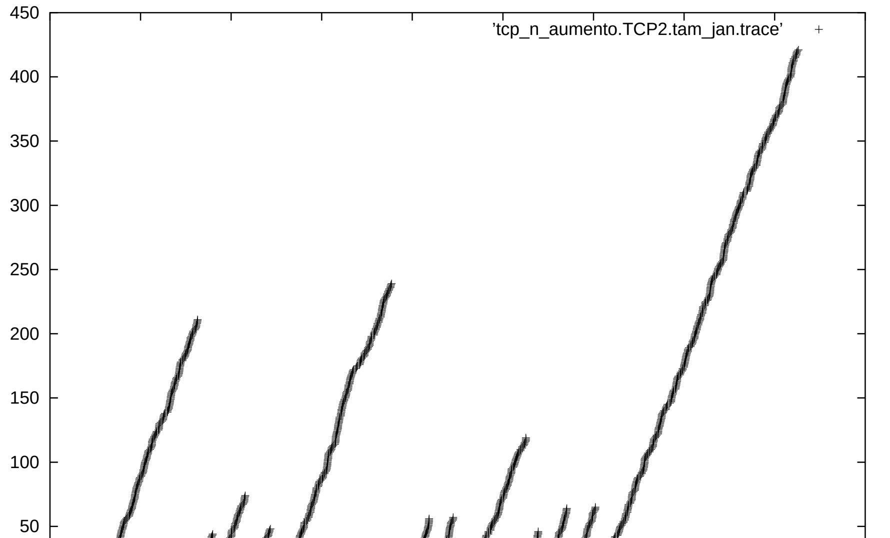

Figura 5.9: Evolução da Janela de congestionamento com probabilidade de perda com distribuição exponencial

Capítulo 6

Conclusões

A necessidade da constante interferência do projetista no processo de decisão da partição *hardware/software* de um protocolo de comunicação, utilizando apenas sua experiência, torna a escolha da partição *hardware/software* muito subjetiva e o processo de implementação muito lento e mais sujeito a erros. É importante que este processo seja automatizado, para que o projetista possa tomar decisões precisas baseado em medidas e critérios objetivos. Com esta finalidade a primeira contribuição deste trabalho é desenvolver uma metodologia para a automatização do processo de escolha da partição *hardware/software* no projeto de protocolos de comunicação baseada nos conceitos de *hardware/software codesign*, na modelagem por redes de Petri, em uma ferramenta de modelagem e análise de desempenho de sistemas e em um método de otimização baseado em um algoritmo genético, que minimiza uma função objetivo construída a partir do desempenho, especificado através de medidas de **Qos**, e do custo desejado para o protocolo. O custo do *hardware* é analisado por uma ferramenta de síntese de *hardware*. O custo do *software* poderia ser fixado como o valor homem-hora dispendido para o seu desenvolvimento, mas é considerado desprezível. O resultado fornecido pela metodologia possibilita o estabelecimento de um critério de partição *hardware/software* objetivo.

A segunda contribuição deste trabalho é o estabelecimento de um critério para relacionar a saída do AG, ou seja, um conjunto de taxas ótimo, com uma partição *hardware/software* que atenda às especificações de custo e desempenho estabelecidas pelo projetista. Foi estabelecido um critério de partição que se mostrou coerente, ao longo dos exemplos realizados. No entanto, este critério ainda deve ser melhor investigado nas próximas aplicações da metodologia a outros protocolos.

A terceira contribuição deste trabalho foi o desenvolvimento do algoritmo que implementa a metodologia. Devido aos diferentes graus de sensibilidade de cada transição do modelo, ou seja, devido a alguns parâmetros influenciarem mais o valor da função objetivo do que outros, optou-se por se utilizar um algoritmo genético híbrido que varia a faixa de valores possíveis de cada parâmetro conforme a sua sensibilidade. Este algoritmo associa técnicas tradicionais, que fazem uso de derivadas para determinar as faixas de variação permitidas para cada taxa de transição do protocolo, a um algoritmo genético simples. Isto possibilita uma convergência mais rápida do algoritmo genético para o erro especificado para a função objetivo e não provoca uma convergência prematura para um mínimo local.

Apesar do algoritmo seguir uma estrutura básica de um algoritmo genético simples, foi necessário um grande desenvolvimento do mesmo para adaptá-lo à metodologia. Pode-se citar a integração do AG com o Tangram-II como uma contribuição importante.

O custo da implementação em *hardware* pode ser significativamente reduzido caso sejam utilizados dispositivos lógicos programáveis (PLD's), que aproximam o custo do *hardware* com o custo do *software*, pois o *hardware* pode ser desenvolvido e testado em um programa comercial desenvolvido com esta finalidade (por ex., o ALTERA), e sintetizado em *chips* de baixo custo, ao contrário dos FPGA's e ASIC's.

Apesar da metodologia apresentada ter sido direcionada para aplicação em projeto de protocolos, ela é totalmente genérica, de alto nível de abstração e pode ser aplicada a qualquer sistema que seja modelado de forma simples numa rede de Petri. Caso esta tarefa seja difícil, pode-se modelar o sistema diretamente no Tangram-II e continuar o processo a partir desta especificação. A metodologia pode ser utilizada em vários níveis, dependendo da *granularidade* que se deseje: no nível da rede, no nível do protocolo ou mesmo no nível de uma transição. Particularmente no último caso, pode haver necessidade de reparticionar um conjunto de operações representado por uma transição, dividindo-a em várias outras, caso seja verificado na etapa do projeto de *hardware* que uma operação consome, por exemplo, 90% do tempo da transição. Ou seja, o fato de uma transição ter sido selecionada para implementação em *hardware* não significa que **todas as operações** relativas àquela transição serão implementadas em *hardware*.

A literatura disponível sobre a automatização do processo de partição *hardware/software* ainda é muito pobre. Apenas Hidalgo [3] faz uma tentativa semelhante ao processo descrito neste trabalho sem, no entanto, utilizar uma ferramenta de análise de desempenho que forneça medidas **durante** o processo de otimização, o mesmo acontecendo para o custo do *hardware*.

O programa da ferramenta Tangram-II que executa o método escolhido para a solução da cadeia de Markov (**solv**) é bastante rápido e apesar de poder ser chamado até centenas de milhares de vezes durante a execução do algoritmo genético, dependendo do tamanho da população, do número de pontos a serem aproximados e do número de gerações, o algoritmo é executado no máximo em algumas horas, no sistema operacional Linux e num microcomputador Pentium-II 400 MHZ com 128MB de memória RAM. Este é um tempo curto se considerarmos todo o processo de síntese do protocolo, principalmente pelo tempo economizado pelo projetista na escolha da melhor partição *hardware/software*. Anteriormente, observou-se que o mesmo programa executado numa estação ULTRA 1 e sistema operacional **Solaris** demorava alguns dias para executar completamente.

A ferramenta Tangram-II se mostrou eficiente no que se refere à facilidade de modelar a rede de Petri estocástica original, gerar a cadeia de Markov associada e tempo de solução da mesma. As simulações realizadas no segundo exemplo do capítulo 5 foram executadas em alguns segundos. Presume-se que para protocolos mais complexos estas simulações sejam mais demoradas, mas não o suficiente que interfira no andamento do projeto.

A escolha do método de otimização baseado em algoritmos genéticos, devido aos motivos já mencionados no capítulo 4, mostrou-se uma escolha acertada pois não é necessária muita precisão no cálculo da função objetivo porque o critério de seleção baseado numa "roleta" é um critério aproximado. Apesar de demandar um custo computacional elevado, o algoritmo genético é simples de ser implementado, não é tendencioso e evita funções de erro mais complexas como, por exemplo, a função exponencial utilizada no método de otimização *Simulated Annealing* [39].

Através dos experimentos descritos no capítulo 5 foi possível comprovar que a metodologia é possível de ser implementada e possui um grande potencial ainda a ser explorado, principalmente em relação à abrangência das aplicações. Os exemplos apresentados tiveram um **caráter acadêmico**, ou seja, avaliar a viabilidade da metodologia e do critério de partição utilizado, além de servirem para depurar o algoritmo genético que foi desenvolvido e integrá-lo ao Tangram-II. Neste particular, os exemplos alcançaram o seu objetivo. Pôde-se perceber, no entanto, que o protocolo TCP não é um bom candidato para se aplicar esta metodologia. Isto porque o desempenho do protocolo TCP (e seu controle de congestionamento) está diretamente ligado ao número de pacotes perdidos **na rede**. Portanto, mesmo que o protocolo fosse implementado **totalmente** em *hardware* não seria possível garantir um alto desempenho, devido às limitações impostas pelo tráfego **da rede**.

A partir do parágrafo anterior, chega-se à conclusão de que os protocolos que mais se aproveitam da metodologia apresentada são aqueles que possuem um alto grau de processamento nos nós e cujo desempenho está diretamente relacionado à velocidade desse processamento. Entre os protocolos que se pretende estudar no prosseguimento deste trabalho encontram-se os de recuperação de erros para transmissão "Multicast" confiável [42][43] e protocolos relacionados com criptografia de dados [44][45].

Neste trabalho procurou-se avaliar, fundamentalmente, o potencial da metodologia e sua praticidade. Agora que a implementação da metodologia já está sedimentada, pretende-se, em trabalhos futuros, dar maior ênfase à elaboração de uma função objetiva mais complexa, que represente de forma mais exata tanto os parâmetros de qualidade de serviço quanto os parâmetros de custo, pois até o presente momento a otimização baseou-se somente no desempenho desejado. A integração entre a parte do algoritmo correspodente à análise das sensibilidades das taxas e o algoritmo genético propriamente dito ainda será realizada. Esta primeira parte, a determinação das faixas de variação das taxas, ainda está sendo realizada de forma manual. Também será realizada a integração do AG com a ferramenta de síntese de Hardware Synopsys para que se possa demonstrar, com uma experiência piloto completa, a eficiência do processo de automatização, através de medidas de custo e desempenho realizadas sobre o protocolo implementado e sintetizado.

Referências Bibliográficas

[1] D. Gajski e F. Vahid, "Specification and design of embedded software/hardware systems", *IEEE Design and Tests of Computer*, vol. 12, no. 1, pp. 53–67, janeiro 1995.

[2] S. Fischer, J. Wytrebowicz e S. Budkowski, "Hardware/software co-design of communication protocols", in *Proceedinds of IEEE 22nd Euromicro Conference*, 1996.

[3] J. I. Hidalgo e J. Lanchares, "Functional partitioning for hardware/software codesign using genetic algorithm", in *Proceedings of the 23rd Euromicro Conference*, 1997.

[4] S. D. Scott, A. Samal e S. Seth, "Hga: A hardware-based genetic algorithm", in *Proceedings of 1995 ACM/SIGDA Third International Symposium on Field Programmable Gate Arrays*, 1995.

[5] H. Choi, V. Kulkarni e K. Trivedi, "Markov regenerative stochastic petri nets", *Performance Evaluation*, vol. 20, no. 1, pp. 337–357, maio 1994.

[6] R. German e C. Lindemann, "Analysis of stochastic petri nets by the method of supplementary variables", *Performance Evaluation*, vol. 20, no. 1, pp. 317–335, maio 1994.

[7] R. Carmo, L. Carvalho, E. Sousa e Silva, M. Diniz e R. Muntz, "Performance/availability modeling with the Tangram-II modeling environment", *Performance Evaluation*, vol. 33, no. 1, pp. 45–65, junho 1998.

[8] D. R. Figueiredo, "Um módulo de simulação da ferramenta Tangram-II: suporte para medidas com recompensa, recursos de eventos raros e aplicações a modelos de rede multimídia", Tese de Mestrado, Programa de Engenharia de Sistemas - COPPE/UFRJ, julho 1999.

[9] A. P. C. Silva, "Tangram-ii user's manual", tech. rep., Universidade Federal do Rio de Janeiro, outubro 2000. `http://www.land.ufrj.br`.

[10] M. Mitchell, *An Introduction to Genetic Algorithms.* MIT Press, 1996.

[11] H. Horner, "A C++ class library for genetic programming: The Vienna University of Economics - genetic programming kernel", *Internet Draft*, maio 1996. http://www.wu-wien.ac.at/usr/h88/h8850092.

[12] L. Pirmez, *Uma metodologia para síntese de alto nível de protocolos a partir de uma descrição formal.* PhD thesis, Programa de Engenharia Elétrica - COPPE/UFRJ, 1996.

[13] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen e A. Haxthausen, "Lycos: the lyngby co-synthesis system", *Design Automation for Embedded Systems*, vol. 2, no. 2, pp. 1–43, fevereiro 1997.

[14] F. Vahid e T. D. Lee, "Towards a model for hardware and software functional partitioning", in *International Workshop on Hardware/Software Codesign*, 1996.

[15] D. Gajski, N. Dutt, A. Wu e S. Lin, *High-Level Synthesis - Introduction to Chip and System Design.* Kluwer Academic Publishers, 1991.

[16] J. Daveau *et al.*, "Cosmos: An sdl based hardware/software codesign environment", *Hardware/Software Codesign and Coverification, Current Issues in Electronic Modelling*, vol. 8, 1997.

[17] J. L. Peterson, "Petri nets*", *Computing Surveys*, vol. 9, no. 3, pp. 223–252, setembro 1977.

[18] M. Diaz, "Modeling and analysis of communication and cooperation protocols using petri net based models", *Computer Networks*, vol. 10, no. 6, pp. 419–441, junho 1982.

[19] S. Kumar *et al.*, "A framework for hardware/software codesign", *IEEE Computer*, vol. 26, no. 12, pp. 39–45, dezembro 1993.

[20] C. N. Coelho Jr. *et al.*, "Redesigning hardware-software systems", in *Third International Workshop on Hardware/Software Codesign*, 1994.

[21] A. Y. Alomary, "A hardware/software codesign partitioner for asip design", in *ICECS'96*, 1996.

[22] R. K. Gupta e G. Micheli, "Hardware-software cosynthesis for digital systems", *IEEE Design and Tests of Computer*, setembro 1993.

[23] R. K. Gupta, C. N. Coelho e G. Micheli, “Program implementation schemes for hardware-software systems”, *IEEE Computer*, janeiro 1994.

[24] R. Ernst, J. Henkel e T. Benner, “Hardware-software cosynthesis for microcontrollers”, *IEEE Design and Tests of Computer*, dezembro 1993.

[25] N. S. Woo, A. E. Dunlop e W. Wolf, “Codesign from cospecification”, *IEEE Computer*, janeiro 1994.

[26] M. K. Molloy, “Performance analisys using stochastic petri nets”, *IEEE Transactions on Computers*, vol. 31, no. 9, pp. 913–917, setembro 1982.

[27] S. M. Ross, *Stochastic Processes.* John Wiley & Sons, 1983.

[28] F. Vahid, T. D. Le e Y. Hsu, “A comparison of functional and structural partitioning”, in *International Symposium on System Synthesis*, 1996.

[29] A. S. Tanenbaum, *Computer Networks.* Prentice Hall, 1996.

[30] W. Stevens, “Tcp slow start, congestion avoidance, fast retransmit and fast recovery algorithms”, *Internet RFC 2001*, janeiro 1997.

[31] D. E. Comer, *Internetworking with TCP/IP - vol.1.* Prentice Hall, 1995.

[32] L. L. Peterson e B. S. Davie, *Computer Networks: A Systems Approach.* Morgan Kaufmann Publishers, 1996.

[33] M. Schwartz, *Broadband Integrated Networks.* Prentice Hall, 1996.

[34] W. Leland *et al.*, “On the self-similar nature of ethernet traffic (extended version)”, *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, fevereiro 1994.

[35] W. Willinger *et al.*, “Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level”, *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 71–86, fevereiro 1997.

[36] W. C.-W. Cheng, “The tangram graphical interface facility (tgif) manual”, *Internet Draft*, 2000. `http://bourbon.cs.ucla.edu:8801/tgif/`.

[37] J. Koza, *Genetic Programming II.* MIT Press, 1994.

[38] D. Whitley, “The genitor algorithm and selective pressure”, in *Proceedings of 3rd International Conference on Genetic Algorithms*, 1989.

[39] A. Laarhoven, *Simulated Annealing: theory and applications.* D. Reidel, 1987.

[40] M. N. de Miranda, "Aproximação de distribuições de probabilidades não exponenciais por distribuições do tipo fase", Tese de Mestrado, Instituto Nacional de Pesquisas Espaciais - INPE, outubro 1996.

[41] M. Mathis, J. Semke e J. Mahdavi, "The macroscopic behavior of the tcp congestion avoidance algorithm", *Computer Communication Review*, vol. 27, no. 3, pp. 1–16, julho 1997.

[42] J. Nonnenmacher, E. Biersack e D. Towsley, "Parity based loss recovery for reliable multicast transmission", in *Proceedings of ACM SIGCOMM'97*, 1997.

[43] S. Pingali, D. Towsley e J. Kurose, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols", in *ACM SIGMETRICS'94*, 1994.

[44] M. Abadi, "Two facets of authentication", in *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, 1998.

[45] M. Abadi, "Security protocols and specifications", in *In Foundations of Software Science and Computation Structures: Second International Conference, FOSSACS '99*, 1999.

Apêndice A

Método do Gradiente para a Determinação das Faixas dos Parâmetros

Dada uma função escalar de um vetor:

$$f(x) = 0 \tag{A.1}$$

onde $x \in R^n$ e:

$$f : R^n \longrightarrow R \tag{A.2}$$

Seja x^k o valor de x na iteração k. Utilizando o desenvolvimento da função em série de Taylor em torno de x^k, obtém-se:

$$f(x^{k+1}) = f(x^k) + \nabla^t f(x^k)(x^{k+1} - x^k) + ... \tag{A.3}$$

onde:

$$\Delta x = x^{k+1} - x^k \tag{A.4}$$

e

$$\nabla^t f = [\frac{\partial f}{\partial x_1} \;\; \frac{\partial f}{\partial x_2} \;\; ... \;\; \frac{\partial f}{\partial x_n}] \tag{A.5}$$

Truncando-se a série nos termos de primeira ordem e igualando a zero, o valor de x^{k+1} pode ser obtido por:

$$\nabla^t f(x^k).(x^{k+1} - x^k) = -f(x^k) \tag{A.6}$$

e

$$\nabla^t f(x^k).\nabla f(x^k).(x^{k+1} - x^k) = -f(x^k).\nabla f(x^k) \tag{A.7}$$

$$x^{k+1} = x^k - \frac{\nabla f(x^k)}{< \nabla f(x^k).\nabla f(x^k) >}.f(x^k) \tag{A.8}$$

onde

$$\frac{\nabla f(x^k)}{< \nabla f(x^k).\nabla f(x^k) >} \tag{A.9}$$

é a inversa generalizada de $\nabla f(x^k)$. Nos problemas de otimização, o que se deseja determinar é o mínimo (máximo) de uma função, ou seja:

$$\min_x f(x)$$

. No caso particular deste trabalho, a função f é a função objetivo a ser otimizada pelo algoritmo genético, chamada de $\varepsilon(\lambda)$, onde λ é o vetor das taxas da rede de Petri que modela o protocolo. Neste caso, o valor de $\varepsilon(\lambda)$ no ponto ótimo não é necessariamente nulo, logo a equação A.8 deve ser modificada para:

$$\lambda^{k+1} = \lambda^k - \alpha.\frac{\nabla\varepsilon(\lambda^k)}{< \nabla\varepsilon(\lambda^k).\nabla\varepsilon(\lambda^k) >}.\varepsilon(\lambda^k) \tag{A.10}$$

onde $\alpha \in (0, 1]$ é um fator de amortecimento. Dado que no processo de otimização não se conhece o valor analítico de $\nabla\varepsilon$ utiliza-se uma aproximação discreta. No contexto deste trabalho, as componentes do gradiente são denominadas de *sensibilidades*, sendo definidas, no caso geral, por:

$$S^{\varepsilon}_{\lambda_i} = \frac{\varepsilon(\lambda_i + \Delta\lambda_i) - \varepsilon(\lambda_i)}{\Delta\lambda_i} \tag{A.11}$$

O vetor de sensibilidades $S^{\varepsilon}_{\lambda}$, dado por:

$$S^{\varepsilon}_{\lambda} = \begin{bmatrix} S^{\varepsilon}_{\lambda_1} \\ S^{\varepsilon}_{\lambda_2} \\ \cdots \\ S^{\varepsilon}_{\lambda_n} \end{bmatrix} \tag{A.12}$$

é utilizado em substituição ao gradiente. O módulo do vetor de sensibilidades é dado por:

$$\|S_\lambda^\varepsilon\|^2 = (S_\lambda^\varepsilon)^t.S_\lambda^\varepsilon \tag{A.13}$$

Pela série de Taylor, uma variação de $\delta\lambda_i$ no parâmetro λ_i produzirá uma variação no erro dada por:

$$\varepsilon({\lambda_i}^k + \delta{\lambda_i}^k) \backsimeq \varepsilon({\lambda_i}^k) + S_{{\lambda_i}^k}^\varepsilon.\delta{\lambda_i}^k \tag{A.14}$$

Se $S_{\lambda_i}^\varepsilon < 0$ isto implica que $\varepsilon(\lambda_i^k + \delta\lambda_i^k)$ pode ser feito nulo para um valor $\delta\lambda_i^k$ positivo dado pela equação A.10 com $\alpha = 1$, ou seja:

$$\lambda_i^{k+1} = \lambda_i^k - \frac{S_{\lambda_i^k}^\varepsilon}{\|S_{\lambda^k}^\varepsilon\|^2} \cdot \varepsilon(\lambda_i^k) \tag{A.15}$$

Supondo que o limite de variação de λ_i na iteração k é dado por FA_i^k, adotou-se a heurística de estender a faixa de variação de λ_i na iteração $k+1$ até o limite:

$$FA_i^{k+1} = FA_i^k - \frac{\varepsilon(\lambda_i)}{\|S_{\lambda^k}^\varepsilon\|^2} \cdot S_{\lambda_i}^\varepsilon \tag{A.16}$$

onde FA_i^k é a faixa de variação dos parâmetros na iteração k e FA_i^{k+1} é a faixa de variação na iteração $k+1$. Note que no caso de $S_{\lambda_i}^\varepsilon > 0$ o limite será reduzido. Após a determinação das faixas de variação ótimas para os parâmetros executa-se o programa de AG da seção 4.5.

Resumindo o algoritmo:

- Estabelecer uma faixa inicial única e pequena para todos os parâmetros;
- Executar o AG e ao fim de 10 gerações verificar o erro;
- Variar cada uma das taxas de 1%, uma de cada vez, e avaliar o valor de erro produzido por cada parâmetro;

- A partir dos valores obtidos, calcular as sensibilidades em relação a cada parâmetro, segundo a equação A.11;
- Calcular o módulo do vetor de sensibilidades a partir da equação A.13;
- Calcular a nova faixa de variação de cada parâmetro a partir da equação A.16
- Repetir as operações anteriores até que todos os valores calculados para as sensibilidades não alterem mais a faixa corrente