Mestrando: Wellington Carlos Lopes

Dissertação de mestrado
Análise de Desempenho do Protocolo CAN para
Aplicação na Área Agrícola Utilizando Redes de
Petri Coloridas

São Carlos, fevereiro de 2007

WELLINGTON CARLOS LOPES

**Análise de Desempenho do Protocolo CAN para Aplicação na
Área Agrícola Utilizando Redes de Petri Coloridas**

Dissertação apresentada à Escola de Engenharia de
São Carlos da Universidade de São Paulo, como
parte dos requisitos para a obtenção do título de
Mestre em Engenharia Mecânica.

ORIENTADOR: Prof. Dr. Ricardo Yassushi Inamasu

São Carlos

2007

DEDICATÓRIA

Aos meus amados pais Carlos e Sandra, pelo grande incentivo nos estudos, pelo amor, carinho e dedicação.

Ao amigo André Torre Neto, pela amizade, pela confiança depositada em mim e por ter sido um grande incentivador.

Ao amigo Rafael Vieira de Sousa, pela amizade, por sua grande colaboração no trabalho e por ter sido minha referência desde o início do trabalho.

AGRADECIMENTOS

Ao Pai Celestial, por sua constante presença em meu coração.

Aos meus pais e ao meu irmão, pelo seu apoio, amor, carinho e esperança depositados em mim em todos os momentos.

Ao meu orientador Dr. Ricardo Yassushi Inamasu, pela amizade, confiança no trabalho e dedicação nas orientações.

Ao amigo Rafael Vieira de Sousa, pelo seu apoio e incentivo, que foram fundamentais para os bons resultados desse trabalho.

Aos meus grandes amigos, por sua amizade e presença em todas as horas.

Ao Amigo Hilano José Rocha de Carvalho, por sua grande colaboração nesse trabalho.

Aos professores, pesquisadores, funcionários do Departamento de Engenharia Elétrica da EESC USP, pelo apoio recebido.

A todos os amigos do Laboratório de Simulação, pelo companheirismo e pelo trabalho em equipe.

Ao professor Dr. Adenilso da Silva Simão, pela cooperação.

Aos Amigos Ferdinando Monsignore, Luciano Vieira Koengnikan, Maurício Fernando Lima Pereira, pela amizade, colaboração e presença em todas as horas.

A Mariana Henrique Passoni, por seu amor e companheirismo.

Ao amigo Fábio Girardi, pela amizade e incentivo.

RESUMO

LOPES, W.C. (2007), **Análise de Desempenho do Procolo CAN para Aplicação na Área Agrícola utilizando Rede de Petri Colorida**. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007.

Empresas e instituições de pesquisa nacionais e internacionais tem concentrado esforços no sentido de desenvolver e promover a implementação do padrão internacional ISO 11783, baseado no protocolo de comunicação digital CAN (Contoller Network Area), para integração de dispositivos em máquinas e implementos agrícolas. Nesse contexto, é apresentado o desenvolvimento e a aplicação de um modelo baseado no formalismo das redes de Petri como uma ferramenta de suporte para a análise e implantação de uma rede CAN em aplicações com tempos-críticos. O modelo foi elaborado no ambiente para edição e simulação DesignCPN, sendo que a simulação é utilizada para a análise do desempenho e do comportamento de uma rede CAN com diferentes configurações que será embarcada em um veículo autônomo agrícola (VAA).

Palavras-Chave: Automação Agrícola; Controller Area Network(CAN); Rede de Petri

ABSTRACT

LOPES, W.C. (2007), **Analysis of Performance of protocol CAN for Agricultural Field Application using Coloured Petri Net**. M.Sc. Dissertation - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007.

International and national research institutions and companies have focused efforts to develop and promote the implementation of the international standard ISO 11783, based upon the digital communication protocol CAN (Controller Area Network), in order to integrate devices in machines e agricultural implements. Therefore, it is presented the development and the application of a model based on Petri nets' formalism as a tool for the support, analysis and accomplishment of a CAN bus in time-critical applications. The model was constructed in an environment for edition and simulation of Colored Petri nets, the software DesignCPN. The simulation was used for the analysis of the behavior and the performance the CAN bus for several configurations, which will be embedded in an Autonomous Agricultural Vehicle (VAA).

Keywords: agricultural automation; controller area network; petri net

LISTA DE FIGURAS

3.1	Elementos Básicos da Rede de Petri	31
4.1	Interface do Software de Simulação HpSIM	35
4.2	Interface do Software Visual Object ++	36
4.3	Simulação Interativa do Software Design CPN	42
5.1	Modelo de uma Rede CAN representando apenas 1 dispositivo	48
5.2	Modelo de uma Rede CAN com 3 dispositivos	50
5.3	Modelo de Rede de Petri Colorida - Legenda na Tabela 5.1	52
5.4	Estrutura do VAA visto de Cima	62
5.5	Estrutura do VAA visto de Frente	62
5.6	Linha de Plantio na Citricultura	63
5.7	Barramento CAN	64
6.1	Taxa de Ocupação do Barramento CAN - Simulação 1	68
6.2	Taxa de Ocupação do Barramento CAN - Simulação 1	69
6.3	Quantidades de Mensagens Enviadas por Dispositivos - Simulação 1	69
6.4	Quantidade de Mensagens Enviadas por Dispositivos - Simulação 2	72
6.5	Taxa de Ocupação do Barramento por Dispositivos - Simulação 2	73
6.6	Quantidades de Mensagens Enviadas por Dispositivos - Simulação 2	74
6.7	Comparativo da quantidade de Mensagens Enviadas	75

6.8	Comparativo da Taxa de Ocupação do Barramento	75
6.9	Configuração de uma Rede CAN (HOFSTEE; GOENSE, 1999)	77
6.10	Taxa de Ocupação do Barramento - Simulação 3	79
6.11	Taxa de Ocupação do Barramento por Dispositivo - Simulação 3	79
6.12	Quantidade de Mensagens Enviadas por Dispositivo - Simulação 3	80
7.1	Tensão dos Cabos	93
7.2	Rede de Comunicação	93
7.3	Metodo de Arbitragem CAN	95
7.4	Pacote de Dados: CAN 2A	97
7.5	Mensagem CAN 2B	99
7.6	Modelo de Rede de Petri (a) - (MURATA, 1989)	107
7.7	Modelo de Rede de Petri (b) - (MURATA, 1989)	107
7.8	Modelo de Rede de Petri Colorida (a)	109
7.9	Modelo de Rede de Petri Colorida (b)	109
7.10	Modelo de Rede de Petri para um sistema de fabricação flexível com marcas iniciais - (DESROCHERS; AL-JAAR, 1995)	113
7.11	Modelo de Rede de Petri Colorida para um Sistema de Produção Flexível - (DESROCHERS; AL-JAAR, 1995)	114

LISTA DE TABELAS

3.1	Tamanho do Barramento CAN vs. Taxa transmissão	29
5.1	Descrição dos elementos gráficos do Modelo	53
5.2	Estrutura do Arquivo de Análise do Jiter	61
6.1	Configuração da Rede CAN - Simulação 1	68
6.2	Tempo de Espera das Mensagens - Simulação 1	70
6.3	Configuração da Rede CAN - Simulação 2	72
6.4	Tempo de Espera das Mensagens - Simulação 2	74
6.5	Comparativo Tempo de Espera das Mensagens Simulação 1 Vs Simulação 2 . .	76
6.6	Configuração da Rede CAN - Simulação 3	78
6.7	Tempo de Espera das Mensagens - Simulação 3	81
7.1	Normas CAN e suas Áreas de Aplicações	92
7.2	Relação entre PN e CPN	115

LISTA DE ABREVIATURAS E SIGLAS

CAN	Controller Area Network
CPN	Coloured Petri Net
CSMA/CD with NDA	Carrier Sense Multiple Access Collision Detect with Non-Destructive Arbitration
ECU	Electronic Control Unit
GPS	Global Position System
ISO	International Organization of Standardization
PN	Petri Net
SAE	Society of Automotive Engineers
TTPN	Timed Transition Petri Net
VAA	Veículo Agrícola Autônomo

SUMÁRIO

1	Introdução	13
1.1	Motivação	13
1.2	Objetivo	16
1.3	Estrutura do Trabalho	17
2	Revisão Bibliográfica	19
2.1	Controller Area Network (CAN)	19
2.2	Redes de Petri	24
3	Revisão Teórica	27
3.1	CAN - Controller Area Network	27
3.2	Rede de Petri	30
4	Materiais e Métodos	33
4.1	Softwares de Simulação	33
4.1.1	HPSim	34
4.1.2	Visual Object Net ++	35
4.1.3	TimeNet	37
4.1.4	GreatSPN	38
4.1.5	DesignCPN	40

4.1.5.1	Editor	41
4.1.5.2	Simulador	41
4.1.6	Análise de Desempenho	43
4.1.7	Linguagem CPN-ML	43
4.1.8	O padrão ML	44
4.2	Modelos de Simulação	45
5	Desenvolvimento do Trabalho	47
5.1	Desenvolvimento do Modelo de Rede de Petri	47
5.1.1	Modelo de Rede de Petri Estocástica	47
5.1.2	Modelo de Rede de Petri de Alto-Nível	51
5.1.2.1	Coleta das Informações Durante a Simulação	56
5.1.2.2	Quantidade de mensagens geradas por dispositivo	57
5.1.2.3	Quantidade de Mensagens no Buffer	58
5.1.2.4	Quantidade de Acessos ao Barramento	59
5.1.2.5	Intervalo Médio de Acessos (<i>Jitter</i>)	60
5.2	Ambiente Simulado: Aplicação da Rede CAN no Veículo Agrícola Autônomo .	61
5.2.1	Veículo Agrícola Autônomo - VAA	61
5.2.2	Barramento CAN	63
6	Resultados e Discussões	67
6.1	Configuração 1	67
6.1.1	Taxa de Ocupação do Barramento	67
6.1.2	Taxa de Ocupação do Barramento por Dispositivo	69

6.1.3	Quantidade de Mensagens Enviadas por Dispositivo	69
6.1.4	Quantidade de Mensagens no Buffer x Tempo de Espera	70
6.2	Configuração 2	71
6.2.1	Taxa de Ocupação do Barramento	71
6.2.2	Taxa de Ocupação do Barramento por Dispositivo	73
6.2.3	Quantidade de Mensagens Enviadas por Dispositivo	73
6.2.4	Quantidade de Mensagens no Buffer x Tempo de Espera	74
6.3	Comparativo entre as Configurações 1 e 2	75
6.4	Configuração 3	76
6.4.1	Taxa de Ocupação do Barramento	77
6.4.2	Taxa de Ocupação do Barramento por Dispositivo	79
6.4.3	Quantidade de Mensagens Enviadas por Dispositivo	80
6.4.4	Quantidade de Mensagens no Buffer x Tempo de Espera	80
7	Conclusões e Sugestões	83
7.0.5	Conclusões	83
7.0.6	Sugestões para Trabalhos Futuros	84
	Referências	85
	ANEXO A - Controller Area Network(CAN)	89
	O Barramento CAN e o Modelo de Rede ISO-OSI	91
	Camada Física	92
	Camada de Acesso ao Meio	93

Método de Arbitragem	94
Formatos das Mensagens CAN	96
Quadro de Dados	96
Quadro Remoto	99
Quadro de Erros	100
Quadro de Sobrecarga	100
Detecção de Falhas na Comunicação	101

ANEXO B - Rede de Petri **105**

Conceitos Básicos de Rede de Petri	107
Rede de Petri Colorida e Definições	108

1 INTRODUÇÃO

Esse capítulo divide-se em 3 seções que apresentam a motivação (seção 1.1), a estrutura da dissertação (seção 1.3) e os objetivos propostos (seção 1.2).

1.1 Motivação

O propósito da norma ISO 11783 é prover um padrão aberto para interconexão de sistemas eletrônicos embarcáveis através de um Barramento que, por sua vez, é um conjunto formado por fios, conectores e dispositivos de potência para promover a interconexão de dispositivos e permitir a comunicação de dados entre estes. A camada de enlace de dados da norma ISO 11783 adota a versão CAN2.0B do protocolo CAN, como especificado pela ISO 11898, Extended CAN (CAN Estendido) Sousa (2002). A norma ISO 11783, utilizada como referência no desenvolvimento desse trabalho, foi especificada para a área agrícola e outras normas foram desenvolvidas para suportar diferentes aplicações do protocolo CAN.

Outra norma bastante utilizada é a SAE J1939, que foi desenvolvida nos anos 90 pela Sociedade de Engenharia Automotiva (*Society of Automotive Engineers*) com o objetivo de criar um padrão de comunicação e controle para redes CAN aplicadas em veículos pesados, conectando *ECU's* em sistemas utilizados em caminhões, ônibus e carretas. Essa norma, a princípio dedicada para aplicações com motores a diesel, define formato das mensagens de do sistema de freio, transmissão e motor.

Outras especificações basearam-se em conceitos e funções da SAE J1939, como por exemplo: NMEA 2000, ISO 11992 e a também a ISO 11783.

Um dos principais objetivos do desenvolvimento dessas normas para aplicações específicas, senão o principal, é fornecer compatibilidade e interoperabilidade entre diferentes equipamentos, ou ECU's de diferentes fabricantes. Dessa maneira qualquer equipamento que siga a norma de uma rede CAN em funcionamento pode ser adicionado à rede de comunicação sem a necessidade de reconfiguração.

A crescente popularidade de aplicações com o CAN em sistemas de automação e controle é sustentada, entre outros fatores, por este possuir características muito próprias para tais sistemas em relação a outros padrões. Entretanto, uma dificuldade de implementação de uma rede CAN é a definição dos endereços dos nós na rede, que estão diretamente associados às prioridades de acesso à mesma. Essa complexidade aumenta proporcionalmente com a quantidade de sensores instalados que compartilham o mesmo barramento. Assim, quanto maior o número de dispositivos, maior a dificuldade de definição dos endereços de identificação na rede.

A definição dos endereços dos dispositivos para aplicações que utilizam a norma ISO 11783 pode ser considerada mais complexa pois de acordo com a especificação, todos os endereços da rede são fixos e não podem ser alterados durante o funcionamento da rede. A norma J1939 permite que os endereços dos dispositivos sejam dinâmicos e possam ser alterados durante o funcionamento da rede, o que permite maior flexibilidade durante inserção ou exclusão de dispositivos da rede de comunicação.

Pode-se dividir os sensores utilizados em uma rede de automação em dois grupos: sensores cíclicos e sensores acíclicos. Os sensores cíclicos acessam o barramento com frequências definidas como, por exemplo: receptores GPS (*Global Positioning System*) que podem ser configurados normalmente para disponibilizar dados em intervalos de 1 ou 0,1 segundo. Os sensores acíclicos acessam o barramento com frequência indefinida, como por exemplo, sensores de presença, que podem ser configurados para acessar o barramento quando os sensores estiverem a uma determinada distância do objeto ou obstáculo.

Métodos de análises de desempenho das redes CAN estão presentes em várias áreas de aplicação, mas ainda não foram formalizadas para aplicações agrícolas. Para uma correta operação, o sistema deverá garantir numa situação ideal que todos consigam acessar o barra-

mento, evitando que o dispositivo que perdeu uma disputa pelo acesso ao barramento perca a mensagem por sobreposição de dado armazenando em uma memória (*buffer*), ou que algum dispositivo não consiga acesso com a frequência desejada pelo sistema de controle. Assim, mostra-se interessante a proposta de estudo para situações como a presença de *Jitter* (tempo médio de espera de uma mensagem na fila de transmissão) no conjunto de mensagens, a alta carga de mensagens sob uma rede e a consideração da viabilidade de aplicação de uma rede CAN em determinadas aplicações agrícolas.

As redes de Petri constituem um formalismo matemático e gráfico bastante eficaz na modelagem de sistemas que apresentam atividades concorrentes e competição na utilização de recursos limitados (MURATA, 1989). O trabalho de Cavalieri *et al.* (1996) aborda com sucesso a utilização de redes de Petri estocásticas na modelagem de uma rede CAN, gerando uma ferramenta de análise para auxiliar o projeto de rede CAN.

Neste trabalho, utiliza-se a abordagem de Cavalieri *et al.* (1996) como orientação ao desenvolvimento de um modelo de rede CAN. Diferencia-se porém por utilizar o formalismo da Rede de Petri Colorida e não o formalismo de Rede de Petri Estocástica Temporizada. A utilização desse tipo de rede de Petri amplia as possibilidades de modelagem, pois permite distinguir os elementos de acordo com suas funções específicas no processo de modelagem e da disponibilidade de um programa de edição e simulação de redes de Petri coloridas que permite a análise de desempenho dos modelos obtidos.

O trabalho fornece suporte a um projeto cooperativo na área relacionada à montagem e análise de uma rede *fieldbus* embarcada (o termo embarcado representa todo e qualquer sistema eletro-eletrônico montado em uma aplicação móvel) baseada no protocolo de comunicação digital CAN para integrar diferentes dispositivos para o sistema de um veículo agrícola autônomo (VAA).

Modelagens desenvolvidas para avaliação de desempenho baseadas em sistemas técnicos são consideradas uma maneira poderosa e que exigem o suporte de poucos equipamentos eletrônicos sofisticados para prever o desempenho e o comportamento desses sistemas antes da execução real, e sua importância aumenta de acordo com a complexidade das

aplicações.

1.2 Objetivo

O objetivo desse trabalho é criar um modelo capaz de simular o funcionamento de redes CAN utilizando Rede de Petri Colorida. Devido a indisponibilidade de ferramentas de apoio para desenvolvimentos de projetos e análises de redes baseadas no protocolo CAN, espera-se que esse modelo seja uma ferramenta de apoio para projetos e análises de redes CAN, e que através da sua utilização seja possível simular a comunicação entre os dispositivos instalados na rede, analisar parâmetros referentes ao desempenho da rede CAN, como: tráfego das informações, quantidade de acessos ao barramento por dispositivo, tempo de espera das mensagens para o envio (*jitter*), quantidade máxima de mensagens no *buffer* aguardando transmissão.

Após o desenvolvimento do modelo de simulação, propõe-se a realização de simulações com configurações dos parâmetros de prioridades diferentes para os dispositivos instalados na rede, com o objetivo de analisar as consequências dessa alteração na rede de comunicação e nos parâmetros de análise.

Além das simulações configuradas com prioridades diferentes para os dispositivos, propõe-se a realização de uma análise da rede de comunicação CAN baseada em outro ambiente para área agrícola. Espera-se também que o modelo desenvolvido seja flexível permitindo que novos nós possam ser inseridos ou excluídos do modelo e também permita realizar alterações das configurações da rede CAN para diferentes ambientes.

Esse trabalho é vinculado ao projeto do VAA de Porto *et al.* (2003), cujo ambiente é módulo integrante para testes e validações do modelo de simulação. Além da análise de desempenho da rede CAN, serão analisados alguns parâmetros referentes ao modelo como, por exemplo: poder de abstração, flexibilidade nas alterações e edições e recursos para a extração dos dados. Espera-se que o modelo desenvolvido possa ser utilizado como ferramenta de apoio em projetos de redes CAN, aplicáveis em diferentes áreas, disponibilizando resultados que contribuam para as definições das frequências de aquisições dos dados e também as prioridades

entre os dispositivos instalados de forma mais precisa.

1.3 Estrutura do Trabalho

A estrutura desse trabalho está dividida em 7 capítulos:

No capítulo 2 são citados trabalhos que possuem abordagens sobre Barramentos CAN, Análises de Desempenho e Redes de Petri, com objetivo de apresentar a literatura atual dessas áreas de pesquisa.

No capítulo 3 são apresentados os conceitos sobre Rede Petri e a comunicação do Barramento CAN. O conhecimento desses conceitos é de fundamental importância para uma maior compreensão desse trabalho .

No capítulo 4 primeiramente é apresentado um comparativo entre 5 softwares de simulações de Rede de Petri diferentes que foram testados durante o desenvolvimento do trabalho. Após esse comparativo, são apresentados dois modelos de simulação que foram desenvolvidos, o primeiro modelo utilizando o formalismo da Rede de Petri Lugar-Transição e o segundo modelo utilizando o formalismo da Rede de Petri Colorida. Por fim, apresenta-se a estrutura e os parâmetros da aplicação do VAA.

O trabalho de desenvolvimento do modelo de simulação utilizando o formalismo da Rede de Petri Colorida é apresentado no capítulo 5.

Todos os resultados e as discussões sobre as análises das redes CAN e suas configurações são apresentados no capítulo 6.

O capítulo 7 apresenta as conclusões e as contribuições desse trabalho.

O anexo A apresenta os conceitos teóricos do protocolo CAN e o anexo B apresenta o formalismo das Rede de Petri Lugar-Transição e Rede de Petri Colorida. O objetivo desses anexos é de servirem de apoio para esclarecimento de dúvidas que possam surgir durante a leitura dessa dissertação e também citar algumas referências que apresentam essas teorias de forma mais completa e detalhada.

2 REVISÃO BIBLIOGRÁFICA

O presente trabalho possui duas vertentes. O protocolo CAN que consiste em um protocolo de comunicação serial e a Rede de Petri que consiste em uma ferramenta de modelagem gráfica e matemática.

Portanto essa revisão bibliográfica foi dividida em duas seções: 2.1 e 2.2.

Na seção 2.1 é apresentada uma revisão bibliográfica sobre o protocolo CAN, citando os principais trabalhos existentes na literatura dos últimos anos sobre o barramento CAN e suas aplicações.

Na seção 2.2 também é apresentada uma revisão bibliográfica sobre Redes de Petri, citando trabalhos que utilizam a Rede de Petri como ferramenta para modelagens e análises de desempenho.

2.1 Controller Area Network (CAN)

Como pode ser notado na literatura, nas últimas duas décadas foram desenvolvidas diversas tecnologias eletrônicas, computacionais e de comunicação. Essas tecnologias possibilitaram um crescimento na utilização de sistemas de controle baseados em redes, normalmente constituídos por dispositivos eletrônicos (sensores, atuadores) e interconectados através de um barramento *fieldbus*.

Recentemente, têm sido introduzidos barramentos de controle baseados em rede, em muitos sistemas com diferentes aplicações. Normalmente os sistemas de controle são compostos por sensores, atuadores e controladores (JEON *et al.*, 2001).

De acordo com Santos, Vasques e Stemmer (2003) e Fredriksson (1997), a utilização de um sistema *FieldBus* (arquitetura de controle distribuída) permite que os dispositivos sejam interconectados a um custo menor (por possuir quantidade bastante reduzida de cabeamento), com menor complexidade de instalação e manutenção.

Dentro desse contexto existem diversos protocolos de comunicação, e Santos, Vasques e Stemmer (2003) apresentam um comparativo entre os protocolos CAN e o PROFIBUS (padrão *fieldbus* para automação de manufatura e controle de processos).

O barramento CAN, foco dessa pesquisa, foi projetado primeiramente para utilização na indústria automotiva (BOSCH, 2003), com o objetivo de diminuir a complexidade e a quantidade de fios que eram necessários para interligarem dispositivos e controlar a comunicação em tempo-real entre eles.

Após esse projeto, o protocolo CAN foi especificado gerando duas normas: Norma ISO 11898, para aplicações de alta velocidade (ISO11898, 1993), e a Norma ISO 11519 que é dividida em duas partes, para aplicações de baixa velocidade (ISO11519-1, 1994), (ISO11519-2, 1994).

O CAN foi desenvolvido para utilização na indústria automotiva, porém muitos trabalhos estão sendo desenvolvidos para aplicação do CAN em diversas áreas. Algumas áreas de aplicação do protocolo CAN, e suas respectivas especificações, são apresentadas por Sousa (2002):

- Área Agrícola especificada nas normas ISO 11783 e DIN 9684 ,
- Área Naval com especificação NMEA 2000,
- Veículos de grande porte (ônibus, caminhões, etc), com a especificação J1939 .

Muitos trabalhos estão sendo desenvolvidos com o objetivo de estudar, analisar e difundir esses padrões do protocolo CAN apresentados anteriormente como, por exemplo: (STONE, 1997),(GUIMARÃES, 2003),(SPECKMANN; JAHNS, 1999) e (ACLE; REORDA; VIOLANTE, 2006).

A crescente popularidade de aplicações com o CAN em sistemas de automação e controle é sustentada, entre outros fatores, por este possuir características muito próprias para tais sistemas em relação a outros padrões como, por exemplo:

- Taxa de comunicação configurável: Podendo variar de 50 Kbps à 1Mbps;
- Mecanismo de controle CSMA/CD with NDA (Carrier Sense Multiple Access / Collision Detection with Non-Destructive Arbitration), com a finalidade de evitar a colisão de dados;
- A comunicação pode ser implementada utilizando apenas 2 fios, gerando uma redução nos custos;
- Flexibilidade para adicionar e remover dispositivos da rede;
- Protocolo multi-mestre, no qual todos os dispositivos instalados no barramento se alternam nos estados de mestre e escravo durante as trocas de mensagens.

O crescimento na utilização do protocolo CAN na área agrícola pode ser analisado através da literatura nos últimos anos através dos trabalhos de (AUERNHAMMER, 2001), (STRAUSS, 2001), (SOUSA, 2002) e (GUIMARÃES; SARAIVA, 2002).

Devido às suas vantagens e principalmente à sua flexibilidade, tanto de implementação como de configuração da rede, cada vez mais o protocolo CAN está sendo utilizado em processos de controles distribuídos em tempo-real. Essas aplicações geralmente possuem duas características muito particulares que são: Tempo - Crítico e Segurança - Crítica e necessitam de um esforço na validação da arquitetura operacional na etapa do projeto (NAVET; SONG, 2001). Por isso que análises do comportamento e do desempenho dessas redes de comunicação acabam se tornando importantes (JAIN, 1991), (NOLTE; NOLIN; HANSSON, 2005).

De acordo com (TINDELL; HANSSON; WELLINGS, 1994) umas das dificuldades fundamentais na engenharia de sistemas em tempo-real é o desenvolvimento de análises para limitar o comportamento do sistema temporizado.

Em relação ao desempenho em tempo-real, uma das características mais importantes do protocolo CAN é o uso do identificador como prioridade de acesso ao barramento, responsável pelo método de arbitragem do barramento CAN. Esses sistemas são caracterizados pela grande necessidade de sincronismo nas trocas de dados, que devem ocorrer dentro de intervalos de tempo precisos a fim se assegurar que o sistema de controle opere corretamente (CENA; VALENZANO, 2000).

Somente as mensagens dos dispositivos configurados com prioridades mais elevadas podem ser garantidas no barramento CAN (TINDELL; BURNS; WELLINGD, 1995). Isso justifica a necessidade de realizações de simulações e análises de desempenho de projetos de rede CAN com o objetivo de evitar sobreposição, ou tempos de espera muito elevados para as mensagens dos dispositivos com prioridades mais baixas, ou dispositivos acíclicos.

Entre os principais trabalhos relacionados à análise de desempenho de protocolos de redes ponto-a-ponto e de comunicações em tempo-real, destacam-se Tindell, Hansson e Wellings (1994), K.Tindell e Burns (1994) e Tindell, Burns e Wellingd (1995). Nesses trabalhos foram definidos quatro tipos de atrasos por latência que podem ser gerados nas comunicações ponto-a-ponto, são eles: Atraso na geração da mensagem, Atraso no enfileiramento, Atraso na transmissão e Atraso de entrega das mensagens.

Em Tindell, Hansson e Wellings (1994) estudou-se especificamente os atrasos de transmissão e os atrasos de enfileiramento no protocolo CAN, e como resultado obteve-se o pior tempo de resposta para o protocolo. Juntamente com Tindell, Burns e Wellingd (1995) esses trabalhos se tornaram as principais referências em relação à análise de desempenho do protocolo CAN, e a partir dos modelos matemáticos apresentados, diversos outros trabalhos foram desenvolvidos utilizando diferentes abordagens.

Entre esses trabalhos está Sevillano *et al.* (1998), que apresenta um modelo de Markov de simulação utilizando a linguagem SPML, para realizar análise da taxa de ocupação do barramento. Esse trabalho destaca que um dos parâmetros mais importantes para serem analisados em sistemas de controles distribuídos é o canal de comunicação, e quando o interesse é a análise de um protocolo de comunicação em tempo-real, o principal parâmetro de análise é

o tempo de espera das mensagens serem entregues. No seu trabalho Pinho e Vasques (2003), também utilizam-se os modelos matemáticos para analisar a confiabilidade do protocolo CAN em comunicações em tempo-real.

Essas análises de desempenho devem ser realizadas de acordo com as diferentes áreas de aplicação do protocolo CAN; por exemplo Hofstee e Goense (1997) realizaram análises de desempenho do protocolo CAN aplicado na área agrícola, mais especificamente em um implemento agrícola (Máquina Semeadora), seguindo a norma DIN 9684 baseada no padrão CAN-2A. Nesse trabalho foram comparados os tempos de respostas das mensagens e a taxa de ocupação do barramento utilizando diferentes configurações, variando as prioridades dos dispositivos e taxa de comunicação.

Em Hofstee e Goense (1999) realizou-se outro trabalho de análise de desempenho do protocolo CAN aplicado na área agrícola. Nesse trabalho também foram realizadas as análises dos parâmetros de rede de controle em tempo real em um implemento agrícola (trator), porém baseado na norma ISO 11783. Essa norma especifica a versão do CAN 2B que possui o campo de identificação de 29 bits.

O trabalho de Darr, Stombaugh e Shearer (2005) também apresenta um estudo sobre a viabilidade da aplicação de uma rede CAN para comunicação de sistema de controle distribuído em um veículo agrícola autônomo.

Para uma correta operação da rede de comunicação, o sistema deverá garantir numa situação ideal que todos dispositivos (cíclicos e acíclicos) consigam obter acesso ao barramento, evitando que dispositivo possuam um tempo de latência alto ou tenham suas mensagens sobrescritas. Assim, mostra-se interessante a proposta de análises do comportamento da rede de comunicação para situações como: presença de *Jitter* (tempo médio de espera de uma mensagem na fila de transmissão) no conjunto de mensagens, a alta carga de mensagens sob uma rede CAN, grande quantidade de mensagens no *buffer*, e a consideração da viabilidade de aplicação de uma rede CAN.

Muitos trabalhos apresentam análises desses parâmetros de comunicações em sistemas de controle baseados em redes como, por exemplo: Yang, Li e Yang (2005) e Pop, Eles e

Peng (2005). Outros trabalhos demonstram mais especificamente resultados sobre análises de Redes CAN como, por exemplo: os trabalhos de Godoy, Sousa e Inamasu (2005), Obermaisser e Peti (2005).

Como resultado dessa crescente utilização na área agrícola, empresas e instituições de pesquisas nacionais e internacionais tem concentrado esforços no sentido de desenvolver e promover a implementação do padrão ISO 11783 para integração de dispositivos em máquinas e implementos agrícolas. A proposta para elaboração da norma ISO 11783 é prover um padrão aberto para interconexão de sistemas eletrônicos embarcáveis através de um barramento e permitir a comunicação entre eles.

2.2 Redes de Petri

O conceito de redes de Petri foi introduzido por Carl Adam Petri, em sua tese de doutorado em 1962, na faculdade de Matemática e Física da Universidade de Darmstadt na Alemanha (PETRI, 1966).

Rede de Petri é uma ferramenta de modelagem gráfica e matemática, bastante flexível e que possui grande capacidade de representação, tanto dos sistemas mais simples como de sistemas complexos. Utilizando o conceito de Rede de Petri, é possível simular o funcionamento de um sistema ou mesmo realizar análises de seu desempenho. As redes de Petri constituem um formalismo matemático e gráfico bastante eficaz na modelagem de sistemas que apresentam atividades concorrentes e competição na utilização de recursos limitados (MURATA, 1989).

Modelos de Redes de Petri podem ser aplicados em qualquer área ou sistema que possa ser descrito graficamente e que necessite representar atividades paralelas ou concorrentes (MURATA, 1989).

Muitos trabalhos utilizam modelos de Redes de Petri, nas suas mais diversas extensões, para realizar análises de sistemas e protocolos de comunicações. Os trabalhos Lai (1995) e Lai (1997) apresentam um estudo de performance do protocolo CSMA/CD, analisando parâ-

metros como : Atrasos e probabilidades de colisões durante os envios das mensagens através de um modelo de Rede de Petri Estocástica.

Outra extensão de Rede de Petri é utilizada em Marsan, Chiola e Fumagalli (1987), que também apresenta o desenvolvimento de um modelo de simulação que utiliza o formalismo de Rede de Petri com Transição Temporizada (TTPN) para análise de performance do protocolo CSMA/CD.

A extensão de Rede de Petri Colorida também foi utilizada por Bao, Liu e Hong (2003) com o objetivo de desenvolver um modelo de simulação para analisar o comportamento do protocolo TCP.

Neste trabalho, utiliza-se a abordagem de Cavalieri *et al.* (1996) como orientação ao desenvolvimento de um modelo de rede CAN, que aborda com sucesso a utilização de redes de Petri estocásticas na modelagem de uma rede CAN, gerando uma ferramenta de análise para auxiliar em projetos e especificações de redes. Entretanto, utilizou-se como ferramenta de modelagem o formalismo da Rede de Petri Colorida (JENSEN, 1996), (JENSEN, 1997). A utilização dessa extensão de rede de Petri amplia a possibilidades de modelagem, pois permite distinguir os elementos de acordo com suas funções específicas no processo de modelagem e da disponibilidade de um programa de edição e simulação de redes de Petri coloridas que permite a análise de desempenho dos modelos obtidos.

Outras vantagens na utilização da Rede de Petri colorida é a diminuição no tamanho e na complexidade de entendimento dos modelos utilizados para analisar grandes sistemas, além de oferecer recursos para analisar modelos com maior poder de abstração(GIRAULT; VALK, 2003), (MACIEL, 1996).

Esse trabalho fornece suporte a um projeto cooperativo na área relacionada à montagem e análise de uma rede *fieldbus* embarcada (o termo embarcado representa todo e qualquer sistema eletro-eletrônico montado em uma aplicação móvel) baseada no protocolo de comunicação digital CAN para integrar diferentes dispositivos para o sistema de um veículo agrícola autônomo(VAA).

3 REVISÃO TEÓRICA

Nesse capítulo são apresentados os conceitos e definições utilizadas nesse trabalho e também está dividido em duas seções:

Na seção 3.1 apresenta-se de forma bastante sucinta a teoria de comunicações *fieldBus*, mais especificamente do barramento CAN e todos os conceitos de comunicação desse barramento.

Na seção 3.2 é apresentada a revisão teórica sobre as Rede de Petri, seus conceitos e formalismo.

3.1 CAN - Controller Area Network

O CAN é um protocolo de comunicação serial para aplicações em tempo real e foi desenvolvido primeiramente para atender aplicações na área automotiva. O desenvolvimento desse novo protocolo teve como principais motivações acrescentar novas funcionalidades à rede de comunicação e a redução da quantidade de cabos e fios na interligação entre os dispositivos (sensores e atuadores) nos interiores dos veículos. Em 1993 o primeiro padrão internacional CAN foi publicado pela *International Standards Organization* (ISO), através da especificação ISO 11898 (para aplicações com altas taxas de transmissão) e no documento ISO 11519 (para aplicações com baixas taxas de transmissão), documentos que definem e especificam a utilização do barramento CAN na área industrial de maneira geral.

Conforme pôde ser notado na revisão bibliográfica, o protocolo CAN, que a princípio era utilizado basicamente na indústria automotiva, atendendo o objetivo inicial do desenvol-

vimento, passou a ser utilizado nas mais diversas áreas de controle e automação. Essa crescente popularidade é sustentada devido às características muito particulares e muitas vantagens que o protocolo CAN possui em relação a outros padrões de comunicação. No final da década de 90, estimava-se que mais 20 milhões de sistemas utilizavam o barramento CAN (versão 2.0A) nas mais diversas áreas (HOFSTEE; GOENSE, 1999). Dentre as diversas áreas de aplicações, a utilização na área agrícola tem tido um crescimento notável nos últimos anos. Entre os principais responsáveis pela crescente utilização do protocolo CAN em máquinas agrícolas, senão o principal, é o aumento da utilização de sistemas eletrônicos e computacionais embarcados, necessários para aplicação do conceito da Agricultura de Precisão, por exemplo: Monitores e/ou Controladores das Operações em campo, piloto automático, computadores de bordo, entre outros.

Como citado anteriormente, o protocolo CAN possui características bastante particulares e vantagens em relação a outros protocolos, que justificam sua utilização nas mais diversas áreas atualmente como, por exemplo, boa performance em aplicações com tempos-críticos, com taxa de transmissão configurável (podendo chegar a 1Mbit/s).

O protocolo CAN utiliza o conceito multi-mestre, onde todos os dispositivos instalados no barramento se alternam entre os estados de “mestre” e “escravo” durante as trocas de mensagens. Isso permite que somente um dispositivo, no caso o dispositivo “mestre”, consiga acessar o barramento e enviar os pacotes de dados, enquanto que os outros dispositivos recebem e reconhecem o pacote de dados enviados. Os pacotes de dados (mensagens) são enviados ao barramento utilizando *multicast*, caracterizado pelo envio de toda e qualquer mensagem para todos os módulos existentes na rede.

As taxas de transmissões de dados do barramento CAN podem ser configuradas de acordo com o tipo de aplicação, variando de 10 Kbit/s à 1Mbit/s, característica que oferece ao usuário maior flexibilidade para configurar e programar a rede de dispositivos de acordo com as necessidades específicas de cada aplicação. Porém a taxa de transmissão é inversamente proporcional ao tamanho do barramento. Em uma aplicação que utiliza a taxa de transmissão de 1Mbps, o barramento pode ter um comprimento máximo de 40 metros, caso o barramento

possua um tamanho maior, podem ocorrer falhas de comunicações durante as transmissões. Em uma aplicação com taxa de transmissão de 50 Kbit/s, o comprimento pode chegar a 1.000 metros. A Tabela 3.1 representa a relação entre o comprimento da rede (barramento) e a taxa de transmissão dos dados.

Tabela 3.1: Tamanho do Barramento CAN vs. Taxa transmissão

Taxa de Transmissão	Tamanho do Barramento
1 Mbit/s	40 m
500 Kbit/s	100 m
250 Kbit/s	250 m
125 Kbit/s	500 m

Outra característica bastante particular no protocolo CAN é a utilização de um método de arbitragem com conceito CSMA/CD with NDA (Carrier Sense Multiple Access / Collision Detection with Non-Destructive Arbitration), com a finalidade de evitar a colisão de dados durante as transmissões entre os dispositivos instalados na rede. No CSMA/CD-NDA, durante as transmissões todos os dispositivos enviam suas mensagens bit-a-bit e caso ocorra uma disputa pelo barramento, o dispositivo que possuir um endereço com maior prioridade continua a transmissão da mensagem sem ter que reiniciá-la, enquanto que as transmissões dos dispositivos com prioridades menores são interrompidas, suas mensagens são armazenadas em um *buffer* para serem transmitidas posteriormente e os status são alterados para “escravos” (habilitados somente para recepção dos dados).

Para o domínio do modelo desenvolvido para análise dos parâmetros de rede CAN, e da importância dessas análises em sistemas de controle em tempo-real, é de fundamental importância considerar as principais características do barramento CAN que inclui basicamente: tamanho dos pacotes de mensagens, características do método de arbitragem e métodos de detecção de falhas nas comunicações. Essas principais informações estão disponíveis de maneira bastante sintetizadas no Anexo (7.0.6).

Outros trabalhos apresentam a teoria do protocolo CAN mais detalhada e completa como, por exemplo: Sousa (2002) e Guimarães (2003)

3.2 Rede de Petri

O conceito de redes de Petri foi introduzido por Carl Adam Petri, em sua tese de doutorado em 1962, na faculdade de Matemática e Física da Universidade de Darmstadt na Alemanha, Petri (1966), e diversas extensões foram desenvolvidas desde então. Dentre essas extensões podem ser citadas: Redes de Petri Estocásticas, Rede de Petri Temporizadas, Redes de Petri Orientadas a Objetos, entre outras.

Rede de Petri é uma ferramenta de modelagem gráfica e matemática, bastante flexível e que possui grande capacidade de representação, tanto dos sistemas mais simples como de sistemas complexos. Utilizando o conceito de Rede de Petri, é possível simular o funcionamento de um sistema ou mesmo realizar análises de seu desempenho. A Rede de Petri utiliza uma ferramenta gráfica para visualização das estruturas e das relações entre os sistemas modelados, que pode ser comparada com Diagramas de Bloco, Diagramas de Seqüência ou qualquer outro conceito de grafos.

Modelos de Redes de Petri podem ser aplicados em qualquer área ou sistema que possa ser descrito graficamente e que necessite representar atividades paralelas ou concorrentes (MURATA, 1989).

Essa flexibilidade pode ser notada na literatura atual, pois o conceito de Rede de Petri está sendo difundido e aplicado nas mais diversas áreas como: computacional, com elaboração de modelos para análises de Base de Dados Distribuídas (VOSS, 1980) ou modelos para análises de programas paralelos ou concorrentes (MURATA, 1980), na área Industrial, com elaboração de modelos sistema de controle da indústria ou da produção.

Desde o surgimento desse conceito, muitos grupos de pesquisas trabalharam para desenvolver a teoria básica, no qual um modelo de Rede de Petri é composto basicamente por quatro componentes: Lugares, Transições, Arcos e Marca, e chamado de Modelo Condição-Evento. Desde a década de setenta muitos resultados foram obtidos decorrentes das pesquisas realizadas e novos conceitos de Redes de Petri foram criados. Entre esses novos conceitos estão as Redes de Petri temporizadas, que possuem componentes capazes de analisar os comporta-

mentos temporais dos sistemas modelados.

A utilização das Redes de Petri cresceu significativamente na medida em que novos componentes foram criados com o intuito de aumentar o poder de abstração e modelagens.

Um modelo de Rede de Petri possibilita a simulação de diversas partes de sistemas como, por exemplo: concorrência, controle, conflitos, sincronização e compartilhamento.

Um modelo de Rede de Petri é descrito por quatro elementos básicos: Lugar, Transição, Arco e Marca.

Lugares: graficamente são representados por um círculo e modelam os estados do sistema. Representam as condições para que um determinado evento aconteça;

Transições: graficamente são representados por um retângulo e modelam os eventos do sistema, que representam as mudanças de estados;

Arcos: Graficamente são representados por uma seta, interligam um lugar e uma transição, e representam como é a sequência com que acontecem as mudanças de estado dentro do sistema;

Marcas: Graficamente são representadas por um círculo de cor preta e representam o comportamento dinâmico do sistema, indicando os estados ativos, ou seja, que estão sendo utilizados pelo sistema;

A figura 3.1 apresenta os quatro elementos básicos da Rede de Petri:

Figura 3.1: Elementos Básicos da Rede de Petri

O conceito de Rede de Petri pode ser classificado basicamente em dois grupos:

- As Redes de Petri Ordinárias ou “Baixo-Nível”: são classificadas em Redes de Petri Condição-Evento, Redes de Petri Lugar-Transição, e Estocásticas.
- As Redes de Petri de Alto Nível: são classificadas como Redes de Petri Coloridas, Redes de Predicado-Transição e Redes de Petri Orientadas a Objeto.

A divisão das extensões das Redes de Petri em dois grupos: Ordinárias (“Baixo Nível”) e Alto Nível pode ser comparada com a mesma analogia existente na divisão das linguagens de programação de computadores. As Redes de Petri de Baixo Nível podem ser comparadas com as linguagens de baixo nível, como Assembly e as Redes de Petri de Alto-Nível podem ser comparadas com as linguagens C++, Java, entre outras.

Para o desenvolvimento desse trabalho foi utilizada a extensão da Rede de Petri Colorida (*CPN - Coloured Petri Net*). Para maior compreensão e entendimento do trabalho e do modelo de simulação desenvolvido é necessário um conhecimento básico sobre o formalismo das Redes de Petri.

No capítulo 7.0.6 é apresentada de maneira bastante sintetizada a teoria e os elementos básicos das Redes de Petri e da extensão Rede de Petri Colorida.

Maiores detalhes sobre a teoria e o formalismo da Rede de Petri ordinárias (MURATA, 1989) e sobre a extensão da Rede de Petri Colorida estão disponíveis nos trabalhos: Jensen (1996), Jensen (1997) e Jensen, Kristensen e Wells (2006).

4 MATERIAIS E MÉTODOS

Nesse capítulo serão apresentados os softwares utilizados no desenvolvimento do trabalho, suas características como: plataforma, facilidade de uso, ferramentas de edição e extração de dados dos modelos de simulação (seção 4.1).

São apresentados com detalhes na seção 4.2 os dois modelos de simulação que foram desenvolvidos utilizando formalismo da Rede de Petri.

4.1 Softwares de Simulação

Devido à complexidade na proposta de criação do modelo de simulação de uma rede de dispositivos CAN, foram utilizados diversos softwares de simulação de Rede de Petri. Alguns softwares que foram utilizados apresentaram carência de ferramentas necessárias para criação do modelo, para execução das simulações e principalmente análise do modelo proposto nesse trabalho.

Como pode ser observado em ToolsDatabase (2005), há uma lista de ferramentas existentes que podem servir de apoio aos desenvolvimentos de modelos de rede de Petri utilizando diversas extensões.

A seguir serão apresentados os softwares utilizados nesse trabalho, descrevendo características de interface, facilidade de uso e recursos para coleta e análise das informações durante uma simulação.

4.1.1 HPSim

O software HPSim foi o primeiro software utilizado para elaboração do modelo de simulação da rede CAN . A utilização desse software forneceu os recursos necessários para criar modelos utilizando o formalismo da Rede de Petri Lugar-Transição, Redes de Petri Estocásticas, Redes de Petri Temporizadas. Esse software foi desenvolvido para a plataforma Windows, possui distribuição gratuita, e pode ser obtido em HPSim (2005).

Por ser um sistema desenvolvido para plataforma Windows, possui fácil instalação e uma interface bastante simples, o que facilita sua utilização. Devido à essas características, o software HPSim pode ser recomendado para usuários que estejam iniciando os estudos sobre simulações utilizando Rede de Petri.

A ferramenta gráfica possui recursos básicos para criação e edição de modelos de simulação que são apresentadas de forma bastante simples e “amigável” para o usuário.

A figura 4.1 apresenta a interface do sistema HPSim com o modelo de simulação da rede CAN que foi desenvolvido no início desse trabalho

Apesar de todas as facilidades oferecidas durante a criação do modelo de simulação de uma rede CAN para área agrícola, o software HPSim não teve sua utilização continuada, pois ofereceu uma carência de ferramentas para extração dos dados gerados durante as simulações. Após as execuções das simulações, os dados disponibilizados pelo software não foram suficientes para a realização das análises de desempenho propostas nesse trabalho, pois com o término de uma simulação, o usuário obtém acesso somente à informação da seqüência de disparo das transições do modelo.

Por ser um recurso insuficiente para realizar as análises dos parâmetros da Rede CAN, o modelo foi criado utilizando uma outra ferramenta, conhecida como Visual Object Net++, e seus resultados serão apresentados a seguir.

Figura 4.1: Interface do Software de Simulação HpSIM

4.1.2 Visual Object Net ++

O Software Visual Object++ oferece recursos para elaboração de modelos utilizando o formalismo de Rede de Petri Condição-Evento, Rede de Petri Lugar-Transição temporizadas ou não.

Esse software foi desenvolvido para a plataforma Windows, possui uma interface bastante simples e intuitiva, como pode ser observado na figura 4.2.

Como o HPSim, a interface do software Visual Object Net++ oferece os recursos básicos para edição dos modelos de Rede de Petri, porém a falta de ferramentas de extração e análise das informações durante as simulações também impossibilitaram sua utilização nesse trabalho. Como foram encontrados os mesmos problemas do software HPSim, iniciou-se um trabalho de pesquisa para encontrar outra ferramenta que disponibilizasse todos recursos necessários para o desenvolvimento desse trabalho.

Figura 4.2: Interface do Software Visual Object ++

4.1.3 TimeNet

TimeNET é uma ferramenta de modelagem e simulação de sistemas que utiliza o formalismo da Rede de Petri Estocásticas, disponibilizada gratuitamente, e está sendo utilizada em diversas instituições de pesquisa para modelagens dos mais diversos sistemas (TIMENET, 2006).

Essa ferramenta possui versões para a plataforma Sun, com instalação recomendada para SunOS 5.7 e 5.9, e também para a plataforma Linux onde recomenda-se a instalação na distribuição SuSe 7.2.

Assim como muitas ferramentas de modelagem de Rede de Petri, TimeNet foi desenvolvido utilizando uma estrutura composta por módulos, que são divididos basicamente em: Módulo de Edição, de Simulação e de Análise de Desempenho .

O software de modelagem TimeNet e conseqüentemente a utilização da extensão de Rede de Petri Colorida Estocásticas, fornecem maiores recursos de modelagem, como a utilização de transições temporizadas capazes de modelar e produzir atrasos em seus disparos (GERMAN *et al.*, 1995). As transições temporizadas são classificadas como: Exponenciais ou Determinísticas. Os atrasos nas transições exponenciais muitas vezes não representam a realidade em muitos sistemas e podem dificultar as análises de desempenho do sistema que está sendo modelado. As transições Determinísticas são utilizadas para modelar e simular operações com frequências definidas. Essas transições são bastante utilizadas em simulações de sistemas de comunicação onde o tempo e as frequências são atributos importantes para a simulação, principalmente para a realização das análises de desempenho.

Para a elaboração do modelo e da simulação proposta nesse trabalho, a utilização das transições determinísticas é fundamental. Essas transições são necessárias para representar as frequências de acessos dos dispositivos no barramento CAN como, por exemplo, simular os acessos dos sensores como GPS e o sensor de ultra-som, que possuem frequências de acessos definidos ou cíclicos.

Apesar de todas as vantagens oferecidas pelo software TimeNet 3.0, não foi pos-

sível utilizá-lo nesse trabalho devido a sua complexidade de instalação. Durante a instalação, muitos problemas foram enfrentados e direcionaram a pesquisa para utilização de outro software. Dessa maneira não foi possível avaliar a facilidade de uso, interface e as ferramentas de análise de desempenho que estão disponíveis nessa versão do software.

Após a tentativa de utilização do software TimeNet, foi utilizada uma outra ferramenta de modelagem, chamada GreatSPN.

4.1.4 GreatSPN

O software GreatSPN utiliza o formalismo da Rede de Petri Estocástica Generalizadas e a extensão Rede de Petri Colorida para criação, edição e simulação de um modelo de Rede de Petri.

Assim como o software TimeNet 3.0, o GreatSPN2.0 também é composto por vários módulos, entre eles: edição, simulação e análise de desempenho. Essa divisão em módulos permite que diversos grupos de pesquisa participem do desenvolvimento do sistemas, de forma paralela e cooperada, criando novas versões, incorporando novas funcionalidades e mantendo a compatibilidade com as versões anteriores do software. O GreatSPN é software comercial, porém possui uma versão gratuita para instituições de pesquisa e possui versões para instalações nas plataformas Unix, Linux ou Solaris.

A instalação dessa ferramenta é um pouco mais complexa, mas durante esse trabalho, não apresentou nenhum problema que pudesse impossibilitar sua utilização. A instalação foi realizada seguindo a recomendação dos desenvolvedores, no sistema Operacional Linux, utilizando a distribuição Slackware 9.0.

Além dos recursos gráficos como: lugar, transição e arcos, o GreatSPN disponibiliza muitos outros recursos para elaboração de um modelo de Rede de Petri. Entre os recursos, destacam-se os arcos inibidores, as transições determinísticas e exponenciais.

A interface gráfica do software GreatSPN é bastante simples e intuitiva, como as outras ferramentas apresentadas nesse trabalho. Todos os elementos gráficos para a constru-

ção de um modelo de simulação estão disponíveis em uma barra de ferramentas e podem ser inseridos facilmente na área de edição.

Esse software encontra-se na versão 2.0 e além de fornecer ferramentas para criação e edição de modelos, fornece recursos para análises estruturais e ferramenta para análises das propriedades da Rede de Petri como, por exemplo, a propriedade de alcançabilidade e vivacidade. Estão disponíveis também recursos para extrair e analisar informações das simulações. Essas informações estão disponíveis para os usuários, através de arquivos textos no formato de relatórios ou também na forma de gráficos.

Outra recurso interessante e importante durante o trabalho de criação de um modelo de simulação, é a possibilidade de realizar simulações interativas ou a simulações rápidas. A simulação interativa é executada passo a passo de maneira que o usuário possa analisar e validar a sequência de disparo e o comportamento do modelo.

Durante a simulação do modelo, o software disponibiliza para o usuário algumas informações como: valores dos contadores, tempo de simulação, tipo de marcas, quantidade de marcas em um lugar, valores das variáveis de controle, entre outras. Durante uma simulação rápida, o software não apresenta nenhum tipo de informação para o usuário, somente executa a simulação internamente e disponibiliza os resultados logo após atingir algum critério de parada, como o tempo da duração da simulação, que pode ser definido pelo usuário.

Durante a utilização das transições determinísticas, foram encontrados diversos problemas que impossibilitaram a execução da simulação e conseqüentemente a utilização desse software, entre eles, o comportamento acíclico da transição responsável pela criação das mensagens no modelo.

Pela necessidade, imposta nesse trabalho, de utilização das transições determinísticas, para simular as frequências de criação das informações dos dispositivos da rede CAN, optou-se por não dar continuidade ao desenvolvimento desse trabalho utilizando essa ferramenta. Devido a esse problema, não foi possível avaliar as ferramentas de extração e análise dos dados das simulações.

4.1.5 DesignCPN

Design/CPN é ferramenta para modelagem e simulação de sistemas que utilizam o formalismo da Rede de Petri Colorida (CHRISTENSEN; JORGENSEN; KRISTENSEN, 1997). Esse software foi desenvolvido pela empresa Meta Software Corporation, Cambridge EUA em parceria com o grupo de CPN na universidade de Aarhus, Dinamarca. A utilização desse software se justifica por este disponibilizar uma grande quantidade de ferramentas que facilitam a elaboração de modelos, manipulação e principalmente a extração das informações coletadas durante as simulações. Existem duas versões do software, uma versão compatível com o Windows e outra versão para o sistema operacional Linux.

O Design CPN é formado, basicamente, por quatro ferramentas ou pacotes (DESIGNCPN, 2006):

- Editor: Ferramenta para construção, edição e análises de sintaxe dos modelos.
- Simulador: Ferramenta para realizar as simulações, que podem ser: interativas ou automáticas.
- Gráfico da Ocorrência: Ferramenta para construção e análise de gráficos da ocorrência dos modelos.
- Análise de Desempenho: Ferramenta para análises de desempenho baseado nas simulações.

O pacote suporta também Rede de Petri Hierárquica, isto é, os modelos podem ser elaborados através das relações entre diferentes módulos ou sub-redes.

Utilizando o Design/CPN é possível criar e manipular modelos utilizando dados complexos, para isso a ferramenta permite que funções possam ser escritas utilizando a linguagem de programação funcional ML. Essa particularidade permite que o usuário consiga criar no modelo funções, variáveis de controle, listas de cores, manipular arquivos, entre outras vantagens que uma linguagem funcional pode oferecer.

4.1.5.1 Editor

No Editor de Rede de Petri Colorida, é possível criar objetos gráficos como: lugares, transições, arcos, funções entre outros disponíveis na ferramenta DesignCPN.

Após a elaboração do Modelo, o software traduz todos os objetos gráficos para um modelo matemático que será utilizado nas simulações. Dessa forma o modelo apresentado na interface do sistema tem a função de facilitar o entendimento, a visualização do funcionamento e da estrutura do sistema modelado.

O pacote de Edição permite que o usuário crie modelos seguindo o formalismo das Redes de Petri Hierárquicas e utilizando tanto a técnica *Top-Down*, onde cada transição represente uma atividade complexa e cada sub-rede represente sua atividade, como a técnica *Botton-Up*, onde é necessário criar as páginas com os componentes básicos do sistema e posteriormente criar as camadas mais elevadas, substituindo transições. Como apresentado por Jensen (2005), geralmente os modelos são elaborados utilizando ambas as técnicas.

Outro recurso de fundamental importância, também muito utilizada na elaboração dos modelos de simulação, é a verificação de sintaxes. Essa ferramenta analisa a estrutura do modelo verificando possíveis erros de sintaxe como, por exemplo, a criação de modelos que violam alguma definição do formalismo da Rede de Petri.

4.1.5.2 Simulador

O pacote de simulação permite que os modelos criados sejam simulados e executados. Esse pacote está totalmente integrado com o pacote de Edição de modelos e obtém um modelo matemático, como resultado da análise e conversão de objetos gráficos da interface para realizar as simulações. O usuário necessita informar alguns parâmetros de configuração da simulação, entre eles, o intervalo de tempo que será executado, critérios de parada, entre outros.

A simulação também pode ser executada de duas formas diferentes: Interativa ou Automática. Geralmente utiliza-se primeiramente a simulação interativa, pois durante essa execução é possível analisar o comportamento do modelo e também os valores das variáveis de

interesse. A Figura 4.3 apresenta o ambiente do software Design-CPN durante uma simulação.

Figura 4.3: Simulação Interativa do Software Design CPN

Como apresentado na figura 4.3, o Design CPN permite que as variáveis existentes no modelo possam ser analisadas, permitindo a visualização dos tempos da simulação, os valores dessas variáveis durante a simulação, além de permitir analisar a sequência de disparos das transições e acompanhar o funcionamento do sistema de maneira geral. Esse tipo de simulação possui execução bastante demorada e devido a isso, é utilizada somente para analisar e validar o funcionamento do modelo que está sendo desenvolvido.

Durante a execução da simulação automática, o usuário não consegue visualizar nenhuma informação no modelo. Uma vez iniciada, o software somente interrompe a execução da simulação quando algum critério de parada, informado pelo usuário, seja atendido.

Esse tipo de simulação é muito utilizado para extração de dados, em modelos que já tiveram suas estruturas validadas. Os resultados das simulações podem ser armazenados em relatórios no formato de arquivos texto ou gráficos.

4.1.6 Análise de Desempenho

Os modelos de Rede de Petri muitas vezes são utilizados para analisar parâmetros como tempo máximo para execução de alguma atividade, ou o tempo médio de espera para execução de uma determinada solicitação, ou seja, analisar o comportamento e o funcionamento do modelo de simulação. Para isso, o DesignCPN possui estrutura de dados para coletar e armazenar dados durante as execuções das simulações. Essa estrutura é conhecida como “Coletor de Dados”, ou *Data Collection*, que armazena variáveis dos modelos temporizados, como por exemplo: valores médios, somatórios, maior valor, menor valor, entre outras.

Durante a simulação de um modelo, os coletores de dados atualizam os valores das variáveis de interesse que foram definidas pelo usuário, e no término da simulação armazena todas as informações coletadas em arquivos de “log”. A partir desses arquivos é possível analisar os dados na forma de relatórios e gráficos.

Além da utilização dos Coletores de Dados existentes no software Design/CPN, o usuário pode optar por implementar algoritmos utilizando a linguagem CPN-ML para monitorar os parâmetros do modelo. Neste trabalho foram escritos algoritmos específicos para o monitoramento e armazenamento das informações durante as simulações.

4.1.7 Linguagem CPN-ML

O Design CPN utiliza a linguagem CPN ML que é a extensão da linguagem ML realizada de três formas diferentes (CPNML, 2006).

A primeira extensão facilita as declarações dos tipos mais comuns das listas de cores, permitindo que o usuário possa incluir um número de funções, apenas mencionando o nomes nas declarações da lista de cor.

A segunda extensão permite que o usuário declare variáveis e seus tipos como parte das declarações. O Padrão ML não possui declarações de variáveis, pois um valor pode ser definido a uma variável e esse valor que define o seu tipo. O tipo dessa variável pode ser alterado apenas definindo um novo valor de um tipo diferente. No CPN-ML todas as variáveis

precisam, necessariamente, ter seus tipos definidos.

A terceira extensão permite que o usuário declare variáveis de referência, apenas nos *code segments*, ou segmentos de códigos implementados nas transições. As variáveis da referência são elementos não funcionais do padrão ML.

Uma vantagem na utilização do CPN-ML, que devido à essa extensão da linguagem ML, é a possibilidade de pesquisar grande quantidade de materiais de referência (livros, grupos de discussão, apostilas, entre outros).

4.1.8 O padrão ML

A possibilidade de utilização da linguagem de programação ML em um projeto de Rede de Petri Colorida, além de ser uma linguagem bastante estável e segura, fornece ao usuário recursos para criação de tipos de dados, funções, variáveis, manipulação de exceções e expressões de maneira muito similar as outras linguagens de programação. A sintaxe ML pode ser considerada flexível e extensível, permitindo que os códigos possam ser escritos de maneira muito próxima do padrão matemático. Um exemplo é a utilização do operador ++ para adições de listas ou *multi-sets*.

ML é uma linguagem funcional, permite a definição e a aplicação das funções, que podem ser definidas pelo usuário utilizando o padrão das linguagens convencionais, mas ML permite também que funções possam ser tratadas como valores e computar novas funções a partir deles, através dos operadores.

A ML consiste em uma linguagem com um núcleo (*core*) para programação e um sistema de módulos para a programação em grande escala.

A definição detalhada da Linguagem ML pode ser encontrada em Ullman (1997).

4.2 Modelos de Simulação

Para o desenvolvimento de um modelo que simule a rede CAN apresentada anteriormente, utilizou-se a abordagem de Cavalieri *et al.* (1996) como orientação. Entretanto, no modelo elaborado utiliza-se a descrição formal da Rede de Petri Colorida e não a descrição da Rede de Petri Estocástica Temporizada adotada por Cavalieri *et al.* (1996). A utilização desse tipo de Rede de Petri amplia as possibilidades de modelagem, pois permite distinguir os elementos de acordo com suas funções específicas no processo de modelagem e também um dos principais fatores, senão o principal, da disponibilidade de um programa de edição e simulação de Redes de Petri Coloridas que permite a análise de desempenho dos modelos criados e simulados .

Durante a realização desse trabalho foram criados diversos modelos de Simulação, capazes de representar uma rede de dispositivos CAN. Esses modelos podem ser divididos em dois grupos: Representação dos dispositivos e a Representação do Barramento CAN. Para atingir os objetivos desse trabalho, ao longo do desenvolvimento e dos conhecimentos adquiridos, os modelos criados conseguiram explorar um número maior de recursos disponíveis no Design CPN.

Foram elaboradas duas versões do Modelo de Simulação que serão apresentadas no capítulo 5 que descreve as etapas do desenvolvimento.

5 *DESENVOLVIMENTO DO TRABALHO*

5.1 Desenvolvimento do Modelo de Rede de Petri

Nesse capítulo serão apresentados os modelos de Rede de Petri que foram desenvolvidos nesse trabalho.

O primeiro modelo apresentado na seção 5.1.1 foi desenvolvido utilizando o formalismo da Rede de Petri Estocástica e na seção 5.1.2 é apresentado o segundo modelo desenvolvido utilizando o formalismo da Rede de Petri Colorida.

5.1.1 Modelo de Rede de Petri Estocástica

Para facilitar o entendimento, será apresentada apenas parte do primeiro modelo de simulação que foi criado, que representa a estrutura de apenas um dispositivo (Figura 5.1) e destaca os elementos gráficos da Rede de Petri.

O lugar **Sensor1** representa um dispositivo CAN que necessita enviar alguma informação para o barramento. No modelo, o tempo de disparo da transição *EnterData_S1*, determinada pela expressão “@+3”, define a frequência com que dados são coletados pelo sensor. Isso denota que a marca *Data_S1*, correspondente a um dado coletado pelo sensor 1, é depositada no lugar *DataS1*, permanecendo no mesmo por 3 unidades de tempo¹. O lugar *BufferSensor1* modela o *buffer*, dentro do qual mensagens não-enviadas são armazenadas em decorrência da disputa pelo acesso ao barramento. O lugar *CanBusS1* modela o estado do barramento: se o

¹ São chamadas unidades de tempo, pois os tempos podem ser interpretados de diferentes formas: como segundos, minutos ou horas desde que sejam mantidas as proporções nas definições.

Figura 5.1: Modelo de uma Rede CAN representando apenas 1 dispositivo

barramento estiver sendo acessado por um dos três sensores, a marca presente é *Can_Bus_Busy*, caso contrário, a marca presente é *Can_Bus_Free*; isto é, o barramento se encontra livre.

O lugar *Buffer1State*, por sua vez, modela o estado dos *buffers* do Sensor1. Estes iniciam com uma marca *NotFull*, já que os *buffers* em estado inicial estão vazios. O lugar **CheckS01** modela a contagem de marcas presentes nos *buffers*. Tão logo um número máximo, definido pelo usuário, de mensagens são armazenadas, a transição *COI* está habilitada para o disparo. O disparo da mesma resulta na deposição de uma marca *Full* no lugar **Buffer1State**.; isto é, o sensor então sai da disputa pelo barramento.

Caso o dispositivo obtenha acesso ao barramento, a transição *EnterDataS1Bus* dispara após 2 unidades de tempo, definido para representar o tempo que a mensagem fica ocupando o barramento. A marca passa para o lugar **CanBusS1** e posteriormente passa pela transição *ExitDataS1Bus*, que é responsável por retirar a marca do lugar **Can_Bus**, mudando a marca existente para *Can_Bus_Free* (que volta o estado do barramento para livre) e do lugar **Can_Bus_S**. A figura 5.2 apresenta o modelo que representa o barramento CAN, contendo 3 dispositivos .

As principais Declarações dos nós (*node declaration*) implementados no modelo da rede CAN são:

```

1  color DataSensor = with Data_S1 | Data_S2 | Data_S3 timed ;
2
3  color CanBus = with Can_Bus_Busy | Can_Bus_Free ;
4
5  color S_State = with Full | Not_Full ;
6
7  color I = int ;
8
9  color Var_Can_Bus : CanBus ;
10
11 var i1; i2; i3 : I ;
12
13 var Var_Data_S1 , Var_Data_S2, Var_Data_S3 : DataSensor ;
```

Para realização das análises de desempenho do modelo apresentado, definiram-se três coletores de dados *DCSensor1*, *DCSensor2* e *DCSensor3* responsáveis pelo armazenamento

Figura 5.2: Modelo de uma Rede CAN com 3 dispositivos

das informações durante a simulação e disponibilizá-las através dos relatórios nos formatos textos ou gráficos.

Como pode ser observado, o modelo desenvolvido possui uma complexidade de entendimento muito elevada, característica capaz de limitar sua utilização em outras aplicações, além de possuir alta complexidade de edição que dificulta que dispositivos sejam inseridos ou excluídos do barramento simulado. Porém, conforme apresentado em Lopes *et al.* (2005), o modelo conseguiu atingir os objetivos voltados para análise do desempenho de uma rede CAN.

Com o propósito de elaborar um modelo mais simples e capaz de explorar uma quantidade maior de recursos oferecidos pelo formalismo da Rede de Petri Colorida e pelo Software de simulação DesignCPN, foi desenvolvido um novo modelo utilizando o conceito de multi-marcas. O modelo será apresentado a seguir (5.1.2).

5.1.2 Modelo de Rede de Petri de Alto-Nível

Com a possibilidade de utilizar lugares com diversas marcas, o modelo desenvolvido possui uma única estrutura capaz de representar uma quantidade diversificada de dispositivos configurados pelo usuário. No modelo anterior (Modelo de Rede de Petri Estocástica), dentro de cada estrutura, que representava um dispositivo, era utilizada apenas uma marca ou cor que o identificava. Com essa limitação no Modelo de Rede de Petri Estocástica (figura 5.2), foi desenvolvido um modelo de Rede de Petri de Alto-Nível, utilizando o conceito de “Multi-Marcas” que será apresentado na figura 5.3, onde todos os dispositivos simulados compartilham uma única estrutura de lugares, transições e arcos, o que facilita sua edição e entendimento, além de permitir simular um número variado de sensores.

A tabela 5.1 apresenta uma legenda com a descrição básica de cada elemento do modelo CAN.

O modelo apresentado segue os padrões definidos pela Rede de Petri Colorida, e do software de simulação Design CPN, onde os lugares são identificados pelos textos em negrito. As marcas, ou cores possuem seus nomes apresentados em itálico. Para o modelo “Multi-

Figura 5.3: Modelo de Rede de Petri Colorida - Legenda na Tabela 5.1

Tabela 5.1: Descrição dos elementos gráficos do Modelo

Identificação	Tipo	Nome	Descrição
1	Lugar	Sensor	Representa geração de um pacote de dados de um dispositivo (Sensor e/ou Atuador)
2	Transição	EnterData	Atribui uma frequência de acesso para os pacotes gerados
3	Lugar	DataS	Realimentação para o Lugar “Sensor”
4	Transição	DataSTrue	Controle na geração das mensagens
5	Lugar	BufferSensor	Representa a Fila de Mensagens no Buffer
6	Transição	DefineDataSensorList	Monta a lista de mensagens armazenadas no Buffer
7	Lugar	DataSensor_List	Atualiza o Buffer de Mensagens
8	Transição	DefinePriorityDataSensor	Método de Arbitragem
9	Lugar	AccessDataSensorPriority	Arbitragem - Define qual mensagens irá obter o acesso ao Barramento
10	Transição	EnterDataSensor	Arbitragem - Define quanto tempo a mensagem ocupará o barramento
11	Lugar	CanBusBusy	Representa que o Status do Barramento está Ocupado ou Livre
12	Transição	ExitDataSensor	Essa transição retira a mensagem do Barramento (o recebimento pelo destinatário)
13	Lugar	CanBus	Representa o Barramento CAN
14	Lugar	AccessControl	Define qual mensagem terá acesso ao barramento

Marca”, foram utilizadas as seguintes declarações:

```

1 DataSensor = with Data_S1 | Data_S2 | Data_S3 |
2             NOData_S1 | NOData_S2 | NOData_S3 timed;
3 color CanBus = with Can_Bus_Busy | Can_Bus_Free;
4 color Int = int;
5 color DataSensor_list = list Int timed;
6 color IntS = Int timed;
7 color CountDTS = product Int * DataSensor untimed;
8 color ControlAccess = with YES | NO;
9 var a : DataSensor;
10 var L1,L2 : DataSensor_list;
11 var x : Int;
12 val y = 128;
13 val cs1 = ref 0;
14 val cs2 = ref 0;
15 val cs3 = ref 0;

```

As Principais definições de cores são:

- DataSensor: são as cores que diferenciam os dispositivos que serão utilizados na rede de comunicação, no caso temos Data_S1, Data_S2, Data_S3, Data_S4 , Data_S5, Data_S6;
- CanBus: cor que identifica o estado do barramento, que pode ser Can_Bus_Busy para representar que o mesmo esteja ocupado, ou Can_Bus_Free para representar que o barramento esteja livre ;
- Data_Sensor_List: define uma estrutura de lista para armazenar as cores dos diferentes sensores ;

A transição *EnterData*, representa a frequência em que os dados são gerados. Por ser um modelo que utiliza multi-marcas e os dispositivos simulados possuírem frequências de coletas distintas, foi necessário criar uma função para defini-las. Nessa transição, a função $@+F1(x)$ é utilizada para representar essas frequências, onde @+ representa que é uma definição de temporização e a função $F1(x)$ recebe uma cor como parâmetro, verifica a frequência que foi definida para a mesma e retorna um valor inteiro que será utilizado como base para a frequência de acesso.

A função $F1(x)$ está implementada da seguinte maneira:

```

1 fun F1(a) =
2     if a = Data_S1 then 5 else
3     if a = Data_S2 then 10 else
4     if a = Data_S3 then 20 else
5     if a = Data_S4 then 40 else
6     if a = Data_S5 then 30 else
7     12;

```

O parâmetro “a” é cor que a função receberá como parâmetro, caso seja *Data_S1*, que representa o sensor 1, os dados serão gerados a cada 5 unidades de tempo, caso seja *Data_S2*, os dados serão gerados a cada 10 unidades de tempo e assim sucessivamente. Caso a cor não seja de nenhum dos 5 primeiros dispositivos, o retorno é o valor 12 unidades de tempo, para o dispositivo 6.

Com isso, na mesma transição *EnterData*, as frequências serão de @+5, @+10, @+20, @+40, @+30 e @+12, para os sensores 1, 2, 3, 4, 5 e 6 respectivamente.

O lugar **DataS** é utilizado para realimentar a geração das cores no lugar **Sensor**.

Após a geração das informações, as mesmas passam para o lugar **BufferSensor**. Na transição *DefineDataSensorList* é definida uma lista de cores que será utilizada para realizar as prioridades entre os dispositivos. No arco após essa transição é executada a função $F2(a, L1)$. O parâmetro “a” representa a marca que será inserida na lista e o parâmetro $L1$ representa a lista em que a marca será inserida. Para facilitar a manipulação dessa lista, nessa função são inseridos valores inteiros. Os valores inseridos são os inteiros: 1, 2, 3, 4, 5 e 6 para representarem os

identificadores dos sensores.

```

1 fun F2(a,L1) =
2     case a of Data_S1 => 1::L1 |
3     Data_S2 => 2::L1 |
4     Data_S3 => 3::L1 |
5     Data_S4 => 4::L1 |
6     Data_S5 => 5::L1 |
7     Data_S6 => 6::L1;

```

A função F2 verifica o tipo da cor que está sendo passada como parâmetro e adiciona o valor inteiro que a representa, caso a cor seja Data_S1 é adicionado o valor 1 , caso seja igual a Data_S2, é adicionado o valor 2 e assim sucessivamente.

O lugar **DataSensor_List**, representa a lista de cores existentes. Durante a simulação os valores são inseridos na lista na sequência em que são gerados como, por exemplo : L1={ 1,2,2,1,3,2,2,...}.

Na transição *DefinePriorityDataSensor* é retirada a lista de cores e posteriormente é utilizada a função Access(L2).

```

1 fun access L2 =
2     let
3     fun aux e l1 =
4         if (null l1) then
5             e
6         else
7             if (e > (hd l1)) then
8                 aux (hd l1) (tl l1)
9             else
10                aux e (tl l1)
11     in
12         aux (hd L2) (tl L2)
13     end;

```

Essa função acessa a lista e retira a cor que representa o dispositivo de maior prioridade, que é representado pelo menor valor inteiro, por exemplo 1. Caso não tenha nenhuma marca do tipo 1, procura-se o valor 2 e assim sucessivamente, do menor para o maior. Com a utilização dessa função é possível representar o método de arbitragem do barramento CAN.

No lugar **AccessDataSensorPriority** passa a marca com o valor inteiro retirado da lista de cores e é enviado para a transição *EnterDataSensor*, isso significa que o dispositivo ganhou a disputa pelo acesso ao barramento. A transição *EnterDataSensor* é temporizada com 2 unidade de tempo, que simula o tempo em que a mensagem ficará ocupando o barramento, nesse caso 2ms.

Após essa transição os lugares **CanBusBusy** e **CanBus**, representam o status do barramento CAN ocupado por uma mensagem.

Foi utilizada uma função que faz o inverso da função $F2(a,L1)$, chamada de $g(x)$. Essa função passa o valor inteiro que foi retirado da lista de cores e retorna a cor que representa esse valor, por exemplo: caso o valor do parâmetro “x” seja igual a 1, o retorno da função será a cor **Data_S1** , caso seja 2, o retorno será a cor **Data_S2**, e assim sucessivamente. Essa função implementa a operação inversa da função $F2(a,L1)$.

```

1 fun g(x) =
2     case x of 1 => Data_S1 |
3     2 => Data_S2 |
4     3 => Data_S3 |
5     4 => Data_S4 |
6     5 => Data_S5 |
7     6 => Data_S6;
```

Na transição *ExitDataSensor* o barramento é liberado, mudando o status do lugar **CanBus** para **Can_Bus_Free**, em paralelo é retirada a marca no lugar **CanBusBusy**.

Com o barramento com o status de “livre”, outras mensagens podem ser enviadas.

5.1.2.1 Coleta das Informações Durante a Simulação

Para realizar a coleta das informações durante a simulação, não foi possível utilizar a ferramenta disponível pelo Software Design CPN, chamada coletores de dados e que foi utilizada no Modelo 1. Esses coletores podem ser criados automaticamente e armazenam, diversas informações em arquivos ASCII, como: tempo de simulação, contadores, valores máximos e mínimos de marcas, além de dados estatísticos.

Uma característica de impossibilitou a utilização desse recurso no modelo multi-marca, foi que esses coletores não reconhecem as marcas utilizadas no modelo. A utilização dessa ferramenta no modelo 1 só foi possível pois para cada estrutura que representava um dispositivo CAN, foi criado um coletor de dados.

Devido a isso foram criadas diversas funções nas transições do modelo (*code segments*), utilizando a linguagem ML, para a coleta e armazenamento das informações de simulação.

A seguir serão apresentadas as implementações, como as informações são geradas e interpretadas.

5.1.2.2 Quantidade de mensagens geradas por dispositivo

Para armazenamento e análise da quantidade de mensagens geradas pelos dispositivos durante um intervalo de simulação, foi implementado um *code guard* na transição DataS1True.

Logo após a geração das mensagens pelos dispositivos, todas as marcas passam por essa transição e no momento em que a transição recebe uma marca, executa a função do *code guard*:

```
1  Input (a);
2  Action
3      AddCount(a);
4      Outfile := open\_append('AcessandoBuffer');
5      Output(!outfile, 'Bn');
6      Writetime();
7      Output(!outfile, 'Bt');
8      Writedata(a);
9      Close\_out(!outfile);
```

O *Code guard* recebe como parâmetro uma marca e inicia a execução da função. Primeiramente executa a função AddCount(a), com a finalidade de identificar qual é a marca e incrementar uma variável de referência. Por exemplo, foi definida uma variável de referência CS1, que armazena o total de mensagens geradas pelo dispositivo1 e sempre que uma marca

desse dispositivo for gerada, o valor será incrementado.

Depois do incremento dessa variável, a função abre um arquivo chamado “AcessandoBuffer.log”. Com o arquivo aberto inicia-se o processo de escrita das informações de interesse, que são: o tempo da simulação e de qual dispositivo pertence a marca gerada. Para escrita dessas informações há outras duas funções, a `writetime()` e a `writedata(a)`.

A função `writetime()`, apresentada a seguir, simplesmente escreve o tempo da simulação no arquivo:

```
1 fun writetime()=
2
3     output(!outfile,makestring(IntInf.toInt(time())));
```

- `output`: função de escrita no arquivo;
- `!outfile`: indica em qual arquivo será escrito;
- `IntInf.toInt`: faz a conversão do valor de retorno da função `time()`, que é um inteiro infinito, para um valor do tipo inteiro.

A função `writedata(a)`, recebe como parâmetro uma cor “a”, reconhece qual dispositivo gerou essa informação e escreve no arquivo. Conforme apresentado no código a seguir.

```
1 fun writedata(a) =
2     if a = Data\_S1 then output(!outfile,‘DataS1’) else
3     if a = Data\_S2 then output(!outfile,‘DataS2’) else
4     if a = Data\_S3 then output(!outfile,‘DataS3’) else
5     if a = Data\_S4 then output(!outfile,‘DataS4’) else
6     if a = Data\_S5 then output(!outfile,‘DataS5’) else
7     output(!outfile,‘DataS6’);
```

5.1.2.3 Quantidade de Mensagens no Buffer

Outro parâmetro muito importante para análise de desempenho de uma rede CAN, é a quantidade de mensagens no *buffer* durante a comunicação entre os dispositivos. O *buffer* é

uma memória para armazenamento de mensagens que não conseguiram acesso ao barramento CAN, evitando que as mesmas sejam perdidas. Porém o tamanho do *buffer* varia de acordo com a aplicação e com recursos de hardware e software, utilizados para configuração da rede de dispositivos.

Em determinadas configurações de rede, mesmo com um *buffer* para armazenar as mensagens que perderam o acesso ao barramento, pode acontecer que algumas mensagens sejam perdidas. Isso é possível quando o limite de armazenamento do *buffer* é atingido e uma nova mensagem precisa ser armazenada. Nesse caso a nova mensagem gerada irá sobrepor uma mensagem já armazenada.

A sobreposição de mensagens no *buffer* pode ocorrer com dispositivos com um tempo de latência muito elevado e/ou que possuem limitação no tamanho do *buffer*.

Para analisar o comportamento dos dispositivos durante a simulação da rede CAN, foi implementada uma função que escreve em um arquivo texto duas informações: o tempo da simulação e a quantidade de mensagens no *buffer*. Através desse arquivo é possível analisar a quantidade de mensagens que foram armazenadas no *buffer* durante todo o tempo da simulação, observar também algum intervalo de tempo crítico onde possa acontecer sobreposição das mensagens.

No Modelo de Rede de Petri Colorida, foi implementado uma função no *code guard* na transição EnterDataSensor, que escreve em um arquivo ASCII, as informações: tempo da simulação e quantidade de mensagens no *buffer*, em arquivos separados para cada sensor da rede. Para o dispositivo1 da rede, esse método gera um arquivo chamado: “S1BufferSize.log”, para o dispositivo2: “S2BufferSize.log” e assim sucessivamente para todos os dispositivos simulados.

5.1.2.4 Quantidade de Acessos ao Barramento

Outra informação importante para análise do desempenho de uma rede de dispositivos CAN é a quantidade de acessos ao barramento.

Essa informação será utilizada para analisar a taxa de ocupação do barramento du-

rante seu funcionamento, permitindo a análise da quantidade de acessos de todos os dispositivos e também analisar quanto cada dispositivo está ocupando o barramento.

Para obter essas informações, também foi implementada uma função na transição *EnterDataSensor*.

Essa função cria um arquivo contendo o tempo da simulação e a quantidade de mensagens enviadas por dispositivo. Através desse arquivo é possível obter a quantidade total de acessos e calcular a taxa de ocupação do barramento durante diferentes instantes da simulação e não somente analisar os valores finais.

5.1.2.5 Intervalo Médio de Acessos (*Jitter*)

Conforme apresentado por Hofstee e Goense (1997), quando o barramento é usado para finalidades de controle, as mensagens precisam ser entregues ao seu destinatário dentro de um intervalo de tempo definido para que o processo de controle não tenha seu desempenho prejudicado. Devido à importância da análise do tempo de espera da mensagem para ser transmitida no barramento, foi implementada uma maneira de analisar o intervalo médio de acesso por dispositivo simulado.

Na transição *EnterDataSensor*, que simula um acesso de um dispositivo ao barramento, foi implementada uma função com o objetivo de armazenar o tempo e o tipo de mensagem que está sendo enviado para o barramento.

Após a execução dessa função é criado um arquivo com o nome de “SaindoBuffer.log”. Nesse arquivo são escritas somente as duas informações do tempo da simulação e de qual dispositivo refere-se àquela mensagem.

Com o arquivo gerado no início da simulação, contendo os valores de tempo e o tipo de dispositivo, para armazenar a geração das mensagens, é possível analisar o tempo em que a informação foi gerada e comparar com o tempo em que a mesma obteve acesso ao barramento. Essa análise é realizada por dispositivos. A tabela apresenta um exemplo da saída desses arquivos.

Tabela 5.2: Estrutura do Arquivo de Análise do Jiter

Geração da mensagem	Envio no Barramento
300 DataS1	300 DataS1
300 DataS2	302 DataS2
500 DataS3	500 DataS3
500 DataS4	502 DataS4

Como pode ser observado nesse exemplo, as mensagens dos dispositivo2 e dispositivo4 obtiveram um tempo espera de 2 ms para conseguirem o acesso ao barramento. No arquivo “AcessandoBuffer”, as mensagens foram geradas no tempo 300 ms, para o dispositivo 2 e 500 ms para o dispositivo 4. E no arquivo “SaindoBuffer”, essas mensagens foram enviadas para o barramento, no tempo 302 ms e 502 ms respectivamente.

5.2 Ambiente Simulado: Aplicação da Rede CAN no Veículo Agrícola Autônomo

Esse capítulo descreve a arquitetura do VAA, possíveis ambientes de utilização e os parâmetros de configuração utilizados no barramento CAN simulado.

5.2.1 Veículo Agrícola Autônomo - VAA

Como citado anteriormente, o robô autônomo está sendo desenvolvido para diversas aplicações na área agrícola, com aplicações previstas para as áreas como: citricultura, canavieira e grãos (milho, soja). Por possuir uma aplicação voltada para a área agrícola, ambiente bastante imprevisível e que sofre constantes alterações devido as interferências climáticas, por máquinas e por homens, há necessidade de uma maior eficiência do sistema de comunicação, diminuindo ao máximo o tempo de resposta dos dispositivos. O sistema de comunicação inclui aquisição de informações, sensores e câmeras, processamentos e atuação.

A exigência do tempo de resposta pode variar de acordo com a cultura em que o robô for utilizado. Durante a utilização de um veículo autônomo uma das primeiras preocupações é com o quesito segurança, principalmente com prevenção de colisões e de acordo com o ambiente o sistema terá a necessidade de um melhor desempenho.

O trabalho de Freitas *et al.* (2006) apresenta um estudo e o projeto de desenvolvimento de uma estrutura flexível e robusta para utilização do VAA.

As imagens 5.4 e 5.5 apresentam parte dos resultados desse projeto e as dimensões do Veículo Agrícola Autônomo :

Figura 5.4: Estrutura do VAA visto de Cima

Figura 5.5: Estrutura do VAA visto de Frente

Como pode ser observado na figura 5.4, com uma visão superior do projeto, o VAA

possui 2,6 metros de comprimento e a largura ajustável, podendo variar entre 1,5 a 2,6 metros.

A figura 5.5 apresenta outra visão do projeto do VAA que demonstra a altura projetada de cerca de 2 metros e as hastes ajustáveis da largura do veículo. Essa possibilidade de ajustes na largura do VAA é considerada uma característica muito importante por permitir sua utilização em diversas culturas diferentes, pois de acordo com a cultura, as linhas de plantio possuem espaçamentos diferentes. Como por exemplo: O espaçamento entre as linhas de plantio na citricultura fica em torno de 5 metros, com espaçamento entre as plantas de 3 metros (figura5.6).

Figura 5.6: Linha de Plantio na Citricultura

De acordo com as especificações dos motores utilizados no robô, a velocidade média será de 1 Km/h , aproximadamente 0,3 m/s.

5.2.2 Barramento CAN

Esse trabalho está vinculado ao projeto: “Veículo Agrícola Autônomo (VAA): uma plataforma para desenvolvimento de tecnologias de navegação autônoma e para aquisição de dados em Agricultura de Precisão”, e possui o enfoque principal voltado para a análise do comportamento, do tráfego de informações e na utilização da rede de dispositivos no VAA, baseada no protocolo CAN. O modelo de simulação para análise da rede CAN foi desenvolvido utilizando como referência a arquitetura da rede de comunicação que foi proposta, e que está sendo construída, para a o VAA.

Como apresentado em Godoy, Sousa e Inamasu (2005), para montagem da rede de

dispositivos CAN no VAA está prevista a utilização de diversos dispositivos, cerca de aproximadamente 22 dispositivos, entre sensores de ultra-som, bússola digital, câmeras de vídeo, microcomputador, entre outros.

Todos os dispositivos utilizados no VAA podem ser classificados em quatro grupos:

- Sensores
- Atuadores
- Gerenciamento da Rede de Comunicação
- Armazenamento de Dados

Com base nessas informações, foi montado um modelo de simulação contendo 32 dispositivos, com diferentes parâmetros de configuração para cada ambiente simulado.

A figura 5.7 apresenta um exemplo do Barramento CAN com alguns dispositivos que possuem a utilização prevista no VAA.

Figura 5.7: Barramento CAN

Para a configuração dos parâmetros de comunicação do protocolo CAN foi utilizado o modelo matemático apresentado por Tindell, Hansson e Wellings (1994), e aperfeiçoado por Punnenkkat, Hansson e Norstrom (2000), o qual apresenta que o tempo de atraso (*deadline*) de uma mensagem m (chamado de D_m) nunca deve ser maior que o período da mensagem (chamado de T_m).

O tempo de resposta de uma mensagem m (R_m) é definido como o maior tempo entre a geração e enfileiramento da mensagem, e o último tempo de entrega da mensagem no seu destinatário.

O tempo de transmissão (R_m) de uma mensagem m em um barramento CAN pode ser definida pela equação 5.1 .

$$R_m = J_m + W_m + C_m \quad (5.1)$$

O termo J_m representa o tempo médio de espera da mensagem na fila para transmissão (J_{iter}).

O termo W_m representa o atraso na fila no pior caso, isto é, o tempo maior entre a inserção da mensagem na fila de prioridades e o início do envio, ou transmissão.

O termo C_m representa o tempo gasto para a transmissão de uma mensagem m fisicamente sobre o barramento.

A equação 5.2 apresenta C_m :

$$C_m = \left(\left\lceil \frac{34 + 8s_m}{5} \right\rceil + 47 + 8s_m \right) \tau_{bit} \quad (5.2)$$

O termo S_m representa o tamanho da mensagem m em bytes. O termo τ_{bit} representa o tempo do bit no barramento, por exemplo: no barramento configurado com taxa de comunicação de 1Mbit/seg, é $1 \mu s$

O tempo de espera é definido pela equação 5.3:

$$w_m = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m + J_j + \tau_{bit}}{T_j} \right\rceil C_j \quad (5.3)$$

O termo $hp(m)$ representa o conjunto das mensagens de prioridade alta. T_j é a

frequência da mensagem j . B_m é o maior tempo que uma mensagem m pode ser atrasada por mensagens de prioridade baixa, e pode ser definido pela equação 5.4:

$$B_m = \max_{\forall k \in lp(m)} (C_k) \quad (5.4)$$

Onde $lp(m)$ é o conjunto de mensagens de prioridades baixas.

Na equação 5.3 o termo w_m aparece duas vezes, em ambos os lados da equação, e a equação não pode ser reescrita nos termos de w_m . Uma solução possível é desenvolver uma forma de recursão:

$$w_m^{n+1} = B_m + \sum_{\forall j \in hp(m)} \left[\frac{w_m^n + J_j + \tau_{bit}}{T_j} \right] C_j \quad (5.5)$$

Com base no modelo matemático apresentado anteriormente, a configuração do barramento CAN simulado utiliza a versão *Extended* do protocolo CAN, contendo entre outros campos, um quadro de identificação de 29 bits, o quadro de dados com 64 bits. A taxa de comunicação do barramento configurada em 250 Kbps, com o tamanho total de 12 metros. Utilizando essas configurações, o tempo em que uma mensagem ocupa o barramento durante uma transmissão é de 0.641 mseg.

Entre os três ambientes simulados, foram alterados basicamente dois parâmetros: os identificadores dos dispositivos no barramento e as frequências com que as informações são geradas, e os resultados foram comparados e serão apresentados a seguir.

6 RESULTADOS E DISCUSSÕES

Foram realizadas simulações de uma rede CAN contendo 32 dispositivos com diferentes configurações de endereços, conseqüentemente diferentes prioridades e frequências de acesso ao barramento. O tempo total de simulação foi de 10 segundos.

6.1 Configuração 1

Durante a primeira simulação, utilizou-se uma abordagem apresentada por (GODOY; SOUSA; INAMASU, 2005), que sugere a montagem de uma rede CAN contendo 22 dispositivos, onde os dispositivos com prioridades maiores, sejam configurados com frequências mais elevadas. A partir dessa especificação, foram incluídos na rede de simulação outros 10 dispositivos genéricos.

A tabela 6.1 apresenta a configuração da rede CAN para simulação 1.

Utilizando a configuração apresentada na tabela 6.1, foi realizada uma simulação para análise de alguns parâmetros importantes de uma rede CAN: Taxa de ocupação do barramento, Taxa de Ocupação do Barramento por dispositivo, total de mensagens enviadas para o barramento, quantidade de mensagens enviadas por dispositivo, tempo de espera das mensagens no *buffer* de cada dispositivo.

6.1.1 Taxa de Ocupação do Barramento

Conforme apresentado no gráfico 6.1 a taxa de ocupação do barramento foi calculada nos intervalos de 1 segundo. Pode-se observar que no primeiro segundo o barramento teve

Tabela 6.1: Configuração da Rede CAN - Simulação 1

Descrição do Dispositivo	Frequência (ms)	Pacote de Dados (bits)	Prioridade
Controle de Motor de Propulsão 1	50	64	1
Controle de Motor de Propulsão 2	50	64	2
Controle de Motor de Propulsão 3	50	64	3
Controle do Motor de Propulsão 4	50	64	4
Controle do Motor de Guiagem 1	50	64	5
Controle de Motor de Guiagem 2	50	64	6
Controle de Motor de Guiagem 3	50	64	7
Controle de Motor de Guiagem 4	50	64	8
Velocidade / Encoder do Motor de Propulsão 1	100	64	9
Velocidade / Encoder do Motor de Propulsão 2	100	64	10
Velocidade / Encoder do Motor de Propulsão 3	100	64	11
Velocidade / Encoder do Motor de Propulsão 4	100	64	12
Deslocamento Angular do Motor de Propulsão 1	100	64	13
Deslocamento Angular do Motor de Propulsão 2	100	64	14
Deslocamento Angular do Motor de Propulsão 3	100	64	15
Deslocamento Angular do Motor de Propulsão 4	100	64	16
Conjunto de Sensores Ultra-Som Lado Direito	150	64	17
Conjunto de Sensores Ultra-Som Lado Esquerdo	150	64	18
Posicionamento GPS	150	64	19
Localização Bússola Digital	150	64	20
Movimentação da Câmera de Navegação	150	64	21
Controle do PC Industrial	150	64	22
Gerenciamento e Monitoramento dos Dados	250	64	23
Gerenciamento e Monitoramento dos Dados	250	64	24
Administracao de Rede	250	64	25
Dispositivo Genérico 1	250	64	26
Dispositivo Genérico 2	250	64	27
Dispositivo Genérico 3	500	64	28
Dispositivo Genérico 4	500	64	29
Dispositivo Genérico 5	500	64	30
Dispositivo Genérico 6	1000	64	31
Dispositivo Genérico 7	1000	64	32

Figura 6.1: Taxa de Ocupação do Barramento CAN - Simulação 1

a menor taxa de ocupação de toda a simulação, 17,88%. A partir de 2 segundos do funcionamento da rede, a taxa de ocupação do barramento sofreu pouca variação alternando-se entre 19,87% e 19,49% até o tempo final de 10 segundos.

A configuração utilizada não prejudicou o desempenho do barramento CAN, mantendo baixa a taxa de ocupação do barramento durante toda a simulação.

6.1.2 Taxa de Ocupação do Barramento por Dispositivo

Figura 6.2: Taxa de Ocupação do Barramento CAN - Simulação 1

O gráfico 6.2 apresenta o taxa de ocupação do barramento de cada dispositivo instalado na rede.

Os dispositivos configurados com prioridades e frequências de acessos maiores consequentemente foram responsáveis por uma taxa de ocupação do barramento maior. Como pode ser notado do dispositivo 1 ao dispositivo 8 a taxa de ocupação de cada um deles foi de 1,28%.

Do dispositivo 9 ao 16 a taxa de ocupação ficou em torno de 0,63% e quanto menor a prioridade do dispositivo e maior a frequência de acesso as taxas foram menores como, por exemplo os dispositivos 31 e 32 que foram responsáveis por apenas 0,06% da ocupação total do barramento.

6.1.3 Quantidade de Mensagens Enviadas por Dispositivo

Figura 6.3: Quantidades de Mensagens Enviadas por Dispositivos - Simulação 1

O gráfico 6.3 apresenta a mesma relação entre os dispositivos da rede CAN, suas frequências e prioridades, como o gráfico 6.2.

O eixo *X* representa a prioridade de cada dispositivo, com valores inteiros de 1 à 32, e o eixo *Y* representa a quantidade de mensagens enviadas durante os 10 segundos de execução da simulação da rede CAN.

Os oito primeiros dispositivos, com as frequências de acessos configuradas em 50 milissegundos, enviaram 199 mensagens durante os 10 segundos de simulação. À partir do dispositivo 9 ao dispositivo 16, configurados com frequências de acesso de 100 milissegundos, enviaram 99 mensagens.

Os dispositivos 31 e 32 enviaram apenas 9 mensagens.

6.1.4 Quantidade de Mensagens no Buffer x Tempo de Espera

Tabela 6.2: Tempo de Espera das Mensagens - Simulação 1

Prioridade do Dispositivo	Tempo Médio de Espera (ms)	Qtde Mensagens no Buffer
1	0.95	1
2	1.41	1
3	1.60	1
4	2.19	1
5	2.82	1
6	3.35	1
7	3.68	1
8	4.30	1
9	3.68	1
10	4.42	1
11	4.87	1
12	5.56	1
13	7.44	1
14	7.85	1
15	8.56	1
16	9.13	1
17	7.49	1
18	8.21	1
19	8.70	1
20	9.48	1
21	10.01	1
22	10.18	1
23	8.47	1
24	9.48	1
25	9.97	1
26	10.05	1
27	11.30	1
28	14.77	1
29	15.41	1
30	16.05	1
31	16.60	1
32	15.64	1

A tabela 6.2 apresenta a análise de dois parâmetros importantes das redes de comu-

nicação para sistemas de tempo-real, tempo de espera das mensagens e quantidade máxima de mensagens armazenadas no *Buffer*. Esses dois parâmetros podem comprometer o desempenho do sistema de controle gerando erros de comunicação. Para aplicação do VAA a queda de performance pode causar, por exemplo, acidentes ou colisões considerando perdas ou atrasos nas mensagens de controle ou navegação.

Durante essa simulação todos os dispositivos armazenaram somente 1 mensagem no *buffer*, isso significa que nenhuma mensagem foi sobreposta durante o funcionamento da rede CAN. A perda de uma mensagem pode ocorrer por diferentes motivos como, por exemplo: erros de comunicação (envio e recepção) e sobreposição de mensagens no *buffer*. A sobreposição das mensagens ocorre quando a capacidade máxima de armazenamento de mensagens no *buffer* é atingida, de forma que uma mensagem armazenada é descartada e substituída por nova mensagem gerada.

6.2 Configuração 2

Após a análise dos resultados dessa primeira simulação, foi criada outra simulação da rede CAN utilizando uma configuração diferente para as prioridades dos dispositivos. Para a segunda simulação os dispositivos com frequências menores de acessos ao barramento foram configurados com prioridades mais altas e conseqüentemente, os dispositivos com frequências mais elevadas foram configurados com prioridades menores.

A seguir, na tabela 6.3, será apresentada a configuração dos dispositivos da rede CAN durante a segunda simulação.

Utilizando a configuração apresentada na tabela 6.3, realizou-se a simulação da comunicação entre os dispositivos da rede e os resultados serão apresentados a seguir.

6.2.1 Taxa de Ocupação do Barramento

O gráfico 6.4 apresenta da taxa de ocupação do barramento CAN para essa segunda configuração.

Tabela 6.3: Configuração da Rede CAN - Simulação 2

Descrição do Dispositivo	Frequência (ms)	Pacote de Dados (bits)	Prioridade
Dispositivo Genérico 7	1000	64	1
Dispositivo Genérico 6	1000	64	2
Dispositivo Genérico 5	500	64	3
Dispositivo Genérico 4	500	64	4
Dispositivo Genérico 3	500	64	5
Dispositivo Genérico 2	250	64	6
Dispositivo Genérico 1	250	64	7
Administração de Rede	250	64	8
Gerenciamento e Monitoramento dos Dados	250	64	9
Gerenciamento e Monitoramento dos Dados	250	64	10
Controle do PC Industrial	150	64	11
Movimentação da Câmera de Navegação	150	64	12
Localização Bússola Digital	150	64	13
Posicionamento GPS	150	64	14
Conjunto de Sensores Ultra-Som Lado Esquerdo	150	64	15
Conjunto de Sensores Ultra-Som Lado Direito	150	64	16
Deslocamento Angular do Motor de Propulsão 4	100	64	17
Deslocamento Angular do Motor de Propulsão 3	100	64	18
Deslocamento Angular do Motor de Propulsão 2	100	64	19
Deslocamento Angular do Motor de Propulsão 1	100	64	20
Velocidade / Encoder do Motor de Propulsão 4	100	64	21
Velocidade / Encoder do Motor de Propulsão 3	100	64	22
Velocidade / Encoder do Motor de Propulsão 2	100	64	23
Velocidade / Encoder do Motor de Propulsão 1	100	64	24
Controle de Motor de Guiagem 4	50	64	25
Controle de Motor de Guiagem 3	50	64	26
Controle de Motor de Guiagem 2	50	64	27
Controle do Motor de Guiagem 1	50	64	28
Controle do Motor de Propulsão 4	50	64	29
Controle de Motor de Propulsão 3	50	64	30
Controle de Motor de Propulsão 2	50	64	31
Controle de Motor de Propulsão 1	50	64	32

Figura 6.4: Quantidade de Mensagens Enviadas por Dispositivos - Simulação 2

Como pode ser observado no gráfico 6.1 a taxa de ocupação do barramento não sofreu alteração em relação a configuração anterior (Simulação 1). Pode-se observar nessa segunda simulação a menor taxa de ocupação foi registrada no tempo de 1 segundo, 17,88%. A partir de 2 segundos do funcionamento da rede, a taxa de ocupação do barramento sofreu pouca variação alternando-se entre 19,87% e 19,49% até o tempo final de 10 segundos de análise.

A configuração utilizada não prejudicou o desempenho do barramento CAN, mantendo baixa a taxa de ocupação do barramento durante toda a simulação.

6.2.2 Taxa de Ocupação do Barramento por Dispositivo

A figura 6.5 apresenta a taxa de ocupação do barramento por dispositivo durante a comunicação da rede CAN para a segunda configuração dos dispositivos.

Figura 6.5: Taxa de Ocupação do Barramento por Dispositivos - Simulação 2

Através desse gráfico fica claro a alteração das prioridades dos dispositivos instalados na rede, pois os dispositivos com menores prioridades ocuparam a maior parte das comunicações do barramento. Todos os dispositivos foram responsáveis pela mesma taxa de ocupação do barramento durante a primeira simulação.

6.2.3 Quantidade de Mensagens Enviadas por Dispositivo

A figura 6.6 apresenta a quantidade de mensagens enviadas pelos dispositivos na segunda configuração da rede CAN.

Todos os dispositivos mantiveram todos os envios das mensagens da primeira configuração. Nesse gráfico pode ser notado que os valores não sofreram nenhuma alteração, como o parâmetro de taxa de ocupação do barramento por dispositivo. A partir do dispositivo 25 ao 32, cada um foi responsável pelo envio de 199 mensagens. Já os dispositivos 17 ao 23 foram responsáveis pelo envio de 99 mensagens.

Os dispositivos 1 e 2 enviaram apenas 9 mensagens cada.

Figura 6.6: Quantidades de Mensagens Enviadas por Dispositivos - Simulação 2

6.2.4 Quantidade de Mensagens no Buffer x Tempo de Espera

A tabela 6.4 apresenta os resultados do tempo médio de espera das mensagens e a quantidade máxima de mensagens armazenadas no buffer.

Tabela 6.4: Tempo de Espera das Mensagens - Simulação 2

Prioridade do Dispositivo	Tempo Médio de Espera (ms)	Qtde Mensagens no Buffer)
1	1.15	1
2	1.67	1
3	1.65	1
4	2.11	1
5	2.93	1
6	2.09	1
7	2.65	1
8	2.83	1
9	3.06	1
10	4.16	1
11	1.76	1
12	2.14	1
13	2.48	1
14	2.89	1
15	3.68	1
16	4.01	1
17	2.82	1
18	3.45	1
19	3.89	1
20	4.29	1
21	4.94	1
22	5.51	1
23	6.06	1
24	6.56	1
25	4.79	1
26	5.20	1
27	5.54	1
28	6.47	1
29	7.10	1
30	7.47	1
31	8.34	1
32	8.84	1

Todos os dispositivos armazenaram no máximo uma mensagem no *buffer* de memória. Com isso pode-se concluir que não houve sobreposições e perdas de mensagens durante a comunicação do barramento CAN.

Os tempos de espera das mensagens no *buffer* foram relativamente baixos em relação a configuração 1 da rede CAN. O menor tempo de espera foi de 1.15 milissegundos (dispositivo 1), e o maior tempo de espera ficou em torno de 8.84 milissegundos para o dispositivo 32 configurado, com a menor prioridade.

6.3 Comparativo entre as Configurações 1 e 2

Como pode ser notado nos gráficos 6.7 e 6.8 os resultados das análises de taxa de ocupação do barramento, taxa de ocupação do barramento por dispositivos e consequentemente a quantidade de mensagens enviadas por dispositivos, não sofreram alteração nos valores. Pode ser notada apenas alteração nas configurações das prioridades dos dispositivos.

Figura 6.7: Comparativo da quantidade de Mensagens Enviadas

Figura 6.8: Comparativo da Taxa de Ocupação do Barramento

O único parâmetro que sofreu alteração significativa entre as duas simulações foi o tempo de espera das mensagens no *buffer*.

A tabela 6.5, que apresenta o comparativo do tempo médio de espera das mensagens para serem enviadas, demonstra que o tempo médio de espera durante a simulação 2 foi consideravelmente menor que os tempos apresentados na simulação 1.

Tabela 6.5: Comparativo Tempo de Espera das Mensagens Simulação 1 Vs Simulação 2

Prioridade do Dispositivo	Espera Simulação 1 (ms)	Espera Simulação 2 (ms)
1	0,95	1,15
2	1,41	1,67
3	1,60	1,65
4	2,19	2,11
5	2,82	2,93
6	3,35	2,09
7	3,68	2,65
8	4,30	2,83
9	3,68	3,06
10	4,42	4,16
11	4,87	1,76
12	5,56	2,14
13	7,44	2,48
14	7,85	2,89
15	8,56	3,68
16	9,13	4,01
17	7,49	2,82
18	8,21	3,45
19	8,70	3,89
20	9,48	4,29
21	10,01	4,94
22	10,18	5,51
23	8,47	6,06
24	9,48	6,56
25	9,97	4,79
26	10,05	5,20
27	11,30	5,54
28	14,77	6,47
29	15,41	7,10
30	16,05	7,47
31	16,60	8,34
32	15,64	8,84

6.4 Configuração 3

Para essa análise de desempenho foi utilizada a abordagem de Hofstee e Goense (1999) como referência. Nesse trabalho foram realizadas algumas análises do protocolo CAN através um programa de simulação desenvolvido utilizando a linguagem Personal Prosim. Esse programa foi escrito para simulação do funcionamento de uma rede CAN em uma máquina agrícola, mais especificamente um pulverizador, equipada com três barramentos de dados (Barramento do Trator , do Implemento e o barramento Proprietário de um pulverizador). Esse trabalho foi desenvolvido para analisar o desempenho da rede CAN através dois parâmetros: taxa de ocupação do barramento e tempo de espera das mensagens.

A utilização dessa abordagem diferente para análise do protocolo CAN tem como objetivo comparar os resultados de um trabalho já desenvolvido e publicado, em um ambiente diferente do utilizado como referência no desenvolvimento desse modelo de Rede de Petri.

A figura 6.9 e a tabela 6.6 demonstram o ambiente simulado que foi apresentado em (HOFSTEE; GOENSE, 1999).

Figura 6.9: Configuração de uma Rede CAN (HOFSTEE; GOENSE, 1999)

Todos os parâmetros do barramento CAN, como frequência de comunicação de 250 Kbps e pacote de dados de 8 bytes foram mantidos para essa análise.

Com base nessas configurações serão apresentadas as análises de todos os resultados da simulação.

6.4.1 Taxa de Ocupação do Barramento

A figura 6.10 apresenta a análise da taxa de ocupação do barramento durante os 10 segundos de simulação.

A menor taxa de ocupação foi registrada no primeiro segundo, quando foi realizada

Tabela 6.6: Configuração da Rede CAN - Simulação 3

Descrição do Dispositivo	Frequências	Pacote de Dados (bits)	Prioridade
Controle Eletrônico do Motor 2 : EEC2	50 ms	64	1
Controle Eletrônico do Motor 1 : EEC1	30 ms	64	2
Óleo do Motor Nível/Pressão 2	5 s	64	3
Turbo Alimentador	1 s	64	4
Controle Eletrônico do Motor 1 : EEC1	250 ms	64	5
Configuração do Motor	5 s	64	6
Temperatura do Motor	1 s	64	7
Óleo do Motor Nível/Pressão	5 s	64	8
Condições de Exaustão	5 s	64	9
Pressão auxiliar da bomba de água	1 s	64	10
Controle de Velocidade 1 : TSC1	10 ms	64	11
Controle de Velocidade 1 : TSC1	50 ms	64	12
Controle da Transmissão Eletrônica 1: ETC1	10 ms	64	13
Controle da Transmissão Eletrônica 2: ETC2	100 ms	64	14
Óleo da Transmissão	1 s	64	15
Comando do Engate e PTO	100 ms	64	16
Eixo de saída do PTO traseiro	100 ms	64	17
Informações do PTO	100 ms	64	18
Controlador Eletrônico do Eixo 1 : EAC1	500 ms	64	19
Informações do Eixo	1 s	64	20
Controlador Eletrônico do Freio 1 : EBC1	100 ms	64	21
Freios	1 s	64	22
Controle Eletrônico 1 : ERC1	100 ms	64	23
Fluído Arrefecimento	1 s	64	24
Controle da Transmissão 1 : TC1	50 ms	64	25
Controle da Transmissão 1 : TC1	50 ms	64	26
Velocidade e Distância	100 ms	64	27
Controle Direcional/Velocidade	100 ms	64	28
Economia de Combustível	100 ms	64	29
Combustível Alternativo 1	500 ms	64	30
Indicador de Água e Combustível	10 s	64	31
Condições Ambientais	1 s	64	32
Posição do Veículo	5 s	64	33
Energia Elétrica do Veículo	1 s	64	34
Comando de Válvula Auxiliar	100 ms	64	35
Status da Válvula Auxiliar	100 ms	64	36
Comandos do PTO e engates	100 ms	64	37
Status do engate traseiro	100 ms	64	38
Status do engate dianteiro	100 ms	64	39
Comando PTO e engate	100 ms	64	40
Eixo de saída do PTO dianteiro	100 ms	64	41
Informações do PTO	100 ms	64	42
Processo de configuração	26 ms	64	43
Posição do Implemento	5 s	64	44
Posição	250 ms	64	45
Configuração do atuador	5 ms	64	46
Valor do Sensor	33 ms	64	47
Valor do Sensor	33 ms	64	48
Valor do Sensor	33 ms	64	49
Valor do Sensor	33 ms	64	50
Configuração do atuador	5 ms	64	51
Valor do Sensor	33 ms	64	52
Valor do Sensor	33 ms	64	53
Valor do Sensor	33 ms	64	54
Valor do Sensor	33 ms	64	55
Configuração do atuador	250 ms	64	56
Valor do Sensor	33 ms	64	57

a primeira análise (75,51%). À partir da segunda análise, 2 segundos de simulação, a taxa de ocupação sofreu pequenas variações, entre 77% e 78%.

O barramento CAN teve uma taxa de utilização quase 4 vezes maior em relação as outras configurações analisadas, isso se justifica, logicamente, pelo número maior de sensores

Figura 6.10: Taxa de Ocupação do Barramento - Simulação 3

(57) e por possuir dispositivos configurados com frequências de acessos ao barramento muito elevadas (caso dos dispositivos 46 e 51 cerca de 5 milissegundos).

6.4.2 Taxa de Ocupação do Barramento por Dispositivo

O parâmetro de taxa de ocupação do barramento por dispositivo pode ser analisado no gráfico 6.11.

Figura 6.11: Taxa de Ocupação do Barramento por Dispositivo - Simulação 3

A menor taxa de ocupação de um dispositivo foi de 0,01%, registrada pelos dispositivos 4, 7, 15, 20 e outros dispositivos configurados com frequências de acessos de 1 segundo.

Os dispositivos 46 e 51, configurados com as maiores frequências de acessos ao barramento (5 milissegundos), foram responsáveis por 14,10% da ocupação do barramento.

A taxa de ocupação ficou em torno de 7,04% para os dispositivos 11 e 13, ambos

com frequências de acessos de 10 milissegundos.

6.4.3 Quantidade de Mensagens Enviadas por Dispositivo

Figura 6.12: Quantidade de Mensagens Enviadas por Dispositivo - Simulação 3

O gráfico 6.12 apresenta a quantidade de mensagens enviadas por dispositivo. Conforme apresentado no gráfico 6.11, os dispositivos 46 e 51 foram responsáveis por maior parte da ocupação do barramento em consequência de um total de 2199 mensagens enviadas cada.

Os dispositivos 11 e 13 enviaram um total de 1099 mensagens cada.

6.4.4 Quantidade de Mensagens no Buffer x Tempo de Espera

A tabela 6.7 apresenta o número máximo de mensagens armazenadas no buffer de memória dos dispositivos e o tempo médio de espera dessas mensagens no buffer.

Todos os dispositivos armazenaram apenas 1 ou 2 mensagens no *buffer*, com exceção dos dispositivos 46 (que atingiu um total de 7 mensagens armazenadas), o dispositivo 51 (com um total de 9 mensagens armazenadas) e o dispositivo 57 (com um total de 3 mensagens).

De acordo com a capacidade de armazenamento de mensagens no *buffer* dos dispositivos, seja por hardware ou software, esses valores podem representar sobreposição de mensagens em determinados intervalos de tempo e consequentemente perda de informações.

A menor média de tempo de espera das mensagens no *buffer* foi do dispositivo 2, com 0.95 milissegundos, enquanto que as mensagens do dispositivo 44 tiveram a maior média

Tabela 6.7: Tempo de Espera das Mensagens - Simulação 3

Prioridade do Dispositivo	Tempo Médio de Espera (ms)	Qtde Mensagens no Buffer)
1	1.00	1
2	0.95	1
3	1.74	1
4	1.89	1
5	1.89	1
6	3.35	1
7	2.78	1
8	4.63	1
9	5.27	1
10	3.87	1
11	1.08	1
12	2.26	1
13	1.66	1
14	3.84	1
15	6.31	1
16	4.29	1
17	4.87	1
18	5.12	1
19	7.85	1
20	10.41	1
21	6.37	1
22	12.59	1
23	7.02	1
24	14.13	1
25	5.41	1
26	5.91	1
27	8.76	1
28	9.40	1
29	9.92	1
30	14.53	1
31	22.72	1
32	18.74	1
33	23.22	1
34	19.64	1
35	11.57	1
36	12.43	1
37	13.51	1
38	14.45	1
39	14.93	1
40	15.53	1
41	15.61	1
42	17.04	1
43	2.60	1
44	30.91	1
45	13.68	1
46	3.55	7
47	3.31	1
48	3.99	2
49	4.98	2
50	5.49	1
51	5.89	9
52	8.42	2
53	9.85	2
54	11.49	2
55	13.39	2
56	29.51	1
57	14.90	3

do tempo de espera, cerca de 30.91 milissegundos.

7 CONCLUSÕES E SUGESTÕES

Neste capítulo apresenta-se as conclusões desse trabalho e algumas sugestões para trabalhos futuros.

7.0.5 Conclusões

O modelo da rede CAN desenvolvido baseado em Rede de Petri Colorida constitui uma ferramenta para ser utilizada em projetos de redes embarcadas. A simulação com diferentes parâmetros permitiu a verificação do modelo.

Além de fornecer os recursos necessários para análise de desempenho das redes de comunicação, o modelo pode ser considerado bastante flexível, pois permite que dispositivos possam ser inseridos ou excluídos do ambiente configurado, sem exigir esforço maior de edição do modelo da Rede de Petri.

Todos os parâmetros de desempenho da rede de comunicação CAN previstos puderam ser analisados, demonstrando que os recursos de programação e de modelagem disponíveis no software DesignCPN e no formalismo da Rede de Petri Colorida atingiram as expectativas e satisfizeram todas as necessidades do trabalho.

As análises através de simulações permitiram verificar o desempenho da rede de comunicação, demonstrando as consequências nas alterações das prioridades dos dispositivos no parâmetro de tempo de resposta. Com isso, mostrou-se uma ferramenta capaz de facilitar a definição de prioridade e a determinação do tempo de resposta para cada mensagem ou dispositivo, em diversas configurações sem a necessidade da implementação física.

Através do modelo desenvolvido foi possível analisar o comportamento da rede CAN não somente no ambiente previsto do VAA, mas também em outro ambiente com aplicação para área agrícola, demonstrando flexibilidade e suporte para diferentes áreas de aplicação.

Por fim, todos os objetivos definidos nesse trabalho foram atingidos .

7.0.6 Sugestões para Trabalhos Futuros

Como sugestão para trabalhos futuros destacam-se:

Desenvolvimento de novas simulações utilizando ambientes diferentes, ampliando os parâmetros configuráveis como, por exemplo: dispositivos com diferentes tamanhos de pacotes em uma mesma simulação, criação de mensagens de Erros acíclicas e comunicações entre dispositivos de duas ou mais redes CAN.

Ampliação do modelo de simulação, criando sub-modelos que representem outros processos existentes na comunicação do protocolo CAN como, por exemplo: a disputa e troca de endereços entre os dispositivos durante a comunicação.

Implementação de uma rede CAN para que possa ser realizada uma análise comparativa entre os resultados da comunicação CAN simulada e os resultados extraídos da comunicação em um barramento “real”.

REFERÊNCIAS

- ACLE, J. P.; REORDA, M. S.; VIOLANTE, M. Early, accurate dependability analysis of can-based networked systems. *IEEE Design & Test of Computers*, v. 23, p. 38–45, Fevereiro 2006.
- AUERNHAMMER, H. Precision farming: the environmental challenge. *Computer and Eletronics in Agriculture*, v. 30, p. 31–43, 2001.
- BAO, G.; LIU, F.; HONG, L. Modeling and verification of tcp congestion control based on coloured petri nets. *IEEE SMC*, p. 1045–1050, 2003.
- BOSCH, R. *Robert Bosch*. www.bosch.de, Fevereiro 2003.
- CAVALIERI, S.; DISTEFANO, A.; BELLO, L. L.; MIRABELLA, O. Can assessment in time-critical cyclic applications through petri net model. *IEEE IECON 22nd International Conference on Industrial Electronics*, v. 2, n. 5-10, p. 922–927, Aug 1996.
- CENA, G.; VALENZANO, A. Delay analysis of priority promotion systems. *Computer Communications*, v. 23, p. 1252–1262, 2000.
- CHRISTENSEN, S.; JORGENSEN, J. B.; KRISTENSEN, L. M. Design/cpn - a computer tool for coloured petri nets. *Proceedings of the Third International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, v. 127, p. 209 – 223, 1997.
- CPNML. *CPN ML language for declarations and net incriptions*. http://wiki.daimi.au.dk/cpntools-help/cpn_ml.wiki, Fevereiro 2006.
- DARR, M. J.; STOMBAUGH, T. S.; SHEARER, S. A. Controller area network based distributed control for autonomous vehicles. *American Society of Agricultural and Biological Engineers*, v. 48(2), p. 479–490, 2005.
- DESIGNCPN. *Design CPN - Computer Tool for Coloured Petri Nets*. <http://www.daimi.au.dk/designCPN/>, Março 2006.
- DESROCHERS, A. A.; AL-JAAR, R. Y. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis*. [S.l.]: IEEE Press, 1995.
- FREDRIKSSON, L. B. *Distributed Embedded Control Systems in Robotics - Ten Years of Development*. <http://www.kvaser.com>, 1997.
- FREITAS, R. R.; SOUSA, R. V.; SAKAI, R. M. R.; PORTO, A. J. V.; INAMASU, R. Y. Aplicação de ferramentas de prototipagem virtual para desenvolvimento de uma plataforma agrícola robótica. 2º Congresso Brasileiro de Agricultura de Precisão, 2006.
- GERMAN, R.; KELLING, C.; SIMMERMAN, A.; HOMMEL, G. Timenet: a toolkit for evaluating non-markovian stochastic petri nets. *Perfomance Evaluation*, v. 24, p. 69–87, 1995.

- GIRAULT, C.; VALK, R. Petri nets for systems engineering: A guide for modeling, verification and application. *Springer Verlag*, 2003.
- GODOY, E. P.; SOUSA, R. V.; INAMASU, R. Y. Software para simulação de redes can aplicadas em máquinas agrícolas. *Congresso Brasileiro de Agroinformática- SBI-AGRO*, 2005.
- GUIMARÃES, A. A. *Análise da Norma ISO11783 e sua Utilização na Implementação do Barramento do Implemento de um Monitor de Semeadora*. Dissertação (Dissertação) — Escola Politécnica da Universidade de São Paulo, 2003.
- GUIMARÃES, A. A.; SARAIVA, A. M. O protocolo can: Entendendo e implementando uma rede de comunicação serial de dados baseada no barramento "controller area network". *Congresso SAE Brasil*, v. 11, n. SAE 2002-01-3569, 2002.
- HOFSTEE, J. W.; GOENSE, D. Simulation of a can-based tractor-implement field bus according to din 9684. *Journal of Agricultural Engineering Research*, v. 68, p. 89–100, May 1997.
- HOFSTEE, J. W.; GOENSE, D. Simulation of a controller area network-based tractor - implement data bus according to iso 11783. *Journal of Agricultural Engineering Research*, v. 73, p. 383–394, March 1999.
- HPSIM. www.winpesim.de, Novembro 2005.
- ISO11519-1. *Road vehicles Low speed serial data communication - Part 1: General and definitions*. [S.l.], 1994.
- ISO11519-2. *Road vehicles Low speed serial data communication - Part 2: Low-speed controller area network (CAN)*. [S.l.], 1994.
- ISO11898. *Road-vehicle- interchange of digital information - controller area network (can) for high-speed communication*. ISO, 1993.
- JAIN, R. K. *The art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. [S.l.]: Wiley- Interscience, 1991.
- JENSEN, K. *Coloured Petri Nets basic Concepts, Analysis Methods and Pratical Use*. Second edition. [S.l.: s.n.], 1996.
- JENSEN, K. *Coloured Petri Nets basic Concepts, Analysis Methods and Pratical Use*. Second edition. [S.l.: s.n.], 1997.
- JENSEN, K. *Overview of Design/CPN*. <http://www.daimi.aau.dk/designCPN/>, Nov 2005.
- JENSEN, K.; KRISTENSEN, L.; WELLS, L. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools*, 2006.
- JEON, J. M.; KIM, D. W.; CHO, Y. J.; LEE, B. H. An analysis of network-based control system using can (controller area network) protocol. In: . [S.l.]: IEEE International Conference on Robotics & Automation, 2001. p. 3577–3581.
- K.TINDELL; BURNS, A. Guaranteeing message latencies on control area network (can). *CAN Conference*, Setembro 1994.

- LAI, R. Performance modelling for the csma/cd protocol using gspn. *IEEE Catalogue No.95TH8061*, v. 95TH8061, p. 126–130, 1995.
- LAI, R. Performance results for the csma/cd protocol using greatspn. *J. Systems Software*, v. 37, p. 75–90, 1997.
- LOPES, W. C.; CARVALHO, H. J. R.; SOUSA, R. V.; INAMASSU, R. Y.; PORTO, A. J. V. Modelagem e simulação de uma rede can para aplicação na área agrícola utilizando rede de petri coloridas (cpns). In: *V Congresso Brasileiro de Agroinformatica, SBI-AGRO*. [S.l.: s.n.], 2005.
- MACIEL, P. R. *Introdução as Redes de Petri e Aplicações*. Campinas SP, 1996.
- MARSAN, M. A.; CHIOLA, G.; FUMAGALLI, A. Timed petri net model for the accurate performance analysis of csma/cd bus lans. *Computer Communications*, v. 10, n. 6, p. 304–312, 1987.
- MURATA, T. Relevance of network thoery to models of distributed/parallel processing. *Franklin Institute*, v. 310, n. 1, p. 41–50, 1980.
- MURATA, T. Petri nets: Properties, analysis and applications. In: (4), . (Ed.). [S.l.]: Proceedings of the IEEE, 1989. p. 541–580.
- NAVET, N.; SONG, Y. Q. Validation of in-vehicle real-time applications. *Computers in Industry*, v. 46, p. 107–122, June 2001.
- NOLTE, T.; NOLIN, M.; HANSSON, H. A. Real-time server-based communication with can. *IEEE Transactions on Industrial Informatics*, v. 1, p. 192– 201, Agosto 2005.
- OBERMAISSER, R.; PETI, P. Comparison of the temporal performance of physical and virtual can networks. In: . [S.l.]: IEEE ISIE, 2005. p. 1415–1422.
- PETRI, C. A. *Communication with Automata*. [S.l.], 1966. v. 1, n. RADC-TR-65-377.
- PINHO, L. M.; VASQUES, F. Reliable real-time comunication in can networks. *IEEE Transactions on Computers*, v. 52, n. 12, p. 1594–1607, December 2003.
- POP, P.; ELES, P.; PENG, Z. Analysis and optimisation of heterogeneous real-time embedded systems. *IEE Proc.-Comp. Digit. Tech.*, v. 152, n. 2, p. 130–147, Março 2005.
- PORTO, A. J. V.; SOUSA, R. V.; SILVA, A. R. Y.; INAMASU, R. Y. Robô agrícola móvel (ram): uma revisão das pesquisas recentes sobre sistemas de navegação autônoma de robôs e veículos agrícolas. *Congresso Brasileiro da Sociedade Brasileira de Informática Aplicada à Agropecuária e Agroíndustria*, 2003.
- PUNNENKKAT, S.; HANSSON, H.; NORSTROM, C. Response time analysis under errors for can. *Proceedings of the Sixth IEEE Real Time Technology and Applications Symposium (RTAS 2000)*, IEEE Computer Society, p. 258, 2000.
- SANTOS, M. M. D.; VASQUES, F.; STEMMER, M. R. Avaliação das propriedades temporais de duas redes de controle: Can e profibus. *Acta Scientiarum . Technology*, v. 25, n. 2, p. 193–201, 2003.

- SEVILLANO, J. L.; PASCUAL, A.; JIMENEZ, G.; CIVIT-BALCELLS, A. Analysis of channel utilization for controller area networks. *Computer Communications*, v. 21, n. 16, p. 1446–1451, 1998.
- SOUSA, R. V. *Can (Controller Area Network): Uma Abordagem para Automação e Controle na Área Agrícola*. Tese (Dissertação) — Escola de Engenharia de São Carlos da Universidade de São Paulo, 2002.
- SPECKMANN, H.; JAHNS, G. Development and application of an agricultural bus for data transfer. *Computers and Electronics in Agriculture*, v. 23, n. 3, p. 219–237, Setembro 1999.
- STONE, M. L. Dynamic address configuration in saej1939. *SAE Transactions*, p. 310–315, 1997.
- STRAUSS, C. *Implementação e avaliação de uma rede experimental baseada em CAN para aplicações agrícolas*. Dissertação (Dissertação) — Escola Politécnica da Universidade de São Paulo, 2001.
- TIMENET. *Overview of TimeNet*. <http://pdv.cs.tu-berlin.de/timenet/>, Fevereiro 2006.
- TINDELL, K.; BURNS, A.; WELLINGD, A. J. Calculating controller area network (can) message response times. *Control Eng. Practice*, v. 3, n. 8, p. 1163–1169, 1995.
- TINDELL, K. W.; HANSSON, H.; WELLINGS, A. J. Analysing real-time communications: Controller area network (can). *IEEE Computer Society*, p. 259–263, 1994.
- TOOLSDATABASE. *Complete Overview of Petri Nets Tools Database*. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/complete_db.html, Novembro 2005.
- ULLMAN, J. D. *Elements of ML programming*. 2º edição. ed. [S.l.: s.n.], 1997.
- VOSS, K. Using predicate/transition-nets to model and analyze distributed data base systems. *IEEE Trans. on Software Eng.*, v. 6, p. 801–806, Novembro 1980.
- YANG, L.; LI, Y.; YANG, G. Analysis of delay and traffic load in networked control system. In: IEEE/ASME. *International Conference on Advanced Intelligent Mechatronics*. [S.l.], 2005. p. 1425–1430.

ANEXO A - CONTROLLER AREA NETWORK(CAN)

O Barramento CAN e o Modelo de Rede ISO-OSI

O modelo ISO/OSI (*Open System Interconnect*) foi criado em 1977 pela ISO (*International Organization for Standardization*) com o objetivo de criar especificação das conectividades para interligar sistemas de computadores locais e remotos. O modelo é dividido em 7 camadas funcionais hierárquicas. A camada mais alta refere-se a aplicações (software) ou dispositivos enquanto que a camada mais baixa refere-se ao envio e recebimento dos bits na rede de comunicação. A seguir serão apresentadas uma breve descrição das camadas e suas respectivas funcionalidades durante a comunicação entre dispositivos da rede.

1. Camada Física: Define os aspectos elétricos e físicos do hardware utilizado na rede;
2. Camada de Enlace: Gerencia o Enlace de Dados. Responsável pelo acesso ao ambiente físico da rede, como transmissão e reconhecimento de erros.
3. Camada de Rede: Estabelece uma conexão lógica entre dois pontos, cuidando do tráfego e roteamentos dos dados da rede.
4. Camada de Transporte: Controla a transferência de dados e transmissões. Protocolos de transporte (TCP) são utilizados nesta camada.
5. Camada de Sessão: Reconhece os nós da rede local (LAN) e configura a tabela de endereçamentos entre fonte e destino. Estabelece as sessões, no qual o usuário poderá acessar outras máquinas da rede.
6. Camada de Apresentação: Transfere as informações de um software de aplicação da camada de sessão para o sistema operacional. Criptografia, conversão de códigos, compressão e descompressão de dados são algumas das funções desta camada.
7. Camada de Aplicação: É representada pelo usuário final no modelo OSI, selecionando serviços a serem fornecidos pelas camadas inferiores.

A divisão desse modelo em camadas é realizada de forma transparente e durante a comunicação entre dois dispositivos de rede, há uma comunicação entre as suas respectivas

camadas.

O CAN, como outros protocolos de comunicações, não possui especificação para todas as camadas do modelo de rede ISO/OSI. O Protocolo CAN, através da norma ISO 11898, especifica a utilização das duas camadas inferiores do modelo ISO/OSI, as camadas: Física(1) e a camada de Acesso Ao Meio (2).

Essa particularidade do protocolo CAN fornece vantagens na sua utilização, pois aumenta sua flexibilidade em relação à configuração e especificação das camadas de alto nível, permitindo que o mesmo seja otimizado de acordo com o tipo de aplicação. Com essa flexibilidade surgiram diversas especificações para as mais diferentes áreas como, entre elas: Automotiva, Agrícola, Aeronáutica, etc .

A tabela 7.1 apresenta algumas dessas especificações e suas respectivas áreas de aplicações do protocolo CAN.

Tabela 7.1: Normas CAN e suas Áreas de Aplicações

Especificação	Versão CAN	Área de Aplicação
NMEA2000	CAN 2.0B	Naval / Aérea
SAE J1939	CAN 2.0B	Automotiva (Caminhões e Ônibus)
DIN 9684	CAN 2.0A	Agrícola
ISO 11783	CAN 2.0B	Agrícola

Os padrões CAN apresentados acima fornecem especificações para as camadas de alto-nível do modelo ISO/OSI, como: Camada de Rede (3), de Transporte (4), de Sessão (5), Apresentação (6) e de Aplicação (7).

Camada Física

A camada Física é responsável pelo envio e recebimento de bits entre os dispositivos da rede, nessa camada são definidos como os sinais são transmitidos, como os sincronismos serão gerenciados e também quais cabos e conectores serão utilizados na instalação da rede.

Uma rede CAN pode ser montada utilizando apenas dois fios ou cabos, geralmente é utilizado cabos par-trançado para diminuir ruídos durante a comunicação. Esses dois fios são

conhecidos como CAN_H e CAN_L.

Caso algum problema aconteça com algum dos cabos que são utilizados como via de transmissão de dados, como rompimento ou curto circuito, as transmissões entre os dispositivos da rede não sofrem prejuízos. Essa característica está disponível no barramento CAN por possuir um sistema de segurança, que analisa a diferença de tensão nos cabos durante as transmissões e garante sua continuidade.

As figuras 7.1 e 7.2 apresentam consecutivamente: os níveis de tensão dos dados durante a comunicação, e três dispositivos interligados em uma rede CAN:

Figura 7.1: Tensão dos Cabos

Figura 7.2: Rede de Comunicação

Camada de Acesso ao Meio

A camada de Acesso ao Meio é responsável pela construção dos pacotes de dados, manipulá-los e também realizar os controles na transmissão. Essa camada de rede é responsável por realizar a identificação dos pacotes de dados, controlar o acesso ao barramento e também realizar verificações de possíveis erros nas transmissões ou nos conteúdos dos pacotes que estão sendo enviados.

O controle de acesso é realizado utilizando um método de arbitragem, conhecido como CSMA/CD-NDBA (Carrier Sense Multiple Access with Collision Detect - Non-Destructive Bit Arbitration). Esse método utiliza os endereços dos dispositivos da rede para análise das prioridades de acessos, em caso de uma disputa entre dois dispositivos ou mais, o dispositivo que tiver maior prioridade continua enviando sua mensagem enquanto que os dispositivos com prioridades menores armazenam suas mensagens em um buffer de memória para enviá-las quando obtiverem o acesso ao barramento.

Outro controle realizado por essa camada de rede, é o controle de erros, que monitora os problemas nas transmissões dos pacotes e armazena em contadores com o propósito de limitar a quantidade de erros por dispositivos. Com isso, um dispositivo que enfrentou muitos problemas na comunicação pode ser desligado automaticamente da rede.

Método de Arbitragem

Para realizar arbitragem dos acessos ao barramento, o protocolo CAN utiliza o método conhecido como CSMA/CD-NDBA (*Carrier Sense Multiple Access with Collision Detect - Non-Destructive Bit Arbitration*).

Esse método utiliza os endereços (identificadores) dos dispositivos instalados na rede CAN. O identificador com o número binário menor, possui prioridade mais elevada e conseqüentemente os identificadores com número binário maior, possuem menores prioridades. O tamanho desses identificadores varia de acordo com a versão do protocolo CAN que está sendo utilizada (11 bits para o CAN 2A e 29 bits para o CAN 2B).

Com o barramento CAN livre, todos os dispositivos conectados na rede podem iniciar uma transmissão enviando bit-a-bit seus pacotes de dados. Como apresentado anteriormente, os primeiros campos do quadro de dados são os identificadores dos dispositivos. O barramento CAN compara esses bits enviados pelo dispositivo, verificando se são dominantes (bit 0) ou recessivos (bit 1).

Durante a arbitragem, cada dispositivo transmite os bits que compõem seu identificador e o controlador CAN compara-o com o bit do barramento, caso estes bits sejam iguais, o dispositivo continua sua transmissão. Se no momento em que o dispositivo estiver transmitindo um bit recessivo e o controlador CAN detectar um bit dominante no barramento, o mesmo interrompe a transmissão e altera seu estado para "escravo" ou "receptor" e passa somente a receber os dados do barramento.

Caso um dispositivo esteja transmitindo um bit recessivo e outro dispositivo da rede enviar um bit dominante simultaneamente, o bit recessivo será sobrescrito e o dispositivo

com o bit dominante, que possui maior prioridade, continuará enviando o pacote de dados. As mensagens dos dispositivos que perderam a disputa pelo acesso ao barramento são armazenadas em um buffer de memória para um envio posterior, evitando que sejam perdidas.

A figura 7.3 apresenta o exemplo de uma disputa entre 6 dispositivos pelo acesso ao barramento

Figura 7.3: Metodo de Arbitragem CAN

Nessa configuração seis dispositivos iniciam uma transmissão dos pacotes de dados simultânea e conseqüentemente inicia-se o processo de arbitragem do barramento CAN. Primeiramente todos os nós enviam o bit de SOF, que identifica o início de uma transmissão e posteriormente transmitem seus identificadores. Como pode ser observado, os primeiros 4 bits enviados pelos dispositivos possuem o mesmo valor e por isso, todos continuam a transmissão.

A disputa inicia no envio do bit numero 2, onde os dispositivos 4, 5 e 6 enviam o bit 1 (recessivo) e os dispositivos (1, 2 e 3) enviam o bit 0 (dominante). Com isso, os dispositivos com menor prioridade alteram seus status para “escravos”, enquanto que os outros dispositivos continuam a transmissão.

No momento do envio do penúltimo bit do campo de identificação, o dispositivo 1 transmite um bit 0, e no mesmo momento os dispositivos 2 e 3 transmitem o bit 1. Por terem

seus dados sobrescritos pelo bit dominante do dispositivo 1, os dispositivos 2 e 3 interrompem as transmissões e também alteram seus status para escravos, enquanto que o dispositivo 1 obtém o acesso ao barramento e continua transmitindo seu pacote de dados.

Como citado, o mecanismo de arbitragem do protocolo CAN é uma forma de resolver os conflitos e colisões nos acessos ao barramento, além de perdas de mensagens, porém em sistemas com uma grande quantidade de dispositivos instalados na rede CAN ou com aplicação em tempo real, dispositivos com baixas prioridades podem possuir um tempo de latência muito elevado. Nesses casos, a definição dos endereços dos dispositivos no barramento é considerada umas das etapas mais importantes, pois interfere diretamente nas frequências de acessos ao barramento.

Os valores dos identificadores dos dispositivos da rede não podem ser alterados dinamicamente de acordo com as necessidades. Todos os identificadores precisam, necessariamente, serem configurados antes da rede de dispositivos ser inicializada. Dispositivos de uma rede CAN que possuem um tempo de latência muito elevado podem causar problemas no funcionamento geral da rede de comunicação como, por exemplo: perda ou sobreposição de mensagens. Geralmente os dispositivos possuem um tamanho limitado do buffer de memória, e caso o limite de armazenamento seja superado, mensagens serão perdidas ou sobrescritas. Esse problema fica ainda mais crítico em aplicações em tempo-real.

Formatos das Mensagens CAN

O protocolo CAN possui quatro tipos de quadro ou mensagens: Quadro de Dados, Quadro Remoto, Quadro de Erro e Quadro de Sobrecarga. A seguir serão apresentadas as estruturas de cada mensagem.

Quadro de Dados

O protocolo CAN utiliza quadros dos dados para transmitir informações entre os dispositivos instalados na rede. Os quadros de dados são sub-divididos em campo de arbitra-

gem, campo de identificação da mensagem, campo de dados e campos de controle, responsáveis pela transferência de diferentes informações.

O barramento CAN possui dois formatos de mensagens, que se diferenciam somente no campo de identificação: a mensagem padrão, ou (Standard frame) que possui um campo de identificação de 11 bits, conhecido como CAN 2.0A e a mensagem estendida, ou (*extended frame*) que possui o campo de identificação de 29 bits, conhecido como CAN 2.0B.

Utilizando formato padrão de mensagens (CAN 2.0A), que possui o campo de identificação de 11 bits, é possível obter no máximo 2048 (2^{11} Bits) mensagens em uma rede. No formato estendido (CAN 2.0B), que possui o campo de identificação de 29 bits, essa possibilidade é de aproximadamente 500 milhões (2^{29} Bits).

A figura 7.4 a seguir apresenta a estrutura das mensagens no formato 2.0A.

Figura 7.4: Pacote de Dados: CAN 2A

A seguir estão descritos os campos que compõem uma mensagem CAN:

SOF (Star of frame): Único bit que marca o início do quadro de dados. Quadro de Arbitragem: Contem o campo de identificação (11 bits) e o campo RTR (*remote transmission request*) de 1 bit que indica se a mensagem é um quadro de dados ou se é um quadro Remoto.

Identificador: O identificador de mensagem possui 11 bits e é o campo responsável pelo controle de acesso ao barramento.

RTR: Um bit que indica o tipo de mensagem, 0 (zero) significa que é pacote de

dados, e 1 (um) significa que é um pacote remoto

Campo de Controle: Possui um bit da extensão do identificador, um bit reservado (r0), e tamanho do pacote de dados em bytes.

Identificador (Ide): Um bit que indica o tipo do quadro. Um bit dominante indica que o quadro é padrão ou um bit recessivo para indicar que o quadro é estendido.

R0: Esse bit é reservado e deve sempre ser dominante.

DLC: Indica o tamanho do campo de dados em bytes.

Campo De Dados: O campo de dados pode conter de 0 a 8 bytes, não possuindo um tamanho fixo.

CRC (Verificação de Redundância Cíclica): Possui 15 bits que o receptor utiliza o valor armazenado nesse campo para verificar se os bits do campo de dados estão corretos.

Campo de Confirmação (ACK): são dois bits, recessivos, enviados pelo dispositivo que transmite o quadro dos dados. Todo dispositivo que receber a mensagem sobrescreve esses dois bits com bits dominantes.

EOF: são 7 bits que indicam o final do quadro da mensagem

O quadro de mensagens CAN 2.0B é semelhante a estrutura das mensagens CAN 2.0A apresentadas acima, diferenciando apenas o campo identificador que possui 29 bits, permitindo um número maior de endereçamento de dispositivos.

A seguir, na figura 7.5 apresenta a estrutura das mensagens no formato CAN 2.0B.

Em relação ao quadro de mensagem do CAN 2.0A, apresentado anteriormente, somente os campos de arbitragem e de controle possuem diferenças. A seguir esta sendo apresentada uma breve descrição das diferenças:

Campo de Arbitragem: Campo com identificador de 29 bits.

Identificador de Mensagem Padrão: 11 bits como no quadro CAN 2.0A.

SRR: O bit SRR é sempre recessivo, indicando que o quadro CAN 2.0B é de menor

Figura 7.5: Mensagem CAN 2B

prioridade em relação aos quadros padrões (CAN 2.0A).

Ide: Indica se o quadro é estendido ou não, caso seja o bit seja recessivo, indica que o quadro é estendido. Caso seja dominante, indica que é um quadro padrão.

Extensão do Identificador da Mensagem: São outros 18 bits adicionais utilizados para identificação da mensagem.

RTR: Possui a mesma função do quadro padrão.

Campo de Controle: Contem dois bits reservados e o tamanho do campo de dados (DLC).

Quadro Remoto

Os quadros remotos são utilizados para solicitar informações de algum dispositivo conectado na rede CAN. Esse quadro possui uma característica muito parecida com o quadro dos dados, porém durante as transmissões, os quadros de dados sempre são dominantes, ou seja, tem maior prioridade para acesso ao barramento em relação aos quadros Remotos. Essa característica é garantida, pois em uma transmissão de um quadro de remoto, o campo RTR é transmitido com um bit recessivo.

Quadro de Erros

O quadro de Erro é transmitido quando um dispositivo conectado a rede CAN detecta uma falha na comunicação e envia uma mensagem para todos os outros dispositivos indicando que foi detectado um erro na transmissão.

Após o recebimento dessa mensagem de erro, o dispositivo que estava enviando o pacote de dados inicia retransmissão.

Há dois tipos de mensagens de erro: Quadro de Erro Ativo: composto por 6 bits dominantes e Quadro de Erro Passivo: composto por 6 bits recessivos. Ambos os quadros violam a regra do *Bit Stuffing*, que permite somente cinco bits consecutivos possuírem o mesmo valor (dominante ou recessivo). Caso aconteça algum desses erros os contadores são incrementados.

Quando um erro é detectado durante a transmissão de uma mensagem no barramento CAN, o dispositivo pode enviar imediatamente um quadro de erro, mesmo sem o término do envio da mesma. Dessa maneira os outros dispositivos conectados na rede detectam esse erro na transmissão.

O Quadro de Erro do protocolo CAN é formado por dois campos:

- **Campo de Erro:** Esse quadro viola a regra do Bit Stuffing enviando uma sequência de seis bits, dominantes ou recessivos, consecutivamente
- **Campo Delimitador:** Esse quadro é formado por oito bits recessivos consecutivos que indica o final do quadro de erro.

Quadro de Sobrecarga

Os quadros da sobrecarga são muito similares aos quadros do erro. O quadro da sobrecarga é transmitido geralmente com o objetivo de retardar a transmissão do quadro seguinte da mensagem com o objetivo de evitar erros nas transmissões e conseqüentemente incrementos nos contadores de erro.

O quadro de sobrecarga também é composto por dois campos. Um campo de sobrecarga composto por seis bits dominantes consecutivos e o delimitador da sobrecarga, composto por oito bits recessivos. O quadro de sobrecarga pode ser transmitido quando um dispositivo receptor necessita atrasar o envio da próxima mensagem, obtendo um tempo maior para processamento da mensagem atual. Caso seja necessário, mais de um quadro de sobrecarga pode ser enviado ao barramento.

Apesar de possuir um formato muito parecido com o quadro de erro, o quadro de sobrecarga não causa retransmissão de mensagens no barramento CAN.

Deteção de Falhas na Comunicação

O protocolo CAN possui uma grande capacidade de se adaptar em condições adversas que podem gerar falhas temporárias ou permanentes durante a comunicação entre os dispositivos. Essas falhas podem ser classificadas em três tipos: Nível de Bit, Nível de Mensagem e Nível Físico. A falha Nível de Bit é sub-dividida em dois tipos: Bit Monitoring e o Bit Stuffing, totalizando 5 métodos capazes de detectar falhas na comunicação. A seguir estão descritos todos os métodos e suas características:

Bit Monitoring: Após a escrita de um bit dominante, o módulo transmissor verifica o estado do barramento. Se o bit lido for recessivo, significará que existe um erro no barramento.

Bit Stuffing: Apenas cinco bits consecutivos podem ter o mesmo valor (dominante ou recessivo). Caso seja necessário transmitir sequencialmente seis ou mais bits de mesmo valor, o módulo transmissor insere, imediatamente após cada grupo de cinco bits consecutivos iguais, um bit de valor contrário. O módulo receptor ficará encarregado de, durante a leitura, retirar este bit chamado de Stuff Bit. Caso uma mensagem seja recebida com pelo menos seis bits consecutivos iguais significa que algum erro aconteceu no barramento ou na transmissão.

CRC ou Cyclic Redundancy Check: Funciona como um *checksum*. O módulo transmissor calcula um valor em função dos bits da mensagem e o transmite juntamente. Os módulos receptores recalculam este CRC e verificam se o valor é o mesmo valor transmitido com a men-

sagem.

Frame Check: Os módulos receptores analisam o conteúdo de alguns bits da mensagem recebida. Os valores destes bits não mudam de mensagem para mensagem e são determinados pelo padrão CAN.

Acknowledgment Error Check: Os módulos receptores respondem a cada mensagem recebida sem erros, escrevendo um bit dominante no campo ACK de uma mensagem de resposta que é enviada ao módulo transmissor. Caso esta mensagem resposta não seja recebida (pelo transmissor original da mensagem), significará que, ou a mensagem de dados transmitida estava corrompida, ou nenhum módulo a recebeu.

Quando uma falha é detectada por um ou mais dispositivos receptores da rede, os mesmos enviam uma mensagem de erro no barramento, informando a todos os dispositivos, que foi encontrado um erro na mensagem e que a mesma deverá ser retransmitida.

Além disso, os dispositivos possuem contadores que são incrementados a cada mensagem que é enviada ou recebida contendo erro.

Em caso de erro na transmissão, os contadores dos módulos receptores são incrementados de um, e os contadores do dispositivo transmissor da mensagem é incrementado oito unidades. Com isso, os valores dos contadores são utilizados como parâmetros para classificar os estados de funcionamento dos dispositivos. Esses estados são: Error Active , Error Passive ou Bus Off.

São classificados como estado de Error Active, os dispositivos com valores entre 1 e 127 , os dispositivos com valores entre 128 e 255 são classificados no estado de Error Passive e os dispositivos com valores superiores a 225 nos contadores, são classificados no estado de Bus Off e são excluídos da rede, onde passam a não enviar e também não receber mensagens da rede de comunicação.

Com exceção de quando o dispositivo atinge o estado de Bus Off, onde o mesmo necessita ser reinicializado para voltar ao seu funcionamento normal, os contadores de erros são decrementados na medida em que cada envio ou recebimento de mensagem ao barramento

é realizado com sucesso. Assim os estados dos dispositivos podem sofrer variações e serem alternados durante o funcionamento da rede.

ANEXO B - REDE DE PETRI

Conceitos Básicos de Rede de Petri

Um modelo de Rede de Petri é descrito, graficamente, basicamente por quatro elementos básicos: Lugar, Transição, Arco e Marca.

Lugares: graficamente são representados por uma Elipse e modelam os estados do sistema. Representam as condições para que um determinado evento aconteça;

Transições: graficamente são representados por um retângulo e modelam os eventos do sistema, que representam as mudanças de estados;

Arcos: Graficamente são representados por uma seta, interligam um lugar e uma transição, e representam como e a sequência com que acontecem as mudanças de estado dentro do sistema;

Marcas: Graficamente são representadas por um círculo de cor preta e representam o comportamento dinâmico do sistema, indicando os estados ativos, ou seja, que estão sendo utilizados pelo sistema;

As figuras 7.6 e 7.7 representam um modelo de Rede de Petri bastante simples e didático que contém os elementos básicos, apresentado por Murata (1989).

Esse exemplo representa a reação química $2H_2 + O_2 \rightarrow 2H_2O$.

Figura 7.6: Modelo de Rede de Petri
(a) - (MURATA, 1989)

Figura 7.7: Modelo de Rede de Petri
(b) - (MURATA, 1989)

Nesse modelo duas marcas (*tokens*) nos lugares de entrada representam duas unidades de H_2 e O_2 e a transição t é habilita (figura 7.6). Após o disparo da transição t as marcas são alteradas representando a reação química (figura 7.7).

Todas as simulações de comportamentos dinâmicos de sistemas, as mudanças dos estados ou marcas seguem uma regra de transição ou também chamada regra de disparo :

- Uma transição t está habilitada a “disparar” se cada um dos locais de entrada P possuir pelo menos $w(P,t)$ marcações, onde $w(P,t)$ representa o peso do arco indo de P a t .
- Uma transição habilitada poderá disparar assim que os eventos relacionados a ela ocorrerem.
- O “disparo” de uma transição t remove $w(P_i, t)$ marcações de cada um local de entrada P_i e adiciona $w(t,P_o)$ marcações a cada local de saída P_o de t .

Descrição Formal da Rede de Petri é definida como $PN = (P, T, F, W, M_0)$, onde :

$P = \{ p_1, p_2, \dots, p_m \}$ é uma lista finita de lugares,

$T = \{ t_1, t_2, \dots, t_n \}$ é uma lista finita de transições,

$F \subseteq (P \times T) \cup (T \times P)$ é uma lista de arcos,

$W: F \rightarrow (1, 2, 3, \dots)$ é uma função de peso,

$M_0: P \rightarrow (0, 1, 2, 3, \dots)$ é a marca inicial,

$P \cap T = \text{“vazio”}$ e $P \cup T \neq \text{“vazio”}$,

Uma Rede de Petri $N = (P, T, F, W)$ que não contenha nenhuma marca inicial é conhecida como N . Uma Rede de Petri que contenha marcações iniciais é conhecida como (N, M_0)

Rede de Petri Colorida e Definições

As redes de Petri Coloridas, ou Colored Petri Nets (CPN), foram criadas no início dos anos noventa pelo professor Kurt Helmer Jensen, na Universidade de Aahus na Dinamarca.

Um dos principais objetivos da utilização da Rede de Petri Colorida (CPN) é a diminuição no tamanho e na complexidade de entendimento dos modelos utilizados para analisar grandes sistemas, além de oferecer recursos para analisar modelos com maior poder de abstração (GIRAULT; VALK, 2003), (MACIEL, 1996).

A Rede de Petri Colorida (CPN) pode ser considerada uma linguagem de modelagem desenvolvida para os sistemas em que a comunicação, sincronização e compartilhamento de recursos são aspectos importantes e críticos para um funcionamento adequado, pois no modelo os recursos das Redes de Petri ordinárias são agregados com os recursos de uma linguagem de programação de alto-nível, disponibilizando primitivas para os processos de interações juntamente com as primitivas para as definições de variáveis, funções e valores, disponíveis nas linguagens de programação de alto-nível.

A representação gráfica de um modelo de Rede de Petri Colorida pode ser considerada bastante intuitiva e simples, o que facilita o entendimento do sistema simulado. Para facilitar o entendimento de modelos mais complexos, a CPN permite que o mesmo possa ser subdividido em “páginas” que interagem durante a simulação.

Uma grande vantagem na utilização da Rede de Petri Colorida é a utilização das marcas coloridas ou *tokens*, individualizados que representam diferentes processos ou recursos em uma mesma rede ou sub-rede. A seguir será apresentada a definição formal das marcas coloridas usando $\mathbb{N}$ para representar uma lista de todos os números inteiros não negativos.

A figura 7.8 apresenta o exemplo do modelo 7.6 utilizando os conceitos das multi-marcas :

Figura 7.8: Modelo de Rede de Petri Colorida (a)

Figura 7.9: Modelo de Rede de Petri Colorida (b)

Definição 1: Uma multi-marca m , é uma lista não vazia S , é uma função $m \in [S \rightarrow \mathbb{N}]$, que pode ser representada pela somatória:

$$\sum_{s \in S} m(s) \cdot s$$

Definição 2: Adição, Multiplicação Escalar, Comparação e Tamanho das multi-listas são definidas seguindo, para todos $m, m_1, m_2 \in S_{MS}$ e todos $n \in \mathbb{N}$:

$$1. m_1 + m_2 = \sum_{s \in S} (m_1(s) + m_2(s)) \cdot s$$

$$2. n * m = \sum_{s \in S} (n(s) * m(s)) \cdot s$$

$$3. m_1 \neq m_2 = \exists s \in S : m_1(s) \neq m_2(s)$$

$$m_1 \leq m_2 = \forall s \in S : m_1(s) \leq m_2(s)$$

$$4. |m| = \sum_{s \in S} m(s)$$

Quando o $|m| = \infty$, m é infinito. Senão m é finito. Quando $m_1 \leq m_2$ define-se uma subtração:

$$5. m_2 - m_1 = \sum_{s \in S} (m_2(s) - m_1(s)) \cdot s$$

Definição 3: De acordo com Jensen (1996), define-se, formalmente, uma rede de Petri colorida como sendo uma nômupla $CPN = (\in, P, T, A, N, C, G, E, I)$ que satisfaça as condições:

i. $\in$ é um conjunto não vazio de tipos, também chamado de lista de cores.

ii. P é um conjunto finito de lugares.

iii. T é um conjunto finito de transições.

iv. A é um conjunto finito de arcos tais como:

$$P \cap T = P \cap A = T \cap A = \emptyset$$

v. N é uma função de nó (node). N é definido a partir de A em $P \times T \cup T \times P$.

vi. C é uma função de cor. É definida a partir de C em Σ .

vii. G é uma função de guarda. É definida a partir de T em expressões tais que: $\forall t \in T :$

$[Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma]$ B denota um tipo booleano contendo os elementos falso, verdadeiro e tendo operações padrões a partir de lógica proposicional

viii. E é uma função de expressão de arco (arc expression). E é definida a partir de A em expressões tais que: $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ onde $p(a)$ é um lugar de $N(a)$.

ix. I é uma função de inicialização. I é definido a partir de P em expressões fechadas tais que: $\forall p \in P : [Type(I(p)) = C(p)_{MS}]$

i, ii, iii: O conjunto de tipos determina os valores dos dados, as operações e as funções que podem ser usados nas expressões da rede como : expressões dos arcos, guardas e expressões da inicializações)

iv: Os lugares, transições e arcos são descritos por três conjuntos finitos P, T e A . Com isso, por exemplo, é eliminada a possibilidade de existência de um número infinito de arcos entre dois lugares.

v: A função Nó interliga pares onde o primeiro elemento é o nó de origem e o segundo elemento o nó de destino. Os dois nós precisam, necessariamente, ser de tipos diferentes, um lugar e outro transição, por exemplo.

vi: A função Cor para cada lugar p , para um tipo $C(p)$, isso significa que cada marca no lugar p deve ser um valor pertencente a $C(p)$.

vii: A função Guarda G para cada transição, t , há uma expressão booleana onde todas as variáveis tenham os tipos pertencentes a Σ .

viii: A função Arc Expression E para arco, a , para uma expressão do tipo $C(p)_{MS}$. Isto significa que cada expressão do arco deve avaliar o tipo dos multi conjuntos do lugar adjacente, P .

ix: A função de Inicialização I em cada lugar, p , em uma “expressão fechada” que deve ser do tipo $C(p)_{MS}$

Após a definição da Estrutura das Redes de Petri Coloridas serão apresentadas algumas definições comportamentais dos modelos.

Primeiramente define-se um Disparo.

Definição 4: Um Disparo na transição t é uma função b definida $Var(t)$:

i. $\forall v \in Var(t) : b(v) \in Type(v)$

ii. $G(t) < b >$

Onde $B(t)$ é o conjunto de todos os disparos para t .

Definição 5: Uma marca é um par (p, c) onde $p \in P$ e $c \in C(p)$, enquanto que o disparo é um par (t, b) onde $t \in T$ e $b \in B(t)$. O conjunto de todas as marcas são representadas por TE e o conjunto de todos os elementos de disparo são apresentados por BE . Uma marca é um multi-conjunto de TE enquanto que o step (tempo) seja não vazio e multi-conjunto finito de BE . A Marca inicial M_0 é obtida pela expressão de inicialização:

$\forall (p, c) \in TE : M_0(p, c) = (I(p))(c)$ O conjunto de todas as marcações e passos são representados por X e X respectivamente.

A seguir sera apresentada a definição formal dos disparos (*Enabling*)

Definição 6: Um passo Y é habilitado na marcação M somente se as seguintes propriedades estiverem satisfeitas:

$$\forall p \in P : \sum_{(t,b) \in Y} E(p, t) < b > \leq M(p)$$

Quando (t, b) é habilitado o também t é habilitado. Os elementos de Y são paralelamente habilitados (quando $|Y| \geq 1$).

Quando o passo Y é habilitado na marca M_1 isso deve ocorrer, trocando a marca M_1 para outra marca M_2 , definida por :

$$\forall p \in P : M_2(p) = (M_1(p) - \sum_{(t,b) \in Y} E(p, t) < b >) + \sum_{(t,b) \in Y} E(t, p) < b >$$

M_2 é diretamente realocado para M_1 . Isso é apresentado como: $M_1[Y]M_2$

A expressão $E(p, t) < b >$ representa a marca que foi removida do lugar p após o disparo t com o *binding* b . Analisando todos os elementos $(t, b) \in Y$ obtém-se todas as marcas que foram removidas de p quando Y ocorreu .

Definição 7:

A definição de seqüência finita é a seqüência de marcas e passos:

$$M_1 [Y_1] M_2 [Y_2] M_3 \dots M_n [Y_n] M_{n+1}$$

Com $n \in \mathbb{N}$, e $M_i [Y_i] M_{i+1}$ para todos $i \in 1, 2, \dots, n$ M_1 é a marca inicial. M_{N+1} é a marca final e n representa o tamanho.

Analogamente, uma seqüência de ocorrências infinitas é a seqüência de marcas e passos :

$$M_1 [Y_1] M_2 [Y_2] M_3 \dots$$

Com $M_i [Y_i] M_{i+1}$ para $i > 1$.

A marca M' é alcançável à partir da marca M' se existir uma seqüência inicial de ocorrência finita em M' e final em M'' . O conjunto de marcas alcançáveis por M' é representado por $[M']$. A marca é alcançável se pertence a $[M_0]$

Para exemplificar as vantagens na utilização da Rede de Petri Colorida citadas anteriormente, será apresentado um exemplo de um sistema de manufatura cujo o modelo representa uma célula de fabricação flexível. Esse exemplo é apresentado com mais detalhes em Desrochers e Al-Jaar (1995).

A figura 7.10 apresenta um sistema de uma célula de fabricação flexível, composto por três sub-sistemas

Figura 7.10: Modelo de Rede de Petri para um sistema de fabricação flexível com marcas iniciais - (DESROCHERS; AL-JAAR, 1995)

Utilizando as definições formais da Rede de Petri Colorida o mesmo sistema pode ser remodelado, facilitando seu entendimento, sua edição e a execução das simulações. A figura 7.11 apresenta o mesmo sistema modelado utilizando conceitos de Rede de Petri Colorida.

O modelo apresenta os seguintes lugares e transições:

Figura 7.11: Modelo de Rede de Petri Colorida para um Sistema de Produção Flexível - (DES-ROCHERS; AL-JAAR, 1995)

- p_1 transportes disponíveis, *ACON*
- p_2 robô esquerdo disponível, *LACQ*
- p_3 processamento com dois robôs, *PROC*
- p_4 robôs disponíveis, *AROB*
- t_1 solicitação robô esquerdo, *RL*
- t_2 solicitação robô direito, *RR*
- t_3 final do processo, *PE*

As listas de cores associadas são:

A seguir a tabela 7.2 apresenta a relação entre os lugares e transições dos modelos de Rede de Petri Ordinária (PN) e Rede de Petri Colorida (CPN).

A matriz de incidência A_c do modelo de Rede de Petri Ordinário, apresentada a seguir, também possui uma matriz correspondente para o modelo de Rede de Petri Colorida.

```

1  C(ACON) = {C1, C2, C3 }
2          = {a1i | i = 1, 2, 3} = C(p1); u1 = 3
3  C(LACQ) = {(C1, R1), (C2, R2), (C3, R3)}
4          = {a2i | i = 1, 2, 3} = C(p2); u2 = 3
5  C(PROC) = {(C1, R1, R2), (C2, R2, R3), (C3, R3, R1)}
6          = {a3i | i = 1, 2, 3} = C(p3); u3 = 3
7  C(AROB) = {R1, R2, R3}
8          = {a4i | i = 1, 2, 3} = C(p4); u4 = 3
9  C(RL) = {(C1, R1), (C2, R2), (C3, R3)}
10         = {b1j | j = 1, 2, 3} = C(t1); v1 = 3
11 C(RR) = {((C1, R1), R2), ((C2, R2), R3), ((C3, R3), R4)}
12         = {b2j | j = 1, 2, 3} = C(t2); v2 = 3
13 C(PE) = {C1, R1, R2), (C2, R2, R3), (C3, R3, R1)}
14         = {b3j | j = 1, 2, 3} = C(t3); v2 = 3

```

Tabela 7.2: Relação entre PN e CPN

CPN	Rede de Petri
p1	p1, p4, p7
p2	p2, p5, p8
p3	p3, p6, p9
p4	p10, p11, p12
t1	t1, t4, t7
t2	t2, t5, t8
t3	t3, t6, t9

$$A_c = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

As interseções dos lugares e as transições do CPN resultam em dividir a matriz Ac em blocos. **Estas matrizes quadradas e todas de três dimensões.** Cada bloco dessas matrizes representa uma função de saída ou da entrada, que são as “nomes” dos arcos.

Conseqüentemente estas funções não são uma matriz e sim um produto Cartesiano sobre seus domínios. A função de entrada de $p1$ para $t1$ é :

$$I(p1, t1) : ACON \times RL = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = ID.$$

As outras funções diferentes de zero são :

$$I(p4, t1) : AROB \times RL = ID$$

$$O(p2, t1) : LACQ \times RL = ID$$

$$I(p2, t2) : LACQ \times RR = ID$$

$$I(p4, t2) : AROB \times RR = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$O(p3, t2) : PROC \times RR = ID$$

$$I(p3, t3) : PROC \times PE = ID$$

$$O(p1, t3) : ACON \times PE = ID$$

$$O(p4, t3) : AROB \times PE = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Para completar a descrição estrutural do Modelo CPN :

$$Ac = \begin{bmatrix} -I(p1, t1) & 0 & O(p1, t3) \\ O(p2, t1) & -I(p2, t2) & 0 \\ 0 & O(p3, t2) & -I(p3, t3) \\ -I(p4, t1) & -I(p4, t2) & O(p4, t3) \end{bmatrix}$$

O exemplo citado anteriormente é apresentado com maiores detalhes em Desrochers e Al-Jaar (1995).