Contribuição à Modelagem de Sistemas de Controle via Rede

Jadsonlee da Silva Sá

Tese de Doutorado apresentada à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

José Sérgio da Rocha Neto, D.Sc.
Orientador
Antonio Marcus Nogueira Lima, Dr.
Orientador

Campina Grande, Paraíba, Brasil
©Jadsonlee da Silva Sá, Abril de 2016

"CONTRIBUIÇÃO À MODELAGEM DE SISTEMAS DE CONTROLE VIA REDE"

JADSONLEE DA SILVA SÁ

TESE APROVADA EM 15/04/2016

JOSÉ SÉRGIO DA ROCHA NETO, D.Sc., UFCG
Orientador(a)

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador(a)

ANGELO PERKUSICH, D.Sc., UFCG
Examinador(a)

AMAURI OLIVEIRA, D.Sc, UFBA
Examinador(a)

UBIRAJARA FRANCO MORENO, Dr., UFSC
Examinador(a)

ADOLFO BAUCHSPIESS, Dr., UNB
Examinador(a)

BENEMAR ALENCAR DE SOUZA, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S111c Sá, Jadsonlee da Silva.

Contribuição à modelagem de sistemas de controle via rede / Jadsonlee da Silva Sá. – Campina Grande, 2016.

238 f. : il. color.

Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

"Orientação: Prof. Dr. José Sérgio da Rocha Neto, Prof. Dr. Antônio Marcus Nogueira Lima".

Referências.

1. Engenharia Elétrica – Arquitetura de Redes. 2. Sistema de Controle Via Rede. 3. Controller Area Network. 4. Automatos. I. Rocha Neto, José Sérgio da. II. Lima, Antônio Marcus Nogueira. III. Título.

CDU 621:004.72(043)

יהוה
ישוע
ורוח הקודש

Agradecimentos

A Deus, Jesus e o Espírito Santo, pela minha vida.

À minha esposa Patrícia Sá..."Agora sim! Está é carne da minha carne e osso dos meus ossos".

Aos meus pais Adaildo Jeremias Sá e Maria Auxiliadora da Silva Sá e aos meus irmãos, Jancylee da Silva Sá e Jemylee da Silva Sá, pelo apoio incondicional; aos meus tios, tias, primos e primas, que acreditaram em mim.

Aos meus mais que amigos e amigas na INSEJEC Juazeiro-BA e Campina Grande-PB, em especial: Pra. Irany, Louise, Callebe, Hélder e Gilvaneide, Jucielson e Gicélia, Hérica e Luis, Josiel, Wilson, Claudemir, Elson, Ronaldo "profeta chorão", Ap. Rivanda e Pr. José Bezerra e muitos outros que em suas orações lembraram do meu nome.

Aos professores José Sérgio da Rocha Neto e Antonio Marcus Nogueira Lima, pela orientação durante todos esses anos.

Ao professor Angelo Perkusich, pelo incentivo.

Ao técnico Simões Toledo, pela ajuda na confecção das placas usadas neste trabalho.

A todos os meus professores da graduação, mestrado e doutorado no DEE/UFCG pela sua contribuição na minha formação e aos funcionários da COPELE, pelo apoio..

Aos amigos da UFCG, em especial, Alfranque Amaral, Saulo Dornelas, Jaidilson Jó e Carlos Alberto.

Aos amigos da UNIVASF, pelo apoio e incentivo durante a realização das atividades do meu doutorado.

Resumo

Sistemas de controle via rede são sistemas onde malhas de controle são fechadas por meio de uma rede de comunicação. Esses sistemas podem ser encontrados em várias aplicações de controle complexas tais como, nas aplicações automotivas, em aeronaves, na robótica, na indústria de manufatura entre outras. A utilização de arquiteturas em rede possibilitam uma melhoria na confiabilidade e flexibilidade de operação, redução da quantidade de cabos e dos custos, facilidade de instalação, manutenção e diagnóstico do sistema. Entretanto, arquiteturas em rede introduzem novos desafios, que tornam o projeto e implementação de sistemas de controle uma atividade complexa. Basicamente, redes de comunicação podem causar atrasos variáveis e perdas de mensagens durante os ciclos de operação das malhas de controle, além de outras particularidades, que podem degradar o desempenho e até causar a instabilidade dos processos físicos. Nas duas últimas décadas, vários trabalhos foram desenvolvidos na tentativa de superar esses desafios. Entretanto, na maioria desses trabalhos, as metodologias de projeto são complexas e geralmente a aplicabilidade é questionável, devido a suposições não realistas e, principalmente, ao grande esforço computacional exigido. Além disso, muitas dessas metodologias se aplicam apenas em arquiteturas mais simples e é também comum considerar NCSs com uma única malha de controle, o que não condiz com a realidade das aplicações. Por outro lado, alguns trabalhos têm possibilitado a extensão de técnicas de controle mais simples e já consolidadas na prática no projeto de sistemas de controle via rede. Entretanto, alguns comportamentos causados pela arquitetura em rede ainda são desconsiderados, os quais na prática comprometem o funcionamento do sistema.

Neste trabalho foi desenvolvido um procedimento para projetar sistemas de controle via rede CAN (*Controller Area Network*), considerando a integração das teorias relacionadas, de modo a verificar e mitigar os efeitos causados pela arquitetura em rede no desempenho das malhas de controle. O procedimento consiste de três atividades: projetar controladores PID (Proporcional Integral Derivativo) para processos com uma entrada e uma saída considerando atrasos de controle variantes no tempo, a partir do conceito da margem de jitter; desenvolver um *framework* utilizando modelagem formal em autômatos temporizados e autômatos temporizados com custo, e as abordagens verificação de modelos clássica e estatística, para determinar o comportamento temporal das malhas de controle, de modo a verificar as condições em que os requisitos de tempo serão satisfeitos, e desenvolver uma plataforma experimental para simular em tempo real sistemas de controle via rede CAN utilizando a técnica *hardware in the loop*, incluindo como partes reais uma rede CAN e um injetor de falhas aleatórias. Um estudo de caso para uma NCS CAN com cinco malhas foi realizado e verificou-se o impacto dos parâmetros da rede no desempenho das malhas de controle.

Abstract

Networked control systems (NCSs) are ones where control loops are closed by means of a communication network. These systems can be found in many complex control applications such as the automotive, aircraft, robotics, manufacturing industry among others. The use of network architectures enable greater reliability and flexibility of operation, reducing the amount of cabling and costs, ease of installation, maintenance and system diagnostics. However, network architectures introduce new challenges that make the design and implementation of control systems a complex activity. Basically, communication networks can cause varying delays and loss of messages during the operating cycles of the control loops as well as others particularities that can degrade performance and even cause the instability of the physical processes. In the last two decades, several studies have been developed in an attempt to overcome these challenges. However, most of these works, the design methodologies are complex and generally the applicability is questionable due to unrealistic assumptions and especially the large computational effort required. Furthermore, many of these methodologies apply only in simpler architectures and is also common to consider NCSs with a single control loop, which is not consistent with reality applications. On the other hand, some studies have allowed the extension of control techniques simpler and already established in practice in the design of NCSs. However, some behaviors caused by network architecture are still disregarded, which in practice compromise the functioning of the system.

This work presents a procedure to design networked control systems (NCSs) using CAN Network, considering the integration of related theories, in order to check and mitigate the effects caused by the network architectures in the performance control loops. Firstly, were developed four models of the CAN network in timed automata and priced timed automata considering different network behavior characteristics. Two of those models have been extended to model CAN NCSs and determine the temporal behavior of the control loops. Then, the procedure was developed, that consists of three activities: to design PIDs controllers for processes with one input and one output, considering variants control delays; to develop a framework using formal modeling in timed automata and priced timed automata, and approaches classical and statistical model-cheking to determine the temporal behavior of the control loops in order to verify the conditions under which the time requirements will be satisfied, and develop an experimental plataform to simulate in real-time the CAN NCSs using technique hardware in the loop, including as real parts a CAN network and a random fault injector. A case study to a CAN NCS with five control loops was realized and the impact caused by network in performance control loopd was checked.

Sumário

Glossário

$\bar{Fd}_i$	Fator de degradação de desempenho médio de uma malha i
Δ	Atraso de controle variante
δ_{max}	Máximo atraso de controle variante
$\hat{\rho}_{x_i}$	Taxa de variação do relógio do nó de rede i
$\hat{\rho}_x$	Taxa de variação do relógio x
$\hat{\rho}_x^{max}$	Taxa de variação máxima do relógio x
$\hat{\rho}_x^{min}$	Taxa de variação mínima do relógio x
$\hat{\varphi}_m$	Margem de fase aparente
ω	Frequência angular
ω_b	Largura de banda do sistema contínuo em malha fechada.
ω_c	Frequência de cruzamento do ganho
ω_n	Frequência natural não-amortecida
ρ_x	Variação do relógio x
ρ_x^{max}	Variação máxima do relógio x
ρ_x^{min}	Variação mínima do relógio x
τ	Intervalo máximo de transferência permissível
τ_a	Tarefa de atuação
τ_{bit}	Tempo de transmissão de um *bit*
τ_c	Tarefa de computação
τ_i	Tarefa i
τ_{m_i}	Tarefa periódica relacionada a mensagem m_i
τ_s	Tarefa de aquisição
ε	Coeficiente de amortecimento
ε_r	Coeficiente de amortecimento relativo

φ_m	Margem de fase
A_c	Conjunto de ações
a_i	Instante de ativação da tarefa τ_i
a_{m_i}	Instante de ativação da mensagem m_i
a_{M_j}	Instante de ativação do ciclo de operação de uma malha de controle j
$B(X)$	Conjunto de restrições dos relógios
B_{m_i}	Atraso de bloqueio da mensagem m_i
b_{m_i}	Quantidade de *byte* de dados da mensagem m_i
$C(s)$	Função de transferência de um controlador contínuo no tempo
$C(z)$	Função de transferência de um controlador discreto no tempo
C_{a_j}	Tempo de computação da tarefa de atuação de uma malha de controle j
C_A	Controlador analógico
C_{c_j}	Tempo de computação da tarefa de controle de uma malha de controle j
C_D	Controlador digital
C_i	Tempo de computação da tarefa τ_i
C_i^{max}	Tempo de computação no pior caso da tarefa τ_i
C_i^{min}	Tempo de computação no melhor caso da tarefa τ_i
$C_{m_{ca_j}}$	Tempo de transmissão da mensagem controlador-atuador de uma malha de controle j
C_{m_i}	Tempo de computação da mensagem m_i
$C_{m_{sc_j}}$	Tempo de transmissão da mensagem sensor-controlador de uma malha de controle j
C_{s_j}	Tempo de computação da tarefa de aquisição de uma malha de controle j
D	*Deadline* relativo
$d(t_k)$	Termo derivativo do controlador PID no tempo discreto
D_i	*Deadline* relativo da tarefa τ_i
d_i	*Deadline* absoluto da tarefa τ_i
D_{m_i}	*Deadline* da mensagem m_i
d_{m_i}	*Deadline* absoluto da mensagem m_i
D_{M_j}	*Deadline* fim a fim de uma malha de controle j

E	Conjunto de bordas
$e(t)$	Sinal de erro no tempo contínuo
$e(t_k)$	Sinal de erro no tempo discreto
e_d	Erro derivativo
e_p	Erro proporcional
f	Frequência nominal
F_d	Fator de degradação de desempenho
f_i	Instante de finalização da tarefa τ_i
f_{m_i}	Instante de finazalição da mensagem m_i
f_x	Frequência real
Fd_i	Fator de degradação de desempenho de uma malha i
h	Período de amostragem
$hep(i)$	Conjunto de mensagens com prioridade maior ou igual a i
$hp(i)$	Conjunto de tarefas com maior prioridade que a tarefa τ_i
I	Função que atribui restrições de tempo aos lugares
$i(t_k)$	Termo integral do controlador PID no tempo discreto
J	*Jitter*
J_i	*Jitter* de liberação da tarefa τ_i
J_i^{max}	*Jitter* de liberação no pior caso da tarefa τ_i
J_i^{min}	*Jitter* de liberação no melhor caso da tarefa τ_i
J_{m_i}	*Jitter* de armazenamento da mensagem m_i
J_m	Margem de *jitter*
K	Ganho proporcional
k	Número do ciclo de operação ou transação de uma malha de controle
k_d	Ganho derivativo
k_i	Ganho integral
k_p	Ganho proporcional
L	Atraso constante
l_0	Lugar inicial
L_e	Conjunto finito de lugares

l_i	Instante de liberação da tarefa τ_i
l_{m_i}	Instante de liberação da mensagem m_i
L_m	Margem de atraso
$lp(i)$	Conjunto de mensagens com menor prioridade que a mensagem m_i
M	Valor da sensibilidade máxima usada nas regras AMIGO
m_{ca}	Mensagem controlador-atuador
m_i	Mensagem i
M_p	*Overshoot* máximo
m_{sc}	Mensagem sensor-controlador
N_i	Nó de rede i
O	*Offset*
O_i	*Offset* da tarefa τ_i
O_{M_j}	*Offset* de uma malha de controle j
p	Número de malhas de controle
p	Número de tarefas
$P(s)$	Função de transferência de um processo físico contínuo no tempo
$p(t_k)$	Termo proporcional do controlador PID no tempo discreto
$P(z)$	Função de transferência de um processo físico discreto no tempo
$P_{ZOH}(z)$	Discretização ZOH de P(s)
q	Quantidade de mensagens
Q_{m_i}	Quantidade de instâncias da mensagem m_i no maior período ocupado
$r(t)$	Sinal de referência no tempo contínuo
$r(t_k)$	Sinal de referência no tempo discreto
r_a	Tempo de resposta da tarefa de atuação
r_c	Tempo de resposta da tarefa de controle
r_i	tempo de resposta da tarefa τ_i
R_i^{max}	Tempo de resposta no pior caso da tarefa τ_i
R_i^{min}	Tempo de resposta no melhor caso da tarefa τ_i
$R_{m_{ca_j}}^{max}$	Tempo de resposta no pior caso da mensagem controlador-atuador de uma malha de controle j

$R^{min}_{m_{ca_j}}$	Tempo de resposta no melhor caso da mensagem controlador-atuador de uma malha de controle j
$r_{m_{ca}}$	Tempo de resposta da mensagem controlador-atuador
$R^{max}_{m_i}$	Tempo de resposta no pior caso da mensagem m_i
r_{M_j}	Tempo de resposta fim a fim de uma malha de controle j
$R^{max}_{M_j}$	Tempo de resposta fim a fim no pior caso de uma malha de controle j
$R^{min}_{M_j}$	Tempo de resposta fim a fim no melhor caso de uma malha de controle j
$R^{max}_{m_{sc_j}}$	Tempo de resposta no pior caso da mensagem sensor-controlador de uma malha de controle j
$R^{min}_{m_{sc_j}}$	Tempo de resposta no melhor caso da mensagem sensor-controlador de uma malha de controle j
$r_{m_{sc}}$	Tempo de resposta da mensagem sensor-controlador
r_s	Tempo de resposta da tarefa de aquisição
s	Variável complexa da transformada de Laplace
s_{e_0}	Estado inicial
S_e	Conjunto de estados
S_i	Soma do tempo de computação com o atraso de interferência de uma tarefa τ_i
t	Valor do tempo real
T_d	Tempo de ação derivativa
t_d	Tempo de atraso
$t_{ideal}(t)$	Valor do relógio ideal no instante t
T_{in}	Tempo de ação integral
T_i	Período da tarefa τ_i
t_k	Instante de amostragem k
T_{m_i}	Período da mensagem m_i
t_{m_i}	Maior período ocupado com nível de prioridade i
t_p	Instante de pico
t_r	Tempo de subida
t_s	Tempo de acomodação
U	Fator de utilização do processador

u	Função avaliação dos relógios
$u(t)$	Sinal de entrada do processo no tempo contínuo
$u(t_k)$	Sinal de controle ou sinal de entrada do processo no tempo discreto
U_m	Fator de utilização da rede
W_{m_i}	Atraso de enfileiramento da mensagem m_i
X	Conjunto de relógios
x_i	Relógio do nó de rede i
$Y(s)$	Transformada de Laplace do sinal y(t)
$y(t)$	Sinal de saída do processo no tempo contínuo
$y(t_k)$	Sinal de saída do processo no tempo discreto
$Y(z)$	Transformada Z do sinal y(t)
z	Variável complexa da transformada Z
A	Atuador
A/D	Analógico Digital
AFDX	*Avionics Full Duplex Switched Ethernet*
AMIGO	*Approximate M-constrained Integral Gain Optimization*
ASI	*Actuator Sensor Interface*
CAL	*CAN Application Layer*
CAN	*Controller Area Network*
CMC	*Classical Model Checking*
CSMA/DCR	Carrier Sense Multiple AccessDeterministic Collision Resolution
D/A	Digital Analógico
dB	Decibel
DBM	Difference Bound Matrices
EDF	*Earliest Deadline First*
F-MIGO	*Fractional M_s constrained Integral Gain Optimization*
FGOSC	*Fractional Gain and Order Scheduling Controller*
FIFO	*First In First Out*
FOLIPD	*First Order Lag plus Integrator plus Delay*
FOPDT	*First Order Plus Time Delay*

FOPI	*Fractional Order Proportional Integral*
FOTD	*First Order System with Time-Delay*
FTTCAN	*Flexible Time-Triggered Controller Area Network*
IAE	Integral Absolute Error
IMC	*Internal Model Control*
ISE	*Integral of Square Error*
ISO	International Organization for Standardization
ITAE	*Integral of Time-Weighted Absolute Error*
ITSE	*Integral of Time-Weighted Square Error*
LMI	*Linear Matrix Inequality*
LQG	*Linear Quadratic Gaussian*
MADB	*Maximum Allowable Delay Bound*
MATI	*Maximal Allowable Transfer Interval*
MIMO	*Multiple Input, Multiple Output*
MMC	Mínimo Múltiplo Comum
NA	Nó atuador
NC	Nó controlador
NC/A	Nó Controlador/Atuador
NCSs	*Networked Control Systems*
NCSWT	*Networked Control Systems WindTunnel*
NIF	Nó Injetor de Falhas
NPTA	Networks of Price Timed Automata
NRZ	*Non-Return to Zero*
NS	Nó sensor
NS/C	Nó Sensor/Controlador
OSI	*Open Systems Interconnections*
P	Processo físico
PI	Proporcional Integral
PID	Proporcional Integral Derivativo
ppm	Partes por milhão

PROFIBUS	*Process Field Bus*
PTA	Priced Timed Automata
RTAI	Real-Time Application Interface
S	Sensor
SDTRs	Sistemas distribuídos em tempo real
SHIL	Sistema hardware in the loop
SISO	*Single Input Single Output*
SMC	*Statistical Model Checking*
SOTR	Sistema operacional em tempo real
STRs	Sistemas em tempo real
TDMA	*Time Division Multiple Access*
TOD	*Try-Once-Discard*
TTCAN	*Time-Triggered Controller Area Network*
TTP	*Time-Triggered Protocol*
UP	Unidade de Processamento
WNCSs	*Wireless Networked Control Systems*
ZOH	Segurador de ordem zero

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

Redes de comunicação tornaram-se essenciais na interconexão dos componentes das malhas de controle (sensores, controladores e atuadores) utilizados no controle de processos físicos em várias aplicações na engenharia. Esses sistemas são normalmente chamados de Sistemas de Controle via Rede (NCSs - *Networked control Systems*[1]) (HESPANHA; NAGHSHTABRIZI; XU, 2007).

A utilização de arquiteturas em rede, ao invés das tradicionais arquiteturas ponto a ponto[2], foi impulsionada pelo aumento da complexidade nas aplicações automotivas, em aeronaves, na robótica, na indústria de manufatura entre outras. Nessas aplicações, os sistemas tornaram-se maiores, com várias malhas de controle e com vários pontos de medição e de atuação distribuídos geograficamente. Como conseqüência, houve um aumento na quantidade dos cabos conectando os sensores e atuadores ao computador. Então, problemas com os custos de instalação, dificuldades nos diagnósticos e nas manutenções, além da redução da confiabilidade e flexibilidade de operação do sistema começaram a surgir, tornando as arquiteturas ponto a ponto inapropriados nessas aplicações. Com isso, a adoção de uma arquitetura em rede tornou-se providencial, pois possibilita uma melhoria na confiabilidade e flexibilidade de operação, redução da quantidade de cabos e dos custos, facilidade de instalação, manutenção e diagnóstico do sistema (BROSTER, 2003).

Apesar das vantagens, a utilização de arquiteturas em rede introduzem novos desafios, que tornam o projeto de sistemas de controle uma atividade complexa (WITTENMARK; NILSSON; TöRNGREN, 1995). Basicamente, redes de comunicação podem causar atrasos variáveis e perdas de mensagens durante os ciclos de operação das malhas de controle, além de outras particula-

[1] NCSs são também chamados de sistemas de controle distribuído em tempo real.

[2] Sistemas de controle em malha fechada que utilizam arquiteturas ponto a ponto são chamados de sistemas de controle via computador (ASTROM; WITTENMARK, 1997).

ridades, que podem degradar o desempenho e até causar a instabilidade dos processos físicos. Esses problemas são gerados, principalmente, pelo compartilhamento do meio de comunicação entre as malhas de controle, devido a distúrbios externos aleatórios no meio de comunicação (e.g. interferências eletromagnéticas.), e pelas características da rede (e.g. controle de acesso ao meio, mecanismos de sinalização e recuperação de erros nas mensagens.). De fato, as metodologias existentes para desenvolver sistemas de controle não consideram os efeitos causados pela arquitetura em rede.

Durante as duas últimas décadas, diversos trabalhos foram desenvolvidos na tentativa de superar esses desafios. Um resumo com os principais resultados são apresentados em Tipsuwan e Chow (2003), Yang (2006), Hespanha, Naghshtabrizi e Xu (2007) e You e Xie (2013). Apesar da grande quantidade de soluções, o problema nos NCSs ainda está aberto (ZAMPIERI, 2008). De acordo com Murray et al. (2003), no relatório técnico do painel "Direções futuras em Controle, Dinâmicas e Sistemas", indicou-se NCSs como uma área de pesquisa promissora para o futuro.

Na maioria dos trabalhos desenvolvidos, as técnicas utilizadas, do ponto de vista de controle, são complexas e geralmente a aplicabilidade é questionável, devido a suposições não realistas e, principalmente, ao grande esforço computacional exigido (ERIKSSON; OKSANEN; MIKKOLA, 2009). De acordo com Hespanha, Naghshtabrizi e Xu (2007), muitas dessas técnicas se aplicam apenas em arquiteturas mais simples e é também comum considerar NCSs com uma única malha de controle, o que não condiz com a realidade das aplicações. Por outro lado, alguns trabalhos ((KAO; LINCOLN, 2004), (CERVIN et al., 2004) e (MIRKIN, 2007)) têm possibilitado a extensão de técnicas de controle mais simples e já consolidadas na prática (e.g. controle PID (Proporcional Integral Derivativo)), no projeto e análise de NCSs (SANTOS, 2008), (ERIKSSON, 2008), (BJöRKBOM, 2010) e (NIEMINEN et al., 2011). Entretanto, alguns comportamentos causados pela arquitetura em rede ainda são desconsiderados (e.g. os atrasos gerados pelos mecanismos de sinalização e recuperação de erros nas mensagens), os quais na prática comprometem o funcionamento do sistema.

Nessas técnicas é fundamental determinar o comportamento temporal das malhas de controle durante a fase de projeto, pois são baseadas em suposições envolvendo requisitos de tempo e/ou fornecem limites para os atrasos de controle (tempos de resposta fim a fim) que garantem a estabilidade das malhas de controle. Entretanto, as abordagens normalmente utilizadas para determinar o comportamento temporal geram resultados pessimistas, os quais podem conduzir a um projeto com recursos subutilizados. Além disso, é comum não considerar erros no meio de comunicação causados por exemplo, por interferências eletromagnéticas.

Um outro fator essencial para o desenvolvimento de NCSs é a utilização de ferramentas e técnicas para simulação e teste que envolvem a interação entre redes de comunicação e dinâmicas de sistemas físicos de modo realista. Atualmente, existem algumas ferramentas disponí-

veis tais como, TRUETIME (CERVIN et al., 2003), PiccSIM (KOHTAMAKI et al., 2009), NCSWT (*Networked Control Systems WindTunnel*) (EYISI et al., 2012) entre outras. Entretanto, a maioria delas não simula ou simula de modo impreciso, eventos e dinâmicas específicas da rede de comunicação que ocorrem na prática tais como, erros no meio de comunicação.

O projeto de NCSs requer a consideração de vários fatores originados pela arquitetura em rede que são cruciais no desempenho do sistema. NCSs envolvem um campo com várias disciplinas formado pelas teorias de controle, comunicação e computação em tempo real (MITTER, 2002) e (GRAHAM; KUMAR, 2003). Portanto, a integração dessas teorias é providencial para o desenvolvimento de metodologias capazes de solucionar esses problemas.

1.2 Objetivos

Neste trabalho foi desenvolvido um procedimento para projetar sistemas de controle via rede CAN (*Controller Area Network*) (BOSCH, 1991), considerando a integração das teorias relacionadas, de modo a verificar e mitigar os efeitos causados pela arquitetura em rede no desempenho das malhas de controle.

1.2.1 Objetivos específicos

Especificamente, teve-se como objetivos:

- Desenvolver um *framework* utilizando modelagem formal em autômatos temporizados e autômatos temporizados com custo, e as abordagens verificação de modelos clássica e estatística, para determinar o comportamento temporal de redes CAN, especificamente, os tempos de resposta de mensagens periódicas, considerando *offsets* estáticos e dinâmicos, relógios ideais e reais e a ausência e presença de erros no meio de comunicação ocasionados por falhas aleatórias;

- Modelar três das quatro principais arquiteturas de NCSs CAN, a partir da extensão dos modelos desenvolvidos no objetivo anterior, e utilizar as abordagens verificação de modelos clássica e estatística para determinar o comportamento temporal dessas três arquiteturas, de modo a verificar as condições em que os requisitos de tempo das malhas de controle serão satisfeitas;

- Desenvolver um procedimento para projetar NCSs CAN com controladores PIDs, considerando atrasos de controle variantes no tempo, usando os resultados obtidos no objetivo anterior;

- Desenvolver uma plataforma experimental para simular em tempo real NCSs CAN usando a técnica *hardware in the loop* (LEDIN, 1999), incluindo como partes reais uma rede CAN e um injetor de falhas aleatórias no meio de comunicação.

1.3 Organização do texto

No Capítulo 2 apresenta-se uma breve revisão histórica sobre sistemas de controle. Em seguida, define-se os sistemas de controle via rede, suas principais arquiteturas e os problemas ocasionados pela rede de comuicação. Depois, apresenta-se uma revisão bibliográfica indicando as abordagens, técnicas e suposições geralmente adotadas no projeto de NCSs, e as abordagens usadas para determinar o comportamento temporal de sistemas em rede.

No Capítulo 3 apresentam-se os conceitos e definições básicas sobre as teorias relacionadas aos NCSs usadas neste trabalho. Inicialmente, apresenta-se a teoria clássica de sistemas de controle em malha fechada, abordando a modelagem, projeto e análise de sistemas. Em seguida, os controladores PIDs e os teoremas para análise de estabilidade de sistemas de controle com atrasos variantes são apresentados. Depois, apresenta-se a teoria sobre sistemas em tempo real e a rede de comunicação CAN. Finaliza-se o capítulo apresentando as técnicas de verificação de modelos clássica e estatística e as definições sobre autômatos temporizados e autômatos temporizados com custo.

No Capítulo 4 apresentam-se os modelos em autômatos temporizados da rede CAN considerando diferentes características do comportamento da rede. Para cada modelo são apresentados os trabalhos relacionados e as vantagens dos modelos propostos. Em seguida, estudos de caso usando as técnicas de verificação de modelos clássica e estatística são realizados para determinar os tempos de resposta no melhor e pior caso das mensagens, e os resultados são comparados com a técnica analítica existente. Dois modelos são validados a partir de resultados obtidos em uma plataforma experimental utilizando testes de aderência.

No Capítulo 5, dois modelos da rede CAN são estendidos para modelar três das quatro principais arquiteturas de NCSs CAN. A partir desses modelos, utilizou-se as técnicas de verificação de modelos simbólica e estatística para determinar o comportamento temporal das malhas de controle. Estudos de caso serão realizados e os resultados serão comparados com a técnica analítica.

Capítulo 6 apresentam-se um procedimento para projetar NCSs CAN e uma plataforma para simulação em tempo real de NCSs CAN. Em seguida, apresenta-se um estudo de caso seguindo o procedimento proposto.

Finalmente, no Capítulo 7, apresentam-se as conclusões e os trabalhos futuros.

Capítulo 2

Sistemas de controle via rede

2.1 Introdução aos sistemas de controle

Sistemas de controle em malha fechada são usados para controlar processos físicos de acordo com especificações de desempenho. Basicamente, esses sistemas são constituídos pelo processo a ser controlado e pelos dispositivos sensor, controlador e atuador. As conexões entre esses dispositivos e o processo formam uma malha de controle.

Ao longo dos anos, os avanços em várias áreas possibilitaram a evolução dos sistemas de controle. Nos primeiros sistemas, chamados de sistemas de controle analógico (OGATA, 2000), todos os sinais eram contínuos no tempo, conforme representado na Figura 2.1. De modo contínuo no tempo, o sensor (**S**) convertia o sinal de saída do processo (**P**) em um sinal específico ($y(t)$). Esse sinal era comparado com um sinal de referência ou valor desejado ($r(t)$), e a diferença, chamada de erro de referência ($e(t)$), era aplicada na entrada do controlador analógico ($\mathbf{C_A}$). O controlador gerava um sinal de controle ($u(t)$), que era aplicado na entrada do processo pelo atuador (**A**).

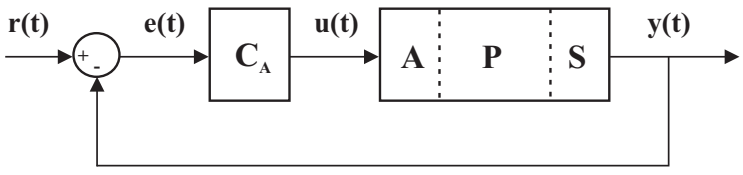


Figura 2.1: Representação básica de um sistema de controle analógico com uma malha.

A partir da década de 20, vários trabalhos voltados para o projeto e análise de estabilidade de sistemas de controle resultaram no desenvolvimento da teoria de controle clássica. Apesar desses avanços, existiam alguns problemas relacionados com o modo de implementação dos controladores analógicos. As equações que geravam as ações de controle necessárias para garantir a estabilidade e desempenho do sistema eram implementadas por meio de circuitos eletrônicos. No entanto, o envelhecimento dos componentes desses circuitos alteravam suas

características, gerando mudanças indesejadas nas ações de controle.

Com os avanços nas teorias de sistemas discretos e a disponibilidade dos primeiros computadores digitais, ocorridos aproximadamente a partir da década de 60, tornou-se possível analisar sistemas complexos com várias entradas e várias saídas baseado nas técnicas de análise e síntese através de variáveis de estado. Esses avanços resultaram no desenvolvimento da teoria de controle moderno e no surgimento dos sistemas de controle via computador ou sistemas de controle digital (ASTROM; WITTENMARK, 1997).

Sistemas de controle via computador são semelhantes aos sistemas de controle analógico. A principal diferença é que o controlador é implementado por um programa (*software*) em um computador. Desse modo, além dos sinais contínuos no tempo provenientes do processo a ser controlado, o sistema contém sinais discretos no tempo devido ao computador. Para realizar a transformação desses sinais, o computador usa conversores A/D (Analógico/Digital) e D/A (Digital/Analógico).

Nos sistemas de controle via computador (Figura 2.2), o ciclo de operação k do controle é iniciado, normalmente, de modo periódico a cada instante $t_k = kh$, $\forall\, k \in \mathbb{Z}^+ = \{0, 1, 2, ...\}$, onde h é o período de amostragem do sinal de saída do processo. Então, o sinal de saída do processo (grandeza física contínua no tempo) é convertido pelo sensor (**S**) e transformado para a forma digital[1] ($y(t_k)$) via o conversor **A/D** integrado ao computador. A diferença entre o sinal de referência $r(t_k)$ e o sinal $y(t_k)$, chamada de erro de referência ($e(t_k)$), é usada para gerar o sinal de controle $u(t_k)$ por meio do controlador ($\mathbf{C_D}$). O sinal de controle é convertido para a forma analógica pelo conversor **D/A** e aplicado na entrada do processo via o atuador (**A**)[2], finalizando o ciclo de operação.

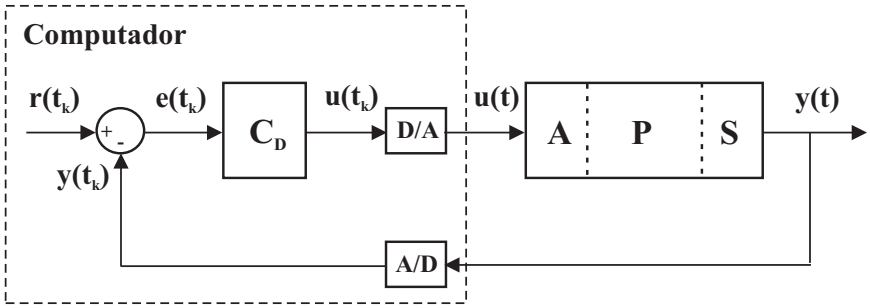


Figura 2.2: Representação básica de um sistema de controle via computador com uma malha.

Do ponto de vista computacional, todas as atividades de uma malha de controle são atribuídas a uma tarefa de tempo real[3] sincronizada pelo relógio do computador. A cada instante t_k, uma instância k dessa tarefa é iniciada e gasta algum tempo executando sequencialmente

[1] Sinal discreto e quantizado pelo conversor A/D.

[2] Esse sinal é mantido constante durante o período h por meio de um segurador de ordem zero implementado no conversor **D/A**.

[3] Programa de computador que realiza atividades específicas com restrições de tempo.

três atividades: aquisição do sinal de saída do processo, execução do algoritmo de controle e aplicação do sinal de controle na entrada do processo. O tempo gasto desde o instante t_k até o instante em que o sinal de controle é aplicado no processo é chamado de atraso de controle.

No projeto de sistemas de controle via computador, é comum assumir amostragem eqüidistante e invariância no tempo. A invariância no tempo requer que os atrasos de controle sejam constantes e que as ações de amostragem e controle ocorram em instantes bem definidos. As variações de tempo, caso existam, devem ser desprezíveis quando comparadas com as dinâmicas do processo ou inclusas no projeto (WITTENMARK; NILSSON; TöRNGREN, 1995). Além disso, assume-se que o periodo de amostragem (h) e a quantização do sinal são suficientemente pequenos, de modo que os sinais digitais obtidos se aproximem da versão contínua do sinal. Assim, os métodos de projeto usados nos sistemas de controle analógico podem ser usados diretamente. O projeto pode ser efetuado de duas formas: projetar o sistema no tempo contínuo e, posteriormente, transformá-lo no tempo discreto usando algum dos métodos disponíveis (e.g. método de Tustin (ASTROM; WITTENMARK, 1997)); ou projetá-lo diretamente no tempo discreto. Ambas as formas podem ser conduzidas usando a teoria de controle clássica ou moderna.

Sistemas de controle via computador predominaram nas diversas aplicações. No entanto, a complexidade atual de várias aplicações tem impossibilitado o uso desses sistemas. Assim, surgiram os Sistemas de Controle via Rede (NCSs).

NCSs são sistemas onde malhas de controle são fechadas por meio de uma rede de comunicação. Nesses sistemas vários nós de rede executam tarefas computacionais relacionadas ao controle dos processos físicos e se comunicam por mensagens transmitidas via um meio de comunicação compartilhado com largura de banda limitada.

Normalmente, uma malha de controle em um NCS é constituída pelo processo a ser controlado e por três nós de rede (nó sensor (**NS**), nó controlador (**NC**) e nó atuador (**NA**)) conectados na topologia barramento. Cada nó possui uma unidade de processamento (**UP**), dispositivos de rede e sensores (**S**) ou atuadores (**A**), conforme as representações na Figura 2.3, além de alguns circuitos básicos (omitidos na Figura) tais como, circuito oscilador, que é responsável por gerar a base de tempo das atividades temporizadas da UP e dos dispositivos de rede.

Os nós executam tarefas, que podem ser de três tipos: tarefa de aquisição (τ_s) - executada no nó sensor; tarefa de controle (τ_c) - executada no nó controlador; e tarefa de atuação (τ_a) - executada no nó atuador. As tarefas podem ser classificadas de acordo com a forma que são ativadas como: guiadas a tempo (*time-driven*) ou guiadas a evento (*event-driven*)[4]. Em uma tarefa guiada a tempo, a ativação ocorre por meio de um evento temporizado, por exemplo, a interrupção de um relógio pré-programado. Enquanto que em uma tarefa guiada a evento, a ativação ocorre via algum evento não-temporizado, por exemplo, uma interrupção gerada pela chegada de uma mensagem no *buffer* de recepção do nó. Geralmente, a tarefa de aquisição

[4]É comum na literatura classificar os nós do mesmo modo. Nó guiado a tempo e nó guiado a evento.

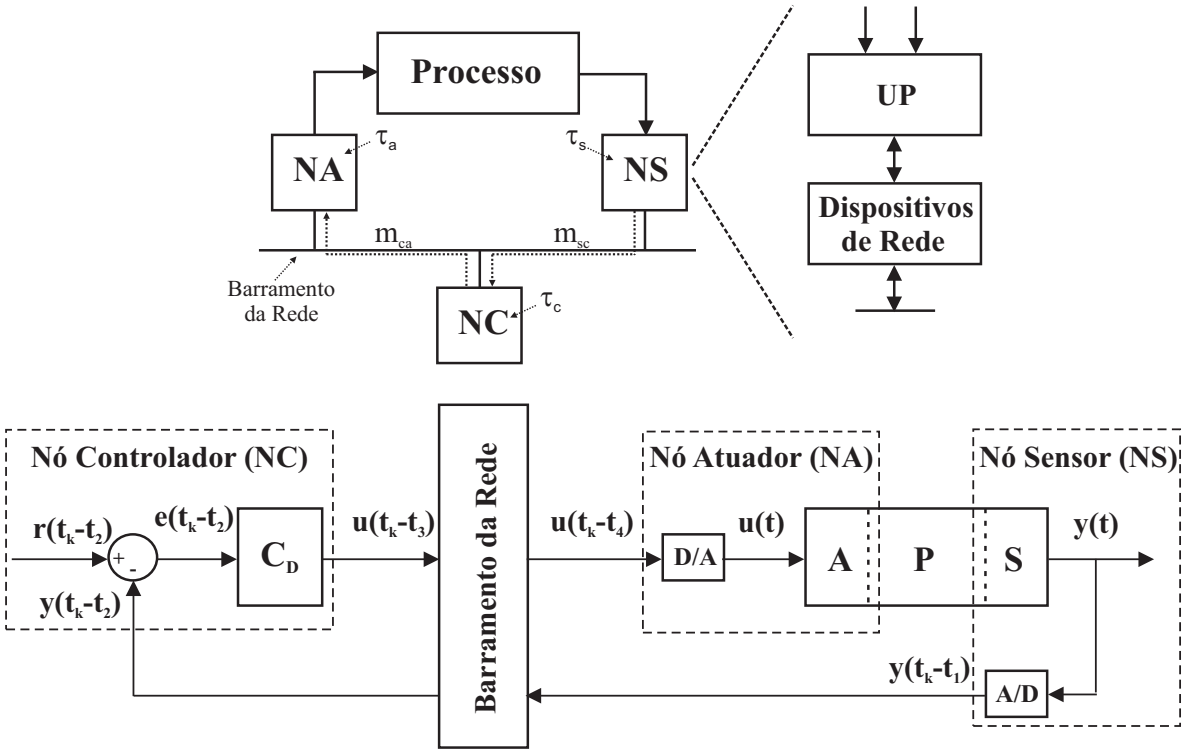


Figura 2.3: Representações básicas de um sistema de controle via rede com uma única malha.

é guiada a tempo e as tarefas de controle e atuação são guiadas a eventos. A partir dessa consideração, garante-se que o sinal de saída do processo será amostrado periodicamente, e que o cálculo da ação de controle e atuação ocorrerão assim que a amostra do sinal de saída do processo e o sinal de controle, respectivamente, estejam disponíveis.

Basicamente, o $k-$ésimo ciclo de operação ou transação de uma malha (ver Figura 2.3) é iniciado periodicamente no instante $t_k = O + kh, \forall\, k \in \mathbb{Z}^+ = \{0, 1, 2, 3,...\}$, onde O, chamado de *offset*, é o instante em que o primeiro ciclo de operação da malha é iniciado, e h é o período de amostragem da malha de controle. O ciclo de operação consiste de uma sequência de tarefas/mensagens a serem executadas/transmitidas em uma ordem fixa. O tempo gasto durante uma transação do ciclo de operação de uma malha é chamado de atraso de controle ou tempo de resposta fim a fim[5]. O instante t_k é marcado por um relógio no nó sensor (**NS**), que inicia a execução da tarefa de aquisição (τ_s). Essa tarefa adquire uma amostra do sinal de saída do processo ($y(t)$) e requisita ao dispositivo de rede a transmissão da mensagem sensor$-$controlador (m_{sc}) contendo essa amostra. Ao receber a mensagem, o nó controlador (**NC**) inicia a execução da tarefa de controle (τ_c). Essa tarefa retira da mensagem a amostra do sinal de saída do processo e executa o algoritmo de controle para gerar o sinal de controle. Em seguida, requisita

[5]O termo atraso de controle é comumente usado pela comunidade de Engenharia de Controle. Enquanto que o termo tempo de resposta fim a fim é usado pela comunidade de Engenharia de Computação. Visto que NCSs são sistemas que requerem conhecimentos dessas engenharias, esses termos serão usados ao longo do texto de forma intercambiável.

a transmissão da mensagem controlador−atuador (m_{ca}) contendo o sinal de controle para o nó atuador (**NA**). Ao receber essa mensagem, a tarefa de atuação (τ_a) é iniciada. Essa tarefa retira o sinal de controle da mensagem e transforma-o em um sinal analógico ($u(t)$), que é aplicado na entrada do processo por meio de um atuador, finalizando o ciclo de operação k.

Há várias diferenças entre os NCSs e os sistemas de controle via computador, que exigem uma reformulação das técnicas de projeto existentes. Do ponto de vista computacional, a tarefa de tempo real da malha de controle é dividida em três tarefas (tarefa de aquisição, tarefa de controle e tarefa de atuação) que são executadas em diferentes nós de rede (nó sensor, nó controlador e nó atuador, respectivamente) conectados por meio de um barramento compartilhado. As atividades de cada tarefa são guiadas por diferentes relógios e devido a rede de comunicação, os atrasos de controle são variantes no tempo. Os sinais digitais em cada nó são atrasados com relação ao instante de amostragem (t_k), devido aos atrasos de computação e comunicação. Assim sendo, tem-se:

$$t_1 = r_s \tag{2.1}$$

$$t_2 = r_s + r_{m_{sc}} \tag{2.2}$$

$$t_3 = r_s + r_{m_{sc}} + r_c \tag{2.3}$$

$$t_4 = r_s + r_{m_{sc}} + r_c + r_{m_{ca}} \tag{2.4}$$

$$t_5 = r_M = r_s + r_{m_{sc}} + r_c + r_{m_{ca}} + r_a \tag{2.5}$$

Onde: r_s é o tempo de resposta da tarefa de aquisição (τ_s); $r_{m_{sc}}$ é o tempo de resposta da mensagem sensor-controlador (m_{sc}); r_c é o tempo de resposta da tarefa de controle (τ_c), $r_{m_{ca}}$ é o tempo de resposta da mensagem controlador-atuador (m_{ca}) e r_a é o tempo de resposta da tarefa de atuação (τ_a). O atraso de controle r_M de um ciclo de operação k em função dos atrasos de computação e comunicação (Eqs. 2.1 a 2.5) está representado pelo diagrama de tempo da Figura 2.4.

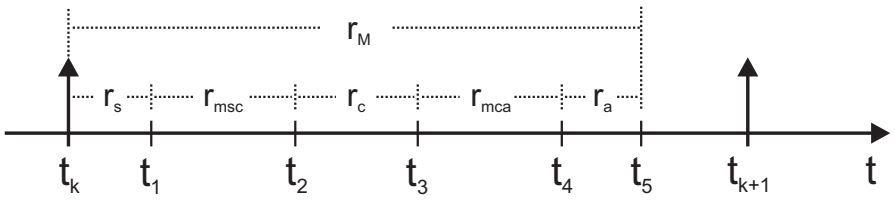


Figura 2.4: Diagrama de tempo do atraso de controle de uma malha com os tempos de resposta das tarefas e mensagens.

2.1.1 Principais arquiteturas de NCSs

De acordo com a disposição do sensor, controlador e atuador, é possível ter outros tipos de nós e, conseqüentemente, diferentes arquiteturas de NCSs, conforme apresentado na Figura 2.5 (YEPEZ; MARTI; FUERTES, 2002) e (SANTOS, 2008).

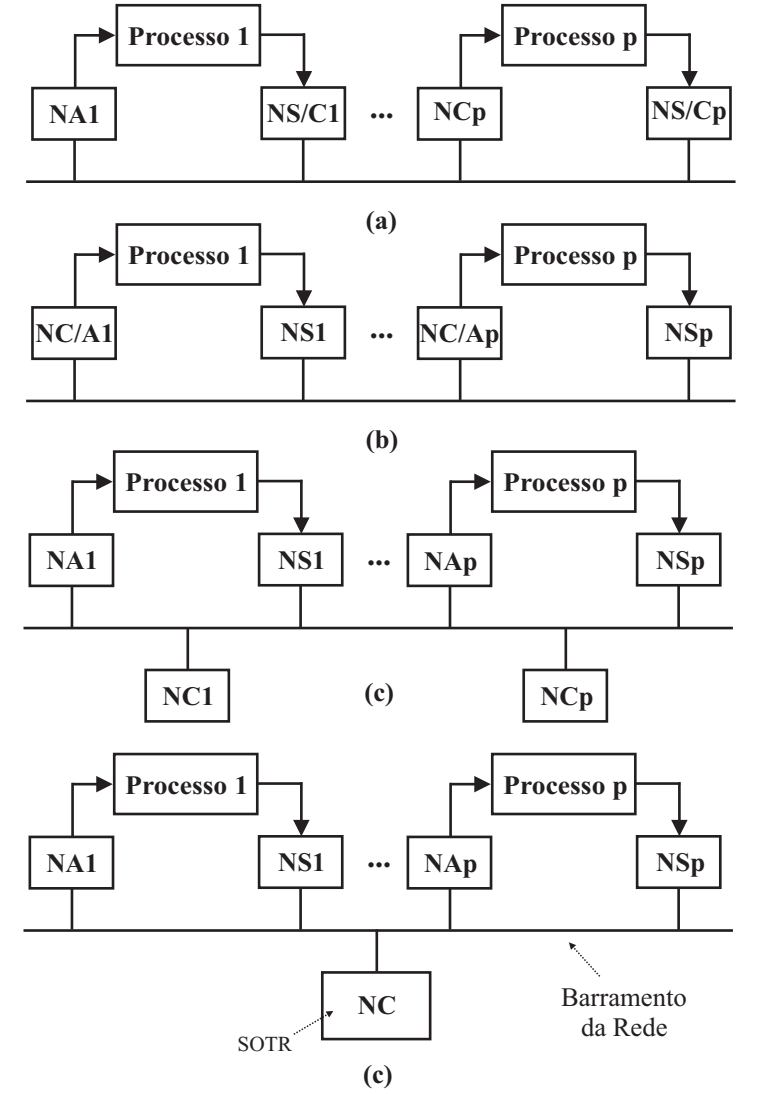


Figura 2.5: Principais arquiteturas de sistemas de controle via rede com p malhas.

Na arquitetura representa na Figura 2.5 (a), o sensor e o controlador estão em um único nó, chamado de nó sensor/controlador (**NS/C**). Enquanto que, na arquitetura representada na Figura 2.5(b), o controlador e o atuador estão juntos em um nó, chamado de nó controlador/atuador (**NC/A**). Nessas arquiteturas, apenas uma mensagem é transmitida durante o ciclo de operação da malha. É possível também ter o sensor e o atuador localizados em um único nó chamado de nó sensor/atuador. As arquiteturas seguintes, Figuras 2.5 (c) e (d), são semelhantes. A

diferença é que na última, todas as tarefas de controle são executadas em um único nó (**NC** - Nó Controlador), ao contrário da arquitetura na Figura 2.5 (c), onde cada malha possui o seu nó controlador específico. Em particular, na arquitetura representada pela Figura 2.5 (d), é necessário usar um sistema operacional em tempo real no nó controlador (**NC**) para escalonar a tarefa de controle de cada malha. Nessas arquiteturas, duas mensagens são transmitidas durante um ciclo de operação da malha. Do ponto de vista prático, as duas últimas arquiteturas são mais comuns e complexas, especificamente a arquitetura na Figura 2.5 (d).

2.1.2 Atrasos de controle

Os atrasos de controle podem ser constantes ou variáveis de modo determinístico ou aleatório, e dependem das características da rede (e.g. controle de acesso ao meio e mecanismos para recuperação de erros em mensagens.), do *hardware* dos nós, das tecnologias de *software* (e.g. características do SOTR[6] tais como, abordagens e políticas de escalonamento.) entre outros parâmetros necessários para o projeto do sistema em rede. Esses atrasos são constituídos pelos atrasos de computação, originados pelas tarefas executadas nas UPs, e pelos atrasos de comunicação, originados pelas mensagens transmitidas via o meio de comunicação compartilhado entre os nós das várias malhas de controle.

Os atrasos de computação podem ser constantes ou variantes de modo determinístico e dependem essencialmente de como as tarefas são executadas nos nós. Em particular, quando mais de uma tarefa é executada no mesmo nó (e.g. nó controlador da arquitetura representada na Figura 2.5 (d)), é necessário ter um sistema operacional em tempo real com um escalonador para escalonar as tarefas, e o tipo do escalonador determinará o comportamento do atraso.

Os atrasos de comunicação dependem da rede usada. Atualmente, existem no mercado diversas tecnologias de rede e uma das formas comuns de classificá-las é com relação ao escalonamento das mensagens, que pode ser *time-triggered*, *event-triggered* ou uma combinação desses dois tipos (TOVAR; PINHO; ALMEIDA, 2002), (ALBERT, 2004) e (METZNER; STIERAND, 2006).

Em redes *time-triggered*, as mensagens são transmitidas em instantes específicos de acordo com uma escala estática produzida durante o projeto do sistema. A escala é constituída por janelas de tempo com duração pré-determinada e alocadas para cada mensagem da rede. Então, as possíveis colisões entre mensagens são evitadas. Nesse tipo de rede, o comportamento temporal das mensagens é determinístico e previsível. No entanto, uma escala deve ser construída para cada possível configuração do sistema, resultando em uma baixa flexibilidade do ponto de vista de evolução do sistema. Além disso, é comum não haver retransmissão de mensagens devido a erros de transmissão, gerando a ausência de informação (sinal do sensor e/ou sinal

[6]Para a arquitetura representada na Figura 2.5 (d).

de controle) durante o ciclo de controle. Mas, é possível criar janelas de tempo maiores para uma eventual retransmissão, resultando em atrasos maiores e com alguma variabilidade. TTP (*Time-Triggered Protocol*) (KOPETZ; BAUER, 2003), (TTAGROUP, 2003) é um exemplo de rede *time-triggered*. Redes do tipo mestre escravo (ASI (*Actuator Sensor Interface*) (KRIESEL; MADELUNG, 1999) e Modbus (MODBUS, 2012)) e passagem de ficha (PROFIBUS (*Process Field Bus*) (PROFIBUS, 2000) e Controlnet (ODVA, 2012)) também seguem um esquema de escalonamento semelhante, onde colisões entre mensagens são evitadas.

Em redes *event-triggered*, as mensagens são escalonadas dinamicamente no meio de comunicação durante a execução do sistema. Algum método de arbitragem associado ao mecanismo de controle de acesso ao meio é usado para solucionar possíveis colisões entre mensagens. Nesse tipo de rede, o comportamento dinâmico na transmissão das mensagens fornece um alto nível de flexibilidade no sentido de que não existe necessidade de reconstruir a escala completa quando ocorrer uma mudança no sistema. Como desvantagem, os tempos de resposta das mensagens podem apresentar uma alta variabilidade, dependendo da carga da rede. CAN (*Controller Area Network*) (BOSCH, 1991) e (ETSCHBERGER, 2001), Ethernet (KUROSE; ROSS, 2003), Devicenet[7] (DEVICENET, 2011), Zigbee (ALLIANCE, 2006) e (BARONTI et al., 2007) são exemplos de redes *event-triggered*.

Nas redes *time-triggered* e *event-triggered* há características dos dois paradigmas de comunicação. O tempo é dividido em ciclos e esses ciclos são particionados em janelas *time-triggered* e *event-triggered*, onde as mensagens são escalonadas. TTCAN (*Time-Triggered* CAN) (LEEN; HEFFERNAN, 2002), FTTCAN (*Flexible Time-Triggered* CAN) (ALMEIDA; FONSECA; FONSECA, 1998), WorldFip (WORLDFIP, 2012), FlexRay (FLEXRAY, 2005), Byteflight (AG et al., 2001), WirelessHART (WIRELESSHART, 2011), (SONG et al., 2008) e ISA100.11a (ISA100, 2012) são exemplos de redes que usam esse paradigma de comunicação.

2.1.3 Perda de mensagens

A perda de mensagens é um dos principais problemas que ocorrem nos sistemas em rede. Esse problema pode ocorrer devido a erros de transmissão nas mensagens, gerados principalmente por interferências eletromagnéticas no meio de comunicação, pela perda do canal de comunicação, por causa das características dos mecanismos de controle de acesso ao meio, gerenciamento de erros em mensagens entre outros.

Em redes com fio, por exemplo, Ethernet (KUROSE; ROSS, 2003), o número de retransmissões sequenciais de uma mesma mensagem, devido a colisões com outras mensagens no meio de comunicação, é limitado em 16 tentativas. Não havendo sucesso na retransmissão, a mensagem é descartada e um erro de excesso de colisão é gerado. Na rede CAN (BOSCH, 1991),

[7]Protocolo das camadas superiores baseada na rede CAN, dedicada para automação industrial.

os mecanismos de controle de acesso ao meio, sinalização e recuperação de erros podem gerar longos períodos de espera para as mensagens serem transmitidas, e isso pode resultar na chegada da mensagem do próximo ciclo de operação (antes que a mensagem do ciclo anterior tenha sido transmitida) e até provocar estouros de capacidade nos *buffers* de transmissão dos nós. Por consequência, esses eventos podem gerar a perda de mensagens.

Em redes sem fio, tais como as baseadas nos padrões 802.15.4 (IEEE, 2006) e 802.11 (IEEE, 1999), o problema da perda de mensagem é mais crítico. Isso ocorre devido as propriedades adversas dos canais de rádio. É comum ocorrer erros de transmissão ocasionados pela perda do sinal de recepção e/ou interferências no meio (WILLIG; MATHEUS; WOLISZ, 2005). Assim como nas redes com fio, as características dos mecanismos de controle de acesso ao meio e gerenciamento de erros em mensagens também contribuem para a perda de mensagens.

2.1.4 Problemas específicos

Em uma malha de controle, a perda de mensagens e, principalmente, atrasos longos de transmissão resultam nos fenômenos da amostragem vazia e rejeição de mensagens (HALEVI; RAY, 1988). A amostragem vazia ocorre quando uma amostra da saída do processo, em um ciclo de operação k, não é recebida pelo nó controlador. A rejeição de mensagem ocorre quando duas amostras da saída do processo, referentes aos ciclos de operação $k-1$ e k, são recebidas pelo nó controlador durante o ciclo de operação k. Devido esses fenômenos estarem relacionados principalmente com os atrasos de tempo, a determinação deles pode ser prevista e até evitada durante a fase de projeto por meio de uma análise do comportamento temporal do sistema. Caso não seja possível evitá-los, é essencial desenvolver meios para detectá-los e tomar decisões durante a operação do sistema, de modo a evitar algum defeito no sistema de controle.

Nos sistemas em rede, cada nó possui um circuito oscilador normalmente de cristal de quartzo com uma frequência nominal conhecida, que é usado para definir a base de tempo do relógio físico do nó. Todas as atividades temporizadas e executadas no nó são guiadas por essa base de tempo. Devido a tolerâncias de fabricação, as frequências de oscilação dos cristais variam ao longo do tempo em torno da frequência nominal. Além disso, fatores como temperatura, nível de tensão elétrica aplicada ao circuito oscilador, envelhecimento do cristal entre outros, também influenciam na variação da frequência. Desse modo, a base de tempo dos relógios de cada nó variam ao longo do tempo e, consequentemente, existirá uma ausência de sincronização entre os relógios dos nós.

Além desses problemas, o projeto de NCSs envolve a escolha de parâmetros da rede e da própria malha de controle que são correlacionados. Por exemplo, em redes com priorização de mensagens (e.g. rede CAN.), é necessário usar políticas de atribuição de prioridade que influenciam diretamente nos atrasos de controle. Um outro parâmetro importante é o período de amostragem, visto que influencia na robustez e no desempenho da resposta transitória em

malha fechada. Enquanto que períodos de amostragem curtos podem assegurar a robustez e o desempenho em sistemas de controle via computador, essa escolha em NCSs aumenta o fator de utilização da rede e, consequentemente, aumentam os atrasos de controle e a variabilidade desses (LIAN, 2001).

2.2 Revisão bibliográfica

Nesta seção, apresenta-se uma revisão bibliográfica indicando as abordagens, técnicas e suposições geralmente adotadas no projeto de NCSs e as abordagens usadas para determinar os atrasos de controle ou tempos de resposta fim a fim.

2.2.1 Projeto de NCSs

Os primeiros trabalho sobre NCSs, desenvolvidos por Halevi e Ray (1988) e Ray e Halevi (1988), consistem de sistemas cujos modelos são discretos e variantes no tempo. A partir da consideração de que o processo e o controlador são invariantes no tempo e as entradas de controle são constantes por partes, o modelo do sistema foi representado por um vetor de estados aumentado, que consiste dos valores passados da entrada e saída do processo, além dos vetores de estado atual do processo e do controlador. Nesse modelo, foi considerado que o sensor e o controlador são guiados a tempo, e o atuador é guiado a eventos. O período de amostragem do sensor e do controlador são idênticos, mas podem ser separados por um intervalo limitado chamado de inclinação de tempo ocasionado pela variação dos relógios dos nós. Além disso, considerou-se que os atrasos de controle devem ser menores do que os períodos de amostragem do sensor e do controlador, mas podem ser maiores que a inclinação de tempo, e o meio de comunicação é livre de erros. Nenhum teste de estabilidade geral foi desenvolvido para o sistema variante no tempo. Mas, para atrasos variantes de modo periódico, uma condição necessária e suficiente para estabilidade assintótica foi estabelecida. Liou e Ray (1990) estenderam esse modelo para utilizar diferentes períodos de amostragem do sensor e controlador (sistema com amostragem multitaxa.). O modelo é também apropriado para sistemas com referências de entrada dependentes no tempo.

Luck e Ray (1990) desenvolveram um algoritmo para compensação de atrasos usando o conceito do observador multipasso e testaram a robustez do algoritmo com relação as incertezas do processo. O sistema foi representado por um modelo discreto e invariante no tempo por meio da introdução de *buffers* nos nós controlador e atuador. Todos os nós são sincronizados e operam no modo guiado a tempo. Considerou-se que os atrasos de controle podem ser maiores do que o período de amostragem. Krtolica et al. (1994) determinaram condições necessárias e suficientes para estabilidade exponencial de média quadrática para sistemas lineares e discretos

no tempo. Os atrasos das mensagens sensor-controlador e controlador-atuador foram considerados como aleatórios e modelados por cadeias de Markov, sob a suposição de que o processo e o controlador são invariantes no tempo. Assim como Luck e Ray (1990), todos os nós são guiados a tempo, mas os atrasos de controle devem ser múltiplos do período de amostragem.

Com o objetivo de minimizar os atrasos induzidos pela rede, Kim, Park e Kwon (1996) definiram o MADB (*Maximum Allowable Delay Bound*)[8], limite máximo de atraso permissível para estabilidade de NCSs usando a teoria de Lyapunov. A partir do MADB, desenvolveram um método para determinar os instantes de amostragem do sensor e um método para alocação da largura de banda da rede, e aplicaram esses métodos utilizando as redes FIP e Profibus em um NCS constituído por três motores DC. Tais métodos foram também descritos em Kim, Park e Kwon (1998). O processo e o controlador foram modelados no tempo contínuo. Considerou-se que a rede é sincronizada e livre de erros de transmissão.

Nos primeiros modelos, era comum usar os nós controladores no modo guiado a tempo, onde o período de amostragem do controlador era definido como o tempo de resposta no pior caso da mensagem sensor-controlador. No entanto, essa configuração de operação pode gerar atrasos maiores do que o necessário (NILSSON, 1998), visto que a mensagem enviada pelo nó sensor pode chegar antes desse período. A partir do trabalho de Nilsson (1998), tornou-se comum considerar nos modelos apenas o sensor guiado a tempo, enquanto que o controlador e o atuador são guiados a eventos.

Nilsson (1998) usou técnicas de controle estocástico na análise de estabilidade e síntese do controlador, e desenvolveu dois modelos para os atrasos aleatórios gerados nas redes CAN e Ethernet. No primeiro modelo, assume-se que os atrasos são representados por uma distribuição de probabilidade constante. No segundo modelo, as funções de distribuição de probabilidade são geradas por uma cadeia de Markov. Considerou-se que os atrasos de controle são menores do que o período de amostragem do sensor. Para cada modelo foi derivado um controlador LQG (*Linear Quadratic Gaussian*) ótimo. Então, a estabilidade do sistema em malha fechada e a avaliação de uma função de custo quadrática foram analizadas. Os controladores utilizam o conhecimento dos atrasos antigos por meio da técnica de *timestamping*. Lincoln e Bernhardsson (2000) estenderam o trabalho de Nilsson (1998) para atrasos de controle maiores do que um período de amostragem para um sistema linear com uma única malha de controle. Um controlador ótimo foi desenvolvido sob um critério linear quadrático. Para mostrar as propriedades do controlador ótimo, dois sistemas foram simulados.

Walsh, Ye e Bushnell (1999) e posteriomente, em Walsh e Hong (2001) e Walsh, Ye e Bushnell (2002), definiram o MADB com um novo nome, MATI (*Maximal Allowable Transfer Interval*), intervalo máximo de transferência permissível (τ), no qual um desempenho desejável

[8]Atraso máximo desde o instante em que o sensor adquire os dados da saída do processo até o instante em que o sinal de controle é aplicado na entrada do processo.

de um NCS pode ser garantido. Considerou-se que o processo e o controlador são contínuos no tempo e a rede é apenas inserida entre o nó sensor e o nó controlador. A lei de controle é definida desconsiderando a rede e assumiu-se que o sistema é exponencialmente estável. Um teorema de estabilidade baseado no segundo método de Lyapunov foi derivado para o caso da transmissão de uma única mensagem[9]. Dois lemas são usados para provar o teorema, o lema de Bellman-Gronwall e o segundo é uma extensão no tempo reverso desse lema. Além disso, desenvolveram o protocolo TOD (*Try-Once-Discard*), que utiliza escalonamento dinâmico baseado no sinal de erro do sistema (*Maximum-Error-First*), para indicar qual nó vencerá a disputa pela acesso ao meio. O protocolo TOD foi comparado com a técnica de escalonamento estático por análise e simulação. De modo a caracterizar o comportamento dos algoritmos de escalonamento, foi introduzido o conceito do máximo aumento permissível no erro em τ segundos. Dois lemas, um para cada escalonador, foram derivados. Para comparar os algoritmos de escalonamento foi simulado uma turbina de gás automotiva usando o método de Monte Carlo.

A partir do trabalho de Walsh, Ye e Bushnell (1999), vários trabalhos foram desenvolvidos na tentativa de melhorar o MADB ou MATI utilizando diferentes técnicas e abordagens tais como, técnicas de análise de estabilidade híbrida[10](BRANICKY; PHILLIPS; ZHANG, 2000) e (ZHANG; BRANICKY; PHILLIPS, 2001), teorema do ganho pequeno (NëSIC; TELL, 2004) e Lyapunov (KIM et al., 2003), (PARK et al., 2002), (MA; CHEN; LIU, 2007), (CARNEVALE; TEEL; NëSIC, 2007b) e (CARNEVALE; TEEL; NëSIC, 2007a).

Zhang (2001) considerou também que o sistema de controle (processo e controlador de realimentação de estados contínuos no tempo) é projetado desconsiderando a rede, e é estável ou satisfaz certas especificações de controle. Além disso, considerou-se que o sensor é guiado a tempo, mas a saída do processo pode ser amostrada periodicamente ou aperiodicamente (sensor guiado a eventos) nos instantes de amostragem, e o controlador e o atuador são guiados a eventos. A arquitetura usada foi a representada pela Figura 2.5(b), ou seja, o controlador e o atuador juntos. Inicialmente, foi realizado um estudo da estabilidade de NCSs com período de transmissão variante no tempo, mas sem atraso de transmissão (os dados dos sensores são enviados instantaneamente) e sem perda de mensagens. Então, condições suficientes para o período de transmissão foram derivadas usando uma função de Lyapunov, de modo a garantir estabilidade exponencial. Métodos de pesquisa exaustiva foram usadas para determinar os limites do período de transmissão (MATI). A partir dos limites de período de transmissão, foi derivada uma análise seguindo o algoritmo de escalonamento taxa monotônica para determinar se o NCS é escalonável. O problema do escalonamento ótimo foi formulado minimizando ou maximizando uma função de medida de desempenho. Um exemplo para encontrar tal escala ótima foi desenvolvido e demonstrou-se que os atrasos geram efeitos negativos no desempenho

[9]Todos os sinais de saída em um NCS são transmitidos em uma única mensagem.

[10]Sistemas com dinâmicas contínuas e a eventos discretos.

e estabilidade do sistema de controle. Em seguida, foi realizado um estudo da estabilidade de NCSs considerando os atrasos na rede usando a noção da região de estabilidade e uma técnica de análise de estabilidade para sistemas híbridos. Um esquema para compensar os atrasos na rede baseado em uma solução no domínio do tempo do sistema em malha fechada foi discutida. Experimentos usando uma rede física foram conduzidos para validar o esquema. Finalmente, realizou-se um estudo da estabilidade com perda de mensagens e múltiplas transmissões de mensagens. Tais cenários foram modelados como sistemas dinâmicos assíncronos e as condições de estabilidade para ambos os cenários foram derivadas. O problema de escalonamento da rede foi novamente estudado, e concluiu-se que, para alguns esquemas de controle a escalonabilidade não é garantida se todas as mensagens forem transmitidas. Entretanto, a perda de certa porcentagem das mensagens podem tornar o sistema escalonável sem perder a estabilidade.

Assim como Nilsson (1998), Zhang (2001) considerou-se que os atrasos de controle são limitados pelo período de amostragem do sensor. Em outros trabalhos, ver por exemplo (SHOU-SONG; QIXIN, 2003), (LI; WANG; SUN, 2004), (YANG et al., 2004), (YANG; WANG, 2005), (PHAT, 2005) e (NAGHSHTABRIZI; HESPANHA; TEEL, 2007), essa suposição foi estendida para considerar atrasos maiores que o período de amostragem do sensor. Entretanto, tais modelos podem resultar no fenômeno da rejeição de mensagens, ou seja, a chegada de mais de uma amostra do processo durante um ciclo de operação.

Depois das pesquisas de Zhang (2001), começou-se a considerar o período de amostragem do sensor variante, ou seja, sensor guiado a eventos (MONTESTRUQUE; ANTSAKLIS, 2004), perdas de mensagens, que são modeladas como fenômenos determinísticos (YUE; HAN; PENG, 2004) e (NAGHSHTABRIZI; HESPANHA, 2005)), ou como fenômenos estocásticos (AZIMI-SADJADI, 2003), (YU et al., 2004b), (HU; YAN, 2007), (GARCíA-RIVERA; BARREIRO, 2007) e (XIONG; LAM, 2007), ou pelo menos ambas as considerações (NëSIC; TELL, 2004), (NAGHSHTABRIZI; HESPANHA, 2006), (NAGHSHTABRIZI; HESPANHA; TEEL, 2007) e (HETEL; DAAFOUZ; IUNG, 2008).

Em geral, o NCS é modelado como um sistema linear, variante e discreto no tempo e as perdas de mensagens são modeladas como fenômenos estocásticos. Na maioria dos trabalhos, os processos são constituídos apenas por uma entrada e uma saída. A análise de estabilidade e síntese do controlador são normalmente baseadas nas teorias de Lyapunov ((XIE; WANG, 2004), (GARCíA-RIVERA; BARREIRO, 2007) e (HETEL; DAAFOUZ; IUNG, 2008).), Lyapunov-krasovskii ((YUE; HAN; PENG, 2004), (NAGHSHTABRIZI; HESPANHA, 2005), (YU et al., 2005), (PHAT, 2005), (NAGHSHTABRIZI; HESPANHA, 2006) e (PAN; MARQUEZ; CHEN, 2006).), e suas extensões ((YU et al., 2004a).). É também comum usar condições LMI (*Linear Matrix Inequality*) na síntese do controlador ((SEILER; SENGUPTA, 2001), (JUNG; LEE; SUB, 2004), (SEILER; SENGUPTA, 2005), (GAO et al., 2010) e (CUENCA et al., 2011).).

Seguindo uma abordagem de análise de estabilidade diferente, Kao e Lincoln (2004) desenvolveram teoremas de estabilidade no domínio da frequência para sistemas lineares, contínuos

ou discretos, e atrasos variantes no tempo. A análise de estabilidade é baseada no teorema do ganho pequeno e pode ser aplicada apenas a sistemas SISO (*Single Input Single Output*), e os atrasos podem ser menores ou maiores que o período de amostragem, mas limitados. Apesar dos teoremas serem apenas suficientes, a estabilidade do sistema é garantida para atrasos constantes, periódicos e aleatórios, em um determinado intervalo. Do ponto de vista prático, os resultados de Kao e Lincoln (2004) são mais interessantes, uma vez que é possível usar as técnicas de controle normalmente usadas na prática no projeto do controlador.

Usando um dos teoremas de estabilidade de Kao e Lincoln (2004), Cervin et al. (2004) propuseram os conceitos da margem de *jitter* e margem de fase aparente para malhas de controle. A margem de *jitter* é definida como o maior valor de atraso variável em função de um atraso constante e conhecido, que pode ser tolerado sem que a malha de controle torne-se instável. A partir desses conceitos e da teoria de escalonamento em tempo real, Cervin et al. (2004) desenvolveram um procedimento para projetar sistemas de controle via computador com atrasos variantes, capaz de garantir a estabilidade e o desempenho dos controladores das malhas. O procedimento de projeto consiste em escolher iterativamente o período de amostragem do sensor, de modo que os controladores das malhas tenham uma mesma degradação de desempenho relativa, levando em consideração os atrasos variáveis. O desempenho das malhas é mensurado usando a margem de fase (FRANKLIN; POWELL; EMAMI-NAEINI, 2002) do sistema no tempo contínuo e a margem de fase aparente do sistema discretizado sob atraso variante.

A partir dos resultados de Kao e Lincoln (2004) e Cervin et al. (2004), vários trabalhos foram desenvolvidos no projeto e análise de NCSs e WNCSs (*Wireless Networked Control Systems*), conforme descritos a seguir.

Santos e Vasquez (2005) utilizaram o conceito da margem de *jitter* e a teoria de escalonamento de mensagens (TINDELL; BURNS; WELLINGS, 1995) para projetar um NCS usando uma rede CAN ideal (sem erros de transmissão) seguindo o mesmo procedimento de projeto proposto por Cervin et al. (2004). Um NCS com três malhas de controle foi projetado utilizando a arquitetura da malha de controle 1 (ver Figura 2.5(a)) e simulados no MATLAB/Truetime (CERVIN et al., 2003). De modo semelhante ao trabalho de Santos e Vasquez (2005), Perez, Moreno e Montez (2006) projetaram um NCS via rede CAN com três malhas de controle usando uma arquitetura onde os sensores, controladores e atuadores são separados (Figura 2.5(c)). Em ambos os trabalhos, considerou-se que o sensor é guiado a tempo e o controlador e atuador são guiados a eventos. O processo e o controlador foram modelados no tempo contínuo e, após a definição do período de amostragem usando o procedimento de projeto de Cervin et al. (2004), os controladores de cada malha foram discretizados usando o método de Tustin.

Santos et al. (2007) adaptaram o procedimento de projeto proposto por Perez, Moreno e Montez (2006) e usaram a técnica de alocação de pólos para a síntese dos controladores digitais para um NCS CAN. Em outro trabalho, Santos, Moreno e Montez (2007) usaram o conceito da

margem de *jitter* para estabelecer um limite superior suficiente do período de amostragem do sensor usando a rede CAN, de modo a garantir a estabilidade das malhas de controle.

Sá et al. (2008) desenvolveram um procedimento baseado no conceito da margem de fase aparente (CERVIN et al., 2004) para avaliar a qualidade de controle de NCSs com a rede Ethernet comutada (TANENBAUM, 2003). A partir desse procedimento, foram realizadas simulações e uma análise para verificar os efeitos do compartilhamento da rede, a taxa de transmissão adequada e o número de nós da rede.

Vários trabalhos foram desenvolvidos usando controladores PIDs e suas variações em conjunto com o conceito da margem de *jitter*. Essas pesquisas foram iniciadas a partir do trabalho de Eriksson e Oksanen (2007), que propuseram uma nova abordagem de sintonia para controladores PID e processos FOLIPD (*First Order Lag plus Integrator plus Delay*) no tempo contínuo considerando atrasos variantes. A abordagem usa otimização multi-objetivo para determinar os parâmetros ótimo do controlador PID, onde o critério de custo ITAE (*Integral of Time-Weighted Absolute Error*) e a margem de *jitter* foram utilizadas como funções objetivas. A robustez com relação a distúrbios foi considerada introduzindo uma restrição que mantém a curva de Nyquist fora do círculo de robustez, semelhante as regras AMIGO (*Approximate M-constrained Integral Gain Optimization*) (ASTROM; HAGGLUND, 2004) e (ASTROM; HAGGLUND, 2006), com parâmetro $M = 1.5$. A partir dessa abordagem novas regras de sintonia foram derivadas e testadas no sistema de controle hidráulico de um trator agrícola. As novas regras de sintonia foram comparadas com outras regras por simulação e na prática, e os resultados obtidos foram melhores em vários aspectos. Usando a mesma abordagem de Eriksson e Oksanen (2007), Eriksson e Johansson (2007a) derivaram regras de sintonia para controladores PIDs contínuos no tempo e processos sem integradores do tipo FOTD (*First Order System with Time-Delay*). Eriksson, Oksanen e Mikkola (2009) derivaram regras de sintonia para um controlador PID e um processo FOLIPD, ambos no tempo contínuo usando a margem de *jitter*. Essas regras foram estendidas por meio de um lema desenvolvido por Mirkin (2007) e aplicadas em um WNCS real constituído por um motor DC e controlador PID discreto capaz de tolerar perdas de mensagens.

Em Eriksson e Koivo (2008), a margem de *jitter* foi usada como métrica de robustez para comparar por simulação controladores (PID, IMC (*Internal Model Control*) e fuzzy) de baixa complexidade em WNCSs (*Wireless Networked Control Systems*) com atrasos variantes no tempo.

A partir da mesma abordagem usada em Eriksson e Johansson (2007a), Bhambhani, Chen e Xue (2008) introduziram um novo projeto de controlador robusto aos atrasos variantes pela otimização dos parâmetros de ganho do controlador FOPI (*Fractional Order Proportional Integral*) baseado no algoritmo F-MIGO (*Fractional M_s constrained Integral Gain Optimization*) para processos FOPDT (*First Order Plus Time Delay*). O processo e o controlador foram modelados no tempo contínuo.

Um controlador PID com margem de *jitter* adaptativa que se adapta as variações dos atrasos ou a perdas de mensagens foi desenvolvido por Pohjola (2008). Assumiu-se que o sensor, controlador e atuador são todos guiados a tempo, e o controlador e o atuador estão juntos em um único nó. O processo e o controlador foram modelados no tempo contínuo. A sintonia é baseada na estimação da variação do atraso causada pela rede durante a execução do sistema. Dois algoritmos para estimação da variação dos atrasos foram desenvolvidos. No primeiro algoritmo, assume-se que ocorre apenas perdas de mensagens e as mensagens contém *timestamping*. O segundo algoritmo é aplicável para qualquer atraso de rede com atrasos desconhecidos. O controlador foi testado por simulação em um WNCS com a rede IEEE 802.15.4 usando a ferramenta PiccSIM, e apresentou melhor desempenho do que um controlador de ganho constante, sintonizado para a máxima variação de atraso.

Björkbom e Johansson (2010) propuseram um controlador PID em rede e um esquema de sintonia baseado no controle IMC, usando a margem de *jitter* para garantir estabilidade na ocorrência de perda de mensagens. Uma heurística de estado permanente foi proposta para longos períodos (maiores do que o limite da margem de *jitter*) de desconexão da rede. Nesse caso, o controlador é substituído pelo controlador heurístico para garantir a estabilidade do sistema. O controlador PID em rede é dividido em duas partes. Primeiramente, um sensor inteligente capaz de realizar alguns cálculos, determina o erro de controle, a integral do erro e a derivada do erro, e transmite esses três termos para o controlador onde o sinal de controle é calculado. Esses três estados do controlador PID podem ser atualizados de modo similar usando um estimador de estados mesmo se mensagens forem perdidas. Se nenhum dado for recebido do sensor, os valores anteriores podem ser mantidos no atuador ou a heurística de interrupção proposta pode ser usada. O efeito das desconexões da rede na estabilidade e desempenho do sistema de controle em malha fechada foram avaliadas e comparadas com esquemas alternativos via simulações. O controlador PID em rede com a heurística de estado permanente obteve melhor desempenho quando comparado com outras técnicas propostas na literatura de NCSs. Considerou-se um processo estável com modelo aproximado de primeira ordem com atraso de tempo e uma arquitetura de rede sem fio onde o controlador e atuador estão juntos e o sensor separado.

Piltz et al. (2010) desenvolveram controladores adaptativos (IMC-PID e PID) para processos MIMO (*Multiple Input, Multiple Output*) em um WNCSs usando a margem de *jitter*. Uma estrutura de controle descentralizada foi usada para que o sistema de controle MIMO fosse decomposto em vários sistemas SISO independentes de modo a usar a teoria da margem de *jitter*. Os parâmetros dos controladores adaptativos foram otimizados por meio dos critérios ITSE (*Integral of Time-Weighted Square Error*) e ISE (*Integral of Square Error*). O controlador foi comparado com outros métodos de controle MIMO descentralizado usando o simulador PiccSIM e o padrão 802.15.4 para rede sem fio, e obteve-se um melhor desempenho que os

outros métodos. Considerou-se uma arquitetura onde o controlador e atuador de cada processo SISO estão juntos e o sensor separado.

Ungan (2010) estudou a aplicação de controladores PID em WNCSs. Dois controladores propostos na literatura, PIDplus (SONG et al., 2006) e o *Jitter Aware* AMIGO (ERIKSSON; JOHANSSON, 2007b), foram analisados e verificou-se que sob condições não confiáveis da rede o desempenho é insatisfatório. Então, para contornar esse problema, um controlador PID baseado em um filtro de Kalman, que requer um simples modelo do processo, foi desenvolvido. Diversas simulações foram realizadas em vários tipos de processos e a robustez do sistema contra perdas de mensagens e incertezas do processo foram testadas obtendo-se resultados satisfatórios sob várias condições da rede. Finalmente, o controlador foi testado na prática em um sistema de controle de um tanque de água usando módulos sem fio Tmoty Sky. A malha de controle possui uma arquitetura onde o controlador e o atuador estão juntos, e o sensor separado.

Em outros trabalhos, controladores PIDs modificados foram desenvolvidos para NCSs com atrasos variantes no tempo usando outras técnicas. Um novo controlador de escalonamento de ganho e ordem fracionária (FGOSC) baseado no controlador PI de ordem fracionária (FOPI) foi desenvolvido por Tejado, Vinagre e Chen (2010). Já Pan, Das e Gupta (2011) desenvolveram uma sintonia ótima para controladores $PI^{\lambda}D^{\mu}$ de ordem fracionária com algoritmos evolucionários para NCSs considerando perdas de mensagens e atrasos aleatórios para processos instáveis. Enquanto que Tran et al. (2013) desenvolveram um controlador PID normalizado usando a abordagem das margens de ganho e fase.

Apesar da quantidade relevante de resultados que usam técnicas de controle mais simples e comuns, a maioria desses trabalhos tem desconsiderado a interdisciplinaridade dos NCSs e a realidade das aplicações. Em geral, os projetos e análises foram realizadas em NCSs com apenas uma malha de controle, arquiteturas mais simples e sob condições ideais, que impossibilitam a verificação de outros problemas gerados pela rede. Além disso, os poucos trabalhos que consideraram a determinação do comportamento temporal, assumiram condições ideais na rede.

2.2.2 Análise do comportamento temporal

A análise do comportamento temporal de um NCS consiste em determinar os tempos de resposta no melhor e pior caso de cada tarefa e mensagem, e consequentemente, os atrasos de controle ou tempos de resposta fim a fim no melhor e pior caso da transação de cada malha de controle. Na análise deve-se considerar as várias condições de projeto e operação do sistema tais como, arquitetura das malhas de controle, modelo das tarefas e mensagens, tipo de escalonador do sistema operacional em tempo real, características da rede, modelo de erros no meio de comunicação entre outros.

Na literatura, existem várias abordagens usadas para determinar os tempos de resposta fim a fim das transações em sistemas semelhantes aos NCSs. As mais usadas são as abordagens

holística, trajetória, *network calculus*, composicional, simulação e verificação de modelos. As quatro primeiras abordagens podem gerar tempos de resposta fim a fim pessimistas, ou seja, maiores do que ocorrem na prática, principalmente nas arquiteturas representadas nas Figuras 2.5 (c) e (d). Vários trabalhos foram desenvolvidos em cada abordagem na tentativa de identificar e reduzir o pessimismo. A seguir, definiremos e apresentaremos uma breve revisão dessas abordagens.

A abordagem holística, desenvolvida por (TINDELL; CLARK, 1994), é baseada na teoria clássica de escalonamento em tempo real. Essa abordagem consiste na integração entre a análise dos tempos de resposta das tarefas em um processador (JOSEPH; PANDYA, 1986), (AUDSLEY et al., 1993) e (TINDELL; BURNS; WELLINGS, 1994) e a análise dos tempos de resposta das mensagens em uma rede de comunicação. Os tempos de resposta fim a fim no pior caso são calculados considerando os tempos de resposta das tarefas e mensagens, ambas no cenário de pior caso, respeitando as relações de precedência entre as tarefas e mensagens do sistema. A ideia básica na integração das análises consiste em considerar o *jitter* de liberação das tarefas e das mensagens. Assume-se que todas as tarefas e mensagens são ativadas no mesmo instante, mas serão liberadas respeitando as relações de precedência. Assim, o *jitter* de uma tarefa ou mensagem com relação de precedência é igual ao tempo de resposta no pior caso da sua tarefa ou mensagem predecessora.

Em seu trabalho, Tindell e Clark (1994) consideraram que as tarefas são periódicas, com prioridades fixas, *deadlines* arbitrários (*deadlines* maiores do que o período da tarefa) e se comunicam por meio de mensagens usando um simples protocolo de rede TDMA. As tarefas são escalonadas por um escalonador preemptivo e os relógios dos nós da rede são sincronizados. A análise foi usada em um sistema hipotético de controle de aeronave em uma rede com três nós, que executam 32 tarefas e transmitem 14 mensagens. Não foi considerado erros de transmissão nas mensagens. Spuri (1996) estendeu a análise holística de Tindell e Clark (1994) para o escalonamento local (em cada nó) de tarefas e mensagens usando a política de atribuição de prioridades EDF e o protocolo de controle de acesso ao meio *timed token*.

Sun e Liu (1996) demonstraram que a análise de Tindell e Clark (1994) é incompleta, pois não verifica os tempos de resposta de todos os eventos que poderão ocorrer no sistema durante o intervalo de tempo, chamado de período ocupado, que deve ser considerado para a análise do pior caso. A partir desse resultado, Sun e Liu (1996) proporam uma nova análise que verifica todos os eventos que poderão ocorrer durante o período ocupado. Contradizendo o trabalho de Sun e Liu (1996), Gutiérrez, Garcia e Harbour (1997) provaram que a análise de Tindell e Clark (1994) é válida e estenderam essa análise para calcular um limite superior dos tempos de resposta local das tarefas e mensagens individualmente, de modo a verificar se os *deadlines* locais das tarefas e mensagens serão ou não satisfeitas. Foi considerado que as mensagens são escalonadas de acordo com a teoria de escalonamento preemptivo com prioridade fixa para

tarefas em um processador.

Bate e Burns (1998) afirmaram que uma abordagem exata para a análise dos tempos de resposta fim a fim pode ser intratável e que a abordagem holística é pessimista. A principal causa do pessimismo da abordagem holística é a consideração de cenários de pior caso que são impossíveis de ocorrerem na prática. Por meio de simulações extensivas usando *offsets* no modelo do sistema, uma nova abordagem para análise de sistemas distribuídos foi desenvolvida, a qual reduz o pessimismo em cerca de 20% com relação a abordagem holística.

Gutiérrez, Garcia e Harbour (1998) melhoraram a análise de escalonamento holística reduzindo o seu pessimismo. O melhoramento na análise consistiu em calcular um limite inferior para o tempo de resposta no melhor caso das tarefas e mensagens, de modo a reduzir o *jitter* de liberação das subsequentes tarefas e mensagens. Essa redução no *jitter* possibilitou reduzir os limites no pior caso dos tempos de resposta, permitindo um aumento da escalonabilidade do sistema. Ao contrário da suposição adotada em seu trabalho anterior (GUTIéRREZ; GARCIA; HARBOUR, 1997), Gutiérrez, Garcia e Harbour (1998) assumiram que as mensagens são escalonadas conforme a teoria de escalonamento não-preemptivo com prioridade fixa, uma vez que as mensagens são indivisíveis e não preemptivas. A análise holística foi também estendida e melhorada por Palencia e Harbour (1998) para considerar *offsets* (estáticos ou dinâmicos) maiores que os períodos das tarefas.

De modo semelhante ao trabalho de Gutiérrez, Garcia e Harbour (1998), Henderson, Kendall e Robson (2001) reduziram o pessimismo da análise holística integrando a análise de tempos de resposta no melhor caso das tarefas e mensagens. Considerou-se que o escalonamento das tarefas é preemptivo com prioridade fixa e que as mensagens são transmitidas de acordo com o protocolo da rede CAN. Nesse trabalho, foi demonstrado que os tempos de resposta no melhor caso para o escalonamento preemptivo de tarefas com prioridade fixa, determinado por Gutiérrez, Garcia e Harbour (1998), são otimistas. Então, foi proposto uma nova análise dos tempos de resposta no melhor caso das tarefas. Entretanto, segundo Redell e Sanfridson (2002), essa análise conduz a um algoritmo numericamente intratável. Em (POP; ELES; PENG, 2002), a análise holística foi usada em sistemas distribuídos implementados com tarefas guiadas a tempo e a eventos, que se comunicam por meio de redes *time* e *event-triggered*.

A **abordagem da trajetória** (MARTIN; MINET, 2005), também baseada na teoria clássica de escalonamento em tempo real, consiste em verificar o escalonamento produzido por todos os nós visitados em uma transação considerando apenas cenários possíveis. Martin, Minet e George (2005) utilizaram essa abordagem para determinar tempos de resposta fim a fim determinísticos em transações escalonadas de acordo com a teoria de escalonamento não preemptivo com prioridade fixa. Uma comparação com a abordagem holística de (TINDELL; CLARK, 1994) foi realizada e os resultados foram menos pessimistas. Essa abordagem tem sido utilizada para determinar os tempos de resposta fim a fim em redes AFDX (*Avionics Full Duplex Switched*

Ethernet) (BAUER; SCHARBARG; FRABOUL, 2009), (BAUER; SCHARBARG; FRABOUL, 2010a) e (HU; LV; HUANG, 2011). Medlej, Martin e Cottin (2012) identificaram as fontes de pessimismo da abordagem da trajetória na rede ethernet chaveada com escalonamento FIFO, que podem afetar na precisão dos limites superiores dos tempos de resposta fim a fim das transações.

A **abordagem *network calculus*** (CRUZ, 1991a), (CRUZ, 1991b) e (BOUDEC; THIRAN, 2012) consiste de uma teoria matemática baseada na álgebra Min-Plus usada em problemas de fluxos encontrados nos sistemas em rede. Essa abordagem tem sido usada para determinar os limites superiores dos tempos de resposta fim a fim. Entretanto, essa abordagem, assim como a holística, pode conduzir a cenários impossíveis e consequentemente a resultados pessimistas. Bauer, Scharbarg e Fraboul (2010b) usaram *network calculus* para calcular os limites superiores dos tempos de resposta fim a fim em redes AFDX e realizou uma comparação com a abordagem da trajetória. Demonstrou-se que a abordagem da trajetória supera a abordagem *network calculus* na maioria dos casos, e sugeriu-se combinar as duas abordagens para obter melhores resultados.

A **abordagem composicional** (RITCHER et al., 2002) consiste em dividir a complexidade da análise do sistema em análises separadas dos componentes individuais e em seguida acoplá-las por meio de fluxos de eventos (*event stream*). Os fluxos de eventos de saída de um componente torna-se o fluxo de eventos de entrada do componente conectado. A análise pode então ser executada iterativamente, alternando a análise de escalonamento local e a propagação do fluxo de eventos entre os componentes. Diferentes representações de fluxos de eventos podem ser usadas para descrever as suas temporizações. Por exemplo, Gresser (1993) usou um sistema de vetor de eventos superposicional, que é propagado usando matrizes de dependência de eventos complexos. Thiele, Chakraborty e Naedele (2000) usaram uma extensão da teoria *network calculus*, chamada de *real-time calculus*. Basicamente, foi usado curvas de chegadas para o fluxo de eventos e curvas de serviços similares para execução da modelagem. Enquanto que Richter (2004) usou modelos de eventos padrões parametrizados baseados na teoria clássica de sistemas em tempo real. Assim como as outras abordagens, as análises usando essa abordagem podem ser pessimistas. A principal razão é uma consideração incompleta das correlações das sequências de eventos. Henia, Racu e Ernst (2007) apresentaram uma técnica que melhora o cálculo do *jitter* de saída por meio da correlação do *jitter* e dos tempos de resposta, de modo a reduzir o pessimismo. Em outro trabalho, Rox e Ernst (2013) verificaram a influência das pilhas de comunicação usando os operadores de composição e decomposição e as correlações entre o instante de ativação das diferentes tarefas para reduzir o pessimismo da abordagem.

Na **abordagem simulação**, uma simulação é executada durante um intervalo de tempo chamado de hiperperíodo, o qual equivale ao MMC (Mínimo Múltiplo Comum) de todos os períodos das transações (AUDSLEY, 1991). Então, os tempos de resposta exatos no melhor caso e no pior caso equivalem, respectivamente, ao menor e maior tempo de resposta das instâncias das transações liberadas durante esse período. Apesar de determinar os tempos de resposta exatos,

esse método torna-se complexo quando há *offsets* dinâmicos e *jitter* de liberação nas transações. Nesse caso, um número desconhecido de simulações da escala deve ser realizado considerando os diferentes *offsets* e os cenários de pior caso para cada instância da transação individualmente (REDELL, 2003). Essa abordagem é exata quando os *offsets* são estáticos e quando não existe *jitter* de liberação nas transações.

A **verificação de modelos clássica** (CMC - *Classical Model Checking*) é uma abordagem usada para verificar propriedades de sistemas de modo automático por meio de ferramentas computacionais (LARSEN; PETTERSSON; YI, 1995). Nessa abordagem, um modelo de estados finito, representando o comportamento do sistema, deve ser desenvolvido usando algum método formal (e.g. autômatos temporizados (ALUR; DILL, 1994), redes de Petri temporizadas (MERLIN, 1974) e (BERTHOMIEU; DIAZ, 1991) entre outras.) e as propriedades a serem verificadas devem ser descritas em uma linguagem de especificação no formalismo da ferramenta. Então, a ferramenta executa algoritmos baseados na pesquisa exaustiva do espaço de estado do modelo e verifica se as propriedades são ou não satisfetas. Uma característica importante dessa abordagem é a capacidade de fornecer um contra-exemplo representando a situação em que a propriedade não foi satisfeita, possibilitando o uso dessa informação para corrigir erros no sistema. A verificação de modelos tem sido implementadas em várias ferramentas tais como, SPIN (HOLZMANN, 1997), Roméo (GARDEY et al., 2005), UPPAAL (LARSEN; PETTERSSON; YI, 1997) entre outras.

Essa abordagem vem sendo usada com sucesso para determinar os tempos de resposta das tarefas em um processador (FERSMAN et al., 2006), (WASZNIOWSKY; HANZALEK, 2008) e (LIME; ROUX, 2009), e das mensagens em sistemas de comunicação (KRAKORA, 2008) e (WASZNIOWSKY; KRAKORA; HANZALEK, 2009). Na determinação de tempos de resposta fim a fim, poucos resultados são encontrados na literatura. Mohalik et al. (2008) usou essa abordagem para analisar os tempos de resposta fim a fim de transações constituídas apenas de tarefas distribuídas, que se comunicam assincronamente por meio de *buffers*, considerando as variações nos relógios dos nós computacionais. Rajeev et al. (2010) proporam uma técnica baseada na verificação de modelos para calcular tempos de resposta no melhor e pior caso e latências fim a fim em sistemas de controle distribuído, onde as tarefas são preemptivas e as mensagens não-preemptivas. O sistema foi modelado usando *calendar automata* na ferramenta SPIN. Recentemente, Mohalik et al. (2013) desenvolveram uma técnica baseada na verificação de modelos para aumentar a escalabidade das aplicações devido ao problema da explosão do espaço de estados. Essa técnica consiste de um modelo abstrato do sistema em *calendar automata* baseado nas equações de tempo de resposta de Joseph e Pandya (1986) para determinar as latências fim a fim.

O problema dessa abordagem é a possibilidade de ocorrer uma explosão do espaço de estados. Esse problema restringe o tamanho e a complexidade do comportamento temporal dos mo-

delos. Em aplicações complexas, é possível que as propriedades a serem verificadas sejam não decidíveis[11]. Apesar dessas restrições, a análise exaustiva do modelo fornece resultados exatos em uma quantidade considerável de aplicações, ao contrário das outras abordagens. Além disso, é possível também verificar propriedades funcionais e caracterizar comportamentos internos do sistema tais como, desvios dentro do código dependentes de dados e *loops*, ao contrário das outras técnicas.

Uma solução para os problemas da verificação de modelos clássica que vem sendo usada recentemente é a **abordagem da verificação de modelos estatísticos** (SMC - *Statistical Model-Checking*) (SEN; VISWANATHAN; AGHA, 2004) e (YOUNES, 2005a). Essa abordagem consiste em monitorar um número de execuções de um modelo com semântica estocástica e, a partir dessas execuções, usar algoritmos estatísticos (e.g. testes de hipótese sequencial e simulação de Monte Carlo) para verificar e estimar o comportamento do sistema com algum nível de confidência. Ao contrário das técnicas que usam uma abordagem exaustiva do espaço de estado, SMC não garante um resultado com 100% de confidência. No entanto, é possível limitar a probabilidade de causar um erro. SMC tem sido implementada com sucesso em algumas ferramentas (e.g. Vesta (SEN; VISWANATHAN; AGHA, 2005), Ymer (YOUNES, 2005b) e UPPAAL SMC (BULYCHEV et al., 2012b) e (DAVID et al., 2014).) e usada em aplicações de alta complexidade que, com as técnicas clássicas de verificação, conduziriam a uma explosão do espaço de estado (DAVID et al., 2011b).

Dentre essas abordagens, apenas a holística (ver os trabalhos de Henderson, Kendall e Robson (2001), Santos e Vasquez (2005), Perez, Moreno e Montez (2006) e Santos (2008)) e a simulação (ver o trabalho de Santos (2008)) foram utilizadas para determinar os tempos de resposta fim a fim em NCSs. Em ambos os trabalhos, considerou-se uma rede CAN onde as mensagens são livres de erros.

2.3 Conclusões

Neste Capítulo apresentou-se uma breve introdução sobre os sistemas de controle em malha fechada, no qual foi descrito a evolução, o funcionamento e as relações entre esses sistemas, do ponto de vista do projeto. Em seguida, a teoria básica sobre sistemas de controle via rede foi apresentada, incluindo uma notação e os termos comumente usados, as principais arquiteturas e os problemas nos NCSs. Por fim, apresentou-se uma revisão indicando as abordagens, técnicas e suposições adotadas no projeto de NCSs e as abordagens usadas para determinar os tempos de resposta fim a fim em NCSs. Nessa revisão foram realizados comentários sobre os principais problemas, as suposições adotadas ao longo do tempo e as tendências atuais.

[11] Os algoritmos executados pela ferramenta não conseguem decidir em um tempo finito se a propriedade é satisfeita ou não.

Capítulo 3

Fundamentação teórica

O desenvolvimento de Sistemas de Controle via Rede (NCSs) envolve o estudo e integração das teorias de sistemas de controle, sistemas em tempo real, redes de comunicação, além de outras teorias relacionadas. Neste Capítulo, apresentam-se inicialmente as tecnologias e suposições adotadas, com o objetivo de delimitar o escopo teórico deste trabalho. Em seguida, as definições e conceitos básicos dessas teorias serão apresentadas.

3.1 Tecnologias e suposições adotadas no trabalho

Dentre as diversas redes e estratégias de controle disponíveis no mercado e na literatura, escolheu-se a rede CAN e o controlador PID, devido a grande aceitação dessas tecnologias nas aplicações. CAN é uma rede de comunicação que foi originalmente projetada para atender as necessidades das aplicações automotivas. Por causa das suas características, da disponibilidade de *hardware* e *software* no mercado[1] e ao seu sucesso na indústria automotiva[2], CAN vem sendo adotada como rede padrão em outras áreas de aplicação. De modo semelhante, os controladores PIDs têm predominado na indústria devido a sua simplicidade e efetividade. Em uma pesquisa, observou-se que em mais de 11000 malhas de controle analisadas em diversos processos na indústria, cerca de 97% eram controladas com PID (ASTROM; HAGGLUND, 1995). Além disso, não existe vantagem significante em usar controladores mais complexos em processos com dinâmicas dominantes de primeira e segunda ordem, os quais são predominantes na maioria das aplicações (ASTROM; HAGGLUND, 1995).

Do ponto de vista da teoria de controle, serão adotadas as seguintes teorias e suposições:

[1] Em 2004, existia pelo menos 15 fabricantes totalizando cerca de 50 famílias diferentes de microcontroladores com rede CAN integrada (DAVIS et al., 2007).

[2] Atualmente, é possível encontrar em carros modernos cerca de 70 nós CAN realizando diversas atividades. Nos Estados Unidos, a agência de proteção ambiental exigiu o uso do CAN, para diagnóstico *on-board*, em todos os carros e caminhões vendidos no país a partir do modelo/ano 2008 (DAVIS et al., 2007).

- Processos lineares e invariantes no tempo com uma única entrada e uma única saída;
- Sensor guiado a tempo, controlador e atuador guiados a eventos;
- Atrasos de controle variantes no tempo e menores que o período de amostragem do sensor;
- Técnicas de controle PID e análise de estabilidade por meio dos teoremas de Kao e Lincoln (2004) e do conceito da margem de *jitter* (CERVIN et al., 2004).

Na determinação do comportamento temporal, considera-se:

- Erros de transmissão nas mensagens devido a injeção de falhas no barramento da rede CAN;
- Modelagem formal usando autômatos temporizados e autômatos temporizados com custo, e as técnicas de verificação de modelos clássica e estatística.

3.2 Sistemas de controle

Para projetar sistemas de controle é fundamental ter um modelo matemático que descreve as dinâmicas do sistema. Na teoria de controle, as dinâmicas dos sistemas são descritas por equações diferenciais que podem ser representadas por modelos em função de transferência e/ou por modelos em espaço de estado. Modelos em função de transferência são usados para representar as relações de entrada-saída de sistemas lineares e invariantes no tempo com uma única entrada e uma única saída. Nessa representação, as equações diferenciais são substituídas por equações algébricas em termos da variável complexa s para sistemas contínuos no tempo ou z para sistemas discretos no tempo. A análise do sistema é realizada no domínio da frequência de acordo com as técnicas de controle clássico. Modelos em espaço de estado podem representar sistemas lineares ou não lineares, variantes ou invariantes no tempo, com uma única ou várias entradas e saídas. Nesse caso, o comportamento dinâmico do sistema é representado por variáveis de estado. A análise do sistema é realizada no domínio do tempo, de acordo com a teoria de controle moderna. Neste trabalho, os sistemas serão representados por modelos em função de transferência.

3.2.1 Modelos em função de transferência

A função de transferência de um sistema linear, invariante e contínuo no tempo (Equação 3.1) pode ser definida como a relação entre a transformada de Laplace do sinal de saída ($y(t)$) e a transformada de Laplace do sinal de entrada ($u(t)$), assumindo todas as condições iniciais nulas. Sendo s a variável complexa e b_0,..., b_m e a_0,..., a_n constantes.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^m + b_1 s^{m-1} + ... + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + ... + a_{n-1} s + a_n} \tag{3.1}$$

No tempo discreto, a função de transferência $G(z)$ de um sistema linear e invariante no tempo com n valores passados da saída y e m valores passados da entrada u, pode ser representado pela Equação 3.2.

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 z^n + b_1 z^{n-1} + ... + b_m z^{n-m}}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + ... + a_n} \tag{3.2}$$

Em ambas as representações ($G(s)$ e $G(z)$), as raízes do numerador e do denominador são, respectivamente, chamadas de zeros e pólos do sistema.

3.2.2 Projeto e análise do sistema de controle

O objetivo do projeto do sistema de controle é garantir determinadas especificações de desempenho. Essas especificações são normalmente relacionadas a estabilidade e a resposta do sistema.

A estabilidade é a primeira especificação a ser considerada no projeto e análise de sistemas de controle. É dito que um sistema linear e invariante no tempo é estável, se a sua saída retornar ao estado de equilíbrio[3] quando o sistema for sujeito a uma condição inicial. Se a saída do sistema não convergir e nem tornar-se ilimitada, é dito que o sistema é marginalmente estável. Quando a saída do sistema diverge ilimitadamente a partir do estado de equilíbrio, sob uma condição inicial, é dito que o sistema é instável.

Na análise de sistemas lineares e invariantes no tempo, a estabilidade pode ser dividida em estabilidade absoluta ou relativa. Na estabilidade absoluta, o objetivo é determinar se o sistema é ou não estável. Enquanto que na estabilidade relativa o objetivo é estabelecer margens dentro das quais o sistema permanece estável. No tempo contínuo, um sistema linear invariante no tempo é estável quando os pólos de malha fechada estão localizados no semi-plano esquerdo do plano s. No tempo discreto, um sistema linear invariante no tempo é estável quando os pólos estão localizados dentro do círculo únitário no plano z.

A resposta de um sistema em malha fechada é o comportamento do seu sinal de saída devido a aplicação de um sinal de entrada, e consiste de uma resposta transitória e uma resposta permanente. A resposta transitória depende dos pólos de malha fechada e ocorre devido ao sistema não reagir instantaneamente a um sinal de entrada ou a perturbações. A resposta permanente refere-se a capacidade do sistema de controle seguir um sinal de referência com erro mínimo.

Em muitos casos práticos, o desempenho do sistema é especificado em termos da resposta transitória a uma entrada em degrau unitário. Por exemplo, considere o sistema de segunda or-

[3] O estado de equilíbrio de um sistema é o estado no qual o sistema permanece na ausência de um sinal de entrada ou perturbação.

dem, representado pela Equação 3.3, onde ζ é o coeficiente de amortecimento e ω_n é a frequência natural não amortecida. Os parâmetros de desempenho para esse sistema são normalmente especificados como tempo de atraso (t_d), tempo de subida (t_r), instante de pico (t_p), *overshoot* máximo (M_p) e tempo de acomodação (t_s) (OGATA, 2000).

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.3}$$

3.2.3 Aproximação de sistemas contínuos no tempo

Em geral, os sistemas de controle são primeiro representados no tempo contínuo. Para uma implementação em computador, é necessário obter uma representação do sistema no tempo discreto. Uma abordagem direta é usar um intervalo de amostragem h e realizar algumas aproximações no tempo discreto do sistema no tempo contínuo.

As dinâmicas de um sistema linear, invariante e contínuo no tempo são descritas por equações diferenciais, que podem ser representadas por funções de transferência. Para obter um versão discreta no tempo desse sistema, ou seja, uma equação a diferenças, aproxima-se as derivadas da equação diferencial por uma diferença avançada (Equação 3.4) ou uma diferença atrasada (Equação 3.5).

$$\frac{dx(t)}{dt} \approx \frac{x(t+h) - x(t)}{h} \tag{3.4}$$

$$\frac{dx(t)}{dt} \approx \frac{x(t) - x(t-h)}{h} \tag{3.5}$$

Essas aproximações correspondem a substituir a variável s por $(z-1)/h$ ou $(z-1)/zh$. A relação entre as variáveis s e z é dada por $z = e^{sh}$. Assim, as aproximações correspondem, respectivamente, as seguintes expansões em série.

$$z = e^{sh} \approx 1 + sh \tag{3.6}$$

$$z = e^{sh} \approx \frac{1}{1 - sh} \tag{3.7}$$

Uma outra aproximação, chamada de aproximação de Tustin, é definida pela Equação 3.8.

$$z = e^{sh} \approx \frac{1 + sh/2}{1 - sh/2} \tag{3.8}$$

A partir desses métodos de aproximação, a função de transferência pulso $H(z)$ ((ASTROM; WITTENMARK, 1997, pp. 57-58)) pode ser obtida substituindo s de $G(s)$, por s', ou seja, $H(z) = G(s')$, onde:

$$s' = \frac{z-1}{h} \tag{3.9}$$

$$s' = \frac{z-1}{zh} \tag{3.10}$$

$$s' = \frac{2}{h}\frac{z-1}{z+1} \tag{3.11}$$

De acordo com o método de aproximação, a região de estabilidade do sistema no plano s será mapeada para uma nova região de estabilidade no plano z. Com a aproximação diferença avançada, é possível que um sistema contínuo no tempo estável seja mapeado em um sistema discreto no tempo instável. Quando a aproximação diferença atrasada é usada, um sistema contínuo no tempo estável sempre resultará em um sistema discreto no tempo estável. Existem também sistemas contínuos no tempo instáveis que serão transformados em sistemas discretos no tempo estáveis. A aproximação de Tustin possui a vantagem de o semiplano s da esquerda ser transformado em um círculo unitário. Sistemas contínuos no tempo estáveis serão, portanto, transformados em sistemas discretos no tempo estáveis, e sistemas contínuos no tempo instáveis serão transformados em sistemas discretos no tempos instáveis.

3.2.4 Controlador PID

O controlador PID é o controlador mais usado em sistemas de controle em malha fechada. O sinal de controle do PID é formado pela combinação das ações proporcional, integral e derivativa, que atuam sobre o erro de referência. Essas ações individuais ou combinadas em dupla são também controladores. Na prática, os controladores mais utilizadas são o PI e PID. Existem muitas variações desses dois controladores, especificamente do controlador PID. Uma quantidade relevante dessas variações é apresentada em O'Dwyer (2006).

A versão clássica do controlador PID no tempo contínuo possui um sinal de controle $u(t)$ descrito pela Equação 3.12, sendo: $e(t)$ é o sinal de erro definido por $e(t) = r(t) - y(t)$; $r(t)$ é o sinal de referência; $y(t)$ é o sinal de saída do processo; K é o ganho proporcional; T_{in} é o tempo de ação integral e T_d é o tempo de ação derivativa.

$$u(t) = K\left(e(t) + \frac{1}{T_{in}}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{3.12}$$

O algoritmo PID clássico definido pela Equação 3.12 pode ser representado pela função de transferência (Equação 3.13), onde $U(s)$ e $E(s)$ são, respectivamente, a transformada de Laplace de $u(t)$ e $e(t)$.

$$C(s) = \frac{U(s)}{E(s)} = K\left(1 + \frac{1}{sT_{in}} + sT_d\right) \tag{3.13}$$

Uma outra representação do algoritmo PID clássico em função de transferência é definida pela Equação 3.14, onde $k_p = K$, $k_i = \frac{K}{T_{in}}$ e $k_d = KT_d$.

$$C(s) = k_p + \frac{k_i}{s} + sk_d \tag{3.14}$$

Para sinais de alta frequência, a ação derivativa possui a desvantagem de tornar o ganho muito alto. Isso significa que ruídos de medição em alta frequência gerarão grandes variações do sinal de controle. O efeito do ruído de medição pode até certo ponto ser eliminado implementando na ação derivativa um filtro de primeira ordem com uma constante de tempo T_d/N, onde N possui valores típicos de 3 a 20 (ASTROM; WITTENMARK, 1997). A função de transferência para o controlador PID com esse filtro de primeira ordem é representada pela Equação 3.15.

$$C(s) = K\left(1 + \frac{1}{sT_{in}} + \frac{sT_d}{1 + sT_d/N}\right) \tag{3.15}$$

Para implementar um controlador PID em um computador, é necessário aproximar as derivadas e a integral que aparecem na lei de controle usando, por exemplo, algum dos métodos descritos na seção 3.2.3.

3.2.5 Conceitos básicos da teoria de controle clássico

Na teoria de controle clássica, os conceitos das margens de atraso, ganho e fase são usados para descrever a sensibilidade de uma malha de controle com relação às várias incertezas do processo. Além disso, as margens são usadas como medidas práticas de estabilidade e existem várias regras de *thumb* associadas a elas. Por exemplo, é recomendado ter uma margem de fase entre 30° e 45° para garantir algum nível de robustez e desempenho do sistema. A seguir, serão apresentados alguns conceitos básicos da teoria de controle clássica.

Definição 1 (Frequência de Cruzamento do Ganho) *A frequência de cruzamento do ganho (ω_c) é o valor de frequência angular no qual o sistema em malha aberta possui ganho de* $0\ dB$.

Definição 2 (Margem de Fase) *Margem de fase (φ_m) é definida como o valor do atraso de fase na frequência de cruzamento de ganho (ω_c) para o qual a fase do sistema em malha aberta torna-se* $180°$.

Definição 3 (Margem de Atraso) *Margem de atraso (L_m) é definida como o maior número L_m para o qual a estabilidade em malha fechada é garantida assumindo um atraso constante igual a L_m.*

Para sistemas de controle contínuos no tempo, a margem de atraso pode ser calculada em função da margem de fase e da frequência de cruzamento de ganho, conforme a Equação 3.16.

$$L_m = \frac{\varphi_m}{\omega_c} \tag{3.16}$$

3.2.6 Análise de estabilidade de sistemas com atrasos variantes

Considere o sistema em malha fechada com uma entrada e uma saída representado nas Figuras 3.1 (a), (b) e (c). Onde: $P(s)$ e $P(z)$ são as funções de transferência do processo, respectivamente, no tempo contínuo e no tempo discreto; $C(s)$ e $C(z)$ são as funções de transferência do controlador, respectivamente, no tempo contínuo e no tempo discreto; Δ indica o atraso de controle variante no tempo; A/D representa um conversor analógico digital que converte uma amostra do sinal de saída do processo a cada período h e D/A um conversor digital analógico com segurador de ordem zero (ZOH). Além disso, o sistema em malha fechada é estável para $\Delta = 0$. A estabilidade do sistema sob um atraso de controle Δ limitado e variante de modo aleatório é determinado pelos teoremas a seguir. As provas desses teoremas são apresentadas em Kao e Lincoln (2004).

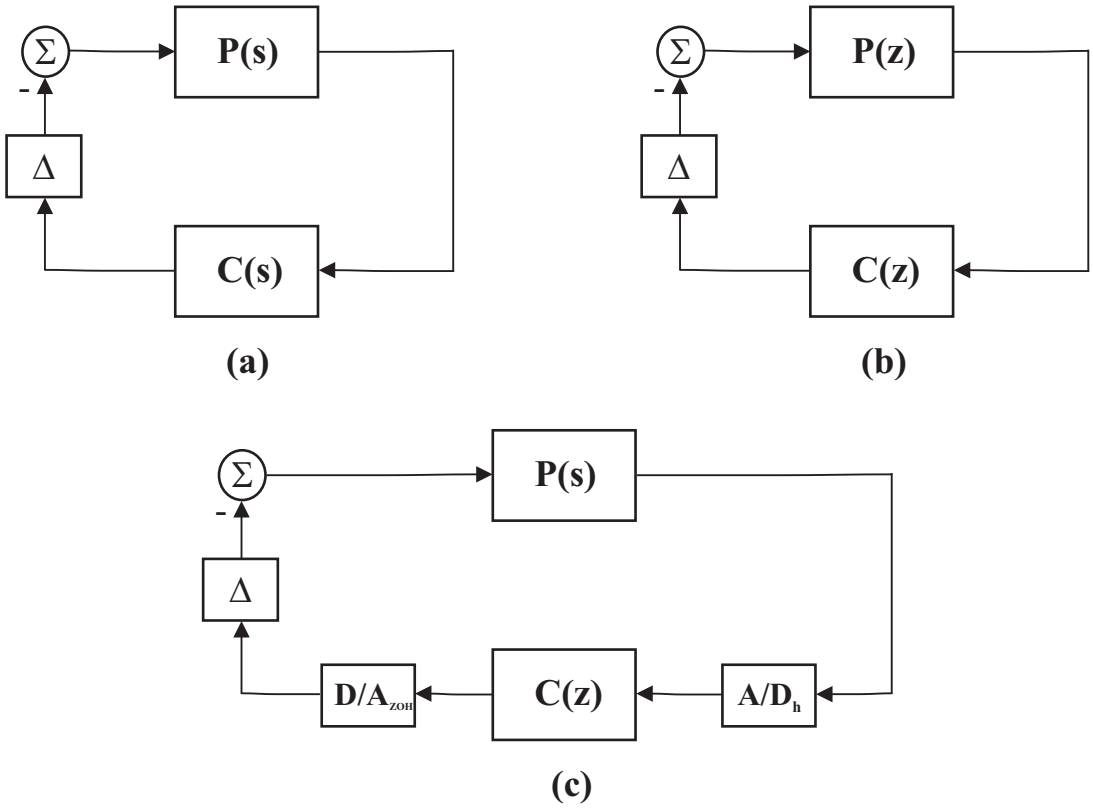


Figura 3.1: Diagramas de blocos de uma malha de controle com atraso Δ e: (a) processo e controlador contínuos no tempo; (b) processo e controlador discretos no tempo e (c) processo contínuo no tempo e controlador discreto no tempo.

Teorema 1 (Caso Contínuo no Tempo) *O sistema em malha fechada representado na Figura 3.1(a) é estavél para qualquer atraso variante no tempo definido por*

$$\Delta(v) = v(t - \delta(t)), \quad 0 \leq \delta(t) \leq \delta_{max} \tag{3.17}$$

se,

$$\left| \frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| < \frac{1}{\delta_{max}\omega}, \quad \forall\, \omega \in [0, \infty]. \tag{3.18}$$

Teorema 2 (Caso Discreto no Tempo) *O sistema em malha fechada representado na Figura 3.1(b) é estavél para qualquer atraso variante no tempo definido por*

$$\Delta(v) = v(n - \delta(n)), \quad \delta(n) \in \{0,1,...,N\} \tag{3.19}$$

se,

$$\left| \frac{P(e^{j\omega})C(e^{j\omega})}{1 + P(e^{j\omega})C(e^{j\omega})} \right| < \frac{1}{N|e^{j\omega} - 1|}, \quad \forall\, \omega \in [0, \infty]. \tag{3.20}$$

Teorema 3 (Caso Contínuo e Discreto no Tempo) *O sistema em malha fechada representado na Figura 3.1(c) é estável para qualquer atraso variante no tempo definido por*

$$\Delta(v) = v(t - \delta(t)), \quad 0 \leq \delta(t) \leq Nh \tag{3.21}$$

onde N é um número real, se

$$\left| \frac{P_{alias}(\omega)C(e^{i\omega})}{1 + P_{ZOH}(e^{i\omega})C(e^{i\omega})} \right| < \frac{1}{\tilde{N}|e^{i\omega} - 1|}, \forall\, \omega \in [0, \infty], \tag{3.22}$$

onde: $\tilde{N} = \sqrt{\lfloor N \rfloor^2 + 2\lfloor N \rfloor g + g}$[4]; $g = N - \lfloor N \rfloor$; $P_{ZOH}(z)$ é a discretização ZOH de $P(s)$ utilizando um segurador de ordem zero, e

$$P_{alias}(\omega) = \sqrt{\sum_{k=-\infty}^{\infty} \left| P\left(j(\omega + 2\pi k)\frac{1}{h} \right) \right|^2} \tag{3.23}$$

Observe que o valor máximo de $\tilde{N}$ deve ser obtido tal que a condição na Equação 3.22 do Teorema 3 seja satisfeita. Por outro lado, veja que o limite admissível para o atraso variável $\delta(t)$ está em função de N. Para facilitar a determinação do valor de N a partir de $\tilde{N}$, Santos (2008) derivou a Equação 3.24. Assim, o maior valor de atraso variante no tempo (Nh) pode ser obtido.

[4] $\lfloor N \rfloor$ é a função "chão"(*floor function*), que retorna o maior inteiro menor ou igual a N.

$$N = \frac{\tilde{N}^2 + \lfloor \tilde{N} \rfloor^2 + \lfloor \tilde{N} \rfloor}{2\lfloor \tilde{N} \rfloor + 1} \tag{3.24}$$

A partir do Teorema 3 apresentado na seção 3.2.6, Cervin et al. (2004) definiram os conceitos da margem de *jitter* e margem de fase aparente. Esses conceitos serão apresentados a seguir.

Margem de *jitter*

A margem de *jitter* é definida como o maior valor de atraso variável em função de um atraso constante e conhecido, que pode ser tolerado sem que a malha de controle torne-se instável. O atraso variável ($\delta(t)$) pode ser dividido em duas partes: uma parte constante ($L \geq 0$), e uma parte variante no tempo chamada de *jitter* ($J(t) \geq 0$), de acordo com a Figura 3.2. Então, do ponto de vista da análise de estabilidade, o processo $P(s)$ pode ser representado pela sua versão atrasada $P(s)e^{-sL}$. Dessa forma, o critério de estabilidade indicado na Equação 3.22 deve ser calculado a partir da versão atrasada do processo, e a estabilidade é garantida para um atraso $\bar{\delta}(t) \in [L, L + Nh]$. O sistema com a versão atrasada de $P(s)$ é representado pela Figura 3.3.

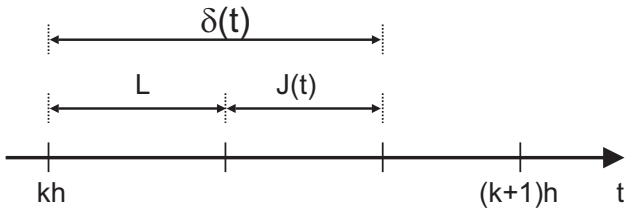


Figura 3.2: Esquema do atraso de controle dividido nas partes constante (L) e variante no tempo (J(t)).

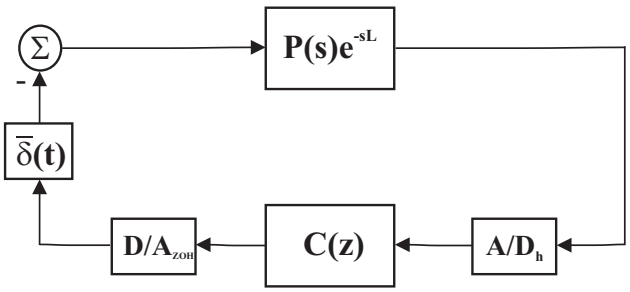


Figura 3.3: Diagrama de blocos modificado da malha de um sistema de controle com atraso $\bar{\delta}(t)$.

De acordo com essas considerações, a seguinte definição para margem de *jitter* foi formulada:

Definição 4 (Margem de *Jitter*) *Considerando o sistema representado na Figura 3.3, a margem de* jitter *é definida como o maior número $J_m(L)$ para o qual a estabilidade em malha fechada é garantida para qualquer atraso variante no tempo $\bar{\delta}(t) \in [L, L + J_m(L)]$.*

A partir da Definição 4, é dito que uma malha de controle é estável se, para todo ciclo de operação, $J(t) \leq J_m(L)$, onde $J(t) = \delta(t) - L$. Em resumo, o critério de estabilidade usando a margem de *jitter* pode ser descrito pelos seguintes passos:

1. Determine o valor mínimo de L e o valor máximo de $J(t)$ para um determinado sistema.
2. Determine o valor máximo de $\tilde{N}$ que satisfaz a Equação 3.22, onde $P(s)$ é substituído por $P(s)e^{-sL}$.
3. Calcule o valor de N por meio da Equação 3.24.
4. Calcule o valor da margem de *jitter* $J_m(L) = Nh$.
5. Verifique se o sistema é estável, ou seja, se $\max\{J(t)\} \leq J_m(L)$.

Margem de fase aparente

Na teoria de controle clássica, é comum usar a margem de fase como medida de desempenho e robustez. No entanto, a margem de fase é definida apenas para sistemas sem *jitter*. Para contornar esse problema, Cervin et al. (2004) generalizaram o conceito estendendo a definição da margem de atraso para sistemas com atraso constante e *jitter*.

Definição 5 (Margem de Atraso para Sistemas com Atraso Constante e *Jitter*) *Considere o sistema representado na Figura 3.3, assumindo um atraso constante L e um jitter $J(t)$, a margem de atraso é definida como o maior número $\hat{L}_m$ para o qual a estabilidade em malha fechada é garantida para algum atraso variante no tempo no intervalo $[L + \hat{L}_m, L + \hat{L}_m + J(t)]$.*

Expressada em termos da função margem de jitter ($J_m(L)$), a margem de atraso é determinada pelo menor valor de $\hat{L}_m$ que resolve a seguinte equação:

$$J_m(L + \hat{L}_m) = J(t) \tag{3.25}$$

A partir da relação entre margem de fase e margem de atraso (Equação 3.16) para sistemas de controle sem *jitter*, uma generalização da margem de fase, denominada de margem de fase aparente, foi definida.

Definição 6 (Margem de Fase Aparente) *Considere o sistema representado na Figura 3.3, assumindo que o atraso constante L e o* jitter *$J(t)$ são conhecidos, a margem de fase aparente é definida como o maior número $\hat{\varphi}_m$ para o qual a estabilidade em malha fechada é*

garantida para algum jitter $\bar{r} \in [L + \hat{\varphi}_m/\omega_c, L + \hat{\varphi}_m/\omega_c + J(t)]$*, onde* ω_c *é a frequência de corte do sistema considerando apenas o atraso constante* L*.*

A margem de fase aparente pode ser interpretada como o menor valor de atraso de fase em ω_c, que reduz a margem de *jitter* ao valor do atraso variável, ou seja, $J_m = J(t)$. Em outras palavras, expressa-se o quanto de atraso de fase o sistema suporta de forma que o critério de estabilidade seja respeitado, e pode ser indicado por:

$$J_m(L + \hat{\varphi}_m/\omega_c) = J(t) \tag{3.26}$$

A estabilidade não pode ser garantida para alguns valores de L e $J(t)$ se $\hat{\varphi}_m \leq 0$. Mas, se $\hat{\varphi}_m > 0$, a estabilidade é garantida. Um fator de degradação de desempenho (Fd) pode ser obtido pela razão entre a margem de fase aparente ($\hat{\varphi}_m$) e a margem de fase (φ_m), conforme indicado pela Equação 3.27.

$$Fd = \frac{\hat{\varphi}_m}{\varphi_m} \tag{3.27}$$

A partir dessa equação, as seguintes observações podem ser feitas:

- Para $Fd > 0$, o sistema é estável.
- Para $Fd \to 1$, o atraso variável causa pouca degradação ao sistema.
- Para $Fd < 0$, o sistema é instável.

Atribuição de *deadline*

Usando o conceito da margem de *jitter*, é possível atribuir um *deadline* relativo D (Equação 3.28) para o atraso no ciclo de operação de uma malha de controle que garante a estabilidade.

$$D = L + J_m(L) \tag{3.28}$$

De modo semelhante, é possível atribuir um *deadline* relativo que, além da estabilidade, garante uma margem de fase aparente. Dado um atraso constante L e uma margem de fase aparente desejável $\hat{\varphi}_m < \omega_c(L_m - L)$, o nível de desempenho é garantido para a seguinte atribuição de *deadline* relativo:

$$D = L + J_m(L + \hat{\varphi}_m/\omega_c) \tag{3.29}$$

3.2.7 Procedimentos para projeto de sistemas de controle com atrasos variantes

A partir dos conceitos da margem de *jitter* e margem de fase aparente, Cervin et al. (2004) desenvolveram um procedimento para projetar sistemas de controle via computador constituído por várias malhas de controle. Esse procedimento é constituído por oito passos:

1. Defina o período de amostragem (nominal) inicial h_i para cada controlador i. Uma regra de *thumb* (ASTROM; WITTENMARK, 1997) é escolher o período de amostragem tal que $\omega_{b_i} * h_i \in [0{,}2, 0{,}6]$, onde ω_{b_i} é a largura de banda do sistema contínuo em malha fechada;

2. Ajuste os períodos linearmente tal que o conjunto de tarefas seja escalonável para uma determinada política de escalonamento. Pode-se usar um teste de escalonabilidade suficiente;

3. Discretize os controladores (método de Tustin) usando os períodos de amostragem atribuídos, conduzindo no conjunto de controladores discretos K(z).

4. Para cada tarefa τ_i, calcule os tempos de resposta no melhor caso (R_i^{min}) e pior caso (R_i^{max});

5. Para cada tarefa τ_i, calcule a margem de *jitter* e a margem de fase aparente, assumindo um atraso constante $L_i = R_i^{min}$ e o *jitter* $J_i = R_i^{max} - R_i^{min}$

6. Para cada tarefa τ_i, calcule a degradação de desempenho relativa Fd_i (Equação 3.27) e o seu valor médio $\bar{dr}_i$;

7. Para cada tarefa τ_i, ajuste o período h_i de acordo com a equação $h_i = h_i + kh_i(Fd_i - \bar{Fd}_i)/\bar{Fd}_i$, onde $k < 1$ é um parâmetro de ganho;

8. Repita a partir do passo 2 até que não seja mais possível obter alguma melhoria. Um possível critério de parada é quando a soma das diferenças dos desempenhos, $\sum |Fd_i - \bar{Fd}_i|$ parar de decrescer.

Santos e Vasquez (2005) e (PEREZ; MORENO; MONTEZ, 2006) usaram o mesmo procedimento de oito passos desenvolvido por Cervin et al. (2004) para projetar, respectivamente, um NCS CAN com a arquitetura representada pelas Figuras 2.5(a) e (c) do Capítulo 2.

Santos (2008) adaptaram o procedimento de Cervin et al. (2004) e Perez, Moreno e Montez (2006) usando a síntese dos controladores via alocação de pólos. O procedimento consiste de onze passos:

1. Defina a equação característica em tempo contínuo $A(s)D(s) + B(s)N(s)$ desejada;

2. Defina as características desejadas adicionais dos controladores (Presença de filtro, integrador, etc.) e a ordem do controlador. Observação: na e nb são definidos em tempo discreto.

3. Arbitre um período de amostragem h_i para cada malha de controle i;

4. Redimensione os períodos linearmente tal que seja possível escalonar todo o conjunto de mensagens para uma determinada rede de comunicação;

5. Faça o mapeamento dos pólos contínuos desejados para o tempo discreto $z = e^{sh_i}$ de modo a definir os pólos discretos desejados;

6. Calcule os controladores de cada uma das malhas resolvendo equação Diophantina $A(z)D(z)+B(z)N(z) = (z - p_1)(z - p_2)...(z - p_{na+nd})$;

7. Para cada malha de controle, calcule o tempo de resposta fim a fim no melhor e pior caso;

8. Calcule a margem de *jitter* e a margem de fase aparente de cada malha de controle;

9. Para cada malha de controle i, calcule a degradação de desempenho relativa Fd_i (Equação 3.27) e o seu valor médio $\bar{Fd_i}$;

10. Para cada malha de controle i, ajuste o período h_i de acordo com a equação $h_i = h_i + kh_i(Fd_i - \bar{Fd_i})/\bar{Fd_i}$, onde $k < 1$ é uma constante;

11. Repita a partir do passo 4 até que não seja mais possível obter alguma melhoria. Um possível critério de parada é quando a soma das diferenças dos desempenhos, $\sum |Fd_i - \bar{Fd_i}|$ parar de decrescer.

3.2.8 Comentários

Sistemas de controle podem ser vistos como sistemas computacionais que necessitam de um tempo específico para realizar as suas atividades com restrições de tempo, de modo a garantir o desempenho e estabilidade dos processos. Portanto, pode-se considerar sistemas de controle como um tipo de sistema em tempo real. Na seção seguinte, será apresentada a teoria sobre sistemas em tempo real.

3.3 Sistemas em tempo real

Sistemas em Tempo Real (STRs) são sistemas onde o funcionamento correto depende não apenas dos resultados lógicos, mas também do instante em que esses resultados são produzidos (BUTTAZZO, 2005). Do ponto de vista computacional, as atividades de um STR podem ser

executadas em uma única Unidade de Processamento (UP) ou distribuída em várias UPs, que podem ser acopladas por meio de uma memória compartilhada ou por meio de uma rede de comunicação. No contexto deste trabalho, pode-se considerar NCSs como um STR onde as atividades são executadas em UPs que se comunicam via rede. Esses sistemas são também chamados de Sistemas Distribuídos em Tempo Real (SDTRs) (ALMEIDA, 1999).

Um SDTR é constituído por nós que se comunicam através de um meio de comunicação compartilhado de acordo com um protocolo de comunicação em rede. Cada nó (Figura 3.4) é formado basicamente por uma UP, dispositivos de rede, além de circuitos básicos (e.g. circuito oscilador.) omitidos na Figura. A UP dispõe de recursos usados para execução das atividades em tempo real tais como, processadores, temporizadores, controlador de interrupções, conversores analógico/digital e digital/analógico, *interface* de comunicação entre outros. Além disso, a UP é usada para gerenciar e controlar os dispositivos de rede, os quais implementam em *hardware* o protocolo de comunicação em rede. Em algumas aplicações, é necessário ter um sistema operacional em tempo real embarcado na UP para gerenciar os recursos.

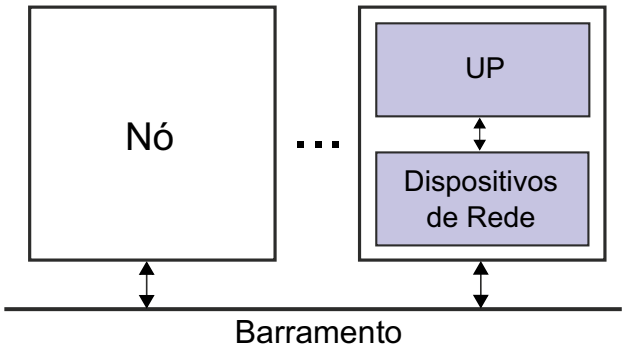


Figura 3.4: Representação básica de um nó em um SDTR.

SDTRs são normalmente modelados por um conjunto de tarefas e mensagens que usam recursos limitados e compartilhados para a realização de atividades específicas sujeitas a requisitos de tempo. O conceito de tarefa e mensagem é uma das abstrações básicas que fazem parte do desenvolvimento de um SDTR. Nesses sistemas, as tarefas são executadas em diferentes UPs e comunicam-se por meio de mensagens transmitidas via o meio de comunicação. Os conceitos que definem tarefas e mensagens são semelhantes. A principal diferença está relacionada com o recurso compartilhado e limitado. No caso das tarefas esse recurso é a UP e no caso das mensagens, o recurso é o meio de comunicação. Existem também outros tipos de recursos que podem ser compartilhados tais como, variáveis e *buffers* de transmissão/recepção. Nas seções seguintes, apresenta-se a teoria básica necessária para entender e projetar SDTRs.

3.3.1 Relógios

Em sistemas computacionais, é comum considerar o conceito Newtoniano de tempo, ou seja, o tempo é considerado uma dimensão externa e contínua que é percebida igualmente em qualquer lugar. Essa noção de tempo, chamada de tempo real e denotada por t, pode ser representada pelo conjunto de números reais positivos. O tempo nos sistemas computacionais é marcado por um relógio. Basicamente, existem três tipos de relógios: ideal, real (físico) e o virtual (GONZáLEZ, 2010). Neste trabalho, consideraremos apenas relógios ideais e reais.

Um relógio ideal é um relógio que sempre reflete o valor de tempo real. Então, se denotarmos um valor do relógio ideal no instante t como $t_{ideal}(t)$, então $t_{ideal}(t) = t$ para todo t. Na prática, relógios ideais não existem fisicamente. Eles são usados apenas teoricamente para representar o tempo real.

Relógios reais são usados para medir fisicamente o tempo real em um sistema computacional. A base de tempo usada por um relógio para medir o tempo real é derivada de um circuito oscilador de cristal de quartzo[5]. Entretanto, devido as tolerâncias de fabricação, as frequências de oscilação dos cristais variam ao longo do tempo em torno da frequência nominal. Além disso, fatores como temperatura, nível de tensão elétrica aplicada ao circuito oscilador, envelhecimento do cristal entre outros, também influenciam na variação da frequência. Desse modo, a base de tempo dos relógios variam ao longo do tempo.

As variações na frequência de um cristal e, consequentemente, em um relógio real x, denotadas por ρ_x, são especificadas em ppm (*parts per million*), e indicam a lentidão ($\rho_x < 0$) e rapidez ($\rho_x > 0$) de um relógio em comparação a um relógio ideal. Considera-se que um relógio real x é não defeituoso se a variação ρ_x for limitada no intervalo $[\rho_x^{min}, \rho_x^{max}]$, onde min e max indicam, respectivamente, os valores mínimo e máximo.

A taxa de variação de um relógio real x, denotada por $\hat{\rho}_x$, varia no intervalo $[\hat{\rho}_x^{min}, \hat{\rho}_x^{max}]$, e é definida pela equação 3.30, onde: f_x é a frequência real e f é a frequência nominal (ideal) do cristal.

$$\hat{\rho}_x = \frac{f_x}{f} = 1 + \rho_x \tag{3.30}$$

Normalmente, as variações indicadas pelos fabricantes são de ± 30 ppm para uma temperatura padrão de 25°C (HENDERSON; KENDALL; ROBSON, 2001), resultando em uma variação de ± 30 μs a cada segundo ($-0{,}000030 \leq \rho_x \leq 0{,}000030$), que equivale a uma variação na frequência de ± 30 Hz a cada 1 MHz. Em outras palavras, isso significa que a frequência de um relógio real com cristal 10 MHz varia no intervalo $[9999700, 10000300]$ (em Hz) e, consequentemente, o relógio possui uma taxa de variação no intervalo [0,999970, 1,000030].

Na prática, em um SDTR, os relógios reais de cada nó serão mais lentos ou mais rápidos

[5] É possível usar ressonadores cerâmicos.

entre si, mesmo tendo cristais com mesma frequência. Isso ocorre pois, as variações nos relógios ocorrem de forma aleatória dentro do intervalo especificado. Na Figura 3.5 apresenta-se um gráfico dos valores de dois relógios reais $x(t)$ e $y(t)$, e um relógio ideal versus o tempo real t, onde ambos os relógios possuem cristais com a mesma frequência. O relógio x é mais rápido e o relógio y é mais lento quando comparado ao relógio ideal.

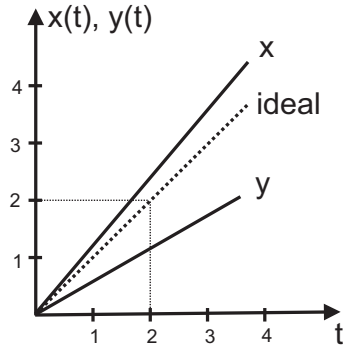


Figura 3.5: Gráfico dos valores dos relógios reais (x e y) e do relógio ideal versus o tempo real t.

3.3.2 Tarefas computacionais

Uma tarefa pode ser definida como uma sequência de instruções pré-determinadas (trecho de código de um programa), que são executadas em uma UP com o objetivo de realizar atividades específicas. Durante a operação de um STR, uma tarefa pode ser requisitada para execução por um número infinito de vezes. Uma requisição da tarefa é chamada de instância.

As tarefas podem ser classificadas de acordo com a freqüência de ativação de suas instâncias, que pode ser periódica, aperiódica ou esporádica. Tarefas periódicas são caracterizadas por um intervalo de tempo constante entre sucessivas ativações de suas instâncias. Esse intervalo de tempo é chamado de período. Tarefas aperiódicas são caracterizadas por um intervalo de tempo desconhecido entre sucessivas ativações de suas instâncias. Enquanto que tarefas esporádicas, consideradas como um tipo de tarefa aperiódica, são caracterizadas pelo conhecimento de um intervalo de tempo mínimo entre sucessivas ativações.

A escolha do tipo de tarefa usada para implementar partes de um STR é baseada nas características da aplicação. Tarefas periódicas podem ser usadas para amostrar os estados de um processo físico. Para manipulação de eventos como um alarme, pode-se usar uma tarefa esporádica quando o intervalo de tempo mínimo entre sucessivas ativações for conhecido. Se o intervalo de tempo mínimo for desconhecido, uma tarefa aperiódica pode ser usada. Em uma aplicação real, as tarefas esporádicas podem ser tratadas como tarefas periódicas considerando o intervalo de tempo mínimo entre sucessivas ativações igual ao período. Para tarefas aperiódicas, normalmente usa-se o conceito de servidores (SPRUNT; SHA; LEHOCZKY, 1989).

Consideraremos a partir daqui apenas tarefas de tempo real periódicas.

As tarefas são ativadas e depois liberadas (prontas para execução). A primeira instância de uma tarefa é ativada após um tempo chamado de *offset*. O *offset* pode ser estático ou dinâmico. No *offset* estático, o instante de ativação é fixado. Enquanto que, no *offset* dinâmico, o instante de ativação varia entre zero e um limite superior conhecido. Após o *offset*, a tarefa é ativada periodicamente.

Uma tarefa pode ser ativada e liberada simultaneamente ou liberada somente após a ocorrência de um determinado evento. Dependendo do modelo de tarefas do sistema, o instante de liberação pode ou não coincidir com o instante de ativação da tarefa. Por exemplo, é possível que a liberação de uma tarefa ocorra apenas quando uma outra tarefa for finalizada. A variação entre os tempos de ativação e liberação é chamado de *jitter* de liberação (AUDSLEY et al., 1993). O *jitter* de liberação pode variar entre um valor mínimo e máximo, chamado de *jitter* de liberação no melhor caso e no pior caso, respectivamente.

O tempo necessário para executar completamente uma instância da tarefa é chamado de tempo de computação. Esse tempo pode variar entre um tempo mínimo e máximo, chamado de tempo de computação no melhor caso e no pior caso, respectivamente.

Toda tarefa de tempo real está sujeita a um requisito de tempo chamado de *deadline*. A princípio, deve-se garantir que toda instância da tarefa seja finalizada antes do seu *deadline*. As conseqüências ao serem finalizadas após o seu *deadline* definem dois tipos de tarefas de tempo real: tarefas críticas (*hard*) e tarefas não-críticas (*soft*). Uma tarefa é crítica se as conseqüências resultarem em falhas severas ao sistema e, se as conseqüências não forem severas, a tarefa é não-crítica.

Em resumo, o comportamento temporal de uma tarefa periódica τ_i, onde i indica o número da tarefa, pode ser descrito por cinco atributos $(T_i, O_i, C_i, J_i, D_i)$. Onde: T_i denota o período; O_i o *offset*; C_i o tempo de computação, que varia entre C_i^{min} e C_i^{max}; J_i o *jitter* de liberação, que varia entre J_i^{min} e J_i^{max}, e D_i o *deadline* relativo da tarefa (relativo a ativação da tarefa). Os símbolos min e max indicam, respectivamente, melhor caso e pior caso. Na Figura 3.6 está representada uma diagrama de tempo contendo as instâncias de uma tarefa periódica τ_i e seus atributos.

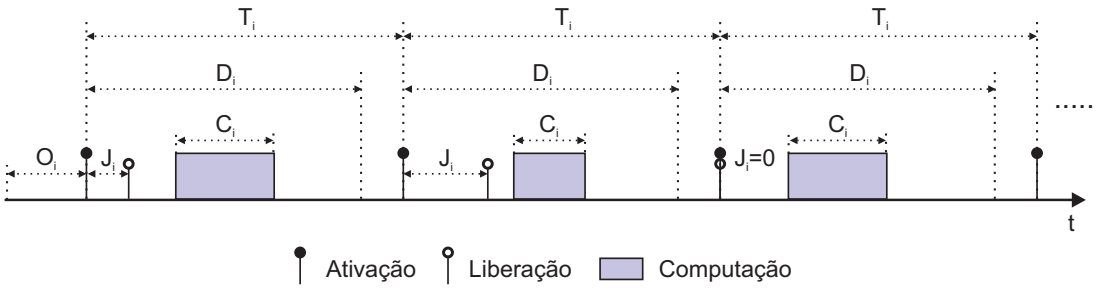


Figura 3.6: Representação das instâncias de uma tarefa periódica e seus atributos.

O fator de utilização (U) de uma UP que executa p tarefas periódicas é definido pela Equação 3.31.

$$U = \sum_{i=1}^{p} \left(\frac{C_i}{T_i} \right) \leq 1 \tag{3.31}$$

Relações de precedência

Em aplicações de tempo real tais como, Sistemas de Controle via Rede (NCSs), a execução de tarefas computacionais e o envio de mensagens não podem ocorrer em ordens arbitrárias. Algumas atividades serão iniciadas somente quando outras atividades específicas forem finalizadas, resultando em relações de precedência. Por exemplo, uma mensagem m_j é precedida por uma tarefa τ_i ($\tau_i \rightarrow m_j$), se a transmissão de m_j puder ocorrer somente após o término da execução de τ_i.

Formalmente, as relações de precedência podem ser representadas na forma de um grafo acíclico orientado, onde os círculos correspondem às tarefas e mensagens do sistema, e os arcos descrevem as relações de precedência entre as tarefas e/ou entre as mensagens. Um exemplo das relações de precedência entre duas tarefas e uma mensagem representadas por um grafo acíclico estão ilustradas na Figura 3.7.

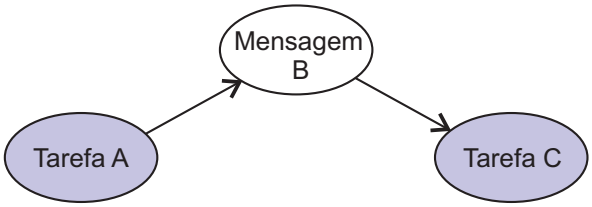


Figura 3.7: Representação das relações de precedência entre duas tarefas e uma mensagem.

3.3.3 Escalonamento de tarefas

Em algumas aplicações, o processador (ou algum outro recurso de uma UP) pode ser compartilhado por mais de uma tarefa. Consequentemente, é necessário decidir qual tarefa poderá usar o recurso em um determinado instante. Esse processo de decisão é chamado de escalonamento e é realizado pelo escalonador do sistema, que produz uma escala de execução de acordo com uma política de escalonamento. Políticas de escalonamento definem critérios ou regras para a ordenação das tarefas. Os escalonadores, que podem ser classificados em *offline* e *online*, usam essas políticas de escalonamento para gerar escalas que se forem realizáveis, garantem o cumprimento dos requisitos de tempo especificados.

Escalonadores

Nos escalonadores *offline*, todas as decisões de escalonamento são feitas durante a fase de projeto. Um tipo de escalonador *offline* é o guiado a tempo (XU; PARNAS, 1993). Nesse tipo de escalonador, uma escala de tempo é dividida em partições e essas partições são atribuídas a cada uma das tarefas durante o projeto do sistema. Cada partição possui um intervalo de tempo equivalente ao tempo de execução da respectiva tarefa. Portanto, o instante de execução de cada tarefa é determinado previamente e o tempo de resposta da tarefa pode ser constante.

Nos escalonadores *online*, as decisões de escalonamento são feitas dinamicamente durante a execução do sistema. O principal tipo de escalonador *online* é o guiado a prioridades. Nesse tipo, as tarefas são escalonadas de acordo com suas prioridades. Esses escalonadores podem ser classificados como preemptivos e não-preemptivos (GEORGE; RIVIERRE; SPURI, 1996). No escalonamento preemptivo, as tarefas com maior prioridade podem suspender a execução de tarefas com menor prioridade. No escalonamento não-preemptivo, uma tarefa não pode ser suspendida durante sua execução. Um diagrama de tempo representando a execução de duas tarefas (X e Y) em uma UP sob escalonamento preemptivo e não-preemptivo está ilustrado na Figura 3.8. Os tempos de computação das tarefas X e Y são constantes e iguais a 3 e 5 unidades de tempo, respectivamente. A tarefa X possui maior prioridade e é ativada e liberada no instante 2, enquanto que a tarefa Y é ativada e liberada no instante 1. Isso significa que o *jitter* de liberação das tarefas é igual a zero. No caso preemptivo, a tarefa Y inicia sua execução no instante 1 e no instante 2 é suspendida por X, que segue executando até o instante 5, onde finaliza sua execução. Nesse mesmo instante, a tarefa Y re-inicia a execução, que é finalizada no instante 9. No caso não-preemptivo, a tarefa X (maior prioridade) não suspende a execução da tarefa Y, que é finalizada no instante 6. Imediatamente, a tarefa X inicia sua execução e finaliza após 3 unidades de tempo. O tempo de resposta das tarefas sob escalonamento online é variável de forma limitada.

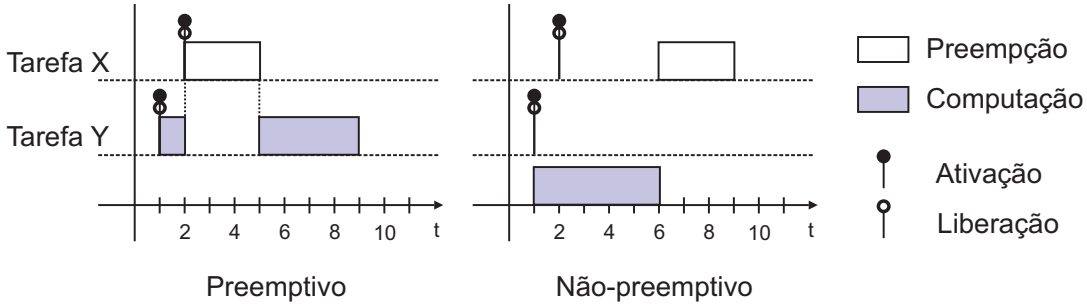


Figura 3.8: Representação da execução de tarefas em um processador sob escalonamento preemptivo e não-preemptivo.

Outros tipos de escalonadores *online* bem simples são o FIFO (*First-In First-Out*) e o Round-Robin (TANENBAUM; WOODHULL, 1997). Nesses escalonadores, as tarefas são execu-

tadas de acordo com a ordem de chegada. Escalonadores FIFO são não-preemptivos e basicamente são implementados por meio de uma fila organizada por ordem de chegada das tarefas. A primeira tarefa da fila inicia a sua execução até finalizar as suas computações. Após a finalização, a próxima tarefa da fila entrará em execução, e assim por diante. Escalonadores Round-Robin são a versão preemptiva dos escalonadores FIFO. Nesse escalonador, as tarefas são escalonadas em uma sequência FIFO, mas cada tarefa pode executar apenas por uma quantidade de tempo limitada, chamada de *time slice* ou *quantum*. Após a execução do seu *quantum*, a tarefa é suspendida e a próxima tarefa da fila inicia a sua execução. A tarefa suspendida será colocada no fim da fila, caso não tenha finalizado as suas computações. Neste trabalho será usado apenas escalonadores *online*.

Políticas de escalonamento

No escalonamento *online* guiados a prioridade, a prioridade de cada tarefa é definida de acordo com uma política de atribuição, que pode ser estática (prioridades fixas) tais como, taxa monotônica (LIU; LAYLAND, 1973) e *deadline* monotônico (LEUNG; WHITEHEAD, 1982), ou dinâmica (prioridades variáveis), por exemplo, EDF (*Earliest Deadline First*) (LIU; LAYLAND, 1973). Políticas de escalonamento com prioridades fixas são fáceis de implementar, enquanto que políticas de escalonamento com prioridades variáveis são melhores teoricamente e de complexa implementação (GEORGE; RIVIERRE; SPURI, 1996).

Na política taxa monotônica, as prioridades das tarefas são atribuídas de acordo com os valores dos seus períodos. Quanto maior o período menor será a prioridade. Na política *deadline* monotônico, as prioridades das tarefas são atribuídas de acordo com os valores dos seus *deadlines*. A tarefa com menor *deadline* possui maior prioridade, enquanto que a tarefa com maior *deadline* possui menor prioridade. Na política EDF, as prioridades das tarefas são atribuídas de acordo com o tempo restante para atingir seus *deadlines*. Quanto menor for esse tempo maior será a sua prioridade. Desse modo, as prioridades são alteradas dinamicamente durante a execução do sistema.

Testes de escalonabilidade

No escalonamento *online* uma análise ou teste de escalonabilidade deve ser usada para determinar se o conjunto de tarefas do sistema é escalonável, ou seja, se os *deadlines* serão satisfeitos. Os testes variam de acordo com o modelo de tarefas e políticas definidas em um problema de escalonamento, e podem ser classificados em necessários, suficientes e exatos.

Testes necessários determinam com precisão se um conjunto de tarefas é não-escalonável. As tarefas que passaram no teste podem ser escalonáveis. Nos testes suficientes, o conjunto de tarefas que passaram no teste são escalonáveis. No entanto, é possível que existam tarefas

escalonáveis dentro do conjunto de tarefas que não passaram no teste. Enquanto que testes exatos determinam precisamente os conjuntos de tarefas escalonáveis e não-escalonáveis.

Existem três diferentes abordagens para testes de escalonabilidade: testes baseados na utilização; testes baseados na demanda e testes baseados nos tempos de resposta. A primeira abordagem usa o fator de utilização do recurso compartilhado pelo conjunto de tarefas (LIU; LAYLAND, 1973) e (LEHOCZKY, 1990). A segunda abordagem é baseada na demanda do processador em um determinado intervalo de tempo (BARUAH; ROSIER; HOWELL, 1990) e (GEORGE; RIVIERRE; SPURI, 1996). A última abordagem é baseada no cálculo do tempo de resposta no pior caso para cada uma das tarefas do conjunto (JOSEPH; PANDYA, 1986) e (AUDSLEY et al., 1993). Os testes baseados nos tempos de resposta são mais usados, pois determinam os limites superiores e inferiores dos atrasos de cada tarefa do sistema. Se esses limites não são maiores que os respectivos *deadlines* de cada tarefa, o sistema é escalonável. Além disso, essa abordagem pode ser usada para tarefas com *deadlines* maiores que o período (*deadlines* arbitrários), tarefas esporádicas e tarefas dependentes (LIMA, 2003). Nas tarefas dependentes há relações de precedência e/ou compartilhamento de recursos. Enquanto que tarefas independentes não há qualquer tipo de relação.

3.3.4 Análise dos tempos de resposta das tarefas

Nesta seção apresenta-se a análise de tempos de resposta para escalonamento preemptivo com prioridade fixa. Essa análise foi desenvolvida por Joseph e Pandya (1986) para calcular os tempos de resposta de tarefas assumindo *jitter* de liberação igual a zero. Audsley et al. (1993) e Tindell, Burns e Wellings (1994) estenderam o trabalho de Joseph e Pandya (1986) para considerar o efeito do *jitter* de liberação das tarefas. Essas análises podem ser usadas para qualquer política de escalonamento com prioridade fixa.

Notação e modelo das tarefas

Considere um único processador e um conjunto com p tarefas independentes e ativadas periodicamente $\tau_1, \tau_2, ..., \tau_p$. Em algum instante, o processador é usado para executar a tarefa pendente (liberada para execução) com maior prioridade. No escalonamento preemptivo, a tarefa pendente de maior prioridade poderá suspender alguma tarefa com menor prioridade que estiver em execução. A tarefa que foi suspendida re-iniciará a execução de onde parou assim que o processador estiver livre e não houver tarefas pendentes com maior prioridade. No escalonamento não-preemptivo, a tarefa pendente de maior prioridade será bloqueada pela tarefa de menor prioridade em execução.

Cada tarefa τ_i, onde i varia de 1 a p, é caracterizada por um período $T_i \in \mathbb{R}^+$, um tempo de computação $C_i \in \mathbb{R}^+$, um *deadline* relativo $D_i \in \mathbb{R}^+$ e um *jitter* de liberação $J_i \in \mathbb{R}^+$. Assume-

se que os *deadlines* das tarefas são menores ou iguais aos seus respectivos períodos, e o tempo de computação é constante e conhecido. A variável i também indica a prioridade da mensagem τ_i. Por conveniência, considera-se que o nível de prioridade decresce de acordo com o número da tarefa, ou seja, τ_1 possui a maior prioridade e τ_p possui a menor prioridade do conjunto de tarefas.

A instância k de uma tarefa τ_i é ativada a cada instante $a_i(k) = kT_i$ e liberada no instante $l_i(k)$ determinado pela Equação 3.32, onde $k \in \mathbb{Z}$. Se τ_i for sempre ativada e liberada no mesmo instante, $J_i = 0$. Caso contrário, τ_i será liberada em algum instante a partir da sua ativação até J_i.

$$l_i(k) = J_i(k) + kT_i \tag{3.32}$$

O *deadline* absoluto da instância k de τ_i é determinada pela Equação 3.33.

$$d_i(k) = kT_i + D_i \tag{3.33}$$

O intervalo de resposta da instância k de τ_i é definido como o espaço de tempo entre o instante de ativação $a_i(k)$ dessa instância e o instante de finalização $f_i(k)$, ou seja, o intervalo semi-aberto à direita $[a_i(k), f_i(k))$. O tempo de resposta desta instância $r_i(k)$ (Equação 3.34) é definido como o comprimento do seu intervalo de resposta.

$$r_i(k) = f_i(k) - a_i(k) \tag{3.34}$$

Uma representação do modelo de tarefas para escalonamento preemptivo está ilustrada na Figura 3.9.

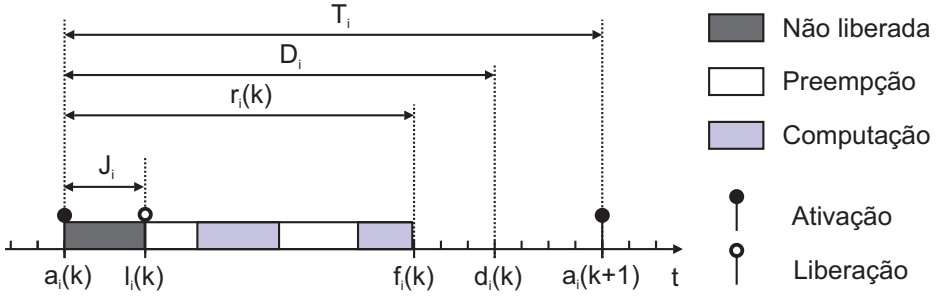


Figura 3.9: Representação do modelo de tarefas no escalonamento preemptivo com prioridade fixa.

O tempo de resposta no pior caso R_i^{max} de τ_i (Equação 3.35) é o maior tempo de resposta de alguma de suas instâncias ocasionado no chamado instante crítico, que é o instante onde todas as tarefas com maior prioridade que τ_i são liberadas simultaneamente (LIU; LAYLAND, 1973). Considera-se que as tarefas são defasadas arbitrariamente. Essa suposição garante a possibilidade de ocorrência do instante crítico.

$$R_i^{max} = \max(r_i(k)) \tag{3.35}$$

Para finalizar o modelo, considera-se algumas suposições comuns na literatura sobre escalonamento em tempo real: as tarefas estão prontas para execução assim que forem liberadas e não podem suspender elas mesmas; as tarefas são preemptadas instantaneamente quando uma tarefa com maior prioridade torna-se pronta para execução; o processador nunca ficará ocioso, exceto se não existirem tarefas prontas para execução; os custos de tempo devido a contexto de chaveamento e escalonamento das tarefas estão inclusos nos tempos de computação, e finalmente, os *deadlines* são críticos, ou seja, cada instância da tarefa deve ser finalizada antes do seu *deadline* absoluto. O conjunto com p tarefas periódicas é escalonável, se e somente se, $R_i^{max} \leq D_i, \forall\ i = 1,...,p$.

Tempos de resposta no pior caso

O tempo de resposta no pior caso de uma tarefa τ_i é derivado da Equação 3.36.

$$R_i^{max} = J_i + S_i \tag{3.36}$$

Onde J_i é o *jitter* de liberação e S_i corresponde a soma do tempo de computação C_i com o atraso de interferência[6], conforme descrito na Equação 3.37, sendo $hp(i)$ o conjunto de tarefas com maior prioridade que a tarefa τ_i. O termo $\left\lceil \frac{S_i+J_j}{T_j} \right\rceil$, correspondente a função "teto"(*ceiling function*), retorna o menor inteiro maior ou igual a $\frac{S_i+J_j}{T_j}$.

$$S_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{S_i + J_j}{T_j} \right\rceil C_j \tag{3.37}$$

A Equação 3.37 pode ser resolvida por meio de uma técnica iterativa baseada em relações de recorrência, de acordo com a Equação 3.38, onde x é a variável iterativa.

$$S_i^{(x+1)} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{S_i^{(x)} + J_j}{T_j} \right\rceil C_j \tag{3.38}$$

O procedimento iterativo começa com um valor inicial $S_i^{(0)} = C_i$, e é finalizado quando $S_i^{(x+1)} = S_i^{(x)}$ ou quando $S_i^{(x)} + J_i > D_i$, onde nessa última condição, τ_i é não-escalonável. Garante-se que $S_i^{(x+1)}$ convergirá se o fator de utilização do processador U for menor ou igual a 1.

[6]O atraso de interferência equivale ao tempo máximo que τ_i é suspendida por tarefas de maior prioridade durante o seu tempo de resposta.

3.4 Rede de comunicação CAN

CAN (*Controller Area Network*) é um rede de comunicação serial desenvolvida pela empresa Robert Bosch GmbH na década de 80, para satisfazer os requisitos da engenharia automotiva (BOSCH, 1991).

O protocolo da rede CAN pode ser descrito em termos do modelo de referência OSI (*Open Systems Interconnections*) por três camadas: a camada de enlace de dados, a camada física, e opcionalmente a camada de aplicação que pode ser especificada pelo projetista do sistema ou por alguns dos protocolos da camada superior baseados no CAN como por exemplo, o CAL (*CAN Application Layer*)/CANopen (PFEIFFER; AYRE; KEYDEL, 2003). A especificação desenvolvida pela Bosch implementa apenas a camada de enlace e parte da camada física, que é descrita pelo padrão ISO 11898-1 (ISO, 1999a). O restante da camada física é definida na ISO 11898-2 para aplicações em alta velocidade (ISO, 1999b), e na ISO 11898-3 para aplicações em baixa velocidade (ISO, 1999c). A camada de sessão para a comunicação CAN, chamado de protocolo TTCAN (*Time-Triggered CAN*) foi definida na ISO 11898-4 (ISO, 2004). Na Figura 3.10 representa-se o modelo OSI e as camadas usadas pelo protocolo da rede CAN.

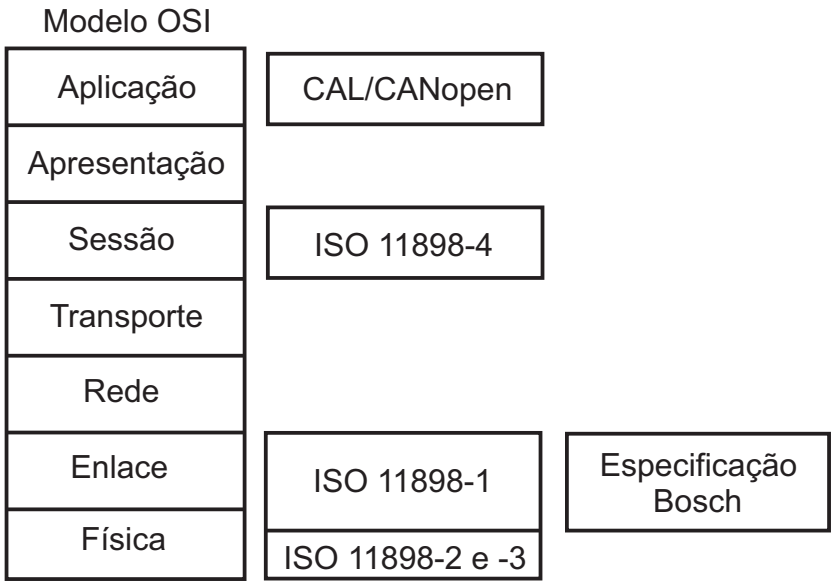


Figura 3.10: Camadas do modelo OSI utilizadas pela rede CAN e os padrões ISO.

Basicamente, uma rede CAN é constituída por nós conectados a um barramento serial que trocam mensagens a uma taxa de transmissão de até 1 Mbits/s. Um nó CAN é constituído por três componentes básicos (Figura 3.11): Unidade de Processamento (UP) e pelos dispositivos de rede (controlador e *transceiver* CAN). Para transmitir uma mensagem, uma tarefa executada na UP prepara os dados a serem transmitidos, armazena-os no *buffer* de transmissão e envia uma requisição de transmissão ao controlador CAN. Assim que o barramento estiver livre, o controlador tentará transmitir a mensagem.

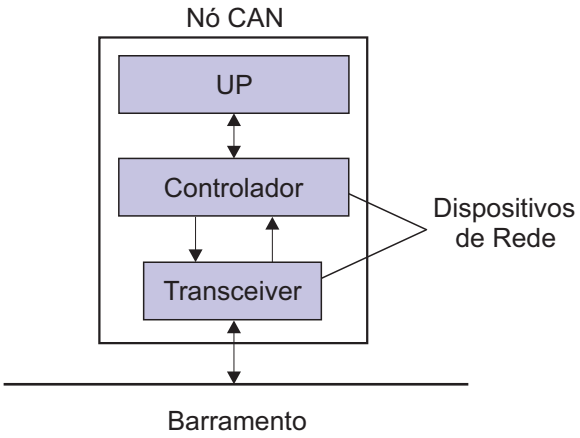


Figura 3.11: Diagrama de blocos de um nó CAN.

3.4.1 Mensagens CAN

Existem quatro diferentes tipos de mensagens: dados, erro, requisição de dados e sobrecarga. As mensagens de dados e de requisição de dados são usadas durante a operação normal da rede. A mensagem de dados é usada para transferir dados de um nó para os demais nós da rede, e a mensagem de requisição de dados é usada para requisitar dados de um determinado nó da rede. As mensagens de erro e de sobrecarga são utilizadas para sinalizar um estado anormal da rede. A mensagem de erro sinaliza a existência de um erro gerado no barramento, e a mensagem de sobrecarga sinaliza que um nó particular não está pronto para receber mensagens naquele momento. Apesar da existência do mecanismo de sinalização de sobrecarga, a maioria dos controladores CAN não requerem um atraso na recepção da próxima mensagem. Apenas mensagens de dados e de erro serão consideradas neste trabalho[7]. Cada mensagem de dados possui em seu cabeçalho um identificador único usado para caracterizar o conteúdo da mensagem e determinar o nível de prioridade no acesso ao barramento.

3.4.2 Escalonamento de mensagens

Na rede CAN é possível que mais de um nó tente transmitir uma mensagem simultaneamente. Para controlar as possíveis colisões durante o acesso ao barramento, CAN utiliza o mecanismo de controle de acesso ao meio CSMA/DCR (*Carrier-Sense Multiple Access/Deterministic Collision Resolution*). Nesse mecanismo, os nós tentam transmitir uma mensagem somente se o barramento estiver livre. Assim que o barramento estiver livre, todos os nós com mensagens prontas para serem transmitidas entram no processo de arbitragem[8]. O nó com a mensagem de maior prioridade vencerá o processo de arbitragem. Os nós que perderam a disputa e outros possíveis nós, tentarão transmitir suas mensagens assim que o barramento estiver livre nova-

[7]Para maiores detalhes dessas mensagens, veja as seções A.2 e A.3 no apêndice A.

[8]Um exemplo do processo de arbitragem é descrito na seção A.6 no apêndice A.

mente.

O controle de acesso ao meio garante que as mensagens serão escalonadas de modo semelhante ao escalonamento não-preemptivo de tarefas em um processador. Quando o recurso compartilhado (barramento) estiver ocupado, a mensagem de maior prioridade pronta para ser transmitida deverá esperar até que o recurso esteja disponível. Isso significa que as mensagens de maior prioridade poderão sofrer bloqueio de mensagens com menor prioridade.

3.4.3 Análise dos tempos de resposta das mensagens

Nesta seção, apresenta-se a análise dos tempos de resposta no pior caso de mensagens CAN. A análise foi desenvolvida por Davis et al. (2007) baseada nos trabalhos de Tindell, Burns e Wellings (1994) e George, Rivierre e Spuri (1996). Essa análise será usada para determinar os tempos de resposta de mensagens CAN e os resultados serão comparados com os resultados obtidos usando as técnicas de verificação de modelos no Capítulo 5.

Notação e modelo das mensagens

Considere uma rede CAN com um único barramento e um conjunto com q mensagens m_1, m_2,..., m_q. Uma mensagem m_i, onde i assume valores de 1 a q, consiste de um número infinito de instâncias $k \in \mathbb{Z}^+ = \{0, 1, 2,...\}$ contendo informações específicas. Assume-se que o controlador de cada nó garante que a sua mensagem com maior prioridade armazenada no *buffer* de transmissão, entrará no processo de arbitragem assim que o barramento estiver livre. Essa suposição é compatível com o funcionamento dos controladores CAN. A variável i também indica a prioridade da mensagem m_i.

Cada mensagem possui uma tarefa periódica relacionada τ_{m_i} com período T_{m_i}. Essa tarefa armazena a mensagem no *buffer* de transmissão do controlador CAN do seu respectivo nó. Então, uma instância k de uma mensagem é ativada a cada instante $a_{m_i} = kT_{m_i}$ e liberada (pronta para transmissão) assim que a tarefa relacionada armazená-la no *buffer* de transmissão. Essa tarefa gasta uma quantidade de tempo igual a J_{m_i}, chamado de *jitter* de armazenamento, que equivale ao tempo de resposta da tarefa τ_{m_i}. Considera-se que não existe controle sobre o instante de ativação da primeira instância de cada mensagem, ou seja, assim que o sistema começa a funcionar, a ativação da primeira instância das mensagens ocorre em um instante arbitrário, e que os relógios de cada nó são ideais.

Toda mensagem possui uma quantidade de *bytes* de dados b_{m_i}, um tempo de transmissão C_{m_i} e um requisito de tempo chamado de *deadline* relativo D_{m_i}. Assume-se que o *deadline* relativo de uma mensagem é menor ou igual ao período T_{m_i}.

A instância k de uma mensagem m_i é liberada no instante $l_{m_i}(k)$ determinado pela Equação 3.39.

$$l_{m_i}(k) = J_{m_i}(k) + kT_{m_i} \tag{3.39}$$

O *deadline* absoluto d_{m_i} da instância k da mensagem m_i é determinado pela Equação 3.40.

$$d_{m_i}(k) = kT_{m_i} + D_{m_i} \tag{3.40}$$

O intervalo de resposta da instância k da mensagem m_i é definido como o tempo entre o instante de ativação $a_{m_i}(k)$ e o instante de finalização $f_{m_i}(k)$ (chegada do último *bit* da mensagem ao nó receptor) dessa instância, ou seja, o intervalo semi-aberto à direita $[a_{m_i}(k), f_{m_i}(k))$. O tempo de resposta $r_{m_i}(k)$ dessa instância (Equação 3.41) é definido como o comprimento do seu intervalo de resposta.

$$r_{m_i}(k) = f_{m_i}(k) - a_{m_i}(k) \tag{3.41}$$

Uma representação do modelo de mensagens está ilustrada na Figura 3.12.

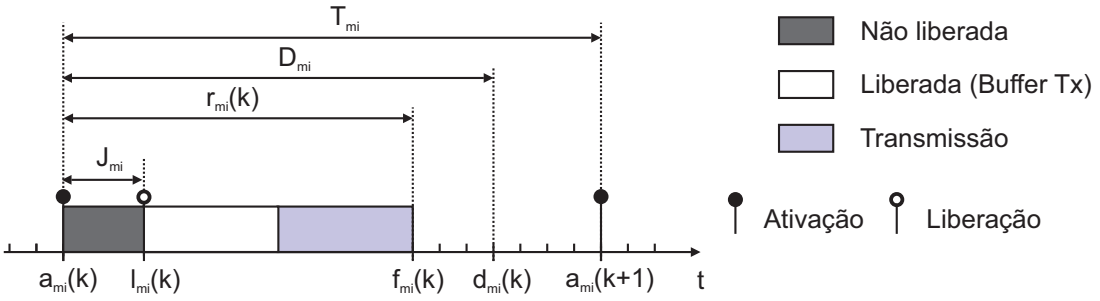


Figura 3.12: Representação do modelo de mensagens CAN.

O tempo de resposta no pior caso $R_{m_i}^{max}$ da mensagem m_i equivale ao maior (Equação 3.42) tempo de resposta de alguma de suas instâncias.

$$R_{m_i}^{max} = \max(r_{m_i}(k)) \tag{3.42}$$

Para finalizar o modelo, considera-se as seguintes suposições: o barramento nunca ficará ocioso, exceto se não existir mensagens prontas para serem transmitidas (liberadas) no *buffer* de transmissão dos nós da rede e os *deadlines* são críticos, ou seja, cada instância da mensagem deve ser finalizada antes do seu *deadline* absoluto. O conjunto com q mensagens ativadas periodicamente é escalonável, se e somente se, $R_{m_i}^{max} \leq D_{m_i}$, $\forall\ i = 1,...,q$. Assume-se que o nível de prioridade decresce de acordo com o aumento do número da mensagem, ou seja, a mensagem m_1 possui a maior prioridade (nível 1) e a mensagem m_q a menor prioridade (nível q).

Cálculo dos tempos de resposta no pior caso

O tempo de resposta no pior caso de uma mensagem m_i é derivado da Equação 3.43.

$$R_{m_i}^{max} = J_{m_i} + W_{m_i} + C_{m_i} \tag{3.43}$$

Onde W_{m_i}, chamado de atraso de enfileiramento, é o tempo máximo que a mensagem m_i poderá esperar no *buffer* de transmissão do controlador antes de iniciar a transmissão, devido outras mensagens estarem ocupando o barramento.

O atraso de enfileiramento consiste de dois tipos de atraso: atraso de bloqueio, originado por uma mensagem com menor prioridade, que poderá estar no processo de transmissão quando a mensagem m_i foi armazenada no *buffer* de transmissão do controlador; e o atraso de interferência, originado pelas mensagens com maior prioridade, que poderão ser transmitidas antes da mensagem m_i.

O máximo atraso de bloqueio ocorre quando a maior mensagem com menor prioridade inicia a transmissão imediatamente antes da mensagem m_i ser armazenada no *buffer* de transmissão. A mensagem m_i deverá esperar até que o barramento esteja livre, antes de entrar no processo de arbitragem. O máximo atraso de bloqueio (B_{m_i}) é determinado pela Equação 3.44, onde $lp(i)$ corresponde ao conjunto de mensagens com menor prioridade que a mensagem m_i.

$$B_{m_i} = \max_{\forall j \in lp(i)} (C_{m_j}) \tag{3.44}$$

A determinação do atraso de enfileiramento de uma mensagem m_i é baseada na definição do período ocupado com nível de prioridade - i (LEHOCZKY, 1990) e (HARBOUR; KLEIN; LEHOCZKY, 1991). Esse período é definido por um intervalo semi-aberto à direita, que inicia em algum instante t^s quando uma mensagem com prioridade i ou maior é enfileirada no *buffer* de transmissão do seu respectivo controlador CAN, e não existem mensagens com prioridade i ou maior esperando para serem transmitidas, as quais foram enfileiradas antes do instante t^s. O período é finalizado no instante t^e quando o barramento torna-se livre e não existem mensagens com prioridade i ou maior esperando para serem transmitidas, as quais foram enfileiradas antes do instante t^e. Durante esse intervalo nenhuma mensagem com prioridade menor que i é capaz de vencer a arbitragem e ser transmitida. O atraso de enfileiramento no pior caso para uma mensagem m_i com prioridade i, ocorre para alguma instância dessa mensagem enfileirada durante o período ocupado que inicia imediatamente após o início da transmissão da maior mensagem com menor prioridade que i. O maior período ocupado com nível de prioridade - i inicia no chamado instante crítico (LIU; LAYLAND, 1973), instante onde a mensagem m_i é armazenada simultaneamente com todas as mensagens com maior prioridade, em seus respectivos *buffers* de transmissão.

O comprimento t_{m_i} do maior período ocupado com nível de prioridade - i é determinado pela seguinte relação recursiva (3.45), onde $hep(i)$ é o conjunto de mensagens com prioridade maior ou igual à mensagem m_i.

$$t_{m_i}^{(x+1)} = B_{m_i} + \sum_{\forall j \in hep(i)} \left\lceil \frac{t_{m_i}^{(x)} + J_{m_j}}{T_{m_j}} \right\rceil C_{m_j} \tag{3.45}$$

O procedimento de iteração começa com um valor inicial $t_{m_i}^{(0)} = C_{m_i}$, e é finalizado quando $t_{m_i}^{(x+1)} = t_{m_i}^{(x)}$. Devido o lado direito da Equação 3.45 ser uma função monotônica não-decrescente de t_{m_i}, a relação de recorrência convergirá se o fator de utilização $U_m(hep(i))$ para mensagens com prioridade maior ou igual a m_i for menor ou igual a 1 (Equação 3.46).

$$U_m(hep(i)) = \sum_{\forall j \in hep(i)} \left(\frac{C_{m_j}}{T_{m_j}} \right) \leq 1 \tag{3.46}$$

Para determinar o maior atraso de enfileiramento de uma mensagem m_i é necessário calcular e verificar qual é o maior atraso de todas as suas instâncias enfileiradas e transmitidas durante o maior período ocupado com nível de prioridade - i. O número de instâncias Q_{m_i} é derivado de (3.47).

$$Q_{m_i} = \left\lceil \frac{t_{m_i} + J_{m_i}}{T_{m_i}} \right\rceil \tag{3.47}$$

Na análise seguinte, a variável k será usada para representar uma instância da mensagem m_i. A primeira instância do período ocupado corresponde a $k = 0$, e a última instância, $k = Q_{m_i} - 1$. O atraso de enfileiramento da instância k da mensagem m_i é determinado pela Equação 3.48, onde $hp(i)$ corresponde ao conjunto de mensagens com prioridade maior que a mensagem m_i e τ_{bit} (tempo de *bit*) representa as diferenças de tempo para o início da arbitração nos diferentes nós, devido aos atrasos de propagação e tolerâncias do protocolo.

$$W_{m_i}^{(x+1)}(k) = B_{m_i} + kC_{m_i} + \sum_{\forall j \in hp(i)} \left\lceil \frac{W_{m_i}^{(x)}(k) + J_{m_j} + \tau_{bit}}{T_{m_j}} \right\rceil C_{m_j} \tag{3.48}$$

A relação de recorrência começa com um valor de $W_{m_i}^{(0)}(k) = B_{m_i} + kC_{m_i}$, e termina quando $W_{m_i}^{(x+1)}(k) = W_{m_i}^{(x)}(k)$ ou $J_{m_i} + W_{m_i}^{(x+1)}(k) - kT_{m_i} + C_{m_i} > D_{m_i}$, onde no último caso, a mensagem é não-escalonável. A relação de recorrência sempre convergirá se a condição do fator de utilização do sistema $U_m = \sum_{i=1}^{q} \left(\frac{C_{m_i}}{T_{m_i}} \right) \leq 1$ for satisfeita (SPURI, 1996). Onde q equivale ao número de mensagens do sistema.

A instância k da mensagem m_i ocorre no instante $kT_{m_i} - J_{m_i}$ relativo ao início do período ocupado. Então, o tempo de resposta da instância k da mensagem m_i é determinado pela Equação 3.49.

$$R_{m_i}(k) = J_{m_i} + W_{m_i}(k) - kT_{m_i} + C_{m_i} \tag{3.49}$$

Observe que o termo W_{m_i} em (3.43) equivale ao termo $W_{m_i}(k) - kT_{m_i}$ em (3.49). Finalmente, o tempo de resposta no pior caso da mensagem m_i é derivado da Equação 3.50.

$$R_{m_i}^{max} = \max_{k=0...Q_{m_i}-1}(R_{m_i}(k)) \tag{3.50}$$

Comentários

A análise apresentada na seção anterior para determinar os tempos de resposta no pior caso de mensagens CAN periódicas pode ser facilmente adaptada para determinar os tempos de resposta fim a fim de malhas de controle nas arquiteturas representadas pelas Figuras 2.5(a) e (b) do Capítulo 2. Para a arquitetura representada pela Figura 2.5(a), o *jitter* de liberação seria constante e igual a soma do tempo de computação das tarefas de aquisição e controle. Além disso, o tempo de computação da tarefa de atuação deverá ser somado com o tempo de resposta no pior caso (Equação 3.50). Uma adaptação semelhante pode ser feita para a arquitetura representada na Figura 2.5(b).

3.5 Verificação de modelos clássica

A verificação de modelos clássica (CMC - *Classical Model-Checking*) (LARSEN; PETTERSSON; YI, 1995) é uma abordagem que usa algoritmos matemáticos executados automaticamente por uma ferramenta computacional para verificar a corretude de um sistema de acordo com requisitos especificados. Basicamente, o projetista dispõe como dados de entrada para a ferramenta, a descrição do modelo do sistema, que representa o possível comportamento do sistema real, e uma descrição da especificação dos requisitos, que representa o comportamento desejado. Então, a ferramenta verifica se as propriedades requisitadas são satisfeitas para um determinado estado inicial do modelo, e fornece tais resultados para o projetista. Uma propriedade muito importante da verificação de modelos é a sua capacidade de fornecer um contra-exemplo, que indica a situação na qual os requisitos foram violados. Desse modo, o projetista pode usar essa informação para corrigir erros no sistema. Um diagrama de blocos com a metodologia da verificação de modelos está representada na Figura 3.13.

Os algoritmos de verificação de modelos são baseados na pesquisa exaustiva do espaço de estados. Desse modo, um modelo de estados finito é requisitado e o sistema pode ser verificado apenas para um conjunto fixado de parâmetros. O problema dessa técnica é a explosão do espaço de estados, que pode resultar na necessidade de alocar uma quantidade de memória

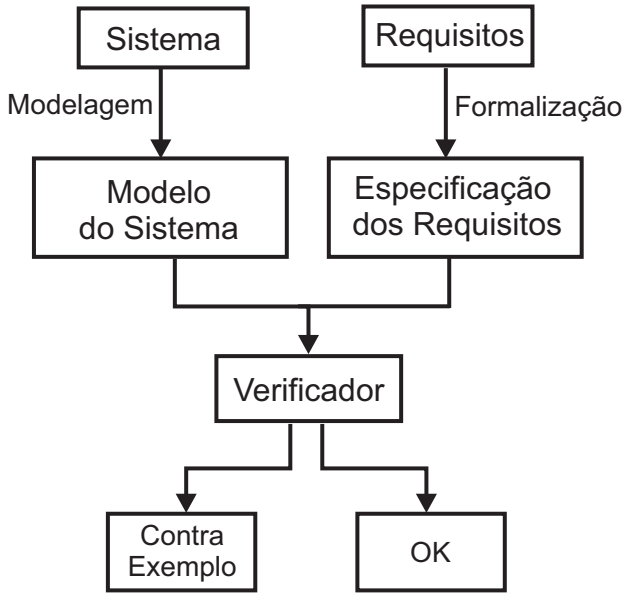


Figura 3.13: Diagrama de blocos com a metodologia da verificação de modelos.

superior a disponível pelo computador que executa a ferramenta. Portanto, é crucial construir modelos compactos. Essa abordagem será utilizada para estudar sistemas modelados em autômatos temporizados.

3.5.1 Autômatos temporizados

A teoria de autômatos temporizados foi desenvolvida por Alur e Dill ((ALUR; DILL, 1990) e (ALUR; DILL, 1994)) para modelar sistemas em tempo real. Uma versão simplificada, chamada de autômatos seguros temporizados (HENZINGER et al., 1994), foi introduzida para especificar propriedades de progresso usando condições chamadas de invariantes locais. Devido a sua simplicidade, autômatos seguros temporizados (doravante chamados de autômatos temporizados) foram adotados em várias ferramentas de verificação tais como, UPPAAL. A seguir, serão apresentados algumas definições de autômatos temporizados. Essas definições foram extraídas de Berhmann, David e Larsen (2004) e Bengtsson e Yi (2004).

Em sua essência, um autômato temporizado é um autômato de estados finito (que é um grafo contendo um conjunto finito de lugares e um conjunto finito de bordas rotuladas) que considera quantitativamente o tempo na transição dos estados. A progressão do tempo ocorre de acordo com um conjunto finito de relógios sincronizados na mesma taxa, onde os quais usam o modelo de tempo denso, ou seja, os valores dos relógios são números reais. Os valores dos relógios podem ser comparados com números inteiros e podem ser resetados quando uma transição for realizada (BENGTSSON; YI, 2004).

Sintaxe de autômatos temporizados

Considere um conjunto de relógios X com um conjunto de restrições $B(X)$. As restrições dos relógios consistem de equações definidas por,

$$\alpha ::= x \sim c \mid x - y \sim c \mid \neg\alpha \mid (\alpha \wedge \alpha)$$

Onde: $x, y \in X$; $c \in \mathbb{N}$; $\neg$ é o operador lógico proposicional que significa negação (não); $\wedge$ é o operador lógico proposicional e[9], e $\sim$ é um dos seguintes símbolos $\{\leq, \geq, =, <, >\}$.

Definição 7 (Autômatos temporizados) *Um autômato temporizado é uma 6-tupla $\{L_e, l_0, X, A_c, E, I\}$, onde: L_e é um conjunto finito de lugares (estados); l_0 é o lugar inicial; X é o conjunto de relógios; A_c é um conjunto de ações; $E \subseteq L_e \times A_c \times B(X) \times 2^X \times L_e$ é um conjunto de bordas*[10]*. Desse modo, pode-se representar a transição de um lugar l para um lugar l', considerando o conjunto $\{l, a, g, r, l'\} \in E$, como $l \xrightarrow{a,g,r} l'$, onde: a é uma ação (canal de sincronização), que fornece sincronização entre autômatos, possibilitando formar uma rede de autômatos temporizados; g é uma restrição de tempo chamada de guarda e r é um conjunto de relógios a serem resetados; e $I : L_e \rightarrow B(X)$ é uma função que atribui a cada lugar L_e restrições de tempo $B(X)$, chamadas de invariantes de estado. Essas restrições devem ser satisfeitas quando o sistema estiver nesses lugares.*

Semântica de autômatos temporizados

Uma avaliação do relógio é uma função $u : X \rightarrow \mathbb{R}_{\geq 0}$ a partir do conjunto de relógios X para os reais não-negativos. $\mathbb{R}^X$ denota o conjunto de todas as avaliações dos relógios. A avaliação inicial $u_0(x) = 0$ para todo $x \in X$. Considera-se na notação, guardas e invariantes no conjunto de avaliações dos relógios. Então, $u \in g$ significa que os valores dos relógios sob avaliação u satisfazem o guarda $g \in B(X)$, e $u \in I(l)$ significa que os valores dos relógio sob avaliação u satisfazem a invariante no lugar l.

A semântica é definida como um sistema de transição, onde um estado consiste do lugar atual (l) e dos valores atuais dos relógios (u), ou seja, do par (l, u). Existem dois tipos de transição entre estados: transição por ação (ocorre quando uma borda habilitada é executada) e transição por tempo (ocorre quando o lugar permanece o mesmo e o valor do relógio é incrementado).

Definição 8 (Semântica de Autômatos Temporizados) *Considere um autômato temporizado (Definição 7). A semântica é um sistema de transição rotulada $\langle S_e, s_{e_0}, \rightarrow \rangle$, onde*

[9] Por exemplo, considere que n é um número natural, $n < 4 \wedge n > 2 \Leftrightarrow n = 3$.

[10] $\subseteq$ é um operador da teoria de conjuntos. $A \subseteq B$ significa que cada elemento de A é também elemento de B.

$S_e \subseteq L_e \times \mathbb{R}^C$ é o conjunto de estados, $s_{e_0} = (l_0, u_0)$ é o estado inicial, e $\rightarrow \subseteq S_e \times \{\mathbb{R}_{\geq 0} \cup A_c\} \times S_e$ é a relação de transição tal que[11]*:*

- $(l, u) \xrightarrow{d} (l, u + d)$ *se para todo* $d' : 0 \leq d' \leq d \Rightarrow u + d' \in I(l)$, *e*
- $(l, u) \xrightarrow{a} (l', u')$ *se existe* $l \xrightarrow{a,g,r} l'$, $u \in g$, $u' = [r \mapsto 0]u$ *e* $u' \in I(l)$,

onde $d \in \mathbb{R}_{\geq 0}$, $u + d$ mapeia cada relógio x em C para o valor $u(x) + d$, e $[r \mapsto 0]u$ denota a avaliação do relógio que mapeia cada relógio em r para 0 e mantém u para o conjunto de relógios $X \backslash r$[12]*.*

Redes de autômatos temporizados

Autômatos temporizados podem ser compostos por uma rede de autômatos temporizados através de um conjunto comum de relógios e ações, consistindo de n autômatos temporizados $\mathcal{A}_i = \{L_{e_i}, l_i^0, X, A_c, E_i, I_i\}$, onde $1 \leq i \leq n$. Um vetor de lugares é um vetor $\bar{l} = (l_1,...,l_n)$. As funções invariantes são compostas em uma função comum por meio de vetores de lugares $I(\bar{l}) = \wedge_i I_i(l_i)$. Usaremos $\bar{l}[l_i' \backslash l_i]$ para denotar o vetor onde o ith elemento l_i de $\bar{l}$ foi substituído por l_i'.

Definição 9 (Semântica da Rede de Autômatos Temporizados) *Considere uma rede de autômatos temporizados $\mathcal{A}_i = \{L_{e_i}, l_i^0, X, A_c, E_i, I_i\}$ com n autômatos temporizados. Seja $\bar{l}_0 = (l_0^1,...,l_0^n)$ o vetor lugar inicial. A semântica é definida como um sistema de transição $\langle S_e, s_{e_0}, \rightarrow \rangle$, onde $S_e = (L_{e_1} \times ... \times L_{e_n}) \times \mathbb{R}^X$ é o conjunto de estados, $s_{e_0} = (\bar{l}_0, u_0)$ é o estado inicial, e $\rightarrow \subseteq S_e \times S_e$ é a relação de transição definida por:*

- $(\bar{l}, u) \rightarrow (\bar{l}, u + d)$ *se para todo* $d' : 0 \leq d' \leq d \Rightarrow u + d' \in I(\bar{l})$*;*
- $(\bar{l}, u) \rightarrow (\bar{l}[l_i' \backslash l_i], u')$ *se existe* $l_i \xrightarrow{a,g,r} l_i'$ *tal que* $u \in g$, $u' = [r \mapsto 0]u$ *e* $u' \in I(\bar{l})$*;*
- $(\bar{l}, u) \rightarrow (\bar{l}[l_j' \backslash l_j, l_i' \backslash l_i], u')$ *se existe* $l_i \xrightarrow{c?,g_i,r_i} l_i'$ *e* $l_j \xrightarrow{c?,g_j,r_j} l_j'$ *tal que* $u \in (g_i \wedge g_j)$, $u' = [r_i \cup r_j \mapsto 0]u$ *e* $u' \in I(\bar{l})$.

[11] $\cup$ é o operador da teoria de conjuntos que significa união. Por exemplo, $A \cup B$ é um conjunto que contém todos os elementos de A e B.

[12] $\backslash$ é o operador da teoria de conjuntos que significa exceto. Por exemplo, $A \backslash B$ é o conjunto que contém todos os elementos de A que não estão em B

Representação simbólica do espaço de estados

Uma vez que os relógios são variáveis que assumem valores reais, o espaço de estados de um autômato temporizado é infinito. Para possibilitar a verificação do modelo, abstrações específicas tais como, regiões e zonas, foram definidas para resolver esse problema, gerando um modelo de estados finito. Especificamente, a ferramenta UPPAAL usa o conceito de zonas normalizadas que são representadas e armazenadas em memórias como DBMs (*Difference Bound Matrices*). Detalhes sobre esses conceitos são apresentados em Bengtsson e Yi (2004).

3.6 Verificação de modelos estatísticos

A verificação de modelos estatísticos (SMC) (SEN; VISWANATHAN; AGHA, 2004) e (YOUNES, 2005a) é uma abordagem que foi proposta como uma alternativa para evitar uma exploração exaustiva do espaço de estados do modelo. A metodologia SMC é semelhante a usada na verificação clássica (Ver Figura 3.13). Entretanto, o verificador é constituído por um conjunto de algoritmos: algoritmo para gerar execuções aleatórias de acordo com uma semântica estocástica de um modelo; algoritmo para monitorar as execuções aleatórias com relação a algumas propriedades, e algoritmos estatísticos para dar uma estimativa do resultado com alguma confidência. A ideia básica da abordagem SMC é monitorar um número de execuções do modelo estocástico e, a partir dessas execuções, usar algoritmos estatísticos (incluindo testes de hipótese sequencial ou simulação de Monte Carlo) para verificar e estimar o comportamento do sistema com algum nível de confidência (DAVID et al., 2011a). Essa abordagem será usada para estudar sistemas modelados em autômatos temporizados com custo (PTA - *Priced Timed Automata*). PTAs são autômatos temporizados que podem ter diferentes taxas de relógios em diferentes lugares. A seguir, serão apresentados algumas definições de PTAS e redes de PTAS (NPTAS). Essas definições foram extraídas de David et al. (2011a).

3.6.1 Autômatos temporizados com custo

Seja X um conjunto finito de variáveis chamadas de relógios. Os relógios podem ter diferentes taxas em diferentes lugares. Uma avaliação de relógio sobre o conjunto X é um mapeamento $\nu : X \to \mathbb{R}_{\geq 0}$, onde $\mathbb{R}_{\geq 0}$ é o conjunto dos reais não-negativos. Denota-se por $\mathbb{R}^X_{\geq 0}$ o conjunto de avaliações dos relógios de X. Seja $\bar{r} : X \to \mathbb{N}$ um vetor taxa, que atribui a cada relógio de X uma taxa, sendo $\mathbb{N} = \{1, 2, 3,...\}$. Então, para $\nu \in \mathbb{R}^X_{\geq 0}$ e um atraso $d \in \mathbb{R}_{\geq 0}$, $\nu + \bar{r} \cdot d$ é a avaliação de relógio definida por $(\nu + \bar{r} \cdot d)(x) = \nu(x) + \bar{r}(x) \cdot d$ para algum relógio $x \in X$. Denota-se $\mathbb{N}^X$ como o conjunto de todos os vetores taxa. Se $Y \subseteq X$, a avaliação $\nu[Y]$ é a avaliação que atribui 0 quando $x \in Y$, e $\nu(x)$ quando $x \notin Y$. Um guarda com limite superior (limite inferior) sobre X é uma conjunção finita de simples limites de relógio da forma $x \sim n$,

onde $x \in X$, $n \in \mathbb{N}$, e $\sim \in \{<, \leq\} (\sim \in \{>, \geq\})$. Denota-se os guardas com limite superior (limite inferior) sobre X por $\mathcal{U}(X)$ $(\mathcal{L}(X))$, e escreve-se $\nu \models g$ sempre que ν for uma avaliação de relógio satisfazendo o guarda g. Seja $\Sigma = \Sigma_i \biguplus \Sigma_o$ uma dijunção dos conjuntos das ações de entrada e saída.

Definição 10 (Autômato Temporizado com Custo) *Um autômato temporizado com custo é uma 7-tupla $\mathcal{A} = (L_e, \ell_0, X, \Sigma, E, R, I)$ onde: L_e é um conjunto finito de lugares; $\ell_0 \in L_e$ é o lugar inicial; X é um conjunto finito de relógios; $\Sigma = \Sigma_i \biguplus \Sigma_o$ é um conjunto finito de ações particionados em entradas (Σ_i) e saídas (Σ_o); $E \subseteq L_e \times \mathcal{L}_\urcorner(X) \times \Sigma \times 2^X \times L_e$ é um conjunto finito de bordas; $R : L_e \to \mathbb{N}^X$ atribui um vetor taxa a cada lugar e $I : L_e \to \mathcal{U}(X)$ atribui uma invariante a cada lugar.*

A semântica de redes de PTAs (NPTAs) é um sistema de transição rotulado temporizado onde os estados são pares $(\ell, \nu) \in L_e \times \mathbb{R}^X_{\geq 0}$ com $\nu \models I(\ell)$ e as transições são por atrasos $(\ell, \nu) \xrightarrow{d} (\ell', \nu')$ com $d \in \mathbb{R}_{\geq 0}$ e $\nu' = \nu + R(\ell) \cdot d$, ou discretas $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ se existir uma borda (ℓ, g, a, Y, ℓ') tal que $\nu \models g$ e $\nu' = \nu[Y]$. Escreve-se $(\ell, \nu) \rightsquigarrow (\ell', \nu')$ se existir uma sequência finita de transições de atrasos e transições discretas de (ℓ, ν) para (ℓ', ν').

Assume-se NPTAs com:

- Entradas habilitadas - para todos os estados (ℓ, ν) e ações de entrada $\iota \in \Sigma_i$, para todos os autômatos temporizados j, existe uma borda $(\ell^j, g, \iota, Y, \ell^{j'})$ tal que $\nu \models g$;
- Determinísticos - para todos os estados (ℓ, ν) e ações $a \in \Sigma$, sempre que $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ e $(\ell, \nu) \xrightarrow{a} (\ell'', \nu'')$, $\ell' = \ell''$ e $\nu' = \nu''$, e
- Sem zenos - o tempo sempre diverge. Além disso, diferentes autômatos sincronizam em correspondência a entradas e saídas como uma sincronização *broadcast* padrão.

Uma vez que $\mathcal{A}^j = (L^j_e, X^j, \Sigma^j, E^j, R^j, I^j) (j = 1,...,n)$ são NPTAs, considera-se que são composáveis em uma rede fechada, se e somente se, seus conjuntos de relógios forem disjuntos $(X^j \cap X^k = \emptyset)$ quando $j \neq k$, se possuem o mesmo conjunto de ação $(\Sigma = \Sigma^j = \Sigma^k)$ para todo j, k, e seus conjuntos de ações de saída fornecem uma partição de Σ $(\Sigma^j_o \cap \Sigma^k_o = \emptyset$ para todo $j \neq k$, e $\Sigma = \cup_j \Sigma^j_o)$. Para $a \in \Sigma$, denota-se por $c(a)$ o único j com $a \in \Sigma^j$.

Definição 11 (Composabilidade de NPTAs) *Sejam as NPTAs $\mathcal{A}^j = (L^j_e, X^j, \Sigma, E^j,$ $R^j, I^j)$ $(j = 1,...,n)$ composáveis. Suas composições $(\mathcal{A}_1|...|\mathcal{A}_n)$ resultam na NPTA $\mathcal{A} = (L_e, X, \Sigma, E, R, I)$, onde: $L_e = \times_j L^j_e$; $X = \cup_j X^j$; $R(\ell)(x) = R^j(\ell^j)(x)$ quando $x \in X^j$; $I(\ell) = \cap_j I(\ell^j)$, e $(\ell, \cap_j g_j, a, \cup_j r_j, \ell') \in E$ sempre que $(\ell_j, g_j, a, r_j, l'_j) \in E^j$ para todo $j = 1,...,n$.*

Semântica probabilística de NPTAs

Seja $\mathcal{A}^j = (L_e^j, X^j, \Sigma, E^j, R^j, I^j)(j = 1,...,n)$ uma coleção de NPTAs composáveis. Sob as suposições de entradas habilitadas, disjunção de conjuntos de relógios e ações de saídas, os estados da NPTA composta $\mathbf{A} = (\mathcal{A}_1|...|\mathcal{A}_n)$ podem ser vistas como tuplas $s = (s_1,...,s_n)$ onde s_j é um estado de $\mathcal{A}^j$, ou seja, da forma (ℓ, ν) onde $\ell \in L_e^j$ e $\nu \in \mathbb{R}_{\geq 0}^{X^j}$. A semântica é baseada no princípio de independência entre os componentes. Cada componente decide repetidamente, baseado em uma determinada função densidade de atraso e em uma função probabilidade de saída, quanto atrasar antes de produzir a saída e qual saída enviar nesse instante. Na corrida entre os componentes, o resultado será determinado pelo componente que escolheu a saída após o atraso mínimo. Então, a saída é enviada e todos os outros componentes poderão consequentemente mudar de estado.

Considere um componente $\mathcal{A}^j$ e denote St^j como o conjunto de estados correspondente. Para cada estado $s = (\ell, \nu)$ de $\mathcal{A}^j$, tem-se distribuições de probabilidade para os atrasos e as saídas. A função densidade de atraso μ_s sobre os atrasos em $\mathbb{R}_{\geq 0}$ serão definidas com distribuições uniforme ou exponencial, dependendo da invariante do lugar ℓ. Denota-se por: E_ℓ a disjunção de guardas g tal que $(l, g, o, -, -) \in E^j$ para alguma saída o; $d(\ell, \nu)$ o atraso mínimo antes de habilitar uma saída, ou seja, $d(\ell, \nu) = \inf\{d \in \mathbb{R}_{\geq 0} : \nu + R^j \cdot d \models E_\ell\}$, e $D(\ell, \nu)$ o atraso máximo, ou seja, $D(\ell, \nu) = \sup\{d \in \mathbb{R}_{\geq 0} : \nu + R^j \cdot d \models I^j(\ell)\}$. Se $D(\ell, \nu) < \infty$, a função densidade de atraso μ_s é uma distribuição uniforme em $[d(\ell, \nu), D(\ell, \nu)]$. Caso contrário, ou seja, se $I^j(\ell)$ não atribuir um limite superior nos atrasos possíveis de s, a função densidade de atraso μ_s é uma distribuição exponencial com uma taxa $P(\ell)$, onde $P : L_e^j \to \mathbb{R}_{\geq 0}$ é um componente adicional de taxa da distribuição acrescentado ao NPTA $\mathcal{A}^j$. Para cada estado $s = (\ell, \nu)$, a função de probabilidade de saída γ_s sobre Σ_o^j é a distribuição uniforme sobre o conjunto $\{o : (\ell, g, o, -, -) \in E^j \wedge \nu \models g\}$ sempre que esse conjunto não for vazio (caso contrário uma distribuição de peso pode ser especificada e usada.). Denota-se por s^o o estado após a saída de o. Similarmente, para cada estado s e alguma ação de entrada ι, denota-se por s^ι o estado após ter recebido a entrada ι.

3.7 Conclusões

Neste capítulo apresentou-se a fundamentação teórica que será usada neste trabalho. Iniciou-se o capítulo apresentando as suposições e tecnologias que serão adotadas. Em seguida, foi apresentada a teoria clássica de sistemas de controle, o controlador PID e os teoremas para análise de estabilidade de sistemas com atrasos de controle variantes. Depois, apresentou-se a teoria de sistemas em tempo real com ênfase no escalonamento preemptivo com prioridade fixa e o escalonamento de mensagens em uma rede CAN. Finalizou-se o capítulo apresentando as abordagem verificação de modelos clássica e estatística e as definições de autômatos temporizados.

Capítulo 4

Modelagem formal e análise dos tempos de resposta da rede CAN

Neste Capítulo apresentam-se modelos em autômatos temporizados e autômatos temporizados com custo da rede CAN. Esses modelos foram desenvolvidos na ferramenta UPPAAL com o objetivo de determinar os tempos de resposta de mensagens CAN periódicas utilizando as abordagens verificação de modelos simbólica ou clássica (CMC - *Classical Model-Checking*) e verificação de modelos estatísticos (SMC - *Statistical Model-Checking*). UPPAAL é uma ferramenta para modelagem, simulação e verificação de modelos de sistemas em tempo real baseado na teoria de autômatos temporizados (BERHMANN; DAVID; LARSEN, 2004). Recentemente, UPPAAL foi estendida para realizar a verificação de modelos estatísticos usando como entrada modelos em autômatos temporizados com custo (BULYCHEV et al., 2012b) e autômatos híbridos estocásticos (DAVID et al., 2012). Mais detalhes sobre a ferramenta UPPAAL estão descritos no apêndice B.

Os modelos apresentados neste Capítulo consideram características reais de uma rede CAN, que podem influenciar diretamente na determinação dos tempos de resposta das mensagens. Especificamente, será considerado que os nós poderão transmitir mensagens periódicas com *offset* estático ou dinâmico, ter relógios ideais ou reais e serem conectados a um barramento ideal (sem erros) ou real (com erros). Assume-se que os possíveis erros no barramento são causados apenas por fontes externas tais como, interferências eletromagnéticas. A partir dessas suposições, é possível desenvolver oito modelos, conforme apresentado na Tabela 4.1. Desses possíveis modelos, apenas os modelos 1, 2, 3 e 5 serão desenvolvidos. Será justificado, a partir dos modelos 1 e 2, a não necessidade de desenvolver o modelo 4 para o caso em que as mensagens são periódicas e os *offsets* são dinâmicos. A teoria básica para desenvolver o modelo 6 será apresentada a partir dos modelos 2 e 5. Mas, esse modelo não será desenvolvido. Enquanto que os modelos 7 e 8 não serão desenvolvidos por limitação da ferramenta UPPAAL. Os trabalhos relacionados a cada modelo serão apresentados e analisados conforme a necessidade. Na literatura, apenas o

modelo 1 foi desenvolvido (KRAKORA, 2008) e (WASZNIOWSKY; KRAKORA; HANZALEK, 2009).

Tabela 4.1: Combinação com os possíveis modelos da rede CAN.

Modelo	***Offset***	**Relógio**	**Barramento**
1	Dinâmico	Ideal	Ideal
2	Dinâmico	Real	Ideal
3	Dinâmico	Ideal	Real
4	Dinâmico	Real	Real
5	Estático	Ideal	Ideal
6	Estático	Real	Ideal
7	Estático	Ideal	Real
8	Estático	Real	Real

Estudos de caso serão realizados para cada modelo, de modo a verificar se os requisitos de tempo serão cumpridos. Especificamente, os modelos 1 e 3 serão testados utilizando uma plataforma experimental. Esses dois modelos serão estendidos para modelar NCSs CAN no Capítulo 6 deste trabalho. Na seção seguinte será apresentada uma notação e suposições da rede CAN utilizadas ao longo deste Capítulo.

4.1 Notação e modelo da rede CAN

Considere uma rede CAN com um único barramento e n nós. O barramento pode ser considerado ideal (sem erros) ou real (com erros). Cada nó N_i, onde i assume valores de 0 a $n-1$, possui um relógio x_i e transmite uma única mensagem periódica m_i, que consiste de um número infinito de instâncias $k \in \mathbb{Z}^+ = \{0,1,2,3,...\}$. O relógio x_i pode ser ideal (taxa de variação constante e igual a 1) ou real com uma taxa de variação $\hat{\rho}_{x_i}$ no intervalo $[\hat{\rho}_{x_i}^{min}, \hat{\rho}_{x_i}^{max}]$.

Cada mensagem possui uma tarefa periódica relacionada τ_{m_i} com período T_{m_i}, que armazena a no *buffer* de transmissão do controlador CAN do seu respectivo nó. A primeira instância dessa tarefa é iniciada em um instante chamado de *offset* O_{m_i}. O *offset* pode ser estático (O_{m_i} assume algum valor fixo no intervalo $[0, T_{m_i}]$) ou dinâmico (O_{m_i} assume algum valor arbitrário no intervalo $[0, T_{m_i}]$). Em ambos os casos, uma instância k de uma mensagem m_i é ativada a cada instante $a_{m_i}(k) = O_{m_i} + kT_{m_i}$ e liberada (pronta para transmissão) assim que a tarefa relacionada armazená-la no *buffer* de transmissão. Essa tarefa gasta uma quantidade de tempo igual a C_i, chamado de atraso de armazenamento ou *jitter* de liberação da mensagem, que pode assumir algum valor fixo ou variante no intervalo $[C_i^{min}, C_i^{max}]$.

Além disso, toda mensagem possui: um tempo de transmissão C_{m_i} constante ou variante no intervalo $[C_{m_i}^{min}, C_{m_i}^{max}]$, dependente do tempo de transmissão de *bit* (τ_{bit}) e do número de

bytes de dados (b_{m_i}); um *deadline* D_{m_i}, onde $D_{m_i} \leq T_{m_i}$, e um nível de prioridade fixa $p_{m_i} = i$. Assume-se que o nível de prioridade decresce de acordo com o aumento do número da mensagem, ou seja, a mensagem m_0 possui a maior prioridade $p_{m_0} = 0$ e a mensagem m_{n-1} a menor prioridade $p_{m_{n-1}} = n - 1$. A política de atribuição de prioridade *deadline* monotônico será usada em todos os estudos de caso apresentados neste Capítulo.

Os tempos de resposta no melhor e pior caso da mensagem m_i serão denotados, respectivamente, por $R_{m_i}^{min}$ (Equação 4.1) e $R_{m_i}^{max}$ (Equação 4.2), sendo $r_{m_i}(k)$ o tempo de resposta da instância k da mensagem m_i. A letra i, subscrita em vários símbolos e atributos, será em alguns casos omitida nos modelos apresentados nas seções seguintes. Além disso, a letra i poderá ser usada para representar o nó, a mensagem transmitida por esse nó e a prioridade da mensagem transmitida por esse nó, uma vez que cada nó transmite apenas uma única mensagem periódica com prioridade fixa. Por exemplo, o nó 0 (N_0) transmite a mensagem 0 (m_0) com prioridade 0 ($p_{m_0} = 0$).

$$R_{m_i}^{min} = \min(r_{m_i}(k)) \tag{4.1}$$

$$R_{m_i}^{max} = \max(r_{m_i}(k)) \leq D_{m_i} \tag{4.2}$$

4.2 Modelo formal da rede CAN com *offset* dinâmico, relógios e barramento ideais

Nesta seção apresenta-se o modelo 1 (Ver Tabela 4.1), onde os *offsets* das mensagens são dinâmicos, o relógio de cada nó CAN é ideal e o barramento é livre de erros (barramento ideal).

4.2.1 Trabalhos relacionados

Os primeiros trabalhos com o objetivo de determinar os tempos de resposta no pior caso de mensagens CAN periódicas, com *offsets* dinâmicos[1] e relógios e barramentos ideais foram desenvolvidos por Tindell e Burns (1994b), Tindell e Burns (1994a) e Tindell, Burns e Wellings (1995), usando uma análise analítica baseada na teoria de escalonamento de tarefas preemptivas com prioridades fixas para sistemas com um único processador (AUDSLEY et al., 1993). A partir dessa análise, foi possível verificar se as mensagens em uma rede CAN satisfazem seus requisitos de tempo. Davis et al. (2007) provaram que essa análise é incompleta e resulta em tempos

[1]Na realidade, o modelo adotado por Tindell em seus trabalhos, considera que não existe controle sobre a ativação da primeira instância da mensagem, de modo a possibilitar a ocorrência do instante crítico (LIU; LAYLAND, 1973). Entretanto, podemos considerar que o uso de *offsets* dinâmicos gerarão instantes de ativação que conduzirão ao instante crítico.

de resposta otimistas. Então, Davis et al. (2007) apresentaram uma análise correta e descreveram como a nova análise pode ser estendida para considerar apropriadamente modelos de erro de transmissão. Zeng et al. (2009) desenvolveram uma análise estocástica para determinar os tempos de resposta de mensagens CAN periódicas.

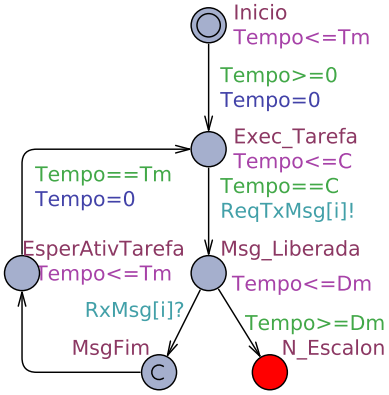
Usando a abordagem verificação de modelos, Krakora (2008) e Waszniowsky, Krakora e Hanzalek (2009) desenvolveram modelos em autômatos temporizados para determinar os tempos de resposta no pior caso de mensagens em uma rede CAN. Nesta seção, apresentaremos um modelo mais compacto baseado nesses modelos, capaz de ser estendido para considerar outros comportamentos e, consequentemente, verificar mais propriedades do sistema.

4.2.2 Modelo em autômato temporizado

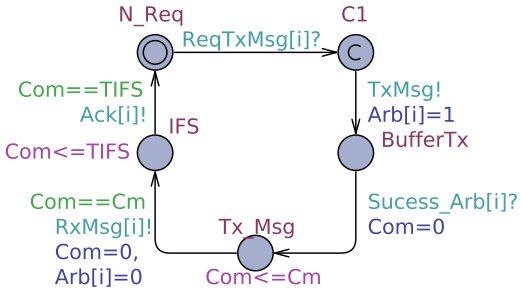
O modelo é constituído por três autômatos (Figura 4.1): tarefa, controlador CAN e arbitragem. Para cada mensagem do sistema, a ferramenta UPPAAL gera automaticamente um autômato tarefa e um autômato controlador CAN. Apenas um autômato arbitragem é gerado para o sistema. O código UPPAAL do modelo é apresentado na Seção C.1 do Apêndice C.

O autômato tarefa (Figura 4.1(a)) modela o comportamento temporal das atividades da tarefa τ_{m_i} relacionada a mensagem m_i. O autômato possui um relógio local *Tempo* usado para marcar o tempo de resposta e o valor dos seguintes atributos: *offset* O_{m_i}; período Tm; tempo de computação C e *deadline* Dm. A primeira instância da tarefa τ_{m_i} é liberada no lugar *Início* com um *offset* variando de 0 a Tm unidades de tempo. O *offset* é modelado pela invariante de estado $Tempo <= Tm$ e o guarda $Tempo >= 0$. As próximas instâncias serão liberadas periodicamente no lugar *EsperAtivTarefa*. Após o tempo de *offset*, a transição para o lugar *Exec_Tarefa* será liberada e o relógio $Tempo$ será zerado. Esse lugar representa a tarefa em execução durante um tempo de computação C, modelado pela invariante $Tempo <= C$ e pelo guarda $Tempo == C$. Quando o relógio $Tempo$ for igual a C unidades de tempo, a transição para o lugar *Msg_Liberada* será liberada e uma requisição para transmitir a mensagem m_i será enviada para o autômato controlador CAN (Figura 4.1(b)) usando o canal *ReqTxMsg[i]!*. O autômato permanecerá nesse lugar até o fim da transmissão da mensagem, que será comunicada pelo autômato controlador CAN por meio do canal *RxMsg[i]?*. Assim que a transmissão da mensagem for finalizada, a transição para o lugar *committed MsgFim* será liberada e, em seguida, ocorrerá a transição para o lugar *EsperAtivTarefa*, onde permanecerá até iniciar um novo período, modelado pela invariante $Tempo <= Tm$ e pelo guarda $Tempo == Tm$. Se o valor do relógio *Tempo* for maior ou igual ao *deadline* Dm, enquanto o autômato estiver no lugar *Msg_Liberada*, uma transição para o lugar *N_Escalon* ocorrerá, indicando que o *deadline* não foi satisfeito.

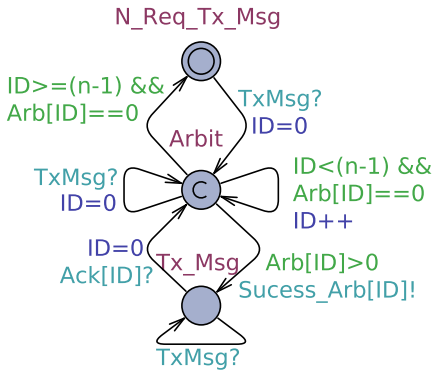
O autômato controlador CAN (Figura 4.1(b)) modela o comportamento temporal da mensagem de dados de acordo com a especificação do protocolo CAN. O autômato possui um



(a) Tarefa.



(b) Controlador CAN.



(c) Arbitragem.

Figura 4.1: Modelo em autômatos temporizados da rede CAN ideal com *offset* dinâmico.

relógio local *Com* usado para marcar o tempo de transmissão da mensagem de dados e o intervalo entre mensagens (intermissão). Inicialmente, o autômato controlador CAN está no lugar inicial *N_Req*. Esse lugar representa o estado em que o controlador CAN do nó não recebeu uma requisição de transmissão de mensagem. Quando o autômato receber uma requisição de transmissão de mensagem via o canal *ReqTxMsg[i]?* enviado pelo autômato tarefa, uma transição do lugar *N_Req* para o lugar committed *C1* e, imediatamente, uma transição para o lugar *BufferTx* será liberada, o sinal *TxMsg!* será enviado para o autômato arbitragem (Figura 4.1(c)) e o *array Arb[i]*, indexado com o valor do identificador da mensagem m_i, será ajustado para '1', indicando que a mensagem está pronta para entrar no processo de arbitragem. Assim que a mensagem vencer o processo de arbitragem, o autômato arbitragem sincronizará com o autômato controlador CAN via o canal *Sucess_Arb[i]?*, a transição para o lugar *Tx_Msg* será liberada e o relógio *Com* será zerado. O autômato permanecerá nesse lugar até o relógio Com marcar Cm unidades de tempo (modelado pela invariante de estado $Com <= Cm$ e o guarda $Com == Cm$), indicando que a mensagem foi transmitida com sucesso. Então, a transição para o lugar *IFS* será liberada, o sinal *RxMsg!* será enviado para o autômato tarefa e o relógio Com e a variável *Arb[i]* serão zerados. O lugar *IFS* representa a intermissão com duração de três tempos de *bit* ($TIFS = 3 * \tau_{bit}$). Assim que o relógio Com marcar três tempos de *bit*, a transição para o lugar committed *N_Req* será liberada e a informação de que o barramento está livre será enviada para o autômato arbitragem por meio do canal *Ack[i]!*.

O controle de acesso ao meio é modelado pelo autômato arbitragem (Figura 4.1(c)). O lugar inicial *N_Req_Tx_Msg* representa o estado onde não existe nenhuma mensagem pronta para ser transmitida. Assim que pelo menos um controlador CAN tiver alguma mensagem pronta para ser transmitida, o autômato receberá um sinal via o canal *TxMsg?* emitido pelo respectivo autômato controlador CAN. Então, a transição para o lugar *committed Arbit* será liberada e a variável *ID* será zerada. A variável *ID*, assim como a variável *i* (usada nos autômatos tarefa e controlador CAN) representam o identificador da mensagem m_i. No lugar *Arbit*, a mensagem com maior prioridade que requisitou uma transmissão será escolhida para ser transmitida. Para isso, o *array Arb[ID]* será verificado item por item, com o índice *ID* variando de 0 a *n-1*, onde *n* equivale ao número de identificadores (mensagens) do sistema. Os identificadores das mensagens são ordenados de forma crescente. A mensagem com maior prioridade possui *ID = 0*, e a mensagem com menor prioridade *ID = n-1*. Então, o primeiro elemento do *array Arb[ID]* com valor maior que 0 vencerá o processo de arbitragem, e um sinal via o canal *Sucess_Arb[ID]!* será enviado para o autômato controlador CAN que transmitirá a mensagem com identificador ID. O autômato permanecerá no lugar *Tx_Msg* até receber o sinal de reconhecimento via *Ack[ID]?*, indicando que a mensagem foi transmitida com sucesso e que o barramento já está livre novamente para iniciar um novo processo de arbitragem. Então, a variável *ID* será zerada e o autômato retornará para o lugar *Arbit*, onde verificará se existe alguma mensagem

pendente para ser transmitida. Caso não exista, o autômato retornará para o seu lugar inicial *N_Req_Tx_Msg*.

Do ponto de vista da rede, as partes do sistema que foram modelados podem ser representadas de modo simplificado pela Figura 4.2. Os três autômatos são representados por retângulos coloridos com os seus respectivos nomes (tarefa, controlador CAN e arbitragem). As conexões entre os autômatos são indicadas por setas rotuladas com o nome dos canais de sincronização (*RxMsg*, *ReqTxMsg*, *Sucess_Arb*, *Ack* e *TxMsg*). Os relógios (*Tempo* e *Com*) são indicados pelos seus nomes e setas dirigidas a partir do bloco Osc, que representa o circuito oscilador responsável pela temporização do nó, até o respectivo autômato.

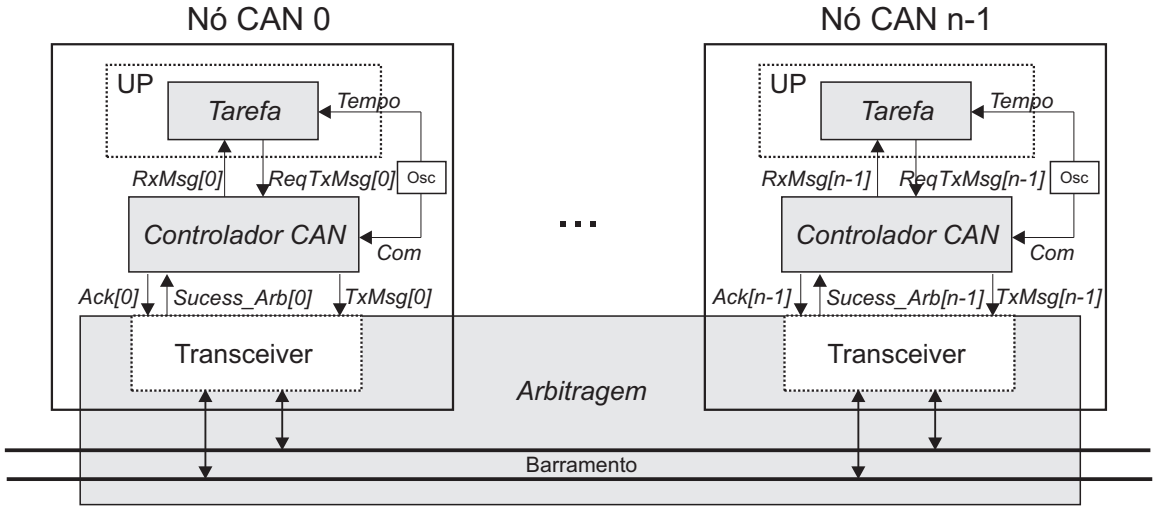


Figura 4.2: Representação da rede CAN com os componentes de cada nó e os três autômatos com os seus relógios e os seus canais de sincronização.

4.2.3 Estudos de caso

Nesta seção, as técnicas de verificação de modelos simbólica e estatística serão usadas para determinar uma análise dos tempos de resposta das mensagens em uma rede CAN, a partir do modelo desenvolvido na seção 4.2.2. A análise será realizada em um estudo de caso de uma rede CAN com oito nós, onde cada nó transmite apenas uma mensagem de dados periódica. Os atributos das mensagens de cada nó estão indicados na Tabela 4.2. Considerou-se em todos os experimentos uma taxa de transmissão igual a 500 Kbits/s ($\tau_{bit} = 2\ \mu$s). Os resultados obtidos no cenário de pior caso por meio da verificação simbólica serão comparados com os resultados obtidos por meio da técnica analítica de escalonamento em tempo real desenvolvida por Davis et al. (2007).

Tabela 4.2: Atributos do conjunto de mensagens (em μs).

$Msg\ (i)$	T_{m_i}	C_i	C_{m_i}	D_{m_i}
0	10000	208	114*τ_{bit}	10000
1	5000	142	82*τ_{bit}	5000
2	10000	142	82*τ_{bit}	10000
3	100000	208	114*τ_{bit}	100000
4	25000	208	114*τ_{bit}	25000
5	20000	142	82*τ_{bit}	20000
6	10000	174	98*τ_{bit}	10000
7	50000	190	106*τ_{bit}	50000

Verificação simbólica versus técnica analítica

As seguintes propriedades lógicas e temporais foram formuladas para verificação do modelo, onde: *TARi* é o autômato tarefa relacionado a mensagem m_i; *N_Escalon* é o lugar do autômato tarefa que representa o não cumprimento do *deadline* da mensagem; *MsgFim* é o lugar do autômato tarefa *TARi* que representa o fim da transmissão da mensagem, *Tempo* é o relógio local do autômato tarefa, *sup* e *inf* são, respectivamente, os operadores *supremum* e *infimum*.

1. *A [] not deadlock*
2. *A [] not TARi.N_Escalon*
3. Qual o tempo de resposta no pior caso da mensagem m_i?
   sup{TARi.MsgFim}:TARi.Tempo
4. Qual o tempo de resposta no melhor caso da mensagem m_i?
   inf{TARi.MsgFim}:TARi.Tempo

A verificação das propriedades foi realizada pela ferramenta UPPAAL versão 4.1.9 em um *desktop* com processador Intel core i7 com 16 GB de memória física e sistema operacional Linux Ubuntu 14.04 de 64 *bits*[2].

As propriedades lógicas 1 e 2 verificam, respectivamente, se não existe *deadlock* no modelo, e se o lugar *N_Escalon* nunca será alcançado, ou seja, se o *deadline* da mensagem m_i foi satisfeito. As propriedades temporais 3 e 4 foram usadas para determinar, respectivamente, os tempos de resposta no pior e melhor caso. Essas propriedades podem ser lidas do seguinte modo: "qual o maior/menor valor do relógio *Tempo*, se o lugar *Msg_Fim* do autômato *TARi* foi alcançado?". Para determinar os tempos de resposta no pior caso usando a técnica analítica, foi

[2]Todas as verificações de propriedades realizadas neste trabalho utilizaram essa configuração.

desenvolvido um programa em linguagem C para resolver as equações apresentadas na Seção 3.4.3.

Três estudos de caso foram realizados sob as seguintes considerações na rede CAN: (1) apenas com as mensagens 0 e 1; (2) com as mensagens 0, 1 e 2 e (3) com as mensagens 0, 1, 2 e 3. Nos três estudos, foi constatado pelas propriedades 1 e 2 que o modelo é livre de *deadlock* e que os *deadlines* das mensagens foram cumpridos. Os tempos de resposta das mensagens usando as técnicas são apresentados nas Tabelas 4.3, 4.4 e 4.5. O tempo total de verificação para determinar os tempos de resposta no melhor caso e pior caso, respectivamente nos casos (1), (2) e (3), foi igual a 1 s, 1,5 s e 259 h. Enquanto que o tempo gasto usando a técnica analítica, em ambos os casos, foi inferior a 1 s.

Tabela 4.3: Tempos de resposta de uma rede CAN com duas mensagens usando a verificação de modelos simbólica e a técnica analítica.

	CMC		**Analítica**
$Msg\ (i)$	$R_{m_i}^{min}$ (μs)	$R_{m_i}^{max}$ (μs)	$R_{m_i}^{max}$ (μs)
0	436	606	606
1	306	540	540

Tabela 4.4: Tempos de resposta de uma rede CAN com três mensagens usando a verificação de modelos simbólica e a técnica analítica.

	CMC		**Analítica**
$Msg\ (i)$	$R_{m_i}^{min}$ (μs)	$R_{m_i}^{max}$ (μs)	$R_{m_i}^{max}$ (μs)
0	436	606	606
1	306	710	710
2	306	710	710

Tabela 4.5: Tempos de resposta de uma rede CAN com quatro mensagens usando a verificação de modelos simbólica e a técnica analítica.

	CMC		**Analítica**
$Msg\ (i)$	$R_{m_i}^{min}$ (μs)	$R_{m_i}^{max}$ (μs)	$R_{m_i}^{max}$ (μs)
0	436	670	670
1	306	774	774
2	306	944	944
3	436	1010	1010

Nos três experimentos, os tempos de resposta no pior caso, em ambas as técnicas, são idênticos. Os tempos de resposta no melhor caso, de acordo com o esperado, são exatamente iguais

a soma dos tempos de computação e transmissão da mensagem. Isso significa que, pelo menos uma vez, as mensagens foram transmitidas imediatamente após serem armazenadas nos seus *buffers* de transmissão, devido o barramento estar livre.

Não foi possível, para um conjunto com mais de quatro mensagens, determinar os tempos de resposta usando a técnica de verificação de modelos simbólica. Isso ocorreu devido a uma explosão do espaço de estados do modelo com mais de quatro mensagens. A verificação com quatro mensagens dá indício de uma possível ocorrência desse problema a partir de cinco mensagens, pois foi observado que o tempo de verificação aumentou exponencialmente de 1,5 s (com três mensagens) para 932400 s ou 259 h (com quatro mensagens). Os resultados usando a técnica analítica para o conjunto com oito mensagens são apresentados na Tabela 4.6. O tempo gasto para obter os tempos de resposta foi inferior a 1 s. Todas as mensagens cumpriram os seus requisitos de tempo, uma vez que os tempos de resposta no pior caso são menores que os *deadline*.

Tabela 4.6: Tempos de resposta de uma rede CAN com oito mensagens usando a técnica analítica.

$Msg\ (i)$	$R_{m_i}^{max}$ (μs)
0	670
1	774
2	944
3	1244
4	1462
5	1566
6	1800
7	1816

Verificação estatística

Para resolver o problema da explosão do espaço de estados, utilizou-se a verificação de modelos estatística. Nesse caso, pretende-se obter as probabilidades de satisfazer *deadlines*. O verificador de modelos estatísticos da ferramenta UPPAAL considera que os atrasos em uma transição, limitados superiormente pela invariante de estado e inferiormente pelo guarda, seguem uma distribuição uniforme.

Foi realizado uma verificação com o conjunto de oito mensagens para determinar a probabilidade do *deadline* da mensagem m_i ser satisfeito. Essa propriedade foi descrita na linguagem de especificação no formalismo da ferramenta UPPAAL, conforme descrito a seguir, onde *TARi* é o autômato tarefa relacionado a mensagem m_i, com i variando de 0 a 7, *Tempo* é um relógio

local, Di é o *deadline* e *MsgFim* é o lugar do autômato tarefa *TARi* que representa o fim da transmissão da mensagem. Em outras palavras, a propriedade verifica qual a probabilidade do lugar *MsgFim* ser alcançado e o valor do relógio *Tempo* ser menor ou igual ao *deadline* Di. No total, foram verificadas oito propriedades, uma por mensagem. Os seguintes parâmetros estatísticos foram utilizados na verificação: $\epsilon = 0{,}01$ e $\alpha = 0{,}01$.

$$Pr[TARi.Tempo <= Di] (<> TARi.MsgFim)$$

A ferramenta retorna um limite inferior e superior de probabilidade para cada propriedade verificada, com confidência de 99%. Os gráficos das funções de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) do conjunto de mensagens são apresentados nas Figuras 4.3 a 4.10. O tempo gasto na verificação foi de 1,42 s. Na função de distribuição acumulada da mensagem 0 (Figura 4.3), observou-se que a probabilidade do tempo de resposta ser inferior a 450 μs é maior que 90%, e que a probabilidade é muito próxima de 100% para um tempo de resposta em torno de 625 μs. Isso significa que o cenário de pior caso, cujo tempo de resposta obtido analíticamente foi igual a 670 μs (ver Figura 4.6), ocorre raramente. É importante notar que o fator de utilização da rede igual a aproximadamente 12%, considerado baixo, contribuiu para a rara ocorrência do cenário de pior caso, uma vez que o barramento está livre em 88% do tempo. Uma análise semelhante pode ser realizada para as mensagens restantes. Na seção 4.4 esse modelo será validado experimentalmente.

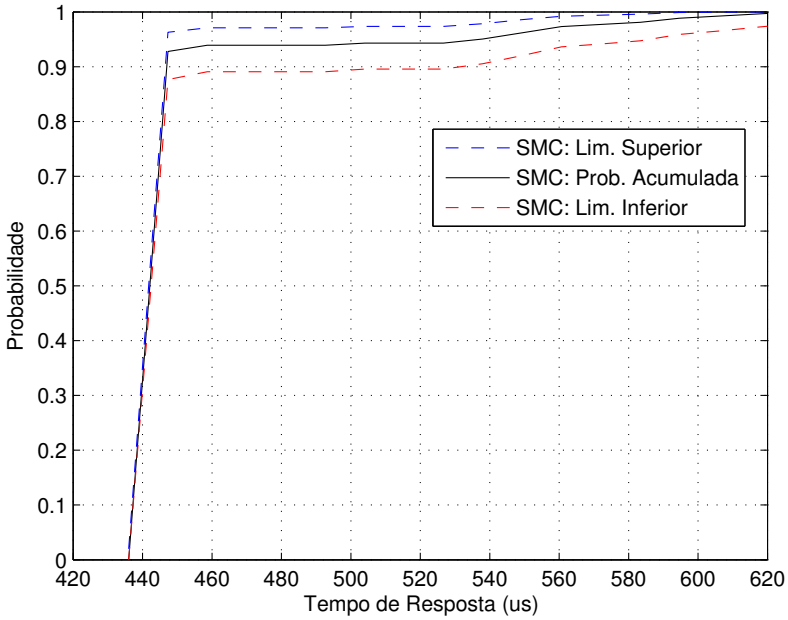


Figura 4.3: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 0.

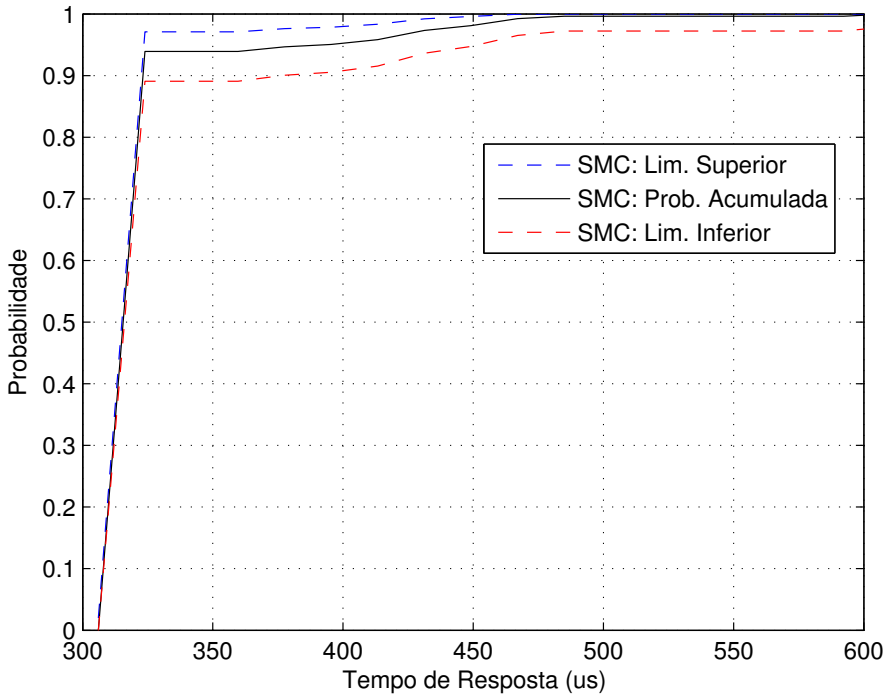


Figura 4.4: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 1.

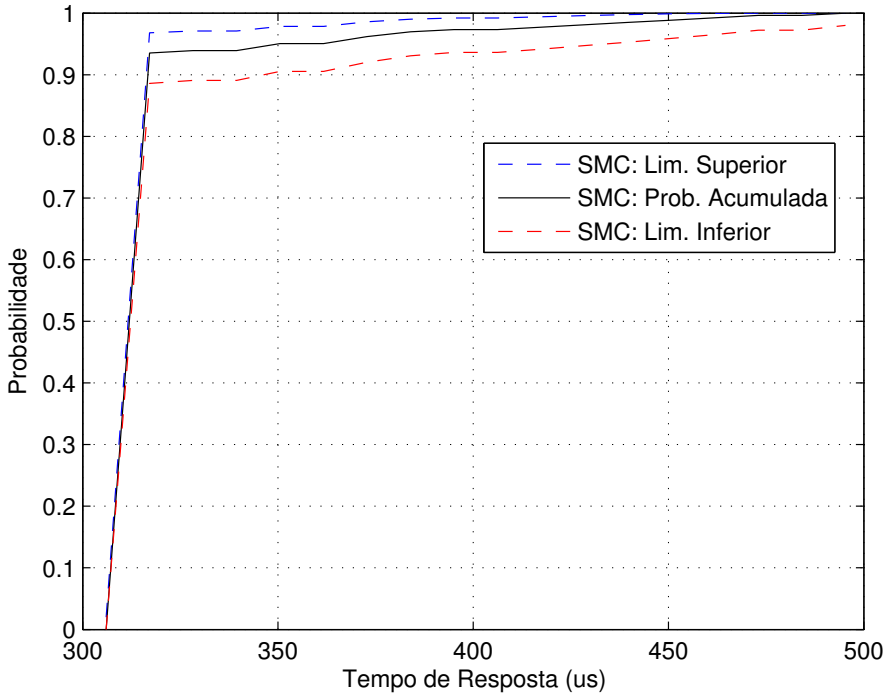


Figura 4.5: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 2.

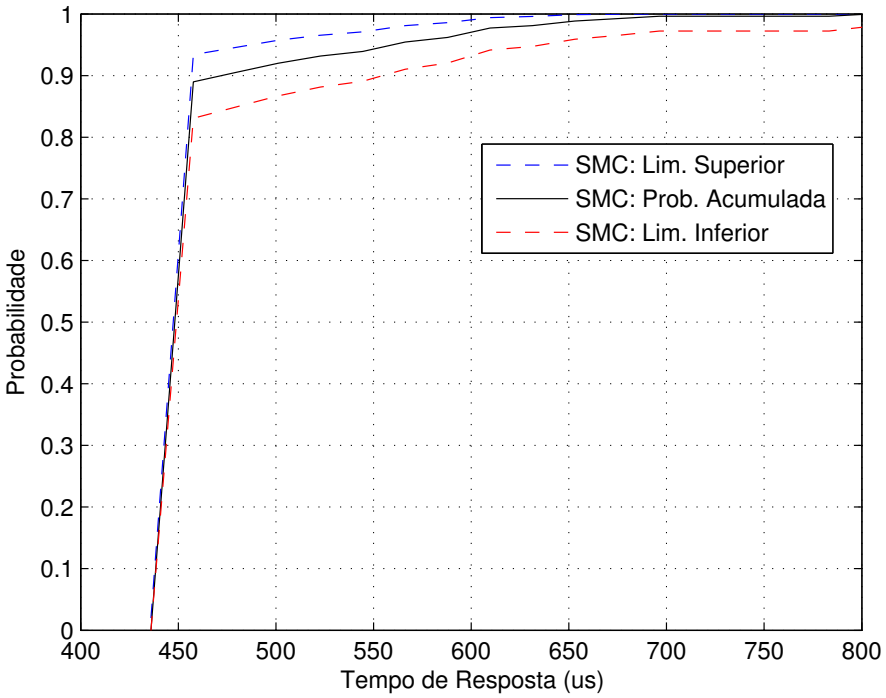


Figura 4.6: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 3.

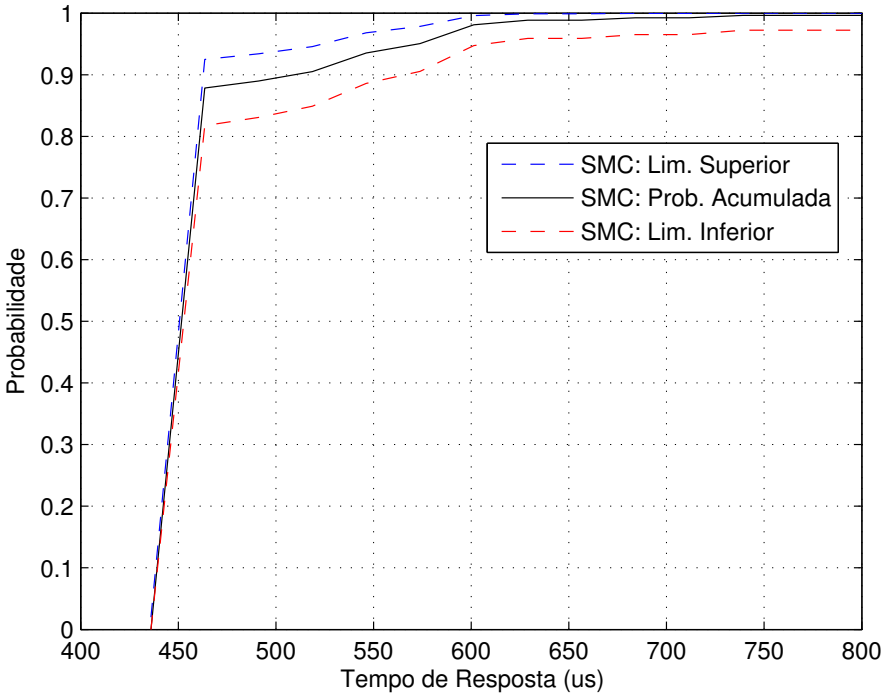


Figura 4.7: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 4.

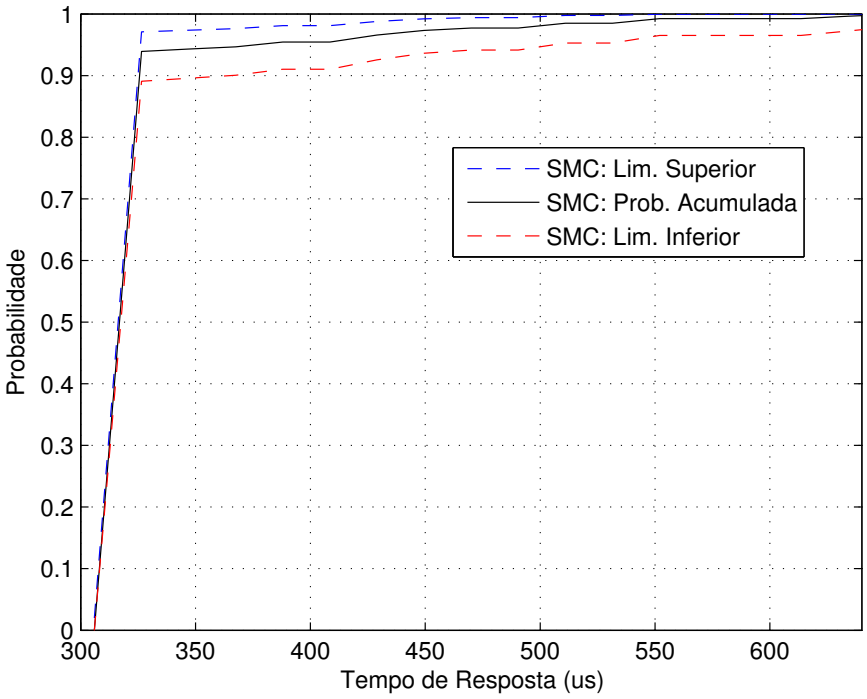


Figura 4.8: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 5.

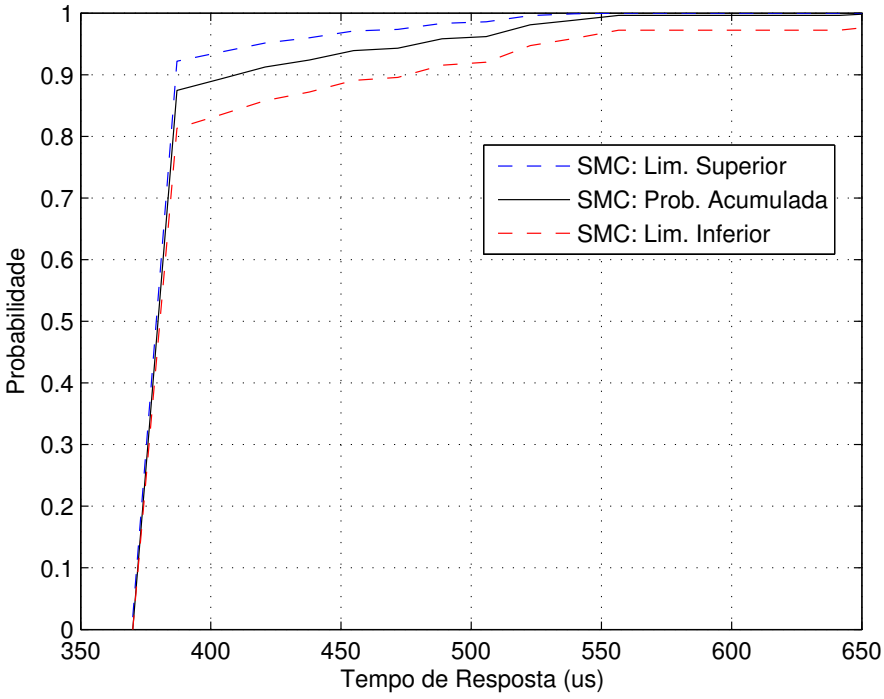


Figura 4.9: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 6.

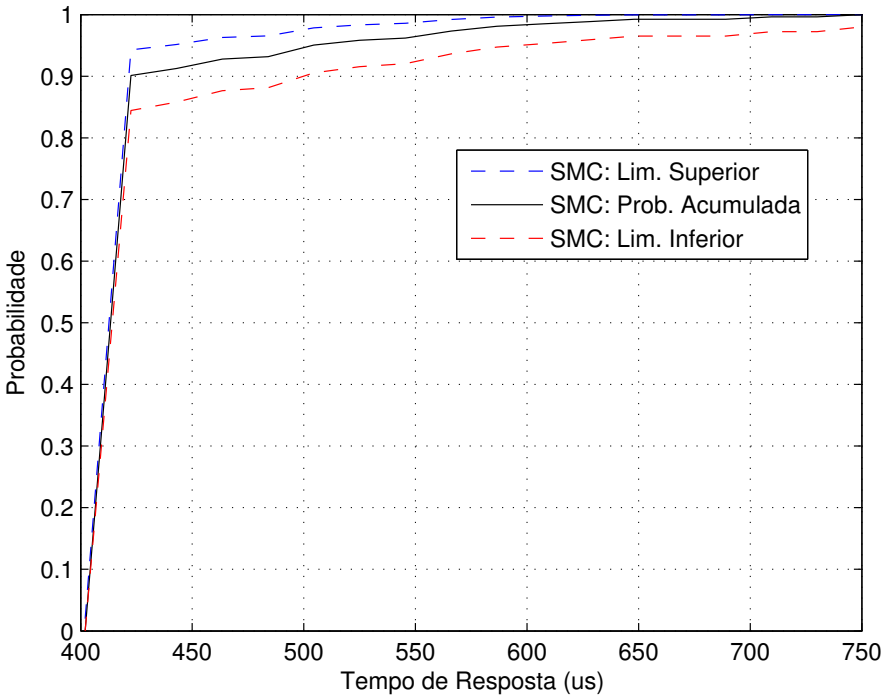


Figura 4.10: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 7.

4.3 Modelo formal da rede CAN com *offset* dinâmico, relógios reais e barramento ideal

Nesta seção apresenta-se o modelo 2 (Ver Tabela 4.1), onde os *offsets* das mensagens são dinâmicos, o relógio de cada nó CAN é real e o barramento é livre de erros (barramento ideal).

4.3.1 Trabalhos relacionados

Apesar da quantidade relevante de estudos na análise dos tempos de resposta de mensagens CAN, poucos trabalhos consideraram as variações dos relógios dos nós. Monot et al. (2012), em uma versão estendida de Monot, Navet e Bavoux (2011), usaram a ferramenta RTaW-Sim (RTAW, 2012) para simular e determinar as distribuições dos tempos de resposta das mensagens considerando as variações dos relógios. Foi demonstrado que para diferentes variações dos relógios os tempos de resposta convergem, e que os tempos de resposta no pior caso são transitórios.

4.3.2 Modelo em autômato temporizado

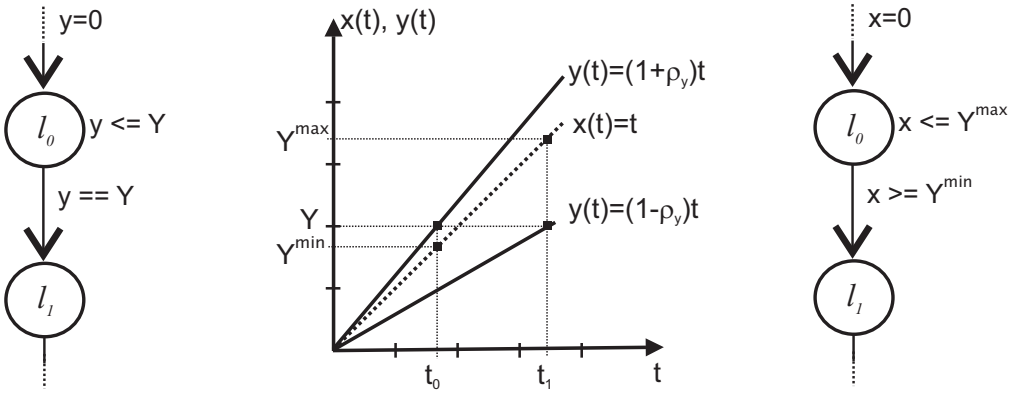
Na teoria de autômatos temporizados, os relógios funcionam de modo semelhante a um relógio ideal. Portanto, é necessário usar algum conceito para modelar relógios reais em autômatos temporizados. Neste trabalho será usado o conceito de autômatos temporizados perturbados (ALUR; TORRE; MADHUSUDAN, 2005), devido a sua compatibilidade com a teoria de autômatos temporizados. Navas and Proenza (RODRIGUEZ-NAVAS; PROENZA, 2013) usaram o mesmo conceito para modelar esse tipo de relógio em uma rede CAN. Entretanto, o objetivo não era determinar os tempos de resposta das mensagens em uma rede CAN.

A semântica de um autômato temporizado perturbado permite modelar um relógio x com taxas variantes dentro de um intervalo limitado $[\hat{\rho}_x^{min},\ \hat{\rho}_x^{max}]$. Assim, pode-se dizer que um autômato temporizado é um caso trivial de um autômato temporizado perturbado com taxa de variação constante e igual a 1. Alur et al. (ALUR; TORRE; MADHUSUDAN, 2005) demonstraram que qualquer autômato temporizado perturbado pode ser transformado em um autômato temporizado equivalente, desde que os limites da taxa do relógio sejam constantes e conhecidos. Consequentemente, pode-se transformar relógios reais em relógios ideais.

Para realizar essa transformação, é necessário considerar os relógios reais como relógios ideais e adicionar os valores mínimos e máximos resultantes dos limites inferiores e superiores das taxas, ao longo de um determinado intervalo de tempo. Os limites inferiores e superiores do valor do relógio são modelados, respectivamente, no guarda e na invariante de estado. A seguir, usaremos uma abordagem semelhante à descrita em (RODRIGUEZ-NAVAS; PROENZA, 2013) para realizar essa transformação.

Considere um relógio real y com variação de $\pm\rho_y$ ppm, ou seja, uma taxa variando no intervalo $[1-\rho_y,\ 1+\rho_y]$. Além disso, considere o autômato temporizado perturbado (simplificado) representado na Figura 4.11(a), que modela o tempo de execução de uma tarefa em um processador. O lugar l_0 representa a tarefa em execução e o lugar l_1 a finalização da tarefa. O tempo de execução dessa tarefa é igual a Y unidades de tempo. Então, a transição de l_0 para l_1 ocorrerá após o relógio y atingir o valor Y, conforme modelado pelo guarda $y == Y$.

O gráfico da Figura 4.11(b) representa o comportamento temporal do relógio real y e de um relógio ideal x ao longo do tempo real. O relógio ideal x será usado no processo de transformação. As retas superior e inferior representam, respectivamente, o comportamento do relógio real y quando a taxa de variação é constante e igual a $1+\rho_y$ (taxa máxima), e igual a $1-\rho_y$ (taxa mínima). A reta tracejada representa o comportamento do relógio ideal x. Os instantes t_0 e t_1 correspondem ao valor de tempo real em que o relógio real y atinge o valor Y quando está operando, respectivamente, na máxima e mínima taxa de variação. Uma vez que na prática a taxa de variação não é constante ao longo do tempo, podemos afirmar que o valor de tempo real t no qual o relógio atinge o valor Y corresponde a algum valor no intervalo $[t_0,\ t_1]$. Esses instantes são fundamentais no processo de transformação e podem ser descritos matematicamente



(a) (b) (c)

Figura 4.11: (a) Modelo em autômato temporizado perturbado, (b) gráfico do comportamento do relógio ideal x e relógio real y com taxa de variação mínima e máxima versus o tempo real t, e (c) modelo em autômato temporizado equivalente.

pelas Equações 4.3 e 4.4.

$$t_0 = \frac{Y}{1+\rho_y} \tag{4.3}$$

$$t_1 = \frac{Y}{1-\rho_y} \tag{4.4}$$

Observe na Figura 4.11(c) que o comportamento do relógio real y com taxa variando no intervalo $[1-\rho_y,\ 1+\rho_y]$ pode ser representado pelo relógio ideal x, por meio da determinação dos valores de x no intervalo $[t_0,\ t_1]$. De acordo com a Figura 4.11(b), os valores de x nos limites desse intervalo são Y^{min} e Y^{max}, onde min e max indicam, respectivamente, o limite inferior e o limite superior. Uma vez que o relógio ideal sempre reflete o valor de tempo real t, podemos afirmar que $t_0 = Y^{min}$ e $t_1 = Y^{max}$. Além disso, como a variação $\rho_y \ll 1$ (da ordem de μs), podemos aproximar a divisão no termo do lado direito das Equações 4.3 e 4.4, resultando nas Equações 4.5 e 4.6. Essa aproximação é importante, pois os valores dos relógios, segundo a definição de autômatos temporizados, podem ser comparados apenas com valores inteiros. Essa restrição é também considerada na ferramenta UPPAAL.

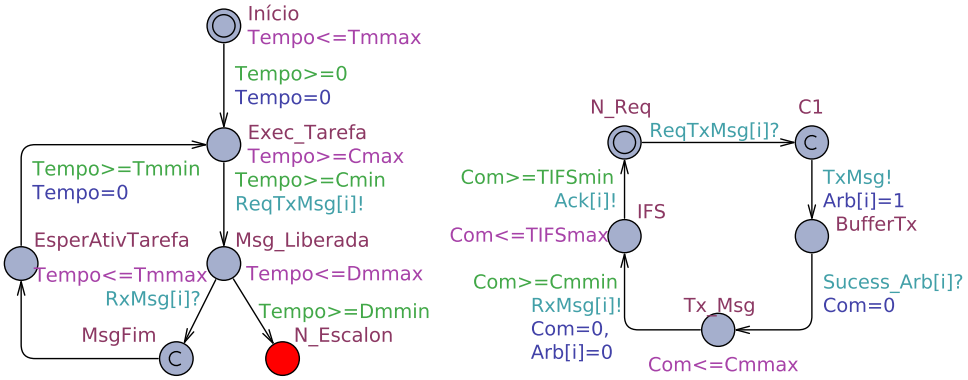
$$Y^{min} = \frac{Y}{1+\rho_y} \approx Y(1-\rho_y) = Y - \rho_y Y \tag{4.5}$$

$$Y^{max} = \frac{Y}{1-\rho_y} \approx Y(1+\rho_y) = Y + \rho_y Y \tag{4.6}$$

Desse modo, o autômato temporizado perturbado da Figura 4.11(a) pode ser transformado no autômato temporizado da Figura 4.11(c). Observe que a transição de l_0 para l_1 poderá ocorrer

em qualquer instante no intervalo $[Y^{min},\ Y^{max}]$.

O modelo da rede CAN considerando relógios reais (modelo 2) é também constituído por três autômatos: tarefa, controlador CAN e arbitragem. O autômato arbitragem é idêntico ao usado no modelo 1 apresentado na seção 4.2.2. Enquanto que os autômatos tarefa e controlador CAN, representados na Figura 4.12, são diferentes apenas na modelagem dos atrasos, devido a variação nos relógios de cada nó. Nesses dois autômatos, um determinado atraso Y é modelado de acordo com as Equações 4.5 e 4.6, resultando nos atrasos Y^{min} e Y^{max}, respectivamente. O atraso Y^{max} é modelado na invariante de estado e o atraso Y^{min} no guarda.



(a) Tarefa.

(b) Controlador CAN.

Figura 4.12: Conjunto de autômatos temporizados de uma rede CAN.

No autômato tarefa (Figura 4.12(a)), o *offset* varia no intervalo de $[0, Tmmax]$, onde $Tmmax$ é a soma do período Tm com a variação positiva $\rho_{Tempo}Tm$ do relógio *Tempo*. O limite inferior do *offset* é igual a zero, pois consideramos que, inicialmente, o relógio *Tempo* está zerado. A modelagem do atributos restantes (tempo de computação, *deadline* e período) variam de um valor mínimo diferente de zero até um valor máximo. Os atributos tempo de transmissão e intervalo entre mensagens (*IFS*) foram modelados de modo semelhante no autômato controlador CAN (Figura 4.12(b)). O código UPPAAL do modelo é apresentado na Seção C.2 do Apêndice C.

4.3.3 Estudos de caso

Nesta seção, as técnicas de verificação de modelos simbólica (clássica) e estatística serão usadas para determinar uma análise dos tempos de resposta das mensagens em uma rede CAN, a partir do modelo desenvolvido na seção 4.3.2. A análise será realizada no mesmo estudo de caso apresentado na seção 4.2.3, cujos atributos das mensagens estão indicados na Tabela 4.2 e taxa de transmissão igual a 500 Kbits/s ($\tau_{bit} = 2\ \mu$s). Para esse conjunto de mensagens serão realizadas comparações entre os resultados obtidos com os modelos sem variações (modelo 1)

e com variações de relógio (modelo 2). Todos os atributos foram especificados em nanosegundos. Devido a ferramenta UPPAAL não aceitar valores não inteiros, foi necessário realizar aproximações para esses valores. Por exemplo, uma variação de 1,9 ns foi aproximada para 2,0 ns.

Verificação simbólica - modelo 1 versus modelo 2

As mesmas propriedades lógicas e temporais formuladas para verificação do modelo 1 foram verificadas. Essas propriedades são apresentadas novamente a seguir, onde: *TARi* é o autômato tarefa relacionado a mensagem m_i; *N_Escalon* é o lugar do autômato tarefa que representa o não cumprimento do *deadline* da mensagem; *MsgFim* é o lugar do autômato tarefa *TARi* que representa o fim da transmissão da mensagem, *Tempo* é o relógio local do autômato tarefa, *sup* e *inf* são, respectivamente, os operadores *supremum* e *infimum*.

1. *A [] not deadlock*
2. *A [] not TARi.N_Escalon*
3. Qual o tempo de resposta no pior caso da mensagem m_i?
   sup{TARi.MsgFim}:TARi.Tempo
4. Qual o tempo de resposta no melhor caso da mensagem m_i?
   inf{TARi.MsgFim}:TARi.Tempo

Dois estudos de caso foram realizados sob as seguintes considerações na rede CAN: (1) apenas com as mensagens 0 e 1, e (2) com as mensagens 0, 1 e 2. Nos dois estudos, foi constatado pelas propriedades 1 e 2 que o modelo é livre de *deadlock* e que os *deadlines* das mensagens foram cumpridos. Na verificação das propriedades 3 e 4, foi considerado dois valores típicos de variações nos relógios: $\pm$ 30 ppm e $\pm$ 50 ppm (HENDERSON; KENDALL; ROBSON, 2000) e (MONOT; NAVET; BAVOUX, 2011). Os tempos de resposta no melhor e pior caso para os dois estudos são apresentados nas Tabelas 4.7 e 4.8. Os tempos de verificação foram iguais a 1 s e 1,5 s, respectivamente, para os estudos (1) e (2).

Tabela 4.7: Tempos de resposta em μs de uma rede CAN com duas mensagens considerando os relógios sem variação (ideal), com variação (real) de $\pm$ 30 ppm e $\pm$ 50 ppm usando a verificação de modelos simbólica.

	Rel. Ideal		**Rel. Real - $\pm$ 30 ppm**		**Rel. Real - $\pm$ 50 ppm**	
$Msg\ (i)$	$R_{m_i}^{min}$	$R_{m_i}^{max}$	$R_{m_i}^{min}$	$R_{m_i}^{max}$	$R_{m_i}^{min}$	$R_{m_i}^{max}$
0	436	606	435,987	606,018	435,979	606,029
1	306	540	305,991	540,016	305,985	540,026

Tabela 4.8: Tempos de resposta em μs de uma rede CAN com três mensagens considerando os relógios sem variação (ideal), com variação (real) de $\pm$ 30 ppm e $\pm$ 50 ppm usando a verificação de modelos simbólica.

	Rel. Ideal		**Rel. Real ($\pm$ 30 ppm)**		**Rel. Real ($\pm$ 50 ppm)**	
$Msg\ (i)$	$R_{m_i}^{min}$	$R_{m_i}^{max}$	$R_{m_i}^{min}$	$R_{m_i}^{max}$	$R_{m_i}^{min}$	$R_{m_i}^{max}$
0	436	606	435,987	606,018	435,979	606,029
1	306	710	305,991	710,021	305,985	710,034
2	306	710	305,991	710,021	305,985	710,034

Nos dois experimentos, observou-se que os tempos de resposta são quase idênticos. Os tempos de resposta para uma variação de $\pm$ 50 ppm são um pouco maiores do que os tempos de resposta para variações de $\pm$ 30 ppm. Esse resultado é justificado pelo fato das mensagens serem periódicas. Com isso, as variações nos relógios deslocam continuamente os instantes de liberação das mensagens ao longo do tempo. Além disso, no modelo, as variações apenas geram um número maior de possíveis instantes de transições temporizadas de um lugar para outro, ao contrário do modelo 1, onde as transições temporizadas ocorrem em instantes fixos. Entretanto, essas variações não influenciam de modo significante nos tempos de resposta.

Ao contrário do modelo 1, não foi possível, para um conjunto com mais de três mensagens, determinar os tempos de resposta usando a técnica de verificação de modelos simbólica. Isso ocorreu devido a uma explosão do espaço de estados do modelo com mais de quatro mensagens. Conforme declarado, um maior número de possíveis instantes de transição foi gerado pelas variações e, portanto, a quantidade de estados do modelo aumentou.

Verificação estatística

Para resolver o problema da explosão do espaço de estados, utilizou-se a verificação de modelos estatísticos para obter uma probabilidade de satisfazer os *deadlines* das mensagens. Foi realizado uma verificação com o conjunto de oito mensagens para determinar a probabilidade do *deadline* da mensagem m_i ser satisfeito. Essa propriedade foi descrita na linguagem de especificação no formalismo da ferramenta UPPAAL, conforme descrito a seguir, onde *TARi* é o autômato tarefa relacionado a mensagem m_i, com i variando de 0 a 7, *Tempo* é um relógio local, Di é o *deadline* e *MsgFim* é o lugar do autômato tarefa *TARi* que representa o fim da transmissão da mensagem. Em outras palavras, a propriedade verifica qual a probabilidade do lugar *MsgFim* ser alcançado e o valor do relógio *Tempo* ser menor ou igual ao *deadline* Di.

Pr[TARi.Tempo <= Di] (<> TARi.MsgFim)

No total, foram verificadas oito propriedades, uma por mensagem. Os gráficos das funções de distribuição acumulada (limites inferior, probabilidade acumulada e superior) do conjunto de

mensagens com confidência de 99% são apresentados nas Figuras 4.13 a 4.20. O tempo gasto na verificação foi de 1,5 s. Em geral, foi observado que as funções de distribuição dos tempos de resposta das mensagens para os casos sem variação e com variação são semelhantes. Desse modo, verificamos que as variações nos relógios também não influenciam nas distribuições dos tempos de resposta. Do ponto de vista da modelagem, esse é um resultado importante, uma vez que nos possibilita desconsiderar as variações nos relógios, reduzindo assim a complexidade e o espaço de estados do modelo.

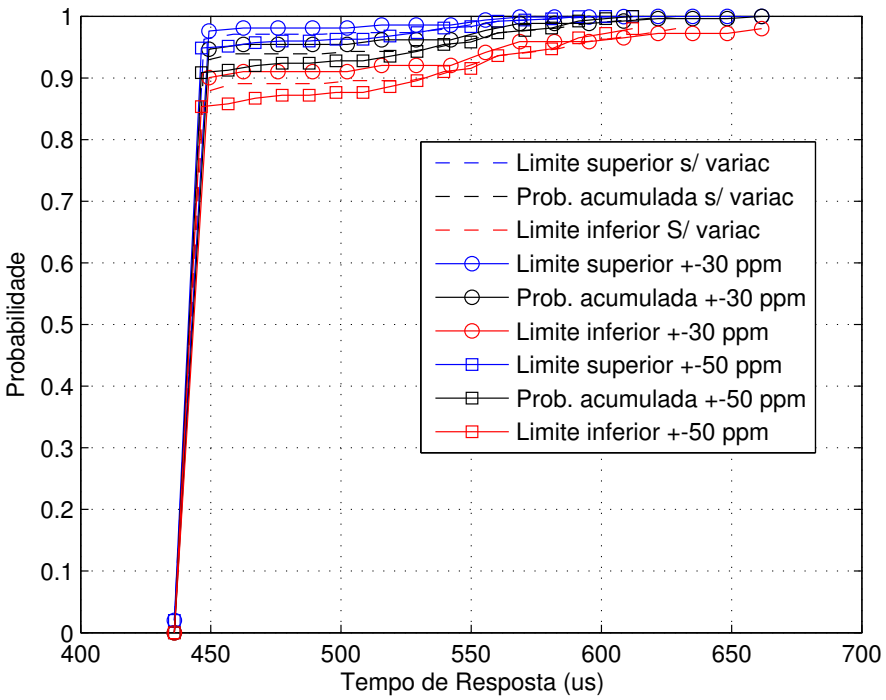


Figura 4.13: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 0.

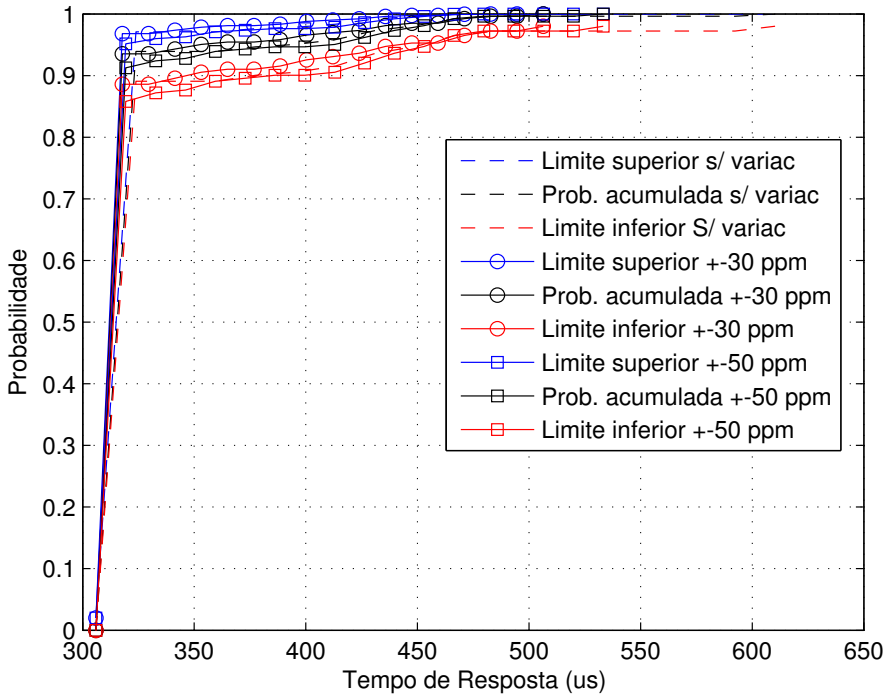


Figura 4.14: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 1 com relógio real.

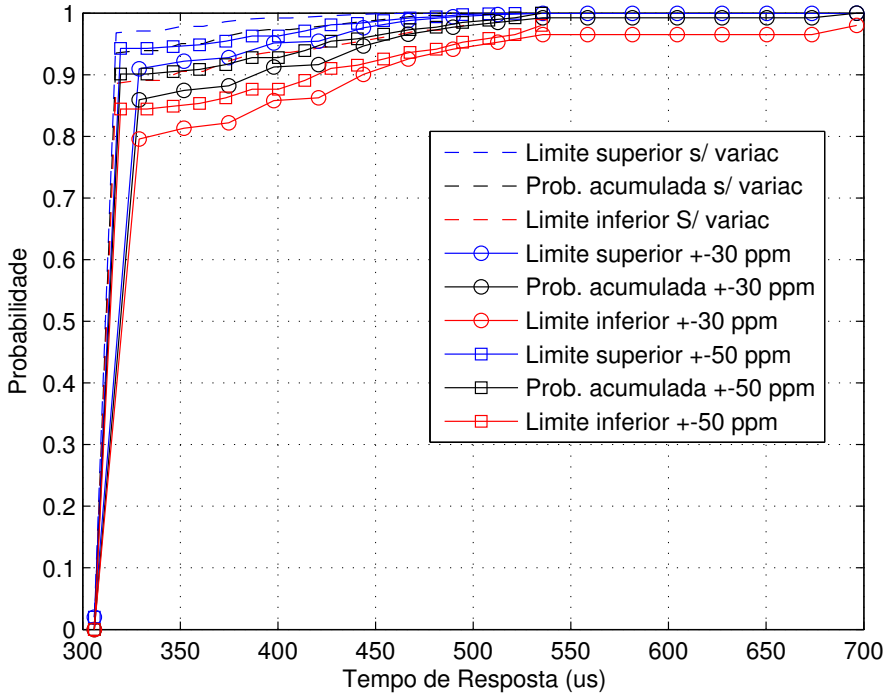


Figura 4.15: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 2 com relógio real.

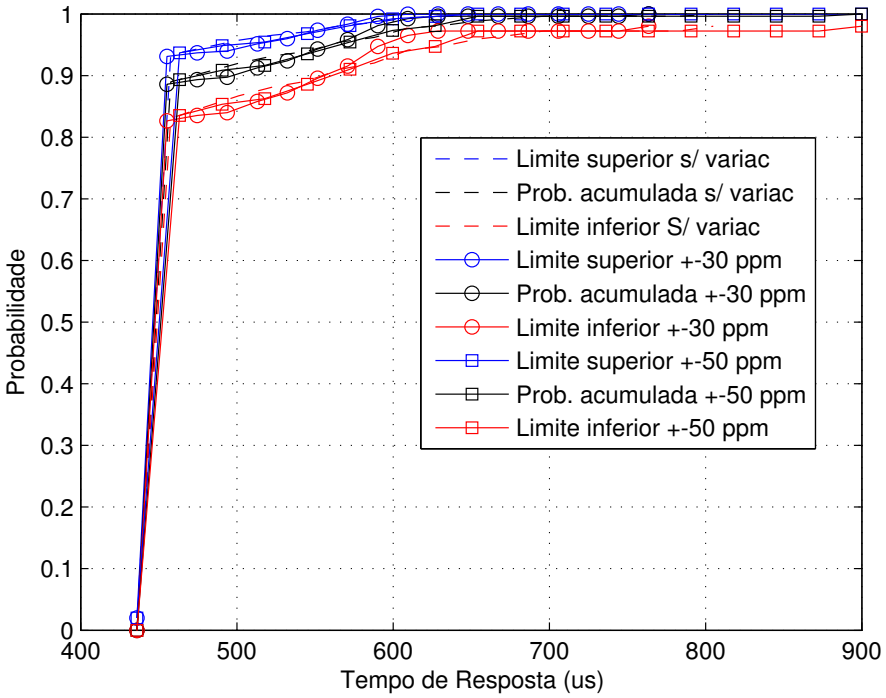


Figura 4.16: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 3 com relógio real.

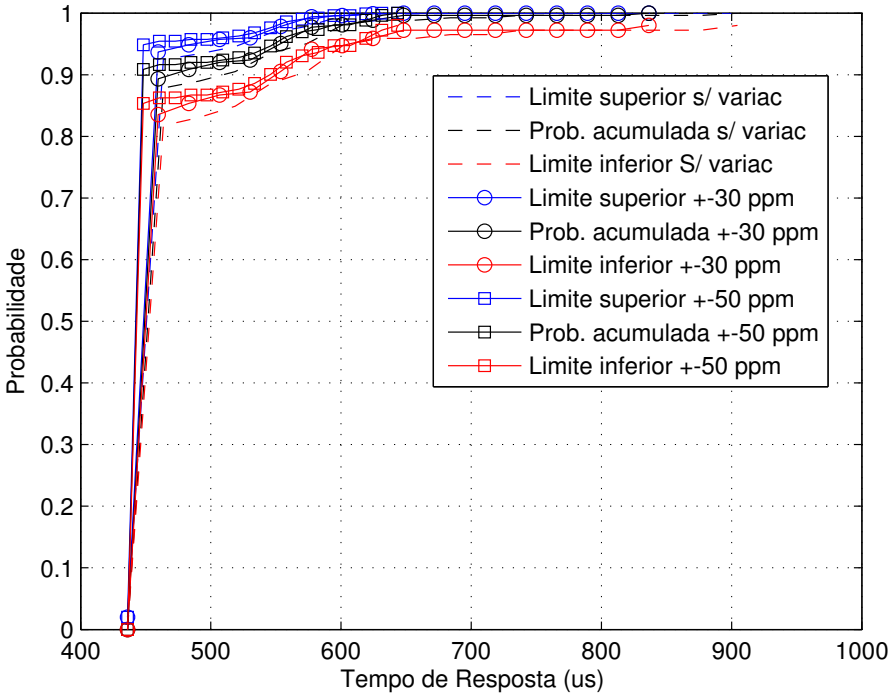


Figura 4.17: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 4 com relógio real.

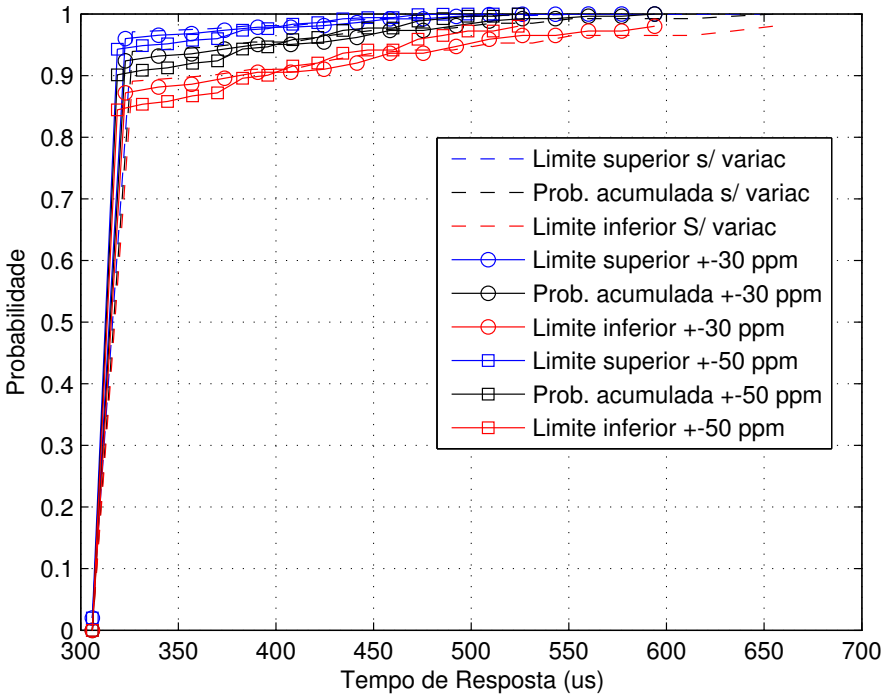


Figura 4.18: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 5 com relógio real.

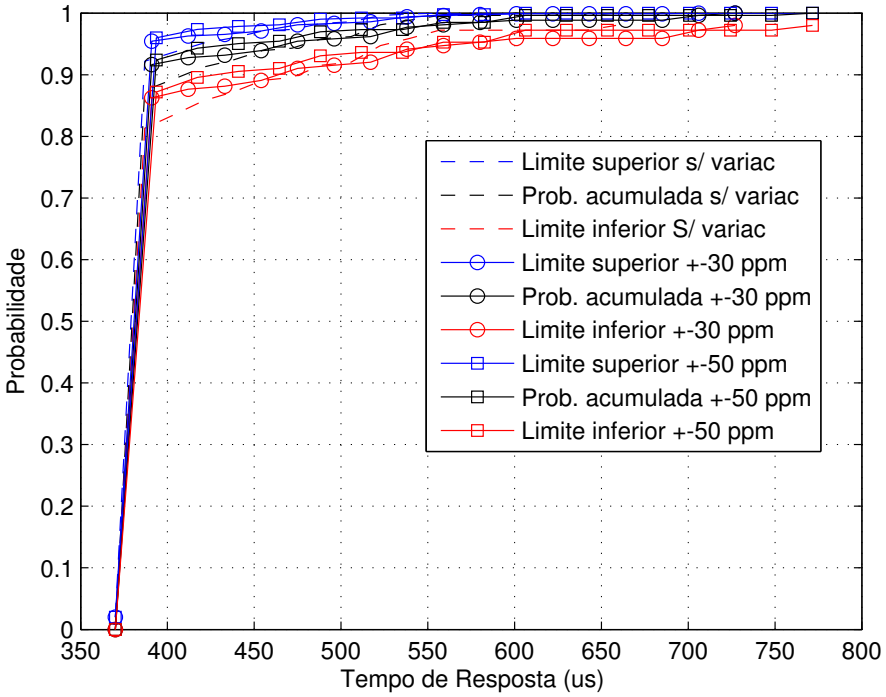


Figura 4.19: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 6 com relógio real.

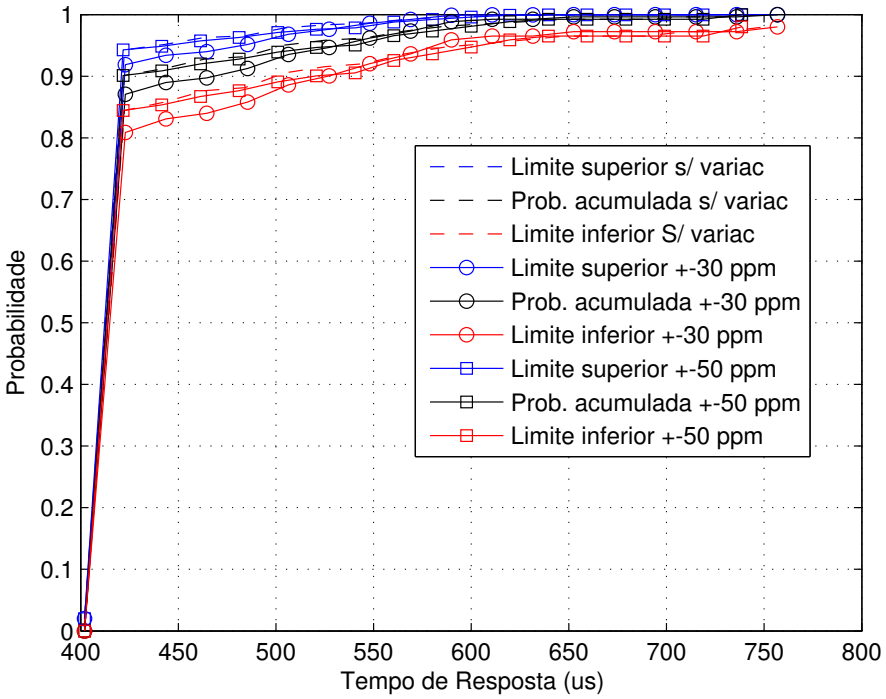


Figura 4.20: Gráfico da função de distribuição acumulada (limite inferior, probabilidade acumulada e limite superior) dos tempos de resposta da mensagem 7 com relógio real.

4.4 Validação do Modelo Formal da Rede CAN com *Offset* Dinâmico, Relógios e Barramento Ideais

Nesta seção será apresentada uma plataforma, chamada de plataforma experimental para rede CAN ideal, para validação do modelo da rede CAN com *offset* dinâmico, relógios e barramento ideais (modelo 1).

4.4.1 Plataforma experimental rede CAN ideal

A plataforma experimental, representada na Figura 4.21, consiste de uma rede CAN com nove nós. Os nós CAN 0 a 7 são formados pelo microcontrolador PIC16F876A (MICROCHIP, 2001), controlador MCP2515 (MICROCHIP, 2005) e *transceiver* MCP2551 (MICROCHIP, 2003). O nó 8 consiste do dispositivo CAN *leaf professional* (KVASER, 2014) conectado via USB a um computador pessoal com sistema operacional linux.

A ideia da plataforma experimental é garantir que cada nó (nós CAN de 0 a 7) envie uma única mensagem periódica com comprimento fixo e livre de erros no barramento, de acordo com o conjunto de mensagens apresentado na Tabela 4.2 da Seção 4.2.3, e armazene os tempos de resposta das mensagens transmitidas ao longo do experimento. O nó 8 será responsável pelo gerenciamento da rede. As seguintes atividades serão executadas por esse nó: envio de

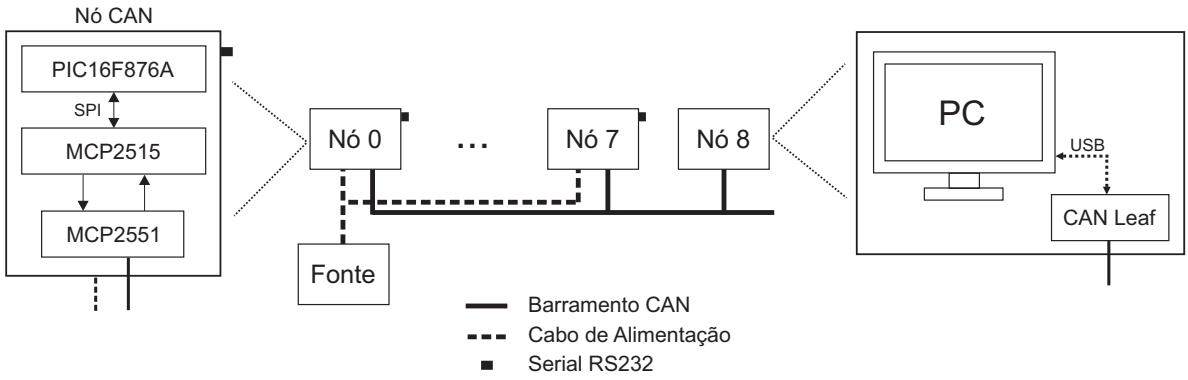


Figura 4.21: Representação da plataforma experimental da rede CAN ideal com os componentes de cada nó.

mensagens informando aos demais nós o início e o fim do experimento (o tempo do experimento é marcado por esse nó); recepção das mensagens transmitidas pelos nós e monitoramento da ocorrência de erros no barramento (sempre que um erro ocorrer, o experimento será finalizado).

Os nós CAN 0 a 7 serão responsáveis por marcar o tempo de resposta de cada instância da sua mensagem periódica. Devido a limitação de memória do microcontrolador, foi utilizado variáveis, que representam intervalos de tempos, as quais são incrementadas sempre que o tempo de resposta da mensagem estiver dentro desse intervalo. A ideia é gerar localmente um histograma dos tempos de resposta com intervalos pré-especificados e com largura fixa. O histograma deverá ter intervalos de tempo suficientes para caracterizar o comportamento dos tempos de resposta das mensagens. Idealmente, os intervalos deveriam ter uma largura igual ao tempo de transmissão de um *bit* (τ_{bit}). Entretanto, devido a limitação de memória, não foi possível utilizar essa largura (τ_{bit}). A partir da memória disponível, foi possível definir 24 intervalos. Assim que o experimento for finalizado, os nós o a 7 transmitem via serial as informações armazenadas.

A largura dos intervalos em cada nó foi definida a partir da observação de resultados experimentais. Os intervalos são definidos pelas Equações 4.7, 4.8 e 4.9, onde τ_{bit} é o tempo de transmissão de um *bit*, NI é o número do intervalo e L_{m_i} é a largura do intervalo da mensagem m_i. A largura do intervalo dos tempos de resposta de cada nó e os intervalos 1, 2, 23 e 24 para um tempo de transmissão de um *bit* igual a 2 μs estão indicados na Tabela 4.9.

$$[0,\ \ C_{m_i} + NI * L_{m_i}] \text{ para } NI = 1 \tag{4.7}$$

$$(C_{m_i} + (NI-1) * L_{m_i},\ \ C_{m_i} + NI * L_{m_i}] \text{ para } 2 \leq NI \leq 23 \tag{4.8}$$

$$(C_{m_i} + (NI-1) * L_{m_i},\ \ \infty) \text{ para } NI = 24 \tag{4.9}$$

Tabela 4.9: Largura do intervalo dos tempos de resposta de cada nó e os intervalos 1, 2, 23 e 24 para $\tau_{bit} = 2$ μs sem o tempo de computação das tarefas.

$Msg\ (i)$	L_{m_i}	$NI = 1$	$NI = 2$	$NI = 23$	$NI = 24$
0	6*τ_{bit}	[0, 240]	(240, 252]	(492, 504]	(504, ∞)
1	9*τ_{bit}	[0, 182]	(182, 200]	(560, 578]	(578, ∞)
2	8*τ_{bit}	[0, 180]	(180, 196]	(516, 532]	(532, ∞)
3	11*τ_{bit}	[0, 250]	(250, 272]	(712, 734]	(734, ∞)
4	8*τ_{bit}	[0, 244]	(244, 260]	(580, 596]	(596, ∞)
5	10*τ_{bit}	[0, 184]	(184, 204]	(604, 624]	(624, ∞)
6	8,5*τ_{bit}	[0, 213]	(213, 230]	(570, 587]	(587, ∞)
7	11*τ_{bit}	[0, 234]	(234, 245]	(696, 718]	(718, ∞)

4.4.2 Experimentos e análise dos resultados

O objetivo do experimento é obter um histograma dos tempos de resposta das mensagens, com os intervalos definidos na Tabela 4.9 da Seção 4.4.1, de modo a gerar a função de distribuição acumulada dos tempos de resposta das mensagens. Um parâmetro importante nesse experimento é o seu tempo de duração, uma vez que os histogramas apresentarão o número de ocorrências dos tempos de resposta das mensagens em cada intervalo. Idealmente, o tempo de duração do experimento deve ser igual ao hiperperíodo das mensagens, ou seja, o mínimo múltiplo comum dos períodos da mensagens, visto que o hiperperíodo é o intervalo onde a escala de tempo das mensagens se repete. Portanto, todos os possíveis tempos de resposta das mensagens ocorrem dentro desse intervalo. Entretanto, como os *offsets* das mensagens são dinâmicos, o hiperperíodo das mensagens torna-se maior e desconhecido. Além disso, as variações nos relógios são também um parâmetro que dificulta o conhecimento do valor do hiperperíodo.

Para resolver o problema da duração do experimento, consideramos que, a partir de uma determinada amostra com duração do experimento igual a t_1, todas as amostras com duração do experimento maior que t_1, pertencem a mesma população. Então, a amostra com duração t_1 é suficiente para caracterizar o comportamento dos tempos de resposta das mensagens. Para isso, foram realizados vários experimentos com diferentes tempos de duração (inicialmente com 60 s de duração) e, a cada duas amostras, foi realizado o teste de aderência Kruskal-Wallis para verificar se as amostras pertencem a mesma população. O teste Kruskal-Wallis é uma versão não-paramétrica[3] do teste ANOVA (*Analysis of Variance*) (JAIN, 1991, pg. 250-253) e uma extensão do teste de Wilcoxon para mais de dois grupos. Esse teste compara as medianas dos grupos das amostras para verificar se a amostras pertencem a mesma população. Devido, o teste

[3]Tipo de teste o qual as distribuições das populações envolvidas não devem pertencer a uma família específica de distribuições de probabilidade, tais como, normal ou exponencial.

ser não-paramétrico e comparar medianas, ao invés de médias, o consideramos adequado para esse problema.

Os testes de aderência foram realizados usando o algoritmo de Kruskal-Wallis, que retorna o valor p para a hipótese nula. A hipótese nula supõe que os dados a serem avaliados pertencem a mesma população (os dados provem de uma mesma distribuição com medianas iguais). Enquanto que, a hipótese alternativa supõe que os dados não são da mesma população. Então, se o valor p for menor que $0{,}01$, o teste rejeita a hipótese nula com um nível de significância de 1%. Caso contrário, $p > 0{,}01$, o teste não rejeita a hipótese nula.

Foram realizados três experimentos com tempos de duração iguais a 60 s, 2 h e 5 h para uma taxa de transmissão de 500 Kbits/s. Uma foto da plataforma experimental é apresentada na Figura 4.22. Os resultados dos testes de aderência Kruskal-Wallis são apresentados na Tabela 4.10. No teste para as amostras de 60 s/2 h, a hipótese nula foi rejeitada para todas as mensagens, uma vez que os valores p foram inferiores a $0{,}01$. Para as amostras de 2 h/5 h, todos as mensagens passaram no teste. Portanto, um tempo de duração do experimento a partir de 2 h é suficiente para caracterizar o comportamento dos tempos de resposta das mensagens.

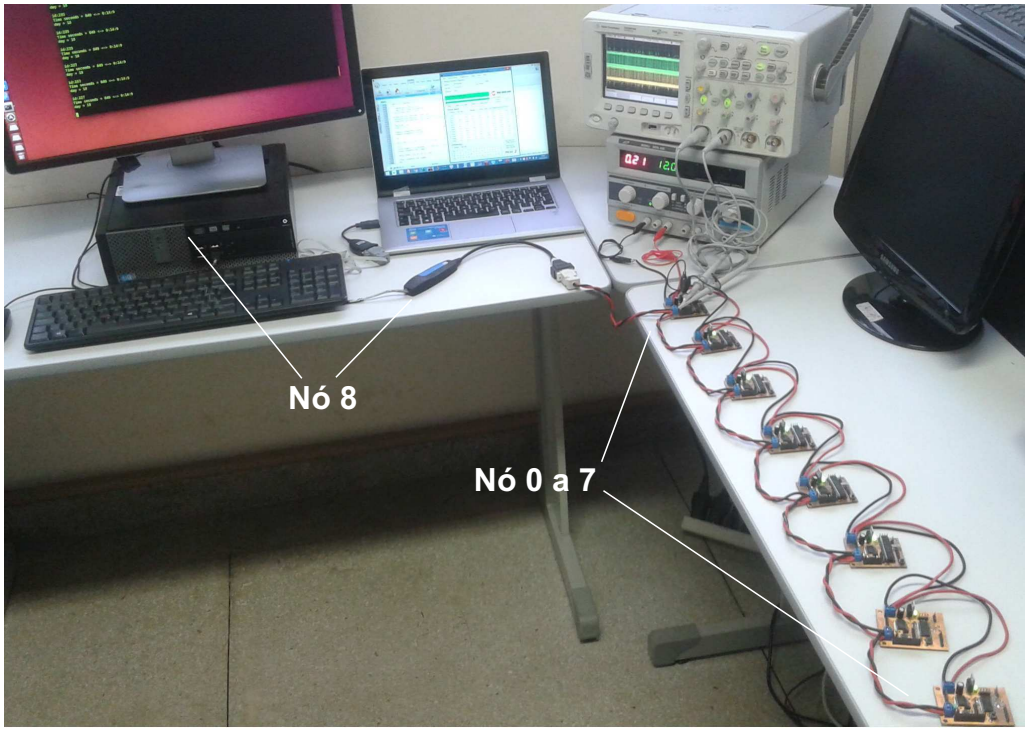


Figura 4.22: Foto da plataforma experimental da rede CAN ideal.

A partir dos resultados obtidos via experimento com duração de 5 h, foram obtidas as funções de distribuição acumulada (FDAs) dos tempos de resposta das mensagens. Esses resultados foram comparados com os resultados obtidos via verificação estatística. Os gráficos das FDAs dos tempos de resposta das mensagens estão representadas nas Figuras 4.23 a 4.30. Nas FDAs obtidas via verificação estatística há um limite superior e inferior com confidência de 99%.

Tabela 4.10: Valor p do teste de aderência Kruskal-Wallis para as amostras de 60 s/2 h e 2 h/5 h em uma rede CAN com *Offset* Dinâmico, Relógios e Barramento Ideais.

$Msg\ (i)$	$Amostras\ 60s/2h$	$Amostras\ 2h/5h$
0	**2,9782e-4**	0,0521
1	**3,0388e-4**	0,0177
2	**4,0422e-8**	0,0221
3	**8,8560e-5**	0,2439
4	**1,5143e-8**	0,0100
5	**6,4032e-4**	0,0382
6	**2,5701e-8**	0,0288
7	**2.7662e-9**	0,0198

Então, para validar o modelo é necessário verificar se as FDAs obtidas experimentalmente são iguais as FDAs obtidas via verificação estatística. Para isso, foi utilizado o teste de aderência de Kolmogorov-Smirnov (teste KS) para duas amostras (JAIN, 1991, pp. 462-465). Esse teste é não-paramétrico e tem como objetivo verificar se duas amostras são definidas pela mesma distribuição de probabilidade. O teste KS é muito sensível as diferenças de posição e forma das FDAs, por isso é muito utilizada nessas verificações.

Para realizar esse teste, foi utilizado o algoritmo de Kolmogorov-Smirnov, que compara as FDAs x e y das amostras em avaliação. É possível também especificar o nível de significância. Nesse teste, a hipótese nula supõe que os dados em x e y são da mesma distribuição. A hipótese alternativa supõe que x e y pertencem a distribuições de probabilidade diferentes. A função retorna os parâmetros h e o valor p. Se h for igual a 1, a hipótese nula é rejeitada com o nível de significância especificado (o valor p é menor que o nível de significância). Se h for zero a hipótese nula é aceita e o valor p terá algum valor maior que o nível de significância e no máximo 1.

Os resultados dos testes (h e valor p) entre a FDA obtida no experimento de 5 h e a FDA obtida via verificação estatística são apresentados na Tabela 4.11. Em todos os testes, o valor h obtido foi igual a zero e, consequentemente, o valor p foi maior que o nível de significância de 0,01. Portanto, o modelo da rede CAN com *offset* dinâmico, relógios e barramento ideais foi validado com sucesso.

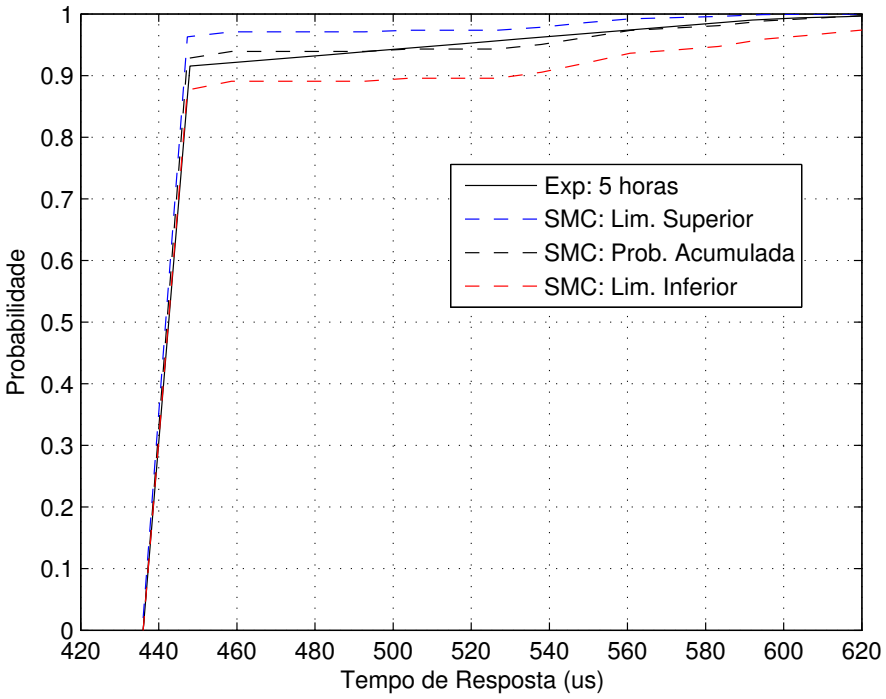


Figura 4.23: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 0 obtidas via experimento de 5 h e verificação estatística.

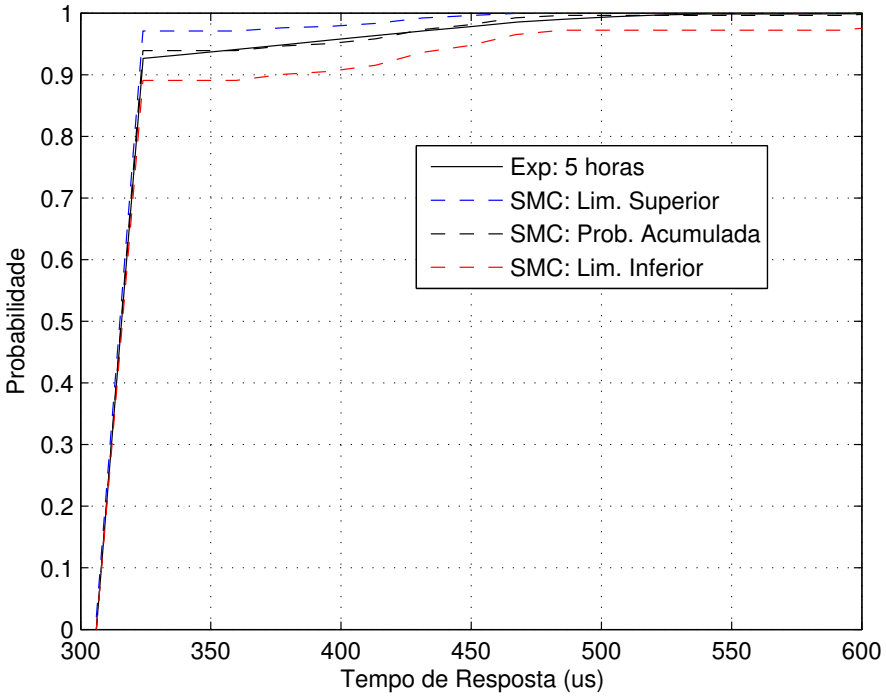


Figura 4.24: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 1 obtidas via experimento de 5 h e verificação estatística.

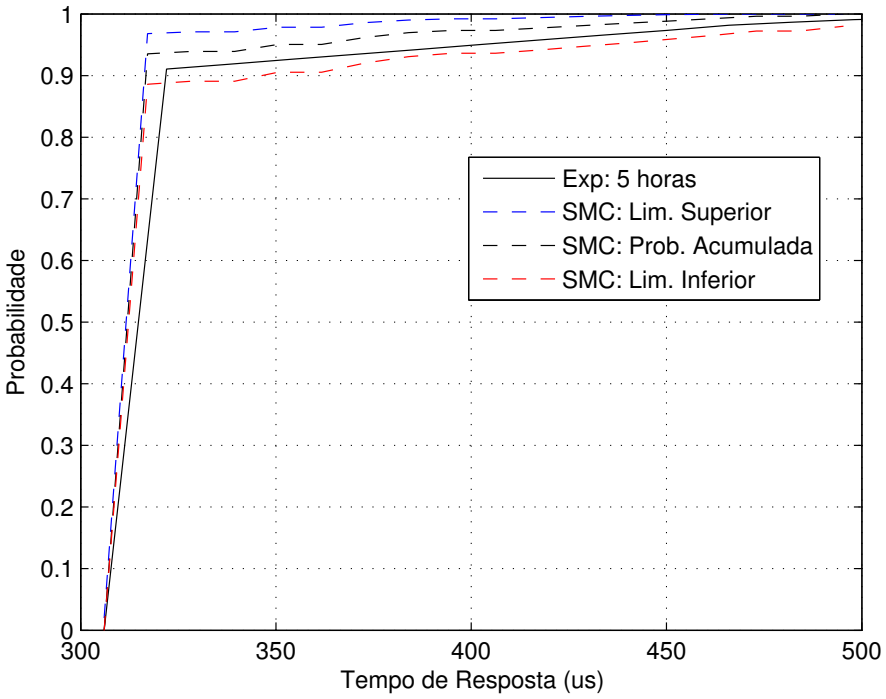


Figura 4.25: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 2 obtidas via experimento de 5 h e verificação estatística.

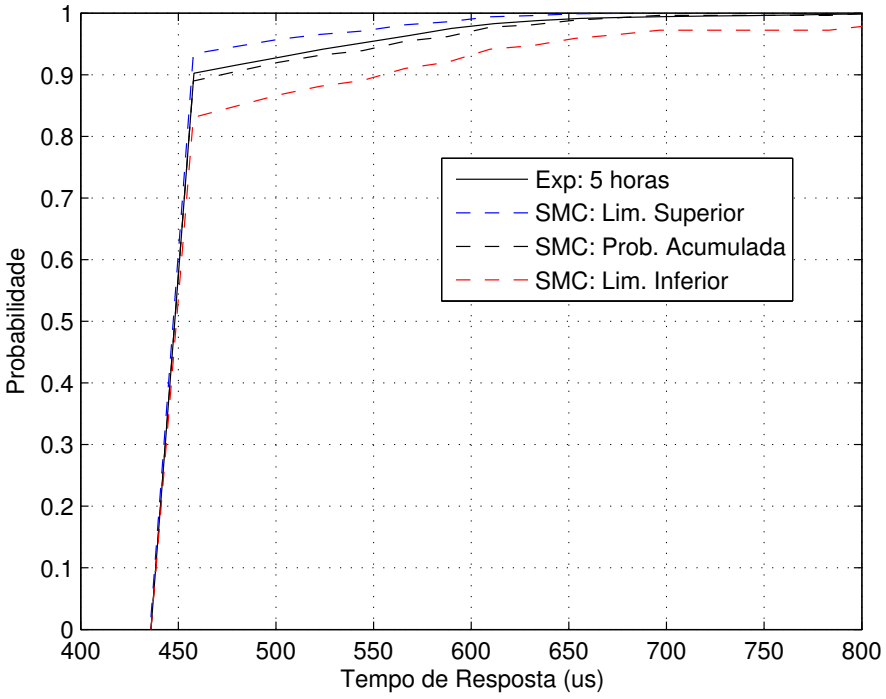


Figura 4.26: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 3 obtidas via experimento de 5 h e verificação estatística.

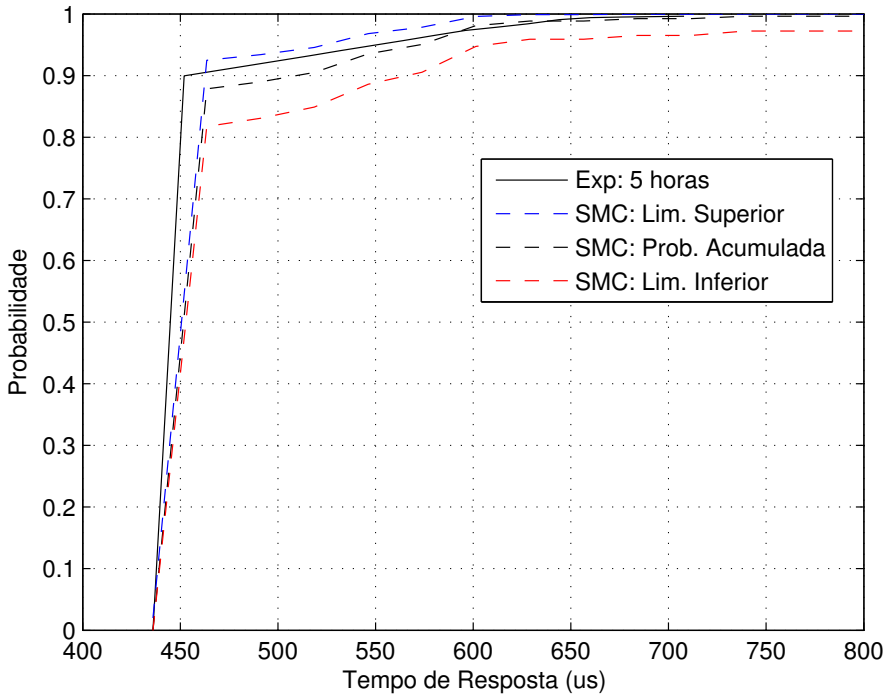


Figura 4.27: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 4 obtidas via experimento de 5 h e verificação estatística.

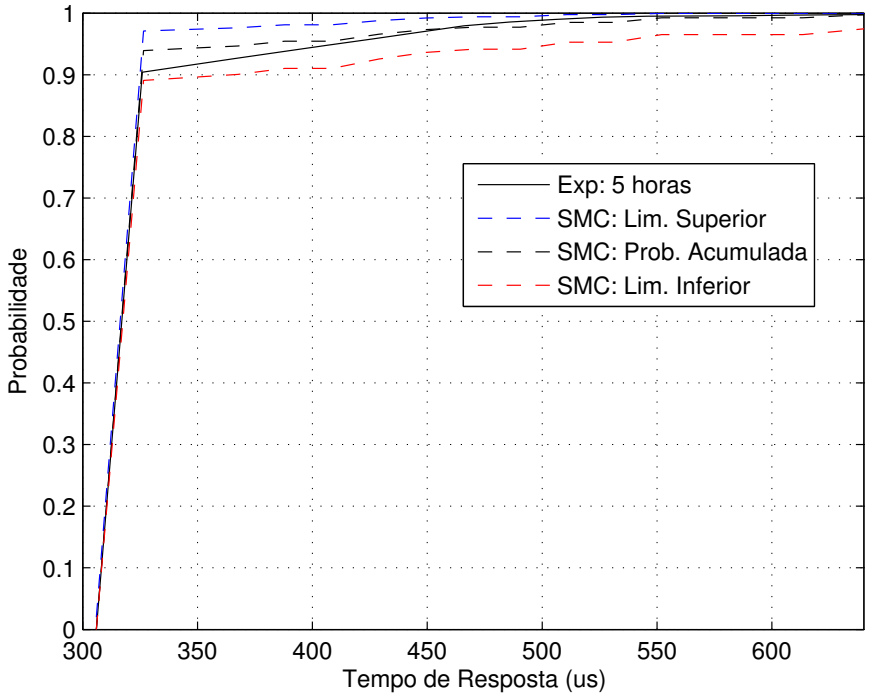


Figura 4.28: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 5 obtidas via experimento de 5 h e verificação estatística.

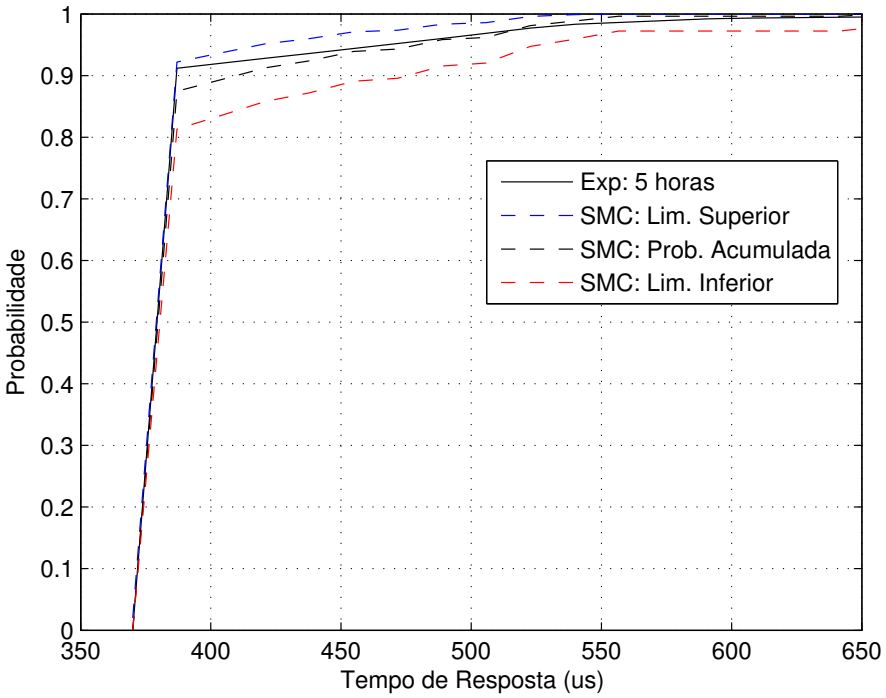


Figura 4.29: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 6 obtidas via experimento de 5 h e verificação estatística.

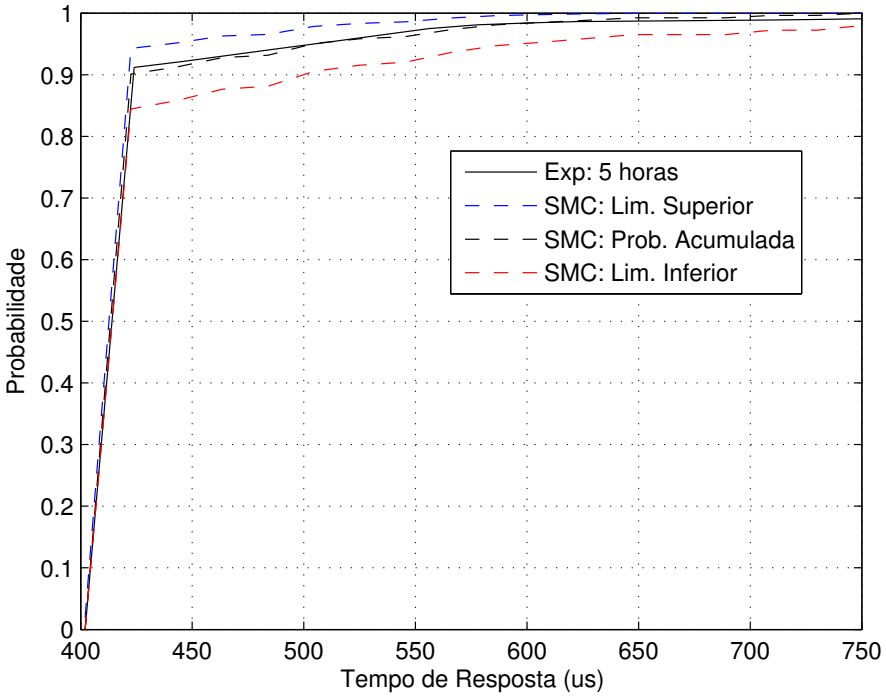


Figura 4.30: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 7 obtidas via experimento de 5 h e verificação estatística.

Tabela 4.11: Valores h e p do teste Kolmogorov-Smirnov para as FDAs dos tempos de resposta obtidas experimentalmente e por verificação estatística em uma rede CAN com *Offset* Dinâmico, Relógios e Barramento Ideais.

	Exp/SMC (FDA)	
$Msg\ (i)$	h	p
0	0	0,9448
1	0	0,4255
2	0	0,7088
3	0	0,9448
4	0	0,0982
5	0	0,4255
6	0	0,0982
7	0	0,2182

4.5 Modelo formal da rede CAN com *offset* dinâmico, relógios ideais e barramento real

Nesta seção apresenta-se o modelo 3 (Ver Tabela 4.1), onde os *offsets* das mensagens são dinâmicos, o relógio de cada nó CAN é ideal e o barramento não é livre de erros (barramento real).

4.5.1 Trabalhos relacionados

Vários trabalhos foram desenvolvidos com o objetivo de determinar os tempos de resposta de mensagens CAN periódicas considerando erros de transmissão devido a interferências no barramento da rede. Todos os resultados são baseados na solução analítica desenvolvida por Tindell e Burns (1994b) para determinar os tempos de resposta no pior caso de mensagens CAN.

O primeiro trabalho nesse contexto foi desenvolvido por Tindell e Burns (1994a), que estendeu o seu trabalho anterior (TINDELL; BURNS, 1994b) para considerar erros de transmissão usando um modelo de erros determinístico. Nesse modelo, considera-se que o número de erros de transmissão pode ser limitado superiormente durante um determinado intervalo de tempo. Desse modo, é possível determinar um limite superior do tempo de resposta de cada mensagem sob condições de erro de transmissão e verificar se os *deadlines* serão satisfeitos. Punnekkat, Hansson e Norström (2000) estenderam o modelo de erros de Tindell e Burns (1994a) para um modelo mais geral, que considera a existência de várias fontes de interferência no modelo.

Navet, Song e Simonot (2000) afirmaram que o modelo de erros de Tindell e Burns (1994a) não é uma consideração geral, visto que erros de transmissão são fenômenos aleatórios e ten-

dem a seguir melhor a leis probabilísticas, ao invés de leis determinísticas. Então, Navet, Song e Simonot (2000) usaram a análise de Tindell e Burns (1994a) e propuseram o conceito de probabilidade de defeito do *deadline* no pior caso (WCDFP − *Worst-Case Deadline Failure Probability*) e um modelo de erros flexível fazendo suposições na freqüência e gravidade (simples e *burst*) de erro, onde a ocorrência de erros segue um processo de Poisson generalizado. Essa análise fornece um método para calcular a probabilidade de uma mensagem não satisfazer o seu *deadline*.

Broster, Burns e Rodrigues-Navas (2002) identificaram duas imprecisões na análise desenvolvida por Navet, Song e Simonot (2000), que aumentam o grau de pessimismo na estimação do WCDFP, e propuseram uma análise que fornece um WCDFP preciso e sem excesso de pessimismo. Essa análise considera que as falhas são aleatoriamente distribuídas seguindo uma distribuição de Poisson.

Sá et al. (2012) desenvolveram um modelo de erros probabilístico para determinar as distribuições dos tempos de resposta considerando o número de retransmissão de mensagens devido a erros, utilizando a técnica de verificação de modelos estatísticos.

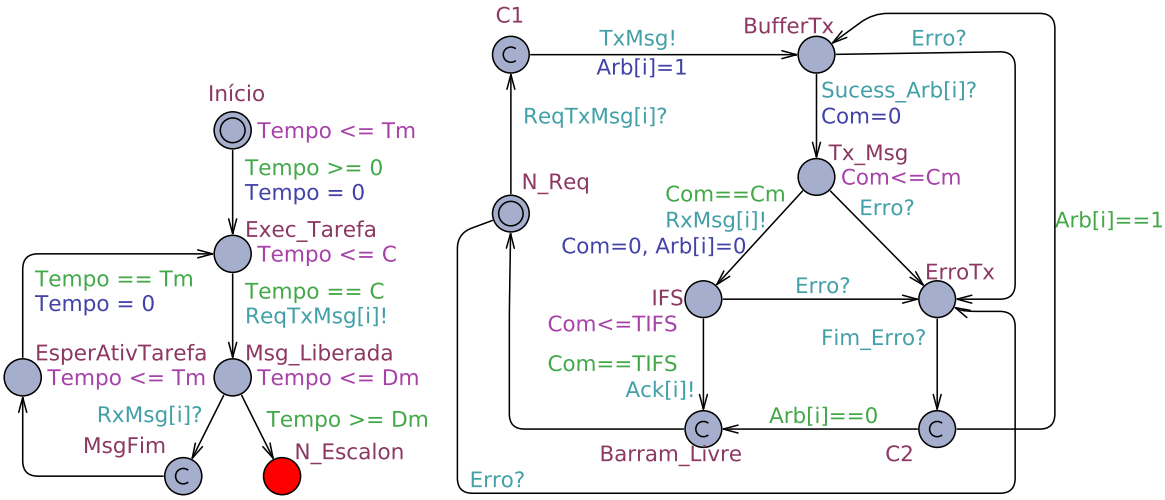
Exceto o trabalho de Sá et al. (2012), todos os trabalhos citados adotam as seguintes suposições, que na prática são incorretas e geram ou podem gerar resultados incorretos: (1) são baseados na análise otimista de Tindell e Burns (1994b)[4]; (2) consideram que o erro na mensagem sempre ocorre no fim da mensagem; (3) desconsideram que pode ocorrer um erro no barramento quando este estiver livre e (4) desconsideram que pode ocorrer um erro durante a intermissão.

4.5.2 Modelo em autômato temporizado

Nesta seção, apresenta-se o conjunto de autômatos temporizados utilizado para modelar a rede CAN com erros no meio de comunicação. O modelo é constituído pelos cinco autômatos (tarefa, controlador CAN, arbitragem, interferência e erros) representados na Figura 4.31. O código UPPAAL do modelo é apresentado na Seção C.3 do Apêndice C.

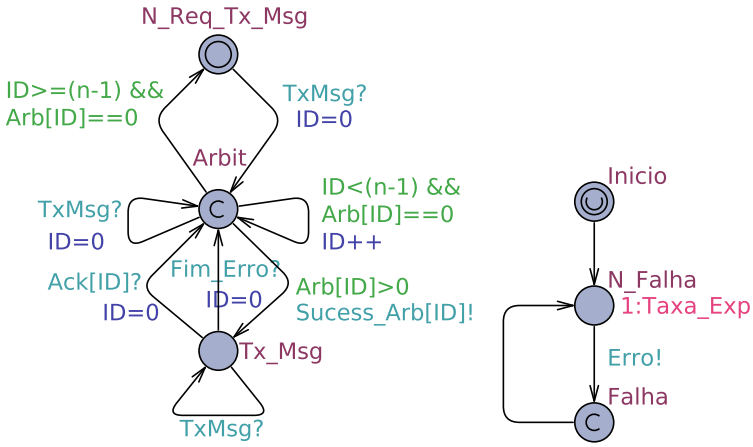
O autômato tarefa (Figura 4.31(a)) modela o comportamento temporal das atividades da tarefa τ_{m_i} relacionada a mensagem m_i. O autômato possui um relógio local *Tempo* usado para marcar o tempo de resposta e o valor dos seguintes atributos: *offset* O_{m_i}; período Tm; tempo de computação C e deadline Dm. A primeira instância da tarefa τ_i é liberada no lugar *Início* com um *offset* variando de 0 a Tm unidades de tempo. O *offset* é modelado pela invariante de estado $Tempo <= T$ e o guarda $Tempo >= 0$. As próximas instâncias serão liberadas periodicamente no lugar *EsperAtivTarefa*. Após o tempo de offset, a transição para o lugar *Exec_Tarefa* será liberada e o relógio $Tempo$ será zerado. Esse lugar representa a tarefa em

[4]Davis et al. (2007) provaram que essa análise é incompleta e resulta em tempos de resposta otimistas.



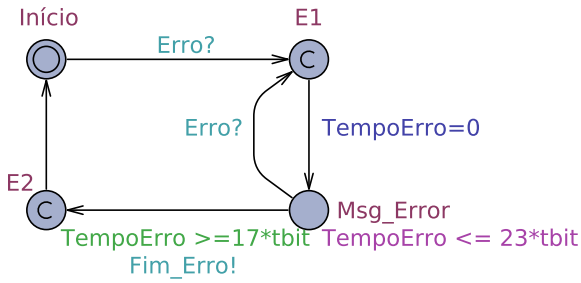
(a) Tarefa.

(b) Controlador CAN.



(c) Arbitragem.

(d) Interferência.



(e) Erros.

Figura 4.31: Modelo em autômatos temporizados da rede CAN com *offset* dinâmico, relógios ideais e barramento real.

execução durante um tempo de computação C, modelado pela invariante $Tempo <= C$ e pelo guarda $Tempo == C$. Quando o relógio $Tempo$ for igual a C unidades de tempo, a transição para o lugar *Msg_Liberada* será liberada e uma requisição para transmitir a mensagem m_i será enviada para o autômato controlador CAN (Figura 4.31(b)) usando o canal *ReqTxMsg[i]!*. O autômato permanecerá nesse lugar até o fim da transmissão da mensagem, que será comunicada pelo autômato controlador CAN por meio do canal *RxMsg[i]?*. Assim que a transmissão da mensagem for finalizada, a transição para o lugar committed *MsgFim* será liberada e, em seguida, ocorrerá a transição para o lugar *EsperAtivTarefa*, onde permanecerá até iniciar um novo período, modelado pela invariante $Tempo <= Tm$ e pelo guarda $Tempo == Tm$. Se o valor do relógio *Tempo* for maior ou igual ao *deadline* Dm, enquanto o autômato estiver no lugar *Msg_Liberada*, uma transição para o lugar *N_Escalon* ocorrerá, indicando que o *deadline* não foi satisfeito.

O autômato controlador CAN (Figura 4.31(b)) modela o comportamento temporal das mensagens de dados e erro de acordo com a especificação do protocolo CAN. O autômato possui um relógio local *Com* usado para marcar o tempo de transmissão da mensagem de dados, mensagem de erro e o intervalo entre mensagens. Inicialmente, o autômato controlador CAN está no lugar inicial *N_Req*. Esse lugar representa o estado em que o controlador CAN do nó não recebeu uma requisição de transmissão de mensagem. Com exceção dos três lugares committed (*C1*, *C2* e *Barram_Livre*) e do lugar *ErroTx*, em todos os outros lugares do autômato há duas bordas de saída (duas possíveis transições) para outro lugar, onde uma dessas transições será liberada sempre que ocorrer um erro no barramento. Esse evento será sinalizado pelo autômato interferência (Figura 4.31(d)) usando o canal broadcast *Erro!*. Quando o autômato receber uma requisição de transmissão de mensagem via o canal *ReqTxMsg[i]?* enviado pelo autômato tarefa, uma transição do lugar *N_Req* para o lugar *BufferTx* será liberada, o sinal *TxMsg!* será enviado para o autômato arbitragem (Figura 4.31(c)) e o *array Arb[i]*, indexado com o valor do identificador da mensagem m_i, será ajustado para '1', indicando que a mensagem está pronta para entrar no processo de arbitragem. Assim que a mensagem vencer o processo de arbitragem, o autômato arbitragem sincronizará com o autômato controlador CAN, via o canal *Sucess_Arb[i]?* e a transição para o lugar *Tx_Msg* será liberada e o relógio *Com* será zerado. O autômato permanecerá nesse lugar até o relógio Com marcar Cm unidades de tempo (modelado pela invariante de estado $Com <= Cm$ e o guarda $Com == Cm$), indicando que a mensagem foi transmitida com sucesso. Então, a transição para o lugar *IFS* será liberada, o sinal $RxMsg!$ será enviado para o autômato tarefa e o relógio Com e a variável $Arb[i]$ será zerada. O lugar *IFS* representa a intermissão com duração de três tempos de *bit* ($TIFS = 3 * \tau_{bit}$). Assim que o relógio Com marcar três tempos de *bit*, a transição para o lugar committed *N_Req* será liberada e a informação de que o barramento está livre será enviada para o autômato arbitragem por meio do canal *Ack[i]!*. Imediatamente, a transição para o lugar inicial *N_Req* será liberada.

Se algum erro no barramento ocorrer quando o autômato estiver no lugar *N_Req* ou no lugar *Tx_Msg* ou no lugar *IFS*, uma transição para o lugar *ErroTx* será liberada. O autômato ficará no lugar *ErroTx* até a mensagem de erro, modelada pelo autômato erros (Figura 4.31(e)), ser transmitida com sucesso. A ocorrência desse evento é indicada pela recepção do sinal *Fim_Erro?* transmitida pelo autômato erro. Após a recepção dessa informação, o autômato passará para o lugar committed *C2*, e duas transições poderão ocorrer: a transição para o lugar *N_Req* precedida pela transição para o lugar committed *Barram_Livre*, ou a transição para o lugar *BufferTx*. A primeira transição será liberada se a condição $Arb[i] == 0$ for satisfeita. Esse é o caso onde um erro ocorreu no barramento e o controlador CAN da mensagem m_i não tinha nenhuma mensagem pronta para ser transmitida ou a mensagem já havia sido transmitida com sucesso. A segunda transição será liberada se a condição $Arb[i] == 1$ for satisfeita. Esse é o caso onde um erro ocorreu no barramento durante a transmissão da mensagem m_i ou quando essa mensagem estava no buffer de transmissão aguardando o resultado do processo de arbitragem.

O controle de acesso ao meio é modelado pelo autômato arbitragem (Figura 4.31(c)). O lugar inicial *N_Req_Tx_Msg* representa o estado onde não existe nenhuma mensagem pronta para ser transmitida. Assim que pelo menos um controlador CAN tiver alguma mensagem pronta para ser transmitida, o autômato receberá um sinal via o canal *TxMsg?* emitido pelo respectivo autômato controlador CAN. Então, a transição para o lugar committed *Arbit* será liberada e a variável *ID* será zerada. A variável *ID*, assim como a variável *i* (usada nos autômatos tarefa e controlador CAN) representam o identificador da mensagem m_i. No lugar *Arbit*, a mensagem com maior prioridade que requisitou uma transmissão será escolhida para ser transmitida. Para isso, o *array Arb[ID]* será verificado item por item, com o índice *ID* variando de 0 a *n-1*, onde *n* equivale ao número de identificadores (mensagens) do sistema. Os identificadores das mensagens são ordenados de forma crescente. A mensagem com maior prioridade possui *ID = 0*, e a mensagem com menor prioridade *ID = n-1*. Então, o primeiro elemento do *array Arb[ID]* com valor maior que 0 vencerá o processo de arbitragem, e um sinal via o canal *Sucess_Arb[ID]!* será enviado para o autômato controlador CAN que transmitirá a mensagem com identificador ID. O autômato permanecerá no lugar *Tx_Msg* até receber o sinal de reconhecimento via *Ack[ID]?*, indicando que a mensagem foi transmitida, ou até receber um sinal via *Fim_Erro?*, indicando que houve um erro na mensagem e que o barramento já está livre novamente para iniciar um novo processo de arbitragem. Em qualquer um dos casos, a variável *ID* será zerada e o autômato retornará para o lugar *Arbit*, onde verificará se existe alguma mensagem pendente para ser transmitida. Caso não exista, o autômato retornará para o seu lugar inicial *N_Req_Tx_Msg*.

As falhas devido a interferências no barramento são modeladas pelo autômato interferência (Figura 4.31(d)). Inicialmente, ocorrerá uma transição instantânea do lugar inicial urgente *Início* para o lugar *N_Falha*. Esse lugar é usado para modelar um período onde não existe falha

no barramento. Enquanto que o lugar committed *Falha* é usado para indicar que houve uma falha no barramento. A transição do lugar *N_Falha* para o lugar *Falha* será liberada em um instante t que segue uma distribuição exponencial com função de distribuição de probabilidade indicada pela Equação 4.10, onde λ é a taxa de ocorrência de erros por unidade de tempo. A taxa está indicada no lugar *N_Falha* pela constante *Taxa_Exp*. A ocorrência de uma falha é indicada pelo envio do sinal broadcast *Erro!* para os autômatos controlador CAN e erros (Figura 4.31(e)). Após a ocorrência de uma falha, o autômato passa do lugar *Falha* para o lugar *N_Falha* imediatamente.

$$Pr(Trans \leq t) = 1 - e^{-\lambda t} \tag{4.10}$$

O autômato erro (Figura 4.31(e)) modela o período de sinalização do erro, que equivale ao tempo de transmissão da mensagem de erro pelos nós da rede. O autômato permanecerá inicialmente no lugar *Início* até ocorrer um erro no barramento, indicado pela recepção do sinal *Erro?*. Então, a transição para o lugar committed *E1* é liberada e, imediatamente, ocorrerá uma transição para o lugar *Msg_Erro* e o relógio *TempoErro* será zerado. Esse lugar modela o período de transmissão da mensagem de erro, com duração de 17 a 23 tempos de *bit*, que é modelado pela invariante de tempo $TempoErro <= 23 * tbit$ e pelo guarda $TempoErro >= 17 * tbit$. Caso não ocorra nenhum erro no barramento, durante a transmissão da mensagem de erro, a transição para o lugar committed *E2* será liberada e o sinal *Fim_Erro!* será enviado para os autômatos controlador CAN e arbitragem, indicando o fim da transmissão da mensagem de erro. Caso ocorra algum erro durante a transmissão da mensagem de erro (lugar *Msg_Erro*), uma transição para o lugar committed *E1* será liberada e, imediatamente, o relógio *TempoErro* será zerado e o autômato retornará novamente para o lugar *Msg_Erro*, onde novamente uma mensagem de erro será transmitida.

4.5.3 Estudos de caso

Foram realizadas verificações com o conjunto de oito mensagens, considerando a taxa de transmissão de 500 K*bits*/s e a taxa de erro de 30 erros/s, para determinar se a probabilidade do *deadline* da mensagem m_i será satisfeita. Essa propriedade foi descrita na linguagem de especificação no formalismo da ferramenta UPPAAL, conforme descrito a seguir, onde *TARi* é o autômato tarefa relacionado a mensagem m_i, com i variando de 0 a 7, *Tempo* é um relógio local, Di é o *deadline* e *MsgFim* é o lugar do autômato tarefa *TARi* que representa o fim da transmissão da mensagem. Em outras palavras, a propriedade verifica qual a probabilidade do lugar *MsgFim* ser alcançado e o valor do relógio *Tempo* ser menor ou igual ao *deadline* Di. No total, foram verificadas oito propriedades, uma por mensagem.

Pr[TARi.Tempo <= Di] (<> TARi.MsgFim)

O tempo gasto na verificação foi de 1,5 s. Os gráficos das funções de distribuição acumulada (limites inferior, médio e superior) do conjunto de mensagens são apresentados nas Figuras 4.32 a 4.39. Na seção seguinte, esse modelo será validado experimentalmente.

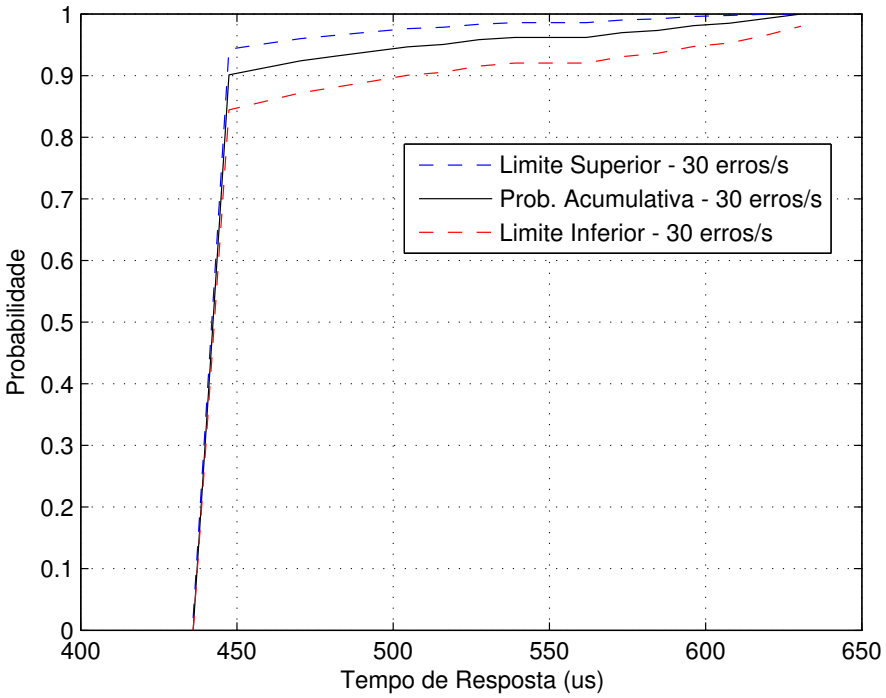


Figura 4.32: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 0 considerando um barramento real com taxa de erro igual a 30 erros/s.

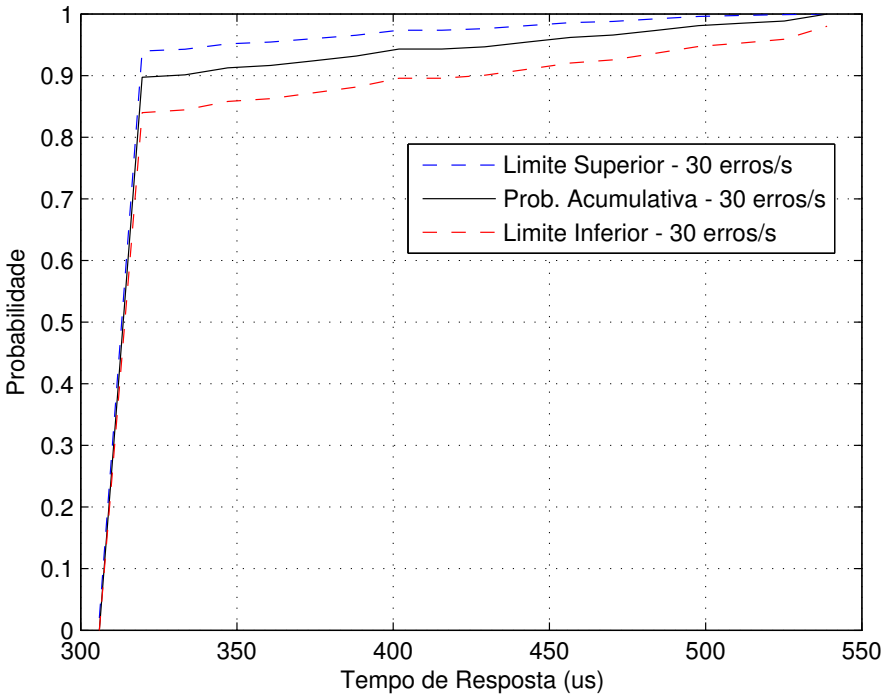


Figura 4.33: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 1 considerando um barramento real com taxa de erro igual a 30 erros/s.

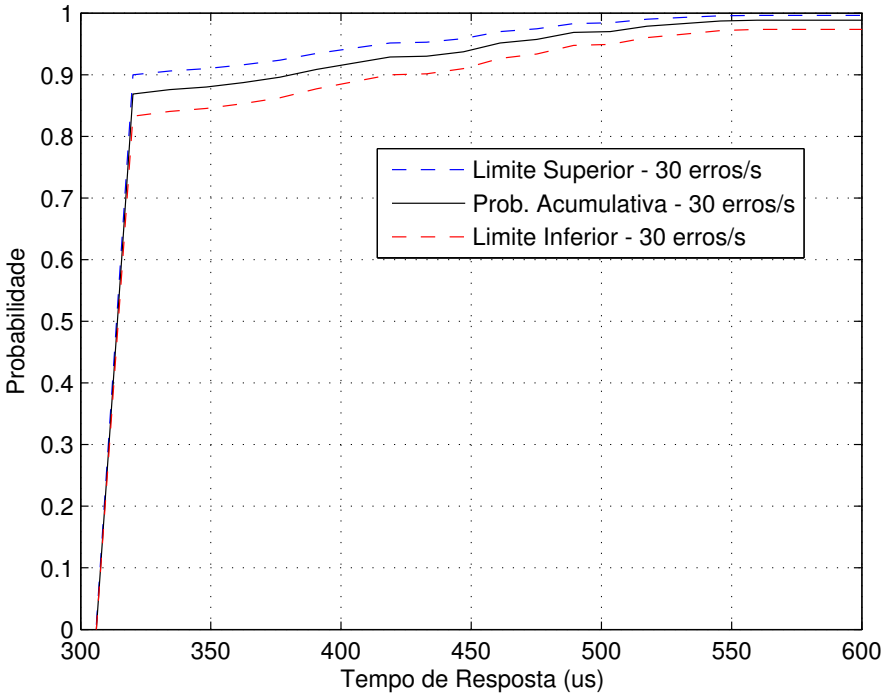


Figura 4.34: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 2 considerando um barramento real com taxa de erro igual a 30 erros/s.

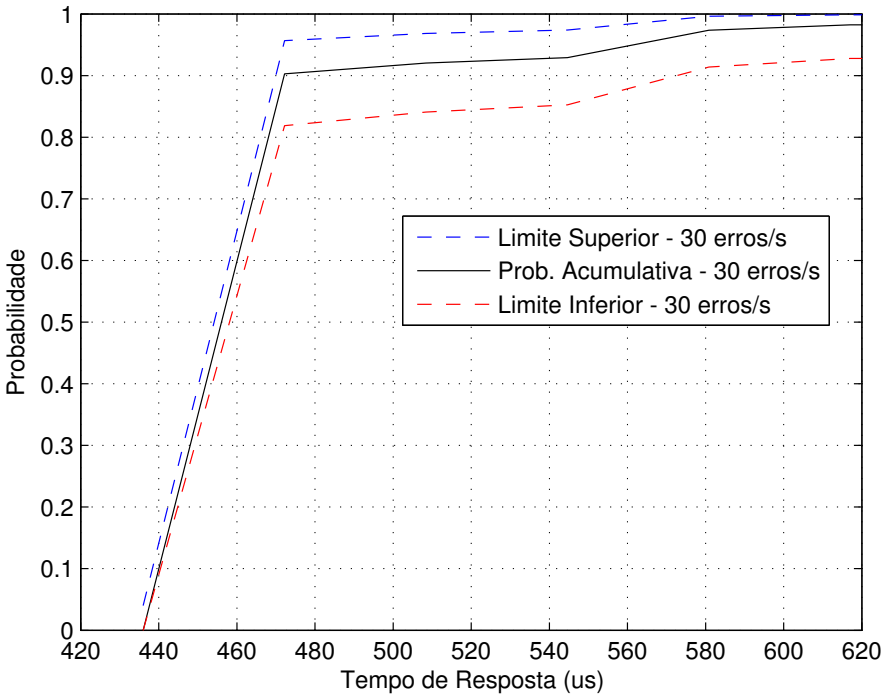


Figura 4.35: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 3 considerando um barramento real com taxa de erro igual a 30 erros/s.

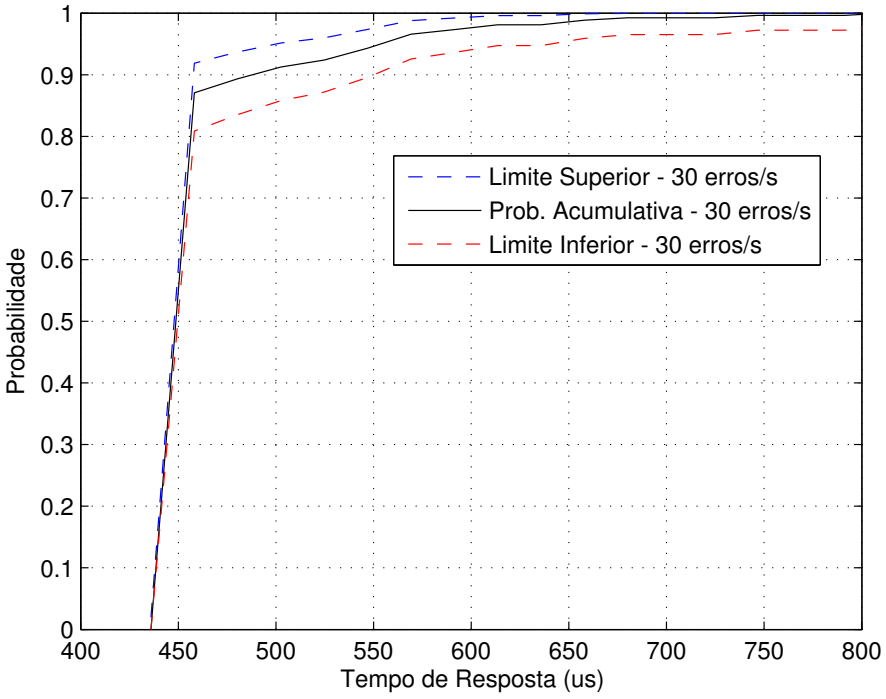


Figura 4.36: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 4 considerando um barramento real com taxa de erro igual a 30 erros/s.

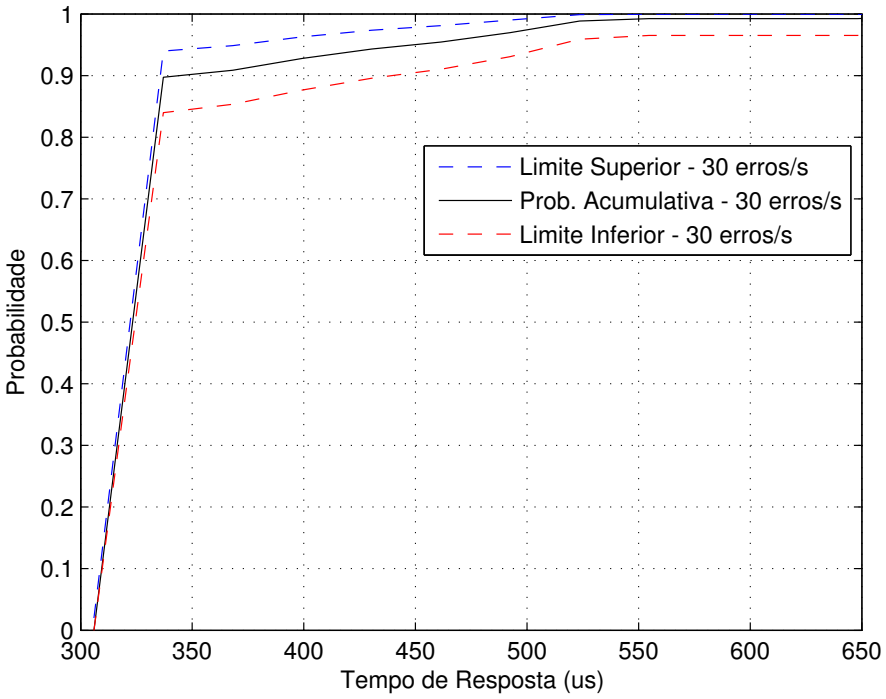


Figura 4.37: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 5 considerando um barramento real com taxa de erro igual a 30 erros/s.

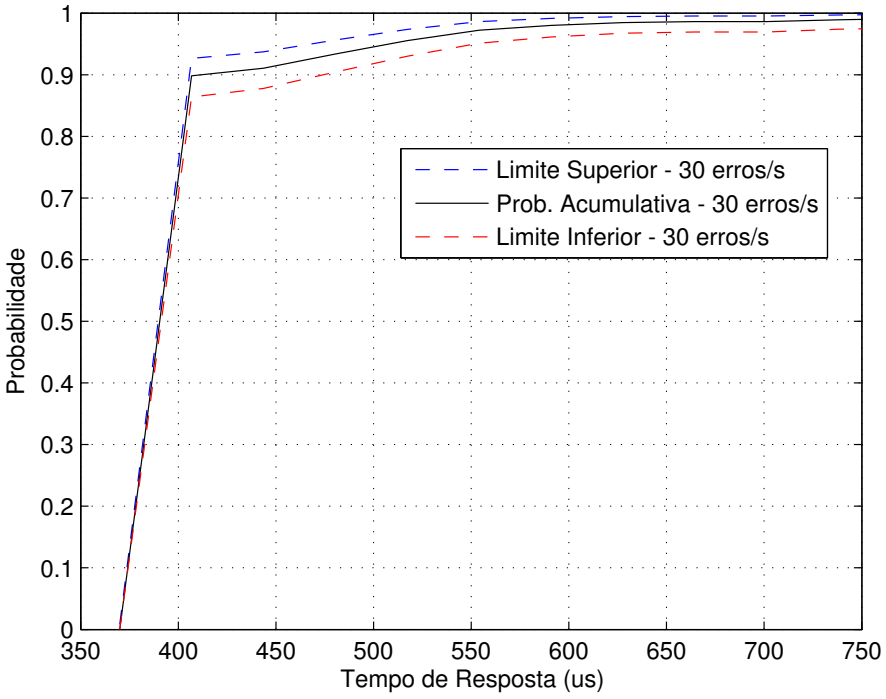


Figura 4.38: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 6 considerando um barramento real com taxa de erro igual a 30 erros/s.

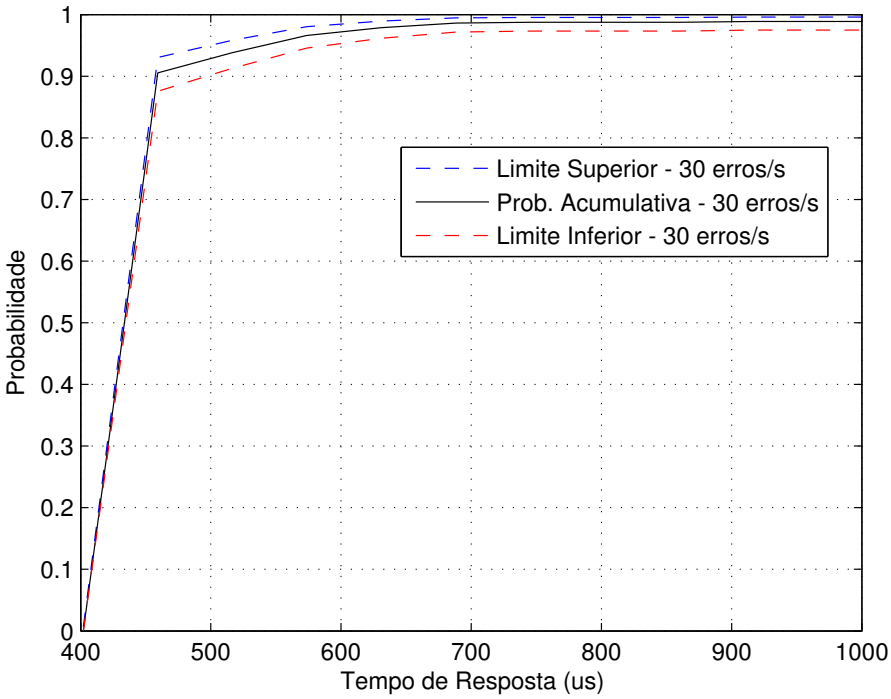


Figura 4.39: Gráfico das funções de distribuição acumulada dos tempos de resposta das mensagem 7 considerando um barramento real com taxa de erro igual a 30 erros/s.

4.6 Validação do modelo formal da rede CAN com *offset* dinâmico, relógios ideais e barramento real

Nesta seção será apresentada uma plataforma, chamada de plataforma experimental para rede CAN real, para validação do modelo da rede CAN com *offset* dinâmico, relógios ideais e barramento real (modelo 3).

4.6.1 Plataforma Experimental Rede CAN Real

A plataforma experimental, representada na Figura 4.40, consiste de uma rede CAN com dez nós. Os nós CAN 0 a 7 são formados pelo microcontrolador PIC16F876A (MICROCHIP, 2001), controlador MCP2515 (MICROCHIP, 2005) e *transceiver* MCP2551 (MICROCHIP, 2003). O nó 8, chamado de injetor de falhas, consiste de uma placa Stellaris LM4F120 (INSTRUMENTS, 2012) baseada no microcontrolador ARM Cortex-M4F de 32 *bits* com unidade ponto flutuante e frequencia de 80 MHz, e um *transceiver* MCP2551. O nó 9 consiste do dispositivo CAN *leaf professional* (KVASER, 2014) conectado via USB a um computador pessoal com sistema operacional linux.

A ideia da plataforma experimental é semelhante à desenvolvida na seção 4.4.1. Nessa plataforma, cada nó (nós CAN de 0 a 7) envia uma única mensagem periódica com comprimento

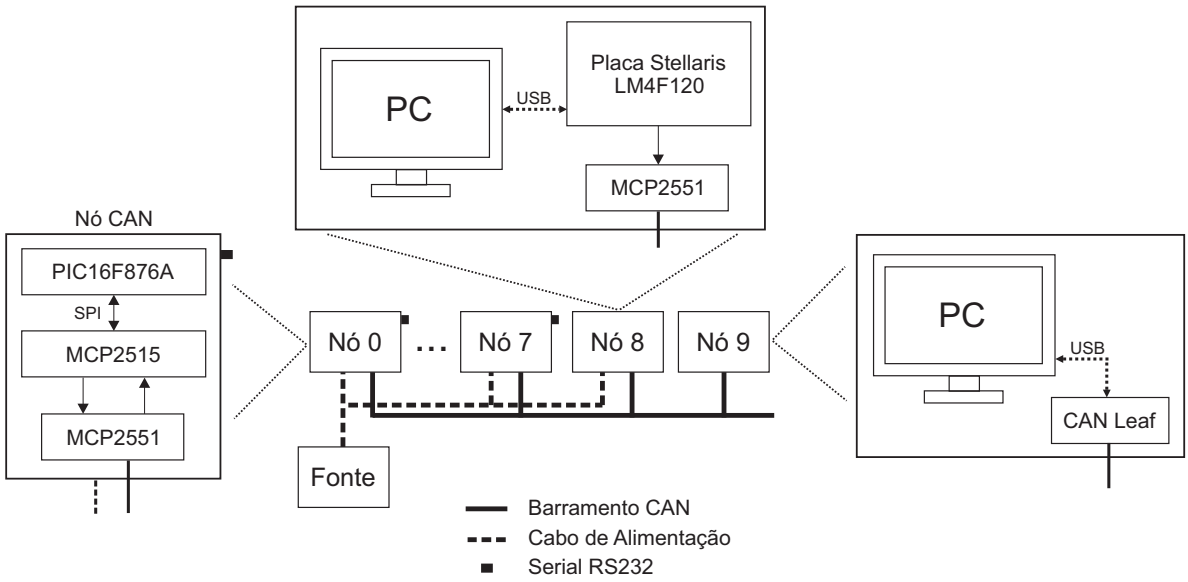


Figura 4.40: Representação da plataforma experimental da rede CAN real com os componentes de cada nó.

fixo, mas o barramento não é livre de erros. Cada um desses nós também será responsável por marcar e armazenar os tempos de resposta das mensagens transmitidas ao longo do experimento, de acordo com um esquema idêntico ao apresentado na seção 4.4.1. O nó 9 será responsável pelo gerenciamento da rede. As seguintes atividades serão executadas por esse nó: envio de mensagens informando aos demais nós o início e o fim do experimento (o tempo do experimento é marcado por esse nó) e recepção das mensagens transmitidas pelos nós.

Os nós CAN de 0 a 7 armazenarão os tempos de respostas das mensagens transmitidas ao longo do experimento de modo idêntico ao apresentado na seção 4.21. A largura dos intervalos em cada nó foi definida a partir da observação de resultados experimentais. Os intervalos são também definidos pelas Equações 4.7, 4.8 e 4.9 (ver seção 4.4.1). A largura do intervalo dos tempos de resposta de cada nó e os intervalos 1, 2, 23 e 24 para um tempo de transmissão de um *bit* igual a 2 μs estão indicados na Tabela 4.12, onde τ_{bit} é o tempo de transmissão de um *bit*, NI é o número do intervalo e L_{m_i} é a largura do intervalo da mensagem m_i.

O componente principal dessa plataforma é o nó injetor de falhas. Na seção seguinte apresenta-se os trabalhos relacionados à injeção de falhas em uma rede CAN. Em seguida, será apresentado o injetor de falhas proposto neste trabalho. É importante distinguir, no contexto deste trabalho, o conceito de falha, erro e defeito (BROSTER, 2003). Uma falha (e.g. interferência eletromagnética) causa uma mudança indesejada no nível do sinal no barramento. Consequentemente, uma mensagem pode ser corrompida (erro). Isso resultará na retransmissão da mensagem, ou seja, um atraso extra é gerado, e o *deadline* poderá não ser cumprido, resultando em um serviço não entregue de acordo com o especificado (defeito).

Tabela 4.12: Largura do intervalo dos tempos de resposta de cada nó e os intervalos 1, 2, 23 e 24 para $\tau_{bit} = 2$ μs sem o tempo de computação das tarefas.

$Msg\ (i)$	L_{m_i}	$NI = 1$	$NI = 2$	$NI = 23$	$NI = 24$
0	6*τ_{bit}	[0, 240]	(240, 252]	(492, 504]	(504, ∞)
1	9*τ_{bit}	[0, 182]	(182, 200]	(560, 578]	(578, ∞)
2	8*τ_{bit}	[0, 180]	(180, 196]	(516, 532]	(532, ∞)
3	8*τ_{bit}	[0, 244]	(244, 260]	(580, 596]	(596, ∞)
4	8*τ_{bit}	[0, 244]	(244, 260]	(580, 596]	(596, ∞)
5	10*τ_{bit}	[0, 184]	(184, 204]	(604, 624]	(624, ∞)
6	8,5*τ_{bit}	[0, 213]	(213, 230]	(570, 587]	(587, ∞)
7	30*τ_{bit}	[0, 272]	(272, 332]	(1536, 1596]	(1596, ∞)

4.6.2 Trabalhos relacionados - injetor de falhas CAN

Rimén e Christmansson (1999) desenvolveram uma ferramenta para injeção de falhas CAN chamada de CANFI. A ferramenta possui duas portas CAN e atua como um repetidor de mensagens CAN entre as portas. Um nó CAN adicional é conectado a uma das portas, enquanto que a segunda porta é conectada a um barramento CAN. Então, todas as mensagens enviadas e recebidas pelo nó adicional passam pelo CANFI, que pode injetar falhas de acordo com eventos específicos.

Novak, Fried e Vacek (2002) desenvolveram em FPGA um gerador capaz de enviar mensagens compatíveis e incompatíveis com a especificação CAN. As mensagens incompatíveis podem ter algum tipo de erro lógico ou temporal e são utilizadas para testar o comportamento da rede em resposta as violações do protocolo CAN.

Motivados pelos trabalhos de Rufino et al. (1998), Tran (1999) e Proenza e Miro-Julia (2000), os quais apresentam algumas vulnerabilidades nos mecanismos de tolerância a falha da rede CAN, devido a cenários específicos de falhas, Rodriguez-Navas, Jimenez e Proenza (2003) descreveram uma arquitetura de um sistema utilizado para injetar, de modo previsível, falhas em *bits* específicos e falhas inconsistentes (falhas que afetam apenas alguns nós da rede) na camada física de uma rede CAN. A arquitetura do sistema consiste de uma ferramenta de software chamada de CANfidant e dos componentes de hardware: controlador de injeção de falhas e injetores individuais de falhas (IFIs). CANfidant tem a função de criar os cenários de falhas e enviá-las para o controlador de injeção de falhas, que as executa por meio dos IFIs de cada nó, que estão localizados entre os controladores CAN e os transceivers. A arquitetura do injetor foi projetada para ser não invasiva e garantir uma alta controlabilidade espacial e temporal. Entretanto, apenas o software CANfidant foi implementado.

Reorda e Violante (2004) desenvolveram um ambiente capaz de injetar falhas *online* em

mensagens com identificadores pré-especificados. O ambiente é constituído por dois módulos: um computador e a placa de desenvolvimento chamada de CANsniffer. O computador atua como *data logger* e é responsável pela configuração e gerenciamento da placa CANsniffer. A placa CANsniffer é utilizada para fazer a interface com a rede CAN e consiste de um dispositivo FPGA e de um *transceiver* CAN. Um *software* no computador controla a placa CANsniffer para monitorar o barramento e indica qual a mensagem com o ID pré-especificado que será injetada a falha e a quantidade de vezes que isso ocorrerá. A falha é gerada pelo envio de seis *bits* com níveis dominantes.

Webermann e Block (2012) descreveram uma abordagem utilizando a tecnologia FPGA para injetar falhas em um barramento CAN. Neste trabalho é proposto um módulo injetor de falhas em VHDL que estende o CAN IP core em um FPGA, que é capaz de gerar todos os tipos de erros em redes CAN. Uma implementação real do sistema não foi apresentada.

Em todos os trabalhos apresentados, observa-se que o principal objetivo é gerar erros diretamente nas mensagens. Mas, na prática, os erros nas mensagens são ocasionados devido a mudanças indesejadas no nível do sinal no barramento. Em geral, essas mudanças indesejadas, chamadas de falhas, são provocadas, normalmente, por interferências eletromagnéticas no barramento. Então, se alguma mensagem estiver sendo transmitida durante a falha, um ou mais *bits* da mensagem serão corrompidos (erro na mensagem) e os mecanismos de detecção de erros da rede CAN detectará o erro na mensagem. Na seção seguinte, será apresentado um injetor de falhas CAN capaz de gerar falhas aleatórias no barramento compatível com a prática.

4.6.3 Injetor de falhas CAN

O injetor de falhas proposto (Sá et al., 2016) é responsável pela geração de falhas aleatórias no barramento. A ideia é garantir que um erro sempre ocorrerá quando uma falha for injetada aleatoriamente no barramento. O tempo entre as falhas seguirá uma distribuição exponencial com média λ.

Estratégia para injeção de falhas

Na rede CAN, um erro ocorrerá sempre que um dos cinco eventos, apresentados a seguir, for detectado pelos nós: alteração no nível lógico do *bit* transmitido; mudança no formato de campos específicos da mensagem (*bits* com valores fixos); violação da regra de inserção de *bits*, ou seja, a transmissão consecutiva de seis *bits* com mesmo nível lógico; sequência CRC recebida diferente da transmitida (*checksum CRC*) e ausência de reconhecimento na mensagem. Independente de qualquer um dos mecanismos para detecção de erros, é fato que um erro somente ocorrerá se houver a alteração do estado atual do sinal no barramento. De acordo com o mecanismo *wired-and*, um modo de alterar o sinal no barramento é através da injeção de um *bit*

dominante. Entretanto, deve-se considerar alguns detalhes do protocolo CAN para garantir a alteração do estado do sinal no barramento.

Inicialmente, considerou-se uma rede CAN onde os nós não estão transmitindo mensagens de dados. Nesse caso, o barramento estará sempre livre. Nesse estado, o nível lógico no barramento é sempre recessivo. Então, se um *bit* dominante for injetado, os nós considerarão que esse *bit* é o *bit* dominante SOF que um nó está transmitindo e que, a partir desse instante, o barramento estará ocupado. Entretanto, como o *bit* injetado não faz parte de uma mensagem e, supondo que nos próximos seis *bits* não haverá uma nova injeção de *bit*, é certo que os próximos seis *bits* serão *bits* recessivos e a regra de inserção de *bits* será violada. Então, os nós transmitirão uma mensagem de erro contendo de seis a doze *bits* dominantes (*flag* de erro) mais oito *bits* recessivos (delimitador de erro). Essa situação está representada na Figura 4.41 para uma rede CAN com taxa de transmissão igual a 500 K*bits*/s (τ_{bit} = 2 μs).

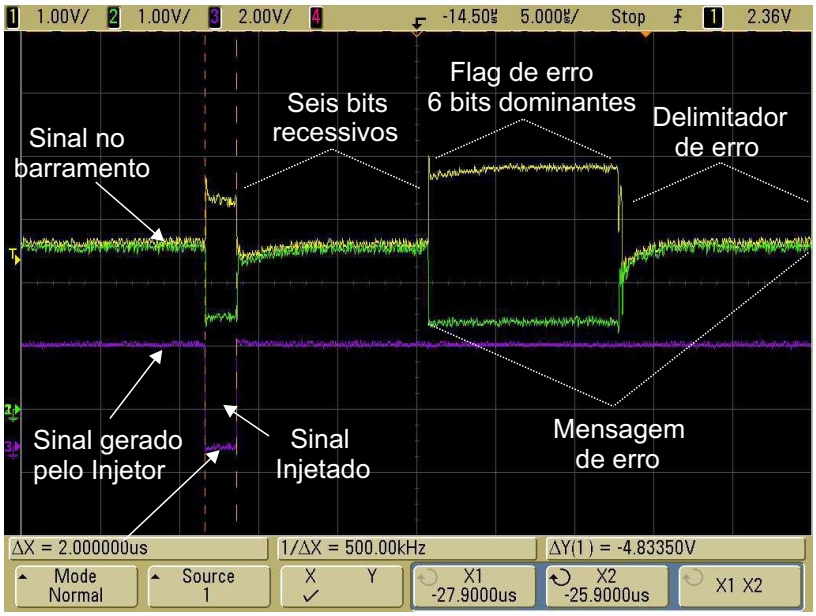


Figura 4.41: Tela do osciloscópio com os sinais no barramento (Linha verde - CAN_H e linha amarela - CAN_L) e no pino de saída da placa Stellaris LM4F120 (Linha roxa) durante a injeção de um *bit* dominante com o barramento livre.

Considerou-se a mesma rede CAN, mas os nós transmitem mensagens de dados. Então, se uma mensagem estiver sendo transmitida durante a injeção da falha, dois eventos poderão ocorrer: (1) se o *bit* da mensagem for também dominante, não ocorrerá um erro, uma vez que não houve mudança no *bit* da mensagem transmitida, ou (2) se o *bit* da mensagem for recessivo, o *bit* dominante injetado sobrescreverá o *bit* da mensagem e, portanto, ocorrerá um erro na mensagem e uma mensagem de erro será imediatamente transmitida pelos nós.

Uma vez que o objetivo do injetor é gerar erros no barramento sempre que uma falha for injetada, pode-se concluir que a injeção de um único *bit* dominante não garantirá a geração de

um erro no barramento. Entretanto, se o injetor sempre enviar uma sequência de seis *bits* dominantes, garante-se que um erro sempre será gerado[5], pois essa sequência de seis *bits* dominantes sempre violará a regra de inserção de *bits* do protocolo CAN. Na Figura 4.42 apresenta-se imagens da tela do osciloscópio com os sinais no barramento e o sinal do injetor de falhas com a sequência de seis *bits* dominantes para o barramento livre (Figura 4.42(a)) e durante a transmissão de uma mensagem (Figura 4.42(b)). Observe em ambas as Figuras que uma mensagem de erro é imediatamente transmitda após o fim da injeção da falha. Na Figura 4.42(b), a mensagem corrompida é imediatamente transmitida logo após o fim da mensagem de erro.

Gerador de falhas aleatórias

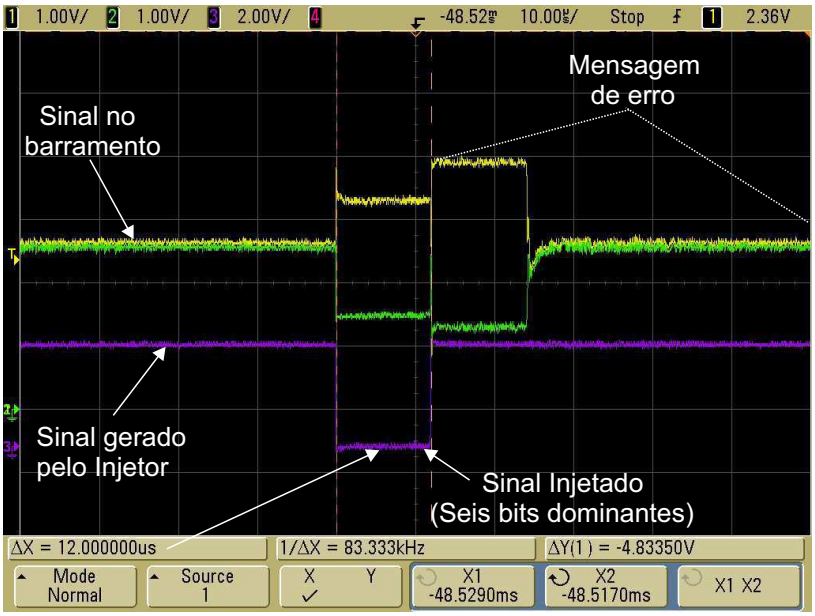
De acordo com a estratégia apresentada na seção anterior, é possível gerar falhas com intervalos determinísticos ou estocásticos. Entretanto, para validar o modelo 3, as falhas serão geradas de modo aleatório seguindo uma distribuição exponencial com média $1/\lambda$, onde λ é a taxa de erros por unidade de tempo.

O gerador de falhas aleatórias foi desenvolvido usando o método da transformada inversa. Esse método baseia-se no fato de que, para uma dada variável aleatória x, com uma função de distribuição acumulada $F(x)$, a variável $u = F(x)$ é uniformemente distribuída entre 0 e 1. Assim, os valores de x podem ser obtidos gerando-se números aleatórios uniformemente distribuídos e, em seguida, calculando-se $x = F^{-1}(u)$. Em outras palavras, esse método consiste em dois passos: gerar números aleatórios distribuídos uniformemente de 0 a 1 e, a partir desses números, calcular a variável aleatória seguindo alguma distribuição, por meio do inverso da sua função de distribuição acumulada.

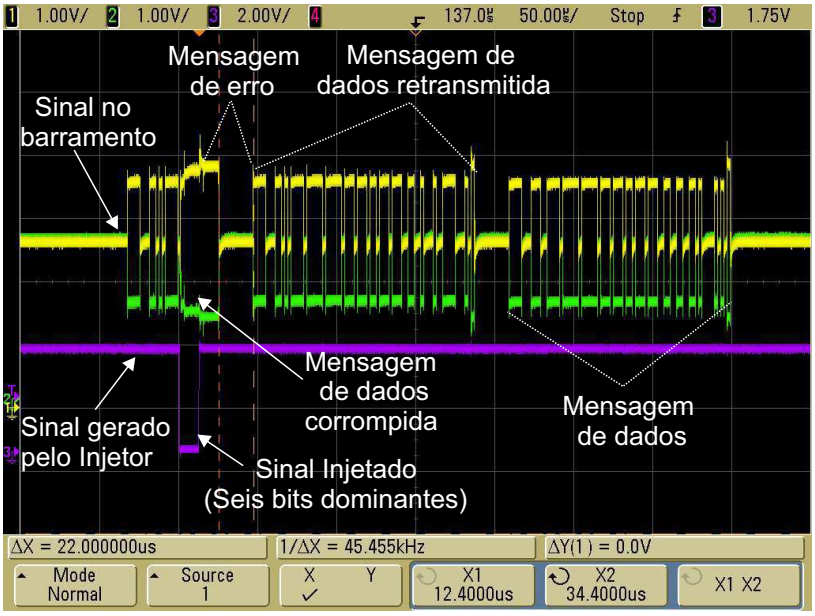
Para geração de números aleatórios distribuídos uniformemente foi utilizado o gerador de L'Ecuyer (DUNN; SHULTIS, 2012, pp. 62-63), que gera números aleatórios distribuídos uniformemente de acordo com a Equação 4.11, onde: z_i é definido pela Equação 4.12, $\hat{z}_i$ é definido pela Equação 4.13, $\%$ é o operador módulo, a_{i-1} e b_{i-1} são chamadas de sementes, e podem assumir valores iguais ou diferentes de 0 a $(2^{32}/2)-1$. Esse gerador possui um período da ordem de 10^{18} e pode ser implementado em um computador de 32 *bits*.

$$u_i = \frac{z_i}{2147483589} \tag{4.11}$$

[5]É importante considerar que a probabilidade de ocorrer um erro não detectado em uma rede CAN é muito pequena. De acordo com a especificação CAN (BOSCH, 1991), "em uma rede operando 8 horas por dia durante 365 dias por ano, e uma taxa de erro de 0,7 s, um erro não detectado ocorrerá a cada 100 anos (média estatística)". Unruh, Mathony e Kaiser (1990) estimaram que o número esperado de erros de transmissão não-detectados em uma rede CAN durante o tempo de vida de um veículo é menor do que 10^{-12}.



(a) Injeção de falhas com o barramento livre.



(b) Injeção de falhas durante a transmissão de uma mensagem CAN.

Figura 4.42: Tela do osciloscópio com os sinais no barramento (Linha verde - CAN_H e linha amarela - CAN_L) e no pino de saída da placa Stellaris LM4F120 (Linha roxa) durante a injeção de uma sequência de seis *bits* dominantes.

$$z_i = \begin{cases} \hat{z}_i, & \hat{z}_i \geq 1 \\ \hat{z}_i + 2147483562, & \hat{z}_i < 1 \end{cases} \tag{4.12}$$

$$\hat{z}_i = ((40001 * a_{i-1}) \ \% \ (2147483563)) - ((40692 * b_{i-1}) \ \% \ (2147483399)) \tag{4.13}$$

Considerando que o tempo entre falhas (t_{falhas}) é uma variável aleatória distribuída exponencialmente e que a função distribuição acumulada é definida pela Equação 4.14, t_{falhas} pode ser definida via o método da transformada inversa pela Equação 4.15. Portanto, os valores t_{falhas_i} podem ser gerados a partir da variável uniforme u_i (Equação 4.11).

$$F(t_{falhas}) = 1 - e^{-\lambda t_{falhas}} \tag{4.14}$$

$$t_{falhas} = -\frac{1}{\lambda} \ln(1 - u) \tag{4.15}$$

Implementação e validação do injetor de falhas

O hardware do injetor de falhas aleatórias, chamado de nó injetor de falhas, consiste da placa de desenvolvimento Stellaris LM4F120 (INSTRUMENTS, 2012) e do transceiver CAN MCP2551 (MICROCHIP, 2003). Essa placa possui um microprocessador ARM Cortex-M4F de 32 bits com unidade ponto flutuante, frequência de clock de até 80 MHz, memória de programa flash de 256 Kbytes e memória de dados SRAM de 32 Kbytes. O gerador de falhas aleatórias descrito na seção anterior foi implementado em linguagem C na placa LM4F120. Basicamente, as falhas são injetadas na rede por meio de um pino de saída digital da LM4F120 conectado ao pino Tx do MCP2551.

Para implementação em tempo real dos atrasos entre falhas, foi utilizado um temporizador de 32 bits da LM4F120 no modo periódico. Esse timer possui resolução de 12,5 ns. Na prática, foi considerado que o menor atraso entre falhas (atraso entre o fim de uma falha e o início da próxima) é igual a 100 ns. Então, consideramos que todo valor de $t_{falhas_i} < 100$ ns será igual a 100 ns. O tempo gasto para calcular uma amostra do atraso aleatório foi de 80 μs, que é considerado um tempo muito alto para injetar falhas no tempo real. Devido a essa limitação do hardware, o firmware da LM4F120 calcula e armazena em uma variável array, antes de efetivamente iniciar a injeção de falhas, quatro mil amostras. Além disso, sempre que o próximo atraso t_{falhas_i} for maior que 100 μs, o microprocessador calculará e armazenará mais um atraso entre falhas. Esse mecanismo garante que sempre existirá uma amostra ordenada do atraso entre falhas considerando as médias entre falhas (0,1 a 0,0333 s/falha ou 10 a 30 falhas/s)) normalmente adotadas (NAVET; SONG; SIMONOT, 2000).

A validação do gerador de falhas aleatórias foi realizada por meio do teste de aderência Kolmogorov-Smirnov para duas amostras (JAIN, 1991). O algoritmo desse teste retorna h e o valor p de um teste de decisão para a hipótese nula de que as duas amostras são da mesma distribuição de probabilidade com um determinado nível de significância α. Se h for igual a 1, o teste rejeita a hipótese nula no nível de significância especificado e retorna p com algum valor menor que α. Caso contrário, retorna p com algum valor maior que α. Então, foram comparadas amostras de diferentes tamanhos e com diferentes médias calculadas pelo LM4F120 e por um programa computacional que produz números aleatórios com distribuição exponencial e uma média específica a partir de um gerador de números aleatórios de distribuição uniforme. Em todos os testes, a hipótese nula não foi rejeitada ($h = 0$), ou seja, as amostras pertencem a mesma distribuição de probabilidade. Os resultados do teste, h e valor p, para uma quantidade de amostras geradas durante experimentos de 60 s, 2 h e 5 h, com média igual a 30 erros/s e nível de significância de 0,01, são apresentados na Tabela 4.13. Todos os teste não rejeitaram a hipótese nula ($h = 0$), que considera as amostras provindas da mesma distribuição de probabilidade.

Tabela 4.13: Valores h e p do teste Kolmogorov-Smirnov para as amostras geradas pelo gerador de falhas e por um programa computacional que gera amostras distribuídas exponencialmente com média 30 erros/s.

Quantidade da amostras	h	p
1800	0	0,5588
216000	0	0,0998
540000	0	0,2316

4.6.4 Experimentos e análise dos resultados

Assim como na seção 4.4.2, o objetivo do experimento apresentado nesta seção é obter um histograma dos tempos de resposta das mensagens, com os intervalos definidos na Tabela 4.12 da aeção 4.6.1, de modo a gerar a função de distribuição acumulada dos tempos de resposta das mensagens. O tempo mínimo de duração do experimento também foi definido de modo idêntico ao apresentado na seção 4.4.2. Consideramos que, a partir de uma determinada amostra com duração do experimento igual a t_1, todas as amostras com duração do experimento maior que t_1, pertencem a mesma população. Então, a amostra com duração t_1 é suficiente para caracterizar o comportamento dos tempos de resposta das mensagens. Para isso, foram realizados vários experimentos com diferentes tempos de duração e, a cada duas amostras, foi realizado o teste de aderência Kruskal-Wallis para verificar se as amostras pertencem a mesma população.

Foram realizados três experimentos com tempos de duração iguais a 60 s, 2 h e 5 h para uma

taxa de transmissão de 500 Kbits/s e uma taxa de erro igual a 30 erros/s. Uma foto da plataforma experimental é apresentada na Figura 4.43. Os resultados dos testes de aderência Kruskal-Wallis são apresentados na Tabela 4.14. No teste para as amostras de 60 s/2 h, a hipótese nula foi rejeitada para todas as mensagens (exceto para a mensagem 7), uma vez que os valores p foram inferiores a $0{,}01$. Para as amostras de 2 h/5 h, todos as mensagens passaram no teste. Portanto, um tempo de duração do experimento a partir de 2 h é suficiente para caracterizar o comportamento dos tempos de resposta das mensagens.

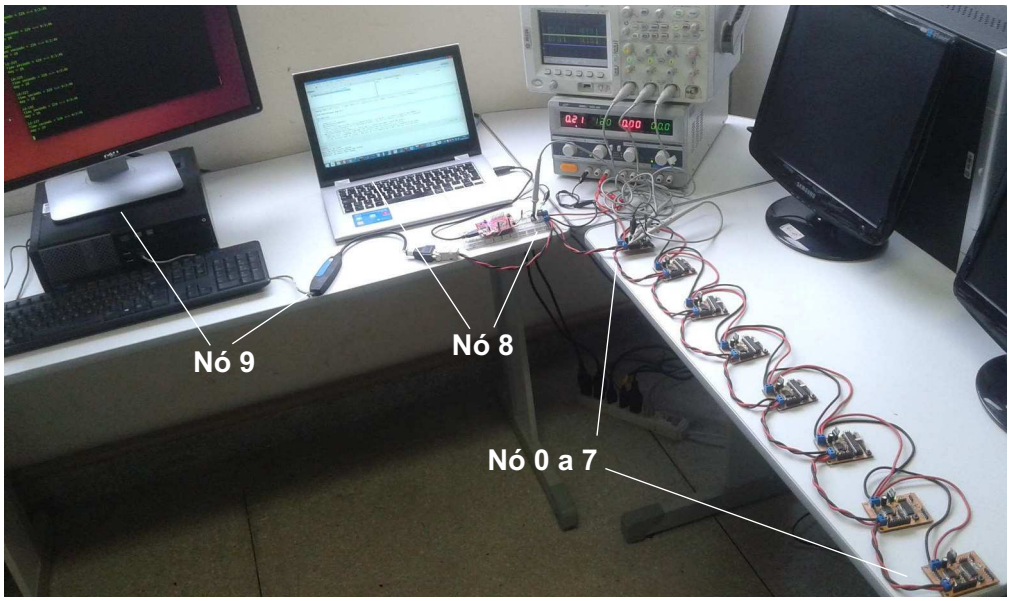


Figura 4.43: Foto da plataforma experimental da rede CAN real.

Tabela 4.14: Valor p do teste de aderência Kruskal-Wallis para as amostras de 60s/2h e 2h/5h em uma rede CAN com *Offset* dinâmico, relógios ideais e barramento real.

$Msg\ (i)$	$Amostras\ 60s/2h$	$Amostras\ 2h/5h$
0	**1,0211e-6**	0,0304
1	**4,3108e-7**	0,0233
2	**2,6664e-8**	0,0158
3	**3,8800e-5**	0,2317
4	**1,4169e-8**	0,2238
5	**3,6474e-5**	0,0870
6	**4,4031e-8**	0,0158
7	0,1123	0,8122

A partir dos resultados obtidos via experimento com duração de 5 h, foram obtidas as funções de distribuição acumulada (FDAs) dos tempos de resposta das mensagens. Esses resultados foram comparados com os resultados obtidos via verificação estatística. Os gráficos das FDAs dos tempos de resposta das mensagens estão representadas nas Figuras 4.44 a 4.51. Nas FDAs obtidas via verificação estatística há um limite superior e inferior com confidência de 99%. Então, para validar o modelo é necessário verificar se as FDAs obtidas experimentalmente são iguais as FDAs obtidas via verificação estatística. Para isso, foi também utilizado o teste de aderência de Kolmogorov-Smirnov (teste KS) para duas amostras.

Os resultados dos testes (h e valor p) entre a FDA obtida no experimento de 5 h e a FDA obtida via verificação estatística são apresentados na Tabela 4.15. Em todos os testes, o valor h obtido foi igual a zero e, consequentemente, o valor p foi maior que o nível de significância de 0,01. Portanto, o modelo da rede CAN com *offset* dinâmico, relógios ideais e barramento real foi validado com sucesso.

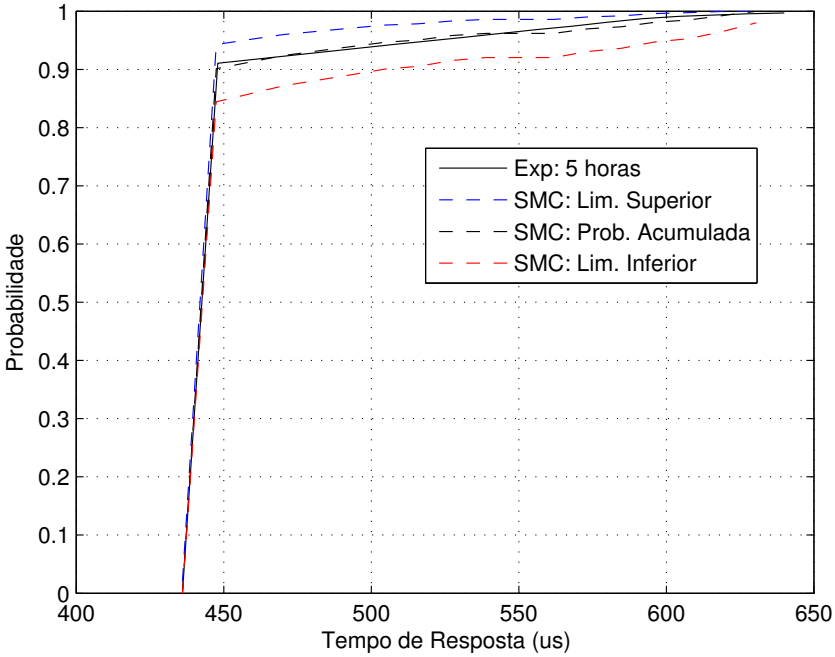


Figura 4.44: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 0 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

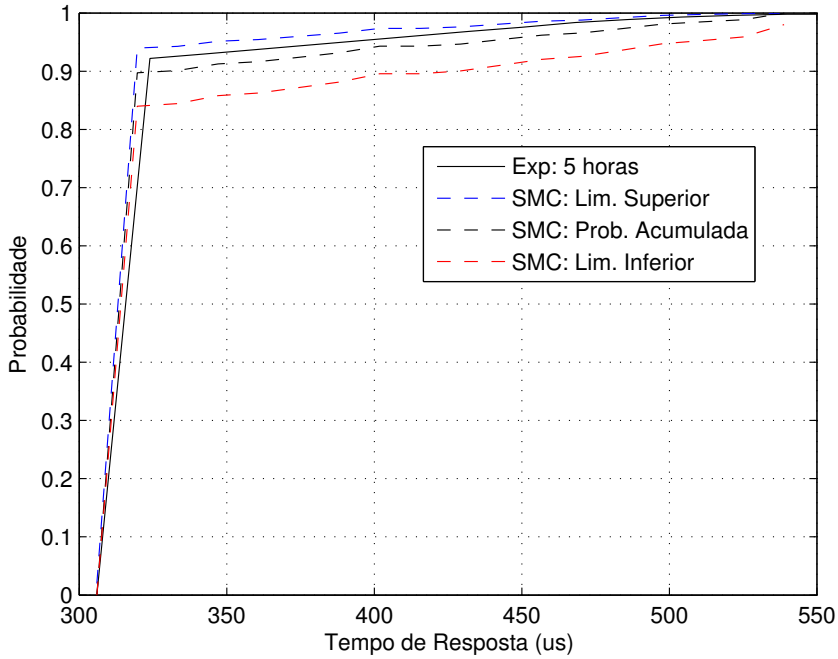


Figura 4.45: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 1 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

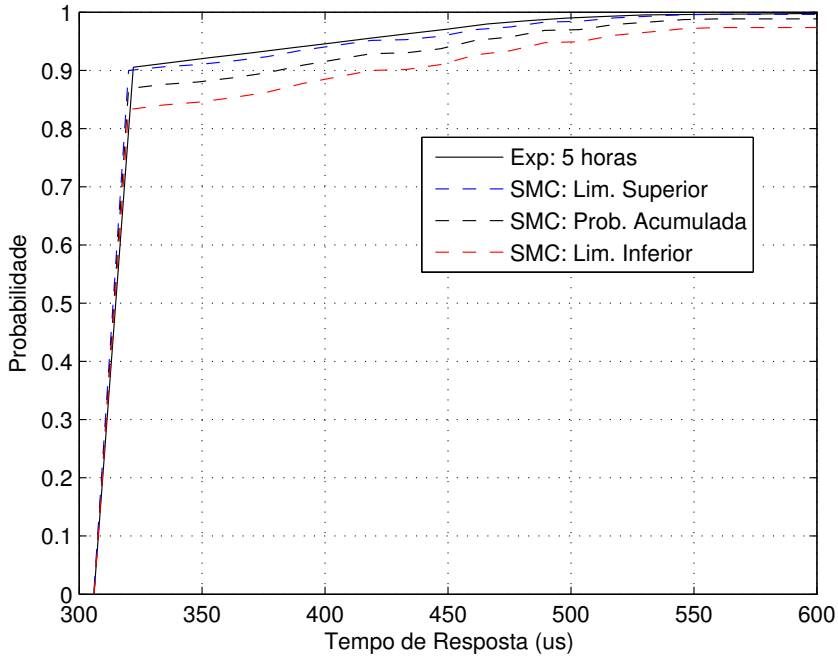


Figura 4.46: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 2 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

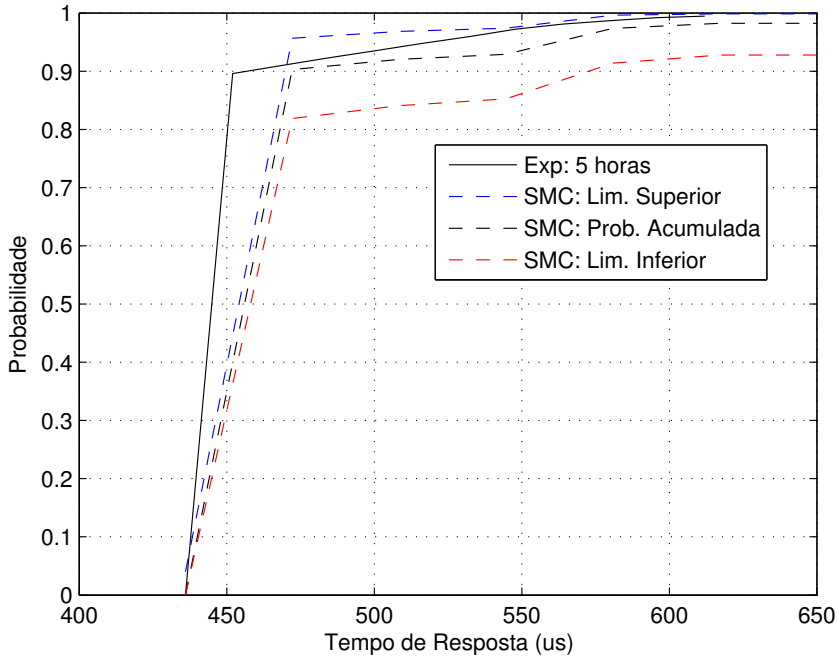


Figura 4.47: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 3 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

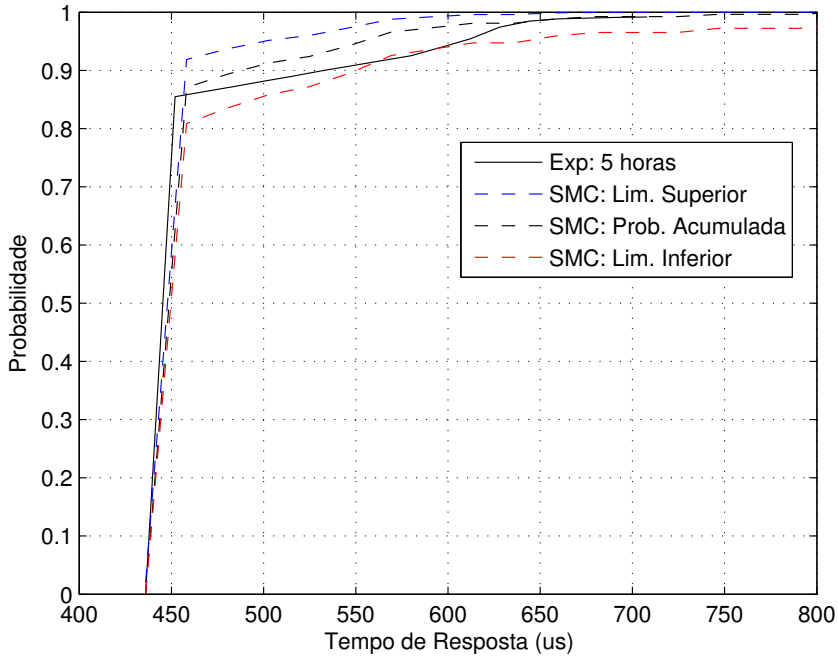


Figura 4.48: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 4 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

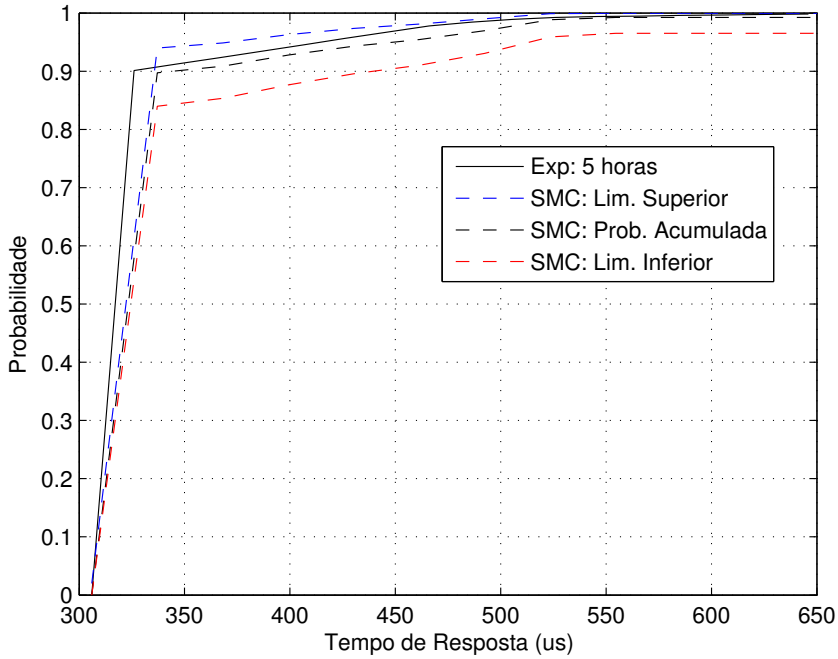


Figura 4.49: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 5 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

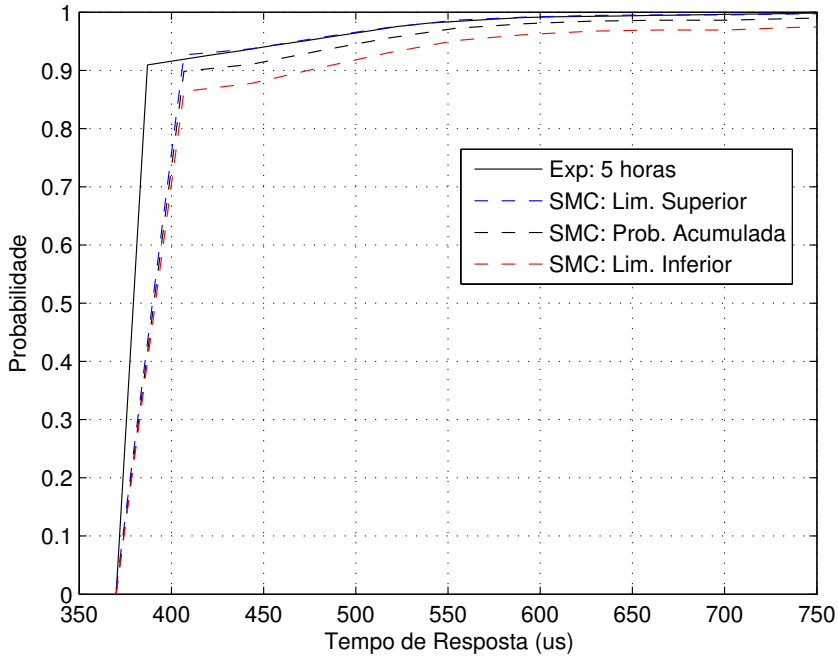


Figura 4.50: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 6 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

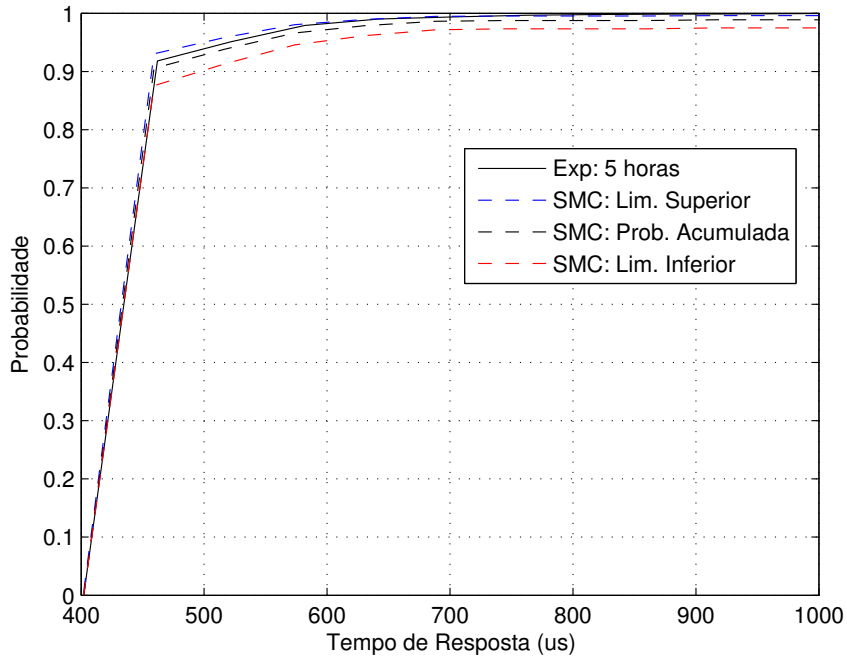


Figura 4.51: Gráfico das funções de distribuição acumulada dos tempos de resposta da mensagem 7 obtidas via experimento de 5 h e verificação estatística para uma rede CAN com barramento real.

Tabela 4.15: Valores h e p do teste Kolmogorov-Smirnov para as FDAs dos tempos de resposta obtidas experimentalmente e por verificação estatística em uma rede CAN com *offset* dinâmico, relógios ideais e barramento real.

	Exp/SMC (FDA)	
$Msg\ (i)$	h	p
0	0	0,9448
1	0	0,9448
2	0	0,2370
3	0	0,4333
4	0	0,4255
5	0	0,9448
6	0	0,0104
7	0	0,0280

4.7 Modelo formal da rede CAN com *offset* estático, relógios e barramento ideais

Nesta seção apresenta-se o modelo 5 (Ver Tabela 4.1), onde os *offsets* das mensagens são estáticos, o relógio e o barramento são ideais.

4.7.1 Trabalhos relacionados

Alguns estudos sobre a análise de tempos de resposta de mensagens CAN com *offsets* estáticos foram desenvolvidos. Em alguns trabalhos usando técnicas analíticas ((SZAKALY, 2003), (DU; XU, 2009), (CHEN et al., 2011)), as análises desenvolvidas são de baixa complexidade computacional, mas os resultados são aproximados. Grenier, Havet e Navet (2008) desenvolveram um algoritmo para atribuição de *offsets* estáticos e verificaram a sua eficiência usando a ferramenta de simulação NETCAR-Analyser (BRAUN; HAVET; NAVET, 2007) para determinar os tempos de resposta exatos no pior caso. Basicamente, a ferramenta realiza uma simulação durante um ou mais intervalos de tempo chamados de hiperperíodo[6], que equivale ao mínimo múltiplo comum de todos os períodos das mensagens (AUDSLEY, 1991), e verifica nesse intervalo o maior tempo de resposta de cada mensagem. Recentemente, Yomsi et al. (2012) adaptaram o trabalho de Palencia e Harbour (1998) para desenvolver uma análise dos tempos de resposta no pior caso para mensagens CAN com *offset* estático. Porém, a análise não é exata. Na literatura, não foi encontrado nenhum modelo em autômatos temporizados da rede CAN que considera os *offset* das mensagens estáticos. Sá et al. (2013) desenvolveram um modelo em autômatos temporizados para modelar a rede CAN com offsets estáticos.

Na seção seguinte apresenta-se um modelo em autômatos temporizados capaz de determinar via verificação simbólica os tempos de resposta exatos no melhor e pior caso. Dois estudos de casos usando a verificação de modelos simbólica serão apresentados.

4.7.2 Modelo em autômato temporizado

O modelo da rede CAN ideal considerando *offset* estático possui os mesmos autômatos descritos no modelo representado na Figura 4.1 (seção 4.2). Os autômatos controlador CAN e arbitragem são idênticos. Enquanto que o autômato tarefa possui uma pequena diferença na modelagem do *offset*, conforme descrito na Figura 4.52. O *offset* é implementado pela invariante de estado $Tempo <= Om$ e o guarda $Tempo == Om$. Isso garante que a primeira instância da tarefa relacionada a mensagem m_i será liberada no lugar *Inicio* com um *offset* estático igual a Om, ao contrário do modelo com *offset* dinâmico, onde o *offset* varia entre 0 e T_m.

[6]É o período em que a escala do conjunto de mensagens se repete ao longo do tempo.

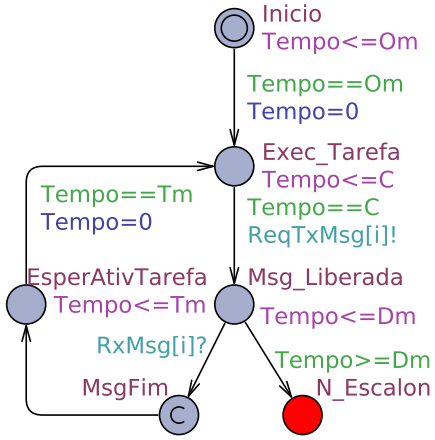


Figura 4.52: Autômato Tarefa da rede CAN ideal com *offset* estático.

Além da modificação apresentada, foi necessário atribuir prioridades aos autômatos do modelo (para cada mensagem), de modo a evitar cenários de inversão de prioridade nas mensagens. Esses cenários foram identificados usando o simulador da ferramenta. Para ilustrar esse problema, considere o conjunto de mensagens, semelhantes aos usados em Davis et al. (2007), apresentado na Tabela 4.16, onde $\tau_{bit} = 8$. Considere também que o *offset* de cada mensagem e o tempo de computação são iguais a zero.

Tabela 4.16: Atributos do conjunto de mensagens (em μs).

$Msg\ (i)$	T_{m_i}	C_{m_i}	D_{m_i}
0	2500	125*τ_{bit}	2500
1	3500	125*τ_{bit}	3500
2	3500	125*τ_{bit}	3500

De acordo com os atributos definidos (Tabela 4.16), veja que as transições *Inicio* $\rightarrow$ *Exec_Tarefa* $\rightarrow$ *Msg_Liberada* e *N_Req* $\rightarrow$ *BufferTx* $\rightarrow$ *Tx_Msg*, respectivamente, dos autômatos tarefa e controlador CAN de cada mensagem, além das transições *N_Req_Tx_Msg* $\rightarrow$ *Arbit* $\rightarrow$ *Tx_Msg*, devem ocorrer no instante de tempo 0. Para esse cenário, observamos, usando o simulador da ferramenta, que o lugar *Tx_Msg* do autômato controlador CAN da mensagem de menor prioridade, poderá ser alcançado antes que as mensagens de maior prioridade nos seus respectivos autômatos controlador CAN, ou seja, essa mensagem poderá ser transmitida antes das mensagens de maior prioridade indicarem que estão prontas para entrar no processo de arbitragem.

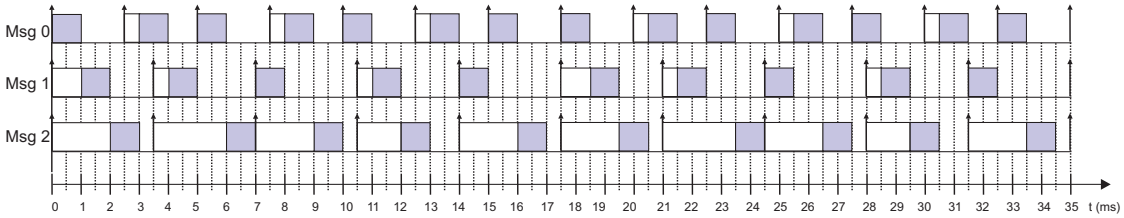
Para resolver esse problema, usamos a capacidade da ferramenta de atribuir prioridades aos autômatos. Desse modo, durante um cenário onde mais de uma transição de diferentes autômatos estiverem habilitadas, a ferramenta executará as transições seguindo os níveis de prioridade de cada autômato. Para definir a prioridade de cada autômato, tomamos como referência a prioridade de acesso ao meio de cada mensagem, e seguimos as relações de precedência entre os

autômatos da mesma mensagem, conforme indicado a seguir, onde: A, CON e TAR são, respectivamente, os autômatos arbitragem, controlador CAN e tarefa, e n o número de mensagens:

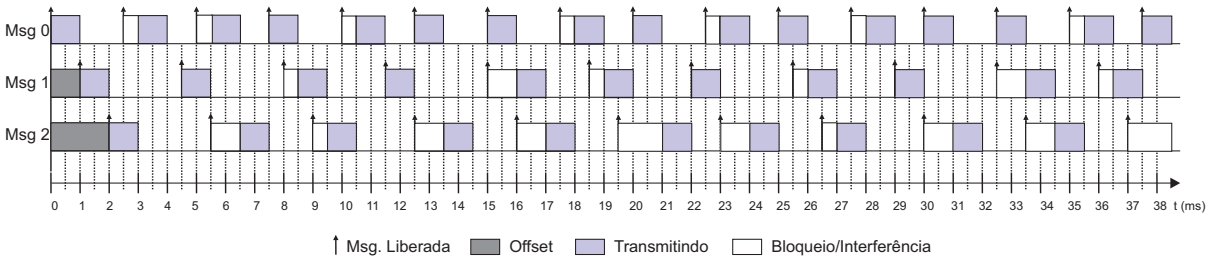
$$A < CON_{n-1} < CON_{n-2} < ... < CON_0 < TAR_{n-1} < TAR_{n-2} < ... < TAR_0$$

Definimos o autômato A com a menor prioridade, de modo a garantir que a transição *Arbit* $\rightarrow$ *Tx_Msg*, que decide qual mensagem será transmitida, nunca ocorrerá antes de todos os autômatos controlador CAN alcançarem o lugar *BufferTx*, ou seja, antes de todas as mensagens prontas para serem transmitidas serem armazenadas no *buffer* de transmissão em um mesmo instante de tempo. O código UPPAAL do modelo é apresentado na Seção C.4 do Apêndice C.

Para verificar o comportamento temporal do modelo, determinamos graficamente os tempos de resposta no melhor e pior caso do conjunto de mensagens da Tabela 4.16, examinando a escala de dois hiperperíodos (Figura 4.53) para duas diferentes configurações de *offset*: $O_{m_0} = O_{m_1} = O_{m_2} = 0$ (*offsets* iguais - Figura 4.53(a)); e $O_{m_0} = 0$, $O_{m_1} = 1\ ms$ e $O_{m_2} = 2\ ms$ (*offsets* diferentes - Figura 4.53(b)). Então, comparamos com os resultados obtidos pela verificação das propriedades *sup{TARi.MsgFim}:TARi.Tempo* e *inf{TARi.MsgFim}:TARi.Tempo*, com e sem a atribuição de prioridades, onde $i = 0$, 1 e 2. Essas propriedades podem ser lidas da seguinte forma: "qual o maior/menor valor do relógio $Tempo$, se o lugar *Msg_Fim* do autômato *TARi* foi alcançado?".



(a) *Offsets* iguais.



(b) *Offsets* diferentes.

Figura 4.53: Escala do conjunto de mensagens com *offsets* estáticos.

Os resultados para o pior caso e melhor caso estão, respectivamente, nas Tabelas 4.17 e 4.18, para a configuração *offsets* iguais. Para a configuração *offsets* diferentes, os resultados para o

pior caso e melhor caso estão, respectivamente, nas Tabelas 4.19 e 4.20. Observe que os tempos de resposta atribuindo prioridades (C/P) aos autômatos são idênticos aos obtidos graficamente (G), ao contrário dos tempos de resposta obtidos com os autômatos sem prioridade (S/P) (com exceção dos tempos de resposta no melhor caso para *offsets* diferentes (Tabela 4.20)). Então, consideramos que a política de atribuição de prioridades aos autômatos do modelo conduz ao comportamento correto da rede, evitando as inversões de prioridade. Outros experimentos para diferentes configurações de *offsets* foram realizados e os resultados foram consistentes.

Tabela 4.17: Tempos de resposta no pior caso (em μs) obtidos graficamente ($R_{m_i}^{max}$(G)), e por verificação de modelos sem ($R_{m_i}^{max}$(S/P)) e com ($R_{m_i}^{max}$(C/P)) prioridade - *Offsets* iguais.

$Msg\ (i)$	$R_{m_i}^{max}$(G)	$R_{m_i}^{max}$(S/P)	$R_{m_i}^{max}$(C/P)
0	1500	**2000**	1500
1	2000	**3000**	2000
2	3500	3500	3500

Tabela 4.18: Tempos de resposta no melhor caso (em μs) obtidos graficamente ($R_{m_i}^{min}$(G)), e por verificação de modelos sem ($R_{m_i}^{min}$(S/P)) e com ($R_{m_i}^{min}$(C/P)) prioridade - *Offsets* iguais.

$Msg\ (i)$	$R_{m_i}^{min}$(G)	$R_{m_i}^{min}$(S/P)	$R_{m_i}^{min}$(C/P)
0	1000	1000	1000
1	1000	1000	1000
2	2500	**1000**	2500

Tabela 4.19: Tempos de resposta no pior caso (em μs) obtidos graficamente ($R_{m_i}^{max}$(G)), e por verificação de modelos sem ($R_{m_i}^{max}$(S/P)) e com ($R_{m_i}^{max}$(C/P)) prioridade - *Offsets* diferentes.

$Msg\ (i)$	$R_{m_i}^{max}$(G)	$R_{m_i}^{max}$(S/P)	$R_{m_i}^{max}$(C/P)
0	1500	**2000**	1500
1	2000	2000	2000
2	2500	2500	2500

Tabela 4.20: Tempos de resposta no melhor caso (em μs) obtidos graficamente ($R_{m_i}^{min}$(G)), e por verificação de modelos sem ($R_{m_i}^{min}$(S/P)) e com ($R_{m_i}^{min}$(C/P)) prioridade - *Offsets* diferentes.

$Msg\ (i)$	$R_{m_i}^{min}$(G)	$R_{m_i}^{min}$(S/P)	$R_{m_i}^{min}$(C/P)
0	1000	1000	1000
1	1000	1000	1000
2	1000	1000	1000

4.7.3 Estudos de caso

Nesta seção, apresentam-se os estudos de casos para um conjunto de 12 mensagens de uma rede CAN instalada no protótipo de um veículo da PSA Peugeot-Citroën (NAVET; SONG; SIMONOT, 2000), e para um conjunto de 17 mensagens de uma simplificação feita por Tindell, Burns e Wellings (1995) de um *benchmark* SAE (*Society Automotive Engineering*). Os resultados para as duas diferentes configurações de *offsets* estáticos foram comparados com os resultados da análise de Davis et al. (2007) para *offsets* dinâmicos. Foram usadas as mesmas propriedades para determinar os tempos de resposta.

Mensagens - PSA Peugeot-Citroën

Os atributos para esse conjunto de mensagens estão indicados na Tabela 4.21. A taxa de transmissão é de 250 K*bits*/s ($\tau_{bit} = 4$) e o fator de utilização da rede é igual a 21,5%.

Tabela 4.21: Atributos do conjunto de mensagens da PSA Peugeot-Citroën (em μs).

$Msg\ (i)$	T_m	C_m	D_m
0	10000	135*τ_{bit}	10000
1	14000	85*τ_{bit}	14000
2	20000	85*τ_{bit}	20000
3	15000	75*τ_{bit}	15000
4	20000	105*τ_{bit}	20000
5	40000	105*τ_{bit}	40000
6	15000	95*τ_{bit}	15000
7	50000	105*τ_{bit}	50000
8	20000	95*τ_{bit}	20000
9	100000	125*τ_{bit}	100000
10	50000	105*τ_{bit}	50000
11	100000	65*τ_{bit}	100000

Na configuração *offset* 1, todos os *offsets* das mensagens são iguais a zero, e na configuração *offset* 2, usamos a seguinte regra para atribuição de *offsets*: $O_{m_0} = 0$ e $O_{m_i} = O_{m_{i-1}} + C_{m_{i-1}}$, resultando nos seguintes valores em μs: $O_{m_0} = 0$, $O_{m_1} = 540$, $O_{m_2} = 880$, $O_{m_3} = 1220$, $O_{m_4} = 1520$, $O_{m_5} = 1940$, $O_{m_6} = 2360$, $O_{m_7} = 2740$, $O_{m_8} = 3160$, $O_{m_9} = 3540$, $O_{m_{10}} = 4040$, $O_{m_{11}} = 4460$.

O tempo de verificação para cada configuração foi de aproximadamente 30 s. As curvas dos tempos de resposta no melhor e pior caso versus o identificador da mensagem estão representadas na Figura 4.54. Observe que para cada configuração de *offset*, os tempos de resposta no pior caso (curvas vermelha e azul com círculos) são menores que os tempos de resposta com *offset* dinâmico (curva preta). Na configuração *offset* 2, obtemos tempos de resposta muito baixos comparados com a configuração 1. Esse resultado deve-se aos valores dos *offsets* e, principalmente, ao baixo fator de utilização da rede, que possibilitou distribuir ao longo do tempo a carga de trabalho na rede. Para a mensagem com identificador 11 (menor prioridade), o tempo de resposta no pior caso para a configuração *offset* 2 foi mais de 85% menor que os tempos de resposta no pior caso com *offset* dinâmico e com a configuração *offset* 1.

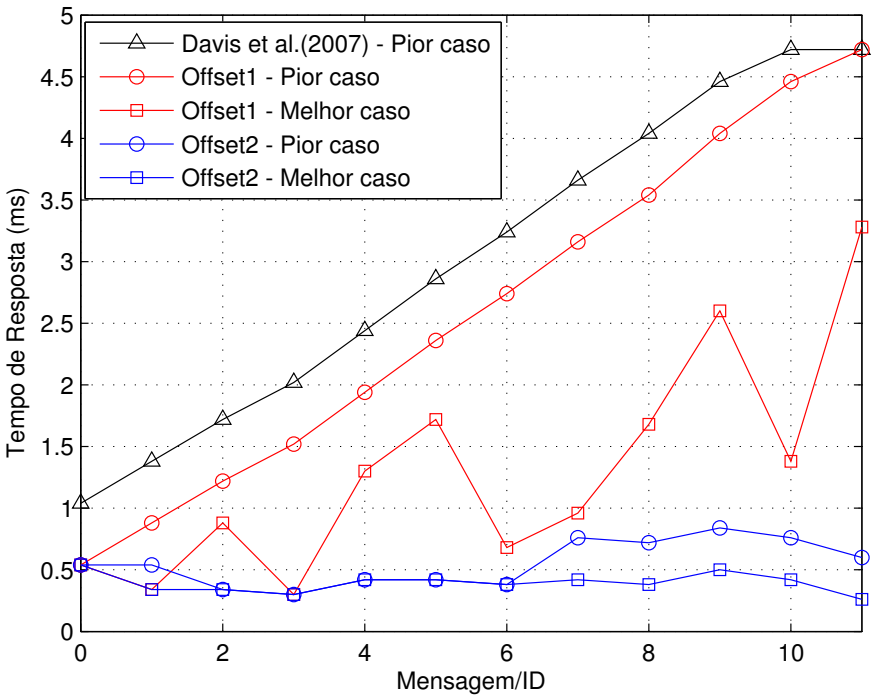


Figura 4.54: Curvas dos tempos de resposta no melhor e pior caso do conjunto de mensagens da PSA Peugeot-Citroën.

Mensagens - *benchmark* SAE

Neste estudo de caso, o fator de utilização da rede é igual a 85%, que é considerado alto. Os atributos para esse conjunto de mensagens estão indicados na Tabela 4.22. A taxa de transmissão é de 125 K*bits*/s ($\tau_{bit} = 8$).

Tabela 4.22: Atributos do conjunto de mensagens do *benchmark* SAE (em μs).

$Msg\ (i)$	T_{m_i}	C_{m_i}	D_{m_i}
0	1000000	65*τ_{bit}	5000
1	5000	75*τ_{bit}	5000
2	5000	65*τ_{bit}	5000
3	5000	75*τ_{bit}	5000
4	5000	65*τ_{bit}	5000
5	5000	75*τ_{bit}	5000
6	10000	115*τ_{bit}	10000
7	10000	65*τ_{bit}	10000
8	10000	75*τ_{bit}	10000
9	10000	75*τ_{bit}	10000
10	100000	65*τ_{bit}	100000
11	100000	95*τ_{bit}	100000
12	100000	65*τ_{bit}	100000
13	100000	65*τ_{bit}	100000
14	1000000	85*τ_{bit}	1000000
15	1000000	65*τ_{bit}	1000000
16	1000000	65*τ_{bit}	1000000

Na configuração *offset* 1, todos os *offsets* das mensagens são iguais a zero, e na configuração *offset* 2 (mesma regra usada no primeiro estudo de caso), os *offsets* das mensagens em μs são : $O_0 = 0$, $O_1 = 520$, $O_2 = 1120$, $O_3 = 1640$, $O_4 = 2240$, $O_5 = 2760$, $O_6 = 3360$, $O_7 = 4280$, $O_8 = 4800$, $O_9 = 5400$, $O_{10} = 6000$, $O_{11} = 6520$, $O_{12} = 7280$, $O_{13} = 7800$, $O_{14} = 8320$, $O_{15} = 9000$, $O_{16} = 9520$.

O tempo de verificação para cada configuração foi de aproximadamente 30 s. As curvas dos tempos de resposta no melhor e pior caso versus o identificador da mensagem estão representadas na Figura 4.55. Assim como no estudo de caso anterior, os tempos de resposta no pior caso (curvas vermelha e azul com círculos) foram menores que os tempos de resposta com *offset* dinâmico (curva preta). Mais uma vez, os resultados com a configuração *offset* 2 foram melhores. Observe que as curvas dos tempos de resposta no melhor e pior caso para ambas

as configurações de *offsets* são parecidas. Acreditamos que esse comportamento foi provocado pelo alto fator de utilização da rede. Apesar do alto fator de utilização, conseguimos obter uma redução significante dos tempos de resposta no pior caso usando *offsets* estáticos.

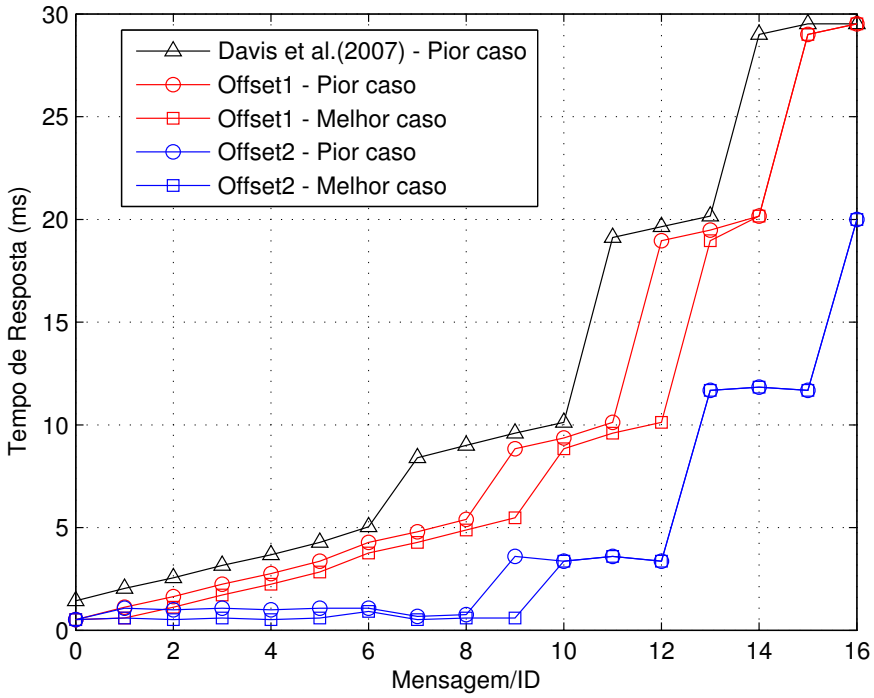


Figura 4.55: Curvas dos tempos de resposta no melhor e pior caso das mensagens do *benchmark* SAE.

4.8 Conclusões

Neste Capítulo foram desenvolvidos quatro modelos da rede CAN usando autômatos temporizados para diferentes características do comportamento da rede, com o intuito de determinar os tempos de resposta no melhor e pior caso utilizando as técnicas de verificação clássica e estatística.

O primeiro modelo, apresentado na seção 4.2 do Capítulo 4, consiste de uma rede CAN com relógios ideais, mensagens com *offsets* dinâmicos e livres de erros no barramento. Um estudo de caso para uma rede CAN com oito mensagens foi realizado. Algumas propriedades lógicas e temporais foram verificadas. Utilizando a verificação de modelos clássica, foi possível determinar os tempos de resposta no melhor e pior caso para no máximo um subconjunto com quatro mensagens. Esses resultados foram comparados com os resultados obtidos via a técnica analítica desenvolvida por Davis et al. (2007). Foi verificado que os resultados são idênticos. A verificação de modelos clássica não conseguiu determinar os tempos de resposta para mais

de quatro mensagens CAN, devido a uma explosão do espaço de estados do modelo. Então, como solução desse problema, foi aplicada a verificação de modelos estatística para o conjunto completo de mensagens. Então, foram obtidos as funções de distribuição acumulada das mensagens. Foi constatado que os tempos de resposta no pior caso ocorrem raramente. De fato, conforme a análise de Davis et al. (2007), o cenário de pior caso ocorre a partir de uma combinação muito específica de eventos. Então, foi verificado que a probabilidade de ocorrência desse cenário é muito baixa. Esse é um resultado importante, pois possibilita projetar redes CAN, do ponto de vista temporal, analisando as probabilidades dos deadlines serem satisfeitos, e não apenas analisando o cenário raro do pior caso. Em seguida, o modelo foi validado utilizando uma plataforma experimental constituída por nove nós CAN. Uma estratégia foi desenvolvida para derivar as funções de distribuição acumulada das mensagens de cada nó, e esses resultados foram comparados com as funções de distribuição acumulada obtidas via verificação de modelos estatísticos. Então, aplicou-se o teste de aderência Kolmogorov-Smirnov para duas amostras com nível de confidência de 99% e verificou-se que as duas funções de distribuição acumulada pertencem a mesma distribuição de probabilidade. Portanto, o modelo foi validado com sucesso.

No segundo modelo, apresentado na seção 4.3, considerou-se que os relógios dos nós da rede CAN são físicos (reais), mensagens com *offsets* dinâmicos e livres de erros de transmissão. Foi apresentado um método para modelar as variações nos relógios dos nós CAN. A partir desse modelos, usou-se a verificação de modelos clássica para determinar os tempos de resposta no melhor e pior caso e a verificação de modelos estatísticos para determinar as distribuições dos tempos de resposta para cada mensagem considerando diferentes taxas de variação dos relógios. Verificou-se, conforme esperado (ver (MONOT et al., 2012)), que as variações nos relógios não influenciam de modo significante nos tempos de resposta e nas distribuições dos tempos de resposta das mensagens. Do ponto de vista da modelagem, esse resultado é importante, uma vez que nos possibilita desconsiderar as variações nos relógios, reduzindo assim a complexidade e o espaço de estados do modelo. Entretanto, essa desconsideração só é válida para modelos onde os *offsets* das mensagens são dinâmicos.

Na seção 4.5, foi desenvolvido um modelo assumindo relógios ideais, *offsets* dinâmicos e a presença de erros no barramento. No modelo, considera-se que os erros no barramento ocorrem aleatoriamente seguindo uma distribuição exponencial com média λ. A partir desse modelo, foi realizado um estudo de caso para uma rede CAN com oito mensagens, taxa de transmissão de 500 kbits/s e taxa de erros igual a 30 erros/s. Esse modelo foi validado com sucesso de modo semelhante ao primeiro modelo. Para isso, utilizou-se a mesma plataforma experimental em conjunto com um injetor de falhas aleatórias. Uma estratégia para injeção de falhas no barramento CAN foi desenvolvida considerando as especificações do protocolo da rede CAN. Para a injeção aleatória de falhas foi desenvolvido um gerador de variáveis aleatórias baseado no

método da transformada inversa e no gerador de L'Ecuyer. Então, esse injetor foi implementado em um microcontrolador.

O último modelo da rede CAN foi apresentado na seção 4.7. Esse modelo considera uma rede CAN com relógios ideais, mensagens com *offsets* estáticos e livres de erros de transmissão. Verificou-se que os tempos de resposta no pior caso são reduzidos significantemente quando as mensagens são escalonadas com *offsets* estáticos usando uma simples regra de atribuição de *offsets*. Para modelar corretamente a rede CAN, demonstrou-se a necessidade de usar a capacidade da ferramenta UPPAAL em atribuir prioridades aos autômatos, afim de evitar inversões de prioridade no acesso ao meio pelas mensagens. Usando a capacidade de simulação do UPPAAL, foi determinado a partir dos cenários que conduzem a inversão de prioridades, uma política de atribuição de prioridades aos autômatos. Para validar essa política e, consequentemente, o comportamento do modelo, determinou-se graficamente os tempos de resposta no melhor e pior caso para um conjunto de mensagens e comparou-se com os resultados obtidos sem e com a política de atribuição de prioridades aos autômatos. Observou-se também que a atribuição de prioridades aos autômatos reduziu o espaço de estados do modelo, de modo a eliminar estados que na prática não ocorrem. Essa redução do espaço de estados possibilitou a verificação, usando a abordagem clássica, de um conjunto com 16 mensagens CAN (*benchmark* SAE). Por outro lado, o uso de prioridades nos autômatos impossibilita (na versão atual da ferramenta UPPAAL) verificar modelos via a abordagem estatística.

Capítulo 5

Modelagem formal e análise dos tempos de resposta de NCSs CAN

Neste Capítulo apresentam-se modelos em autômatos temporizados e autômatos temporizados com custo de três das quatro principais arquiteturas de NCSs CAN. Esses modelos foram desenvolvidos a partir do modelos da rede CAN apresentados no Capítulo 5 deste trabalho. Os modelos foram desenvolvidos com o auxílio da ferramenta UPPAAL, com o objetivo de determinar os tempos de resposta fim a fim de malhas de controle periódicas, assim como, os tempos de resposta das mensagens de cada malha de controle, utilizando as abordagens verificação de modelos simbólica ou clássica (CMC - *Classical Model-Checking*) e verificação de modelos estatísticos (SMC - *Statistical Model-Checking*). Na literatura não foram encontrados modelos de NCSs CAN em autômatos temporizados.

Na seção seguinte será apresentada uma notação e suposições das principais arquiteturas de NCSs CAN utilizadas ao longo deste Capítulo.

5.1 Notação e modelos das arquiteturas de NCSs CAN

Considere um sistema de controle via rede CAN com um único barramento (ideal ou real) e p malhas de controle. O NCS CAN pode ser representado por uma das quatro principais arquiteturas de NCSs. Cada malha de controle j, onde j varia de 0 a $p-1$, possui um processo j e, dependendo da arquitetura (Figura 5.1), dois ou três nós CAN, sendo: nó sensor (NS$_j$), nó controlador (NC$_j$), nó atuador (NA$_j$), nó sensor/controlador (NS/C$_j$), nó controlador/atuador (NC/A$_j$) e nó controlador (NC). Cada nó possui um relógio ideal[1] x_i, onde i varia de 0 a $n-1$.

No NCS CAN representado pela Figura 5.1(a) (arquitetura SC-A), o nó sensor/controlador (NS/C$_j$) executa a tarefa de aquisição (τ_{s_j}) e a tarefa de controle (τ_{c_j}), e transmite a mensagem controlador-atuador (m_{ca_j}) para o nó atuador (NA$_j$). O nó atuador recebe a mensagem m_{ca_j}

[1]Possui taxa de variação constante e igual a 1.

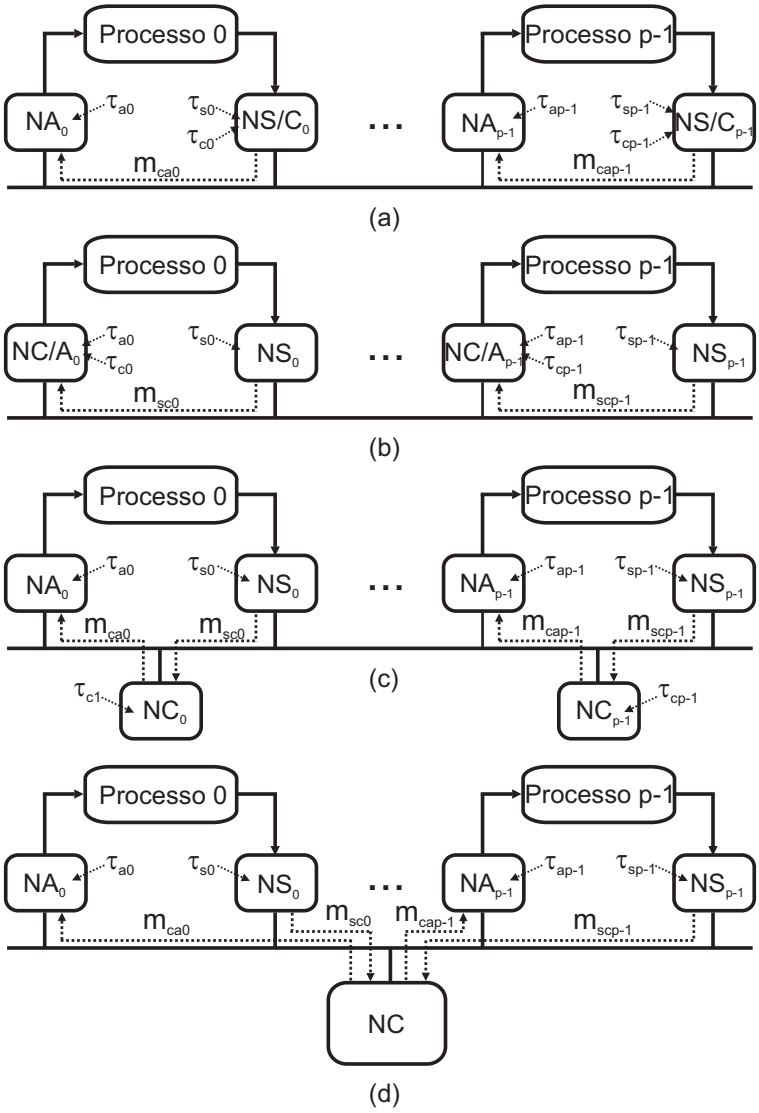


Figura 5.1: Principais Arquitetura de NCSs CAN.

e executa a tarefa de atuação (τ_{a_j}). Enquanto que na arquitetura de NCS CAN representada pela Figura 5.1(b) (arquitetura S-CA), o nó sensor (NS$_j$) executa a tarefa de aquisição (τ_{s_j}) e transmite a mensagem sensor-controlador (m_{sc_j}) para o nó controlador-atuador (NC/A$_j$). O nó controlador-atuador recebe a mensagem m_{sc_j} e executa a tarefa de controle (τ_{c_j}) e a tarefa de atuação (τ_{a_j}).

Na arquitetura de NCS CAN representada pela Figura 5.1(c) (arquitetura S-C-A), o nó sensor (NS$_j$) executa a tarefa de aquisição (τ_{s_j}) e transmite a mensagem sensor-controlador (m_{sc_j}) para o nó controlador (NC$_j$). O nó controlador recebe a mensagem m_{sc_j}, executa a tarefa de controle (τ_{c_j}) e transmite a mensagem controlador-atuador (m_{ca_j}) para o nó atuador (NA$_j$). Enquanto que o nó atuador recebe a mensagem m_{ca_j} e executa a tarefa de atuação (τ_{a_j}). A notação e descrição dos nós, tarefas e mensagens do NCS CAN representado pela Figura 5.1(d) (arquitetura S-Cc-A) é semelhante a do NCS CAN representado pela Figura 5.1(c). A diferença é que a mensagem sensor-controlador de todas as malhas são transmitidas para o nó controlador (NC). O nó NC executa p tarefas de controle (τ_{c_j}) e transmite p mensagens controlador-atuador (m_{ca_j}) para os nós atuadores das respectivas malhas de controle.

O ciclo de operação ou transação k de uma malha j é ativado e liberado periodicamente no instante $a_{M_j}(k) = O_{M_j} + kh_j$, sendo O_{M_j} o *offset* e h_j o período de amostragem da saída do processo ou período da malha de controle j. Considera-se que o *offset* é dinâmico (O_{M_j} assume algum valor arbitrário no intervalo $[0, h_j]$). A partir da liberação do ciclo de operação, sequências de tarefas/mensagens são executadas/transmitidas de acordo com uma relação de precedência pré-definida baseada na arquitetura da malha. Diagramas acíclicos indicando as relações de precedência entre as tarefas e mensagens para as arquiteturas de NCSs estão representadas na Figura 5.2. Os diagramas acíclicos representados nas Figuras 5.2(a) e (b) são usados para representar as relações de precedência nas arquiteturas representadas pelas Figuras 5.1(a) e (b). Enquanto que o diagrama da Figura 5.2(c) representa as relações de precedência nas arquiteturas representadas pelas Figuras 5.1(c) e (d).

As tarefas de aquisição de cada malha de controle j são periódicas. Enquanto que as demais tarefas são esporádicas. A tarefa de aquisição é a tarefa periódica relacionada[2] da mensagem sensor-controlador, exceto na arquitetura representada na Figura 5.1(a), onde a tarefa relacionada é a tarefa de controle. A tarefa de controle, nas outras arquiteturas, é a tarefa relacionada da mensagem controlador-atuador. A tarefa de aquisição de cada malha de controle j possui os seguintes atributos: período T_{s_j}, onde $T_{s_j} = h_j$, *offset* O_{M_j} e tempo de computação C_{s_j}, que pode assumir algum valor fixo ou variável no intervalo $[C_{s_j}^{min}, C_{s_j}^{max}]$. As tarefas de controle e atuação são caracterizadas, respectivamente, apenas o atributo tempo de computação C_{c_j} e C_{a_j}, que podem assumir algum valor fixo ou variável no intervalo $[C_{c_j}^{min}, C_{c_j}^{max}]$ e $[C_{a_j}^{min}, C_{a_j}^{max}]$. Além das tarefas relacionadas, conforme descrito anteriormente, As mensagens são caracteri-

[2]Tarefa que armazena a mensagem no *buffer* de transmissão.

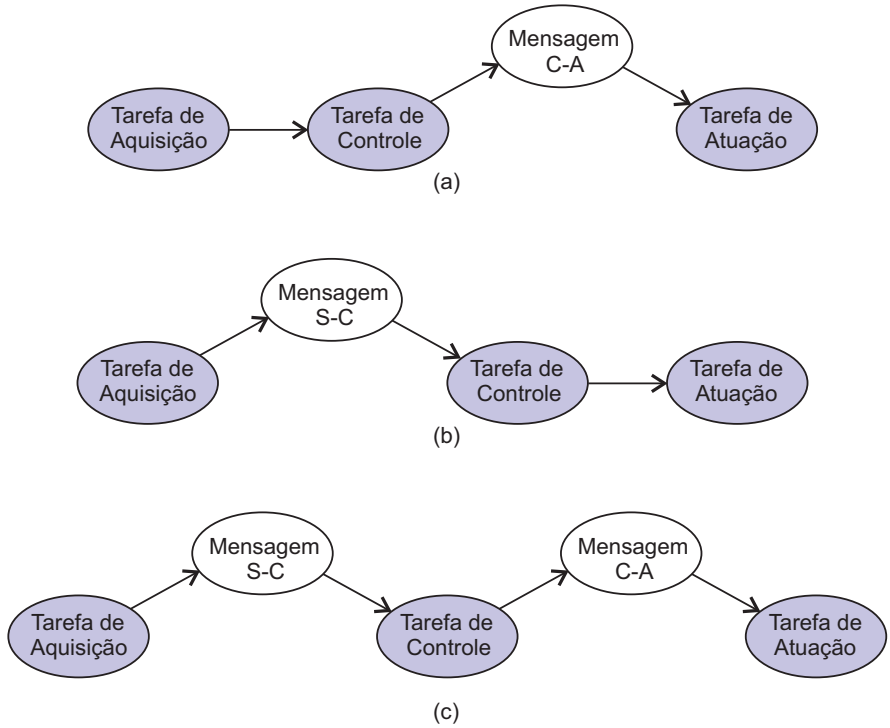


Figura 5.2: Diagrama acíclico com as relações de precedência entre as tarefas e mensagens de uma malha de controle nas principais arquiteturas de NCSs.

zadas apenas pelo atributo tempo de transmissão $C_{m_{sc_j}}$ e $C_{m_{ca_j}}$, respectivamente, constante ou variável no intervalo $[C^{min}_{m_{sc_j}}, C^{max}_{m_{sc_j}}]$ e $[C^{min}_{m_{ca_j}}, C^{max}_{m_{ca_j}}]$.

As prioridades das tarefas e mensagens serão atribuídas de acordo com a política *deadline* monotônico. As tarefas e mensagens da malhas de controle com menor *deadline* fim a fim terão as maiores prioridades. Enquanto que as tarefas e mensagens da malhas de controle com maior *deadline* fim a fim terão as menores prioridades. As prioridades das mensagens da mesma malha de controle serão atribuídas seguindo a ordem das relações de precedência, ou seja, a mensagem sensor-controlador terá maior prioridade que a mensagem controlador-atuador. Uma tarefa τ_i possui uma prioridade p_i no uso do processador, e uma mensagem m_i possui uma prioridade p_{m_i} de acesso ao barramento CAN.

O tempo de resposta fim a fim ($r_{M_j}(k)$) equivale ao tempo gasto desde o início da execução da tarefa de aquisição até o instante em que a tarefa de atuação for finalizada. O tempo de resposta fim a fim no melhor caso, denotado por $R^{min}_{M_j}$ (Equação 5.1), equivale ao menor tempo de resposta fim a fim de algum ciclo de operação k. Enquanto que, o tempo de resposta fim a fim no pior caso, denotado por $R^{max}_{M_j}$, equivale ao maior tempo de resposta fim a fim de algum ciclo de operação k. Para o sistema satisfazer os requisitos de tempo, $R^{max}_{M_j}$ deverá ser menor ou igual ao *deadline* fim a fim (D_{M_j}), conforme indicado na Equação 5.2. Consideramos que o *deadline* fim a fim é menor ou igual ao periodo de amostragem da malha (h_j).

$$R_{M_j}^{min} = \min(r_{M_j}(k)) \tag{5.1}$$

$$R_{M_j}^{max} = \max(r_{M_j}(k)) \leq D_{M_j} \leq h_j \tag{5.2}$$

Denotamos os tempos de resposta das mensagens sensor-controlador e controlador-atuador, respectivamente, por $r_{m_{sc_j}}$ e $r_{m_{ca_j}}$. Sendo assim, o tempo de resposta no melhor caso e pior caso da mensagem sensor-controlador são definidas, respectivamente, pelas Equações 5.3 e 5.4, e o tempo de resposta no melhor caso e pior caso da mensagem controlador-atuador são definidas, respectivamente, pelas Equações 5.5 e 5.6. As letras subscritas j, sc_j e ca_j serão omitidas nos modelos formais.

$$R_{m_{sc_j}}^{min} = \min(r_{m_{sc_j}}(k)) \tag{5.3}$$

$$R_{m_{sc_j}}^{max} = \max(r_{m_{sc_j}}(k)) \tag{5.4}$$

$$R_{m_{ca_j}}^{min} = \min(r_{m_{ca_j}}(k)) \tag{5.5}$$

$$R_{m_{ca_j}}^{max} = \max(r_{m_{ca_j}}(k)) \tag{5.6}$$

Apenas as arquiteturas SC-A (Figura 5.1(a)), S-CA (Figura 5.1(b)) e S-C-A (Figura 5.1(c)) serão modeladas. Além disso, conforme descrito nesta seção, considera-se que o *offset* das malhas de controle é dinâmico, os relógios dos nós são ideais e o barramento pode ser ideal ou real.

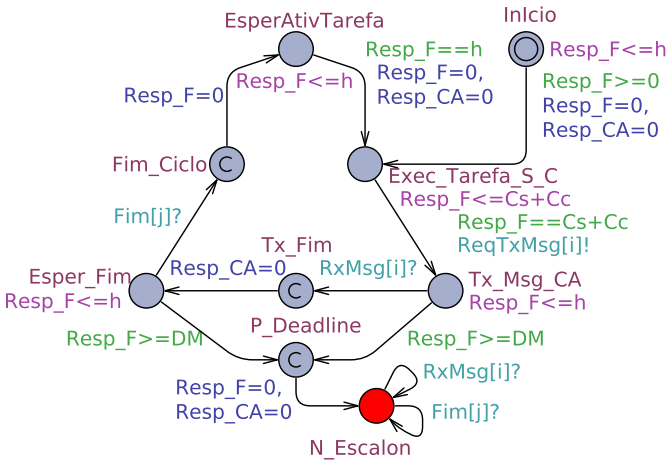
5.2 Modelos formais de NCSs CAN com *offset* dinâmico, relógios e barramento ideais

Nesta seção apresentam-se três modelos em autômatos temporizados de NCSs CAN nas arquiteturas representadas pelas Figuras 5.1(a), (b) e (c). Considera-se que o barramento é livre de erros, os relógios dos nós são ideais e o *offset* de cada malha é dinâmico.

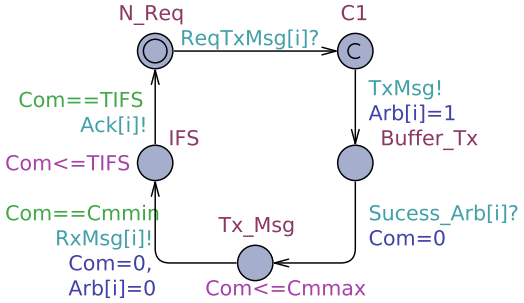
5.2.1 Modelagem da arquitetura SC-A

O modelo do NCS CAN com arquitetura SC-A (Figura 5.3(a)) é constituído pelos autômatos: tarefa aquisição e controle; tarefa atuação; controlador CAN e arbitragem. Cada malha de controle j possui os autômatos tarefa de aquisição e controle, tarefa de atuação e controlador CAN.

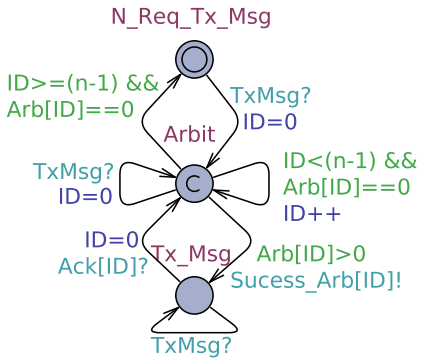
Esses autômatos são gerados automaticamente pela ferramenta UPPAAL. O código UPPAAL do modelo é apresentado na seção D.1 do Apêndice D.



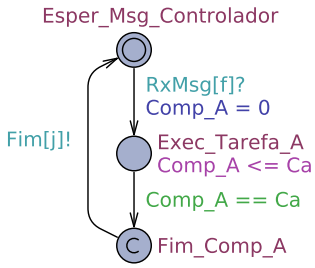
(a) Tarefa aquisição e controle.



(b) Controlador CAN.



(c) Arbitragem.



(d) Tarefa atuação.

Figura 5.3: Conjunto de Autômatos Temporizados para o modelo de NCS CAN na arquitetura SC-A.

O ciclo de operação de uma malha j é iniciado no autômato tarefa de aquisição e controle (Figura 5.3)(a). Esse autômato possui dois relógios locais: *Resp_F* - usado para marcar o período (h - período de amostragem da saída do processo) e o *offset* da malha, e para contar o tempo de resposta fim a fim do ciclo de operação da malha, e *Resp_CA* - usado para marcar o tempo de computação da tarefa de aquisição Cs e contar o tempo de resposta da mensagem controlador-atuador (m_{ca_j}).

O primeiro ciclo de operação é liberado no lugar *Inicio* com um *offset* variando de 0 a h. Os próximos ciclos são iniciados periodicamente no lugar *EsperAtivTarefa*. Após iniciado, a tarefa de aquisição é executada (lugar *Exec_Tarefa_S_C*) e, em seguida, a tarefa de controle. O autômato permanece nesse lugar durante $Cs+Cc$ unidades de tempo e requisita a transmissão da

mensagem controlador-atuador (m_{ca_j}) com identificador *i*, modelada pelo canal *ReqTxMsg[i]!*. A finalização da transmissão dessa mensagem é indicada por um sinal via o canal *broadcast RxMsg[i]?*. Após receber esse sinal, ocorrerá uma transição para o lugar *commited Tx_Fim* e, imediatamente, uma transição para o lugar *Esper_Fim*. Nessa transição, o relógio *Resp_CA* é zerado. O autômato permanece no lugar *Esper_Fim* até receber um sinal por meio do canal *Fim[j]?* enviado pelo autômato tarefa de atuação (Figura 5.3(d)) indicando o fim do ciclo de operação da malha j. Nesse caso, ocorrerá uma transição para o lugar *commited Fim_Ciclo* e, imediatamente, uma transição para o lugar *EsperAtivTarefa*, onde permanecerá até que um novo ciclo de operação da malha seja iniciado, ou seja, quando o relógio *Resp_F* for igual a h. Se a condição *Resp_F* $>$ *DM* for satisfeita durante o ciclo de operação da malha, o autômato alcançará o lugar *N_Escalon* indicando que o *deadline* fim a fim da malha não foi cumprido. A transição para o lugar *N_Escalon* poderá ocorrer em duas situações: quando o autômato estiver no lugar *Tx_Msg_CA* (significa que a mensagem está pronta para ser transmitida ou em transmissão) ou no lugar *Esper_Fim* (Esperando o fim da tarefa de atuação, ou seja, o fim do atual ciclo de operação da malha). Em ambos os casos, ocorrerá uma transição para o lugar *commited P_Deadline* e, imediatamente uma transição para o lugar *N_Escalon*. Nessa última transição, os relógios *Resp_F* e *Resp_CA* são zerados.

O autômato tarefa de atuação (Figura 5.3(d)) possui o relógio local *Comp_A*, que é usado para marcar o tempo de computação da tarefa de atuação (Ca). Esse autômato espera no seu lugar inicial a recepção da mensagem controlador-atuador com identificador *f*, que é modelada pelo canal *RxMsg[f]?*. Após receber a mensagem, a tarefa de atuação é executada durante *Ca* unidades de tempo. Em seguida, envia um sinal via o canal *Fim[j]!* para o autômato tarefa de aquisição e controle indicando que o ciclo de operação da malha j foi finalizado.

A comunicação via rede CAN é modelada por dois autômatos: controlador CAN (Figura 5.3(b)) e arbitragem (Figura 5.3(c)). Para cada mensagem no sistema, a ferramenta UPPAAL gera um autômato controlador CAN. Enquanto que, existe apenas um autômato arbitragem no sistema.

O autômato controlador CAN (Figura 5.3(b)) modela o comportamento temporal da mensagem de dados de acordo com a especificação do protocolo CAN. O autômato possui um relógio local *Com* usado para marcar o tempo de transmissão da mensagem de dados e o intervalo entre mensagens. Inicialmente, o autômato controlador CAN está no lugar inicial *N_Req*. Esse lugar representa o estado em que o controlador CAN do nó não recebeu uma requisição de transmissão de mensagem. Quando o autômato receber uma requisição de transmissão de mensagem via o canal *ReqTxMsg[i]?* enviado pelo autômato tarefa, uma transição do lugar *N_Req* para o lugar committed *C1* e, imediatamente, uma transição para o lugar *BufferTx* será liberada, o sinal *TxMsg!* será enviado para o autômato arbitragem (Figura 4.1(c)) e o *array Arb[i]*, indexado com o valor do identificador da mensagem m_i, será ajustado para '1', indicando que a

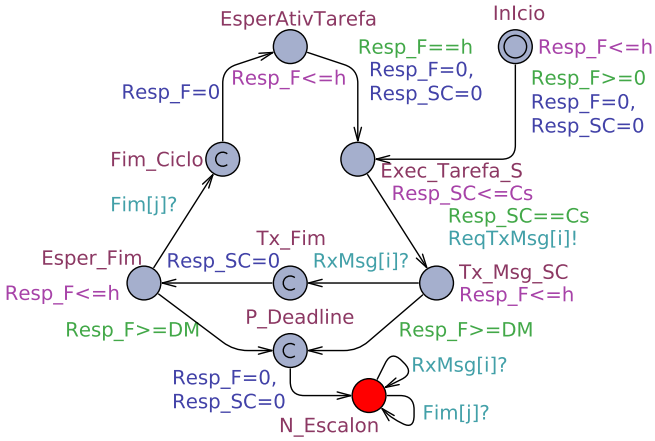
mensagem está pronta para entrar no processo de arbitragem. Assim que a mensagem vencer o processo de arbitragem, o autômato arbitragem sincronizará com o autômato controlador CAN via o canal *Sucess_Arb[i]?*, a transição para o lugar *Tx_Msg* será liberada e o relógio *Com* será zerado. O autômato permanecerá nesse lugar até o relógio Com marcar Cm unidades de tempo (modelado pela invariante de estado $Com <= Cm$ e o guarda $Com == Cm$), indicando que a mensagem foi transmitida com sucesso. Então, a transição para o lugar *IFS* será liberada, o sinal *RxMsg!* será enviado para o autômato tarefa e o relógio Com e a variável *Arb[i]* serão zerados. O lugar *IFS* representa o intervalo entre mensagens com duração de três tempos de bit ($TIFS = 3 * \tau_{bit}$). Assim que o relógio Com marcar três tempos de bit, a transição para o lugar committed *N_Req* será liberada e a informação de que o barramento está livre será enviada para o autômato arbitragem por meio do canal *Ack[i]!*.

O controle de acesso ao meio é modelado pelo autômato arbitragem (Figura 5.3(c)). O lugar inicial *N_Req_Tx_Msg* representa o estado onde não existe nenhuma mensagem pronta para ser transmitida. Assim que pelo menos um controlador CAN tiver alguma mensagem pronta para ser transmitida, o autômato receberá um sinal via o canal *TxMsg?* emitido pelo respectivo autômato controlador CAN. Então, a transição para o lugar committed *Arbit* será liberada e a variável *ID* será zerada. A variável *ID*, assim como as variáveis *i* e *f* (usadas nos autômatos tarefa de aquisição e controle, tarefa de atuação e controlador CAN) representam o identificador da mensagem m_i. No lugar *Arbit*, a mensagem com maior prioridade que requisitou uma transmissão será escolhida para ser transmitida. Para isso, o *array Arb[ID]* será verificado item por item, com o índice *ID* variando de 0 à *n-1*, onde *n* equivale ao número de identificadores (mensagens) do sistema. Os identificadores das mensagens são ordenados de forma crescente. A mensagem com maior prioridade possui *ID = 0*, e a mensagem com menor prioridade *ID = n-1*. Então, o primeiro elemento do *array Arb[ID]* com valor maior que 0 vencerá o processo de arbitragem, e um sinal via o canal *Sucess_Arb[ID]!* será enviado para o autômato controlador CAN que transmitirá a mensagem com identificador ID. O autômato permanecerá no lugar *Tx_Msg* até receber o sinal de reconhecimento via *Ack[ID]?*, indicando que a mensagem foi transmitida com sucesso e que o barramento já está livre novamente para iniciar um novo processo de arbitragem. Então, a variável *ID* será zerada e o autômato retornará para o lugar *Arbit*, onde verificará se existe alguma mensagem pendente para ser transmitida. Caso não exista, o autômato retornará para o seu lugar inicial *N_Req_Tx_Msg*.

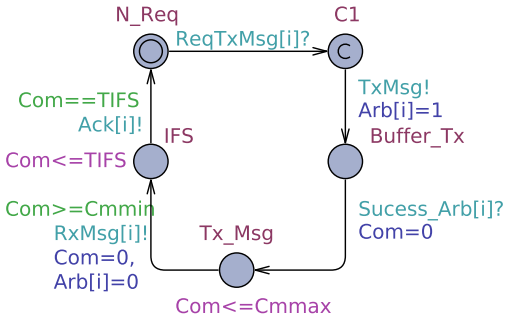
5.2.2 Modelagem da arquitetura S-CA

O modelo do NCS CAN com arquitetura SC-A (Figura 5.4(b)) é constituído pelos autômatos (Figura 5.4): tarefa de aquisição; tarefa de controle e atuação; controlador CAN e arbitragem. Nesse modelo, cada malha de controle j possui os autômatos tarefa aquisição, e tarefa controle e atuação. Esses autômatos são gerados automaticamente pela ferramenta UPPAAL. O código

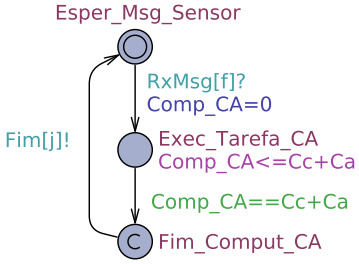
UPPAAL do modelo é apresentado na seção D.2 do Apêndice D.



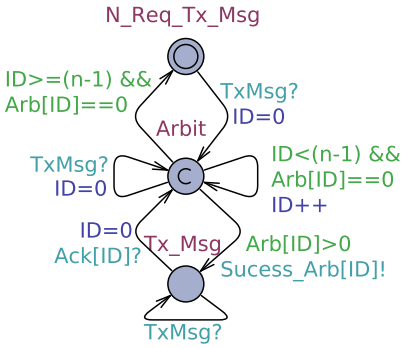
(a) Tarefa aquisição.



(b) Controlador CAN.



(c) Tarefa controle e atuação.



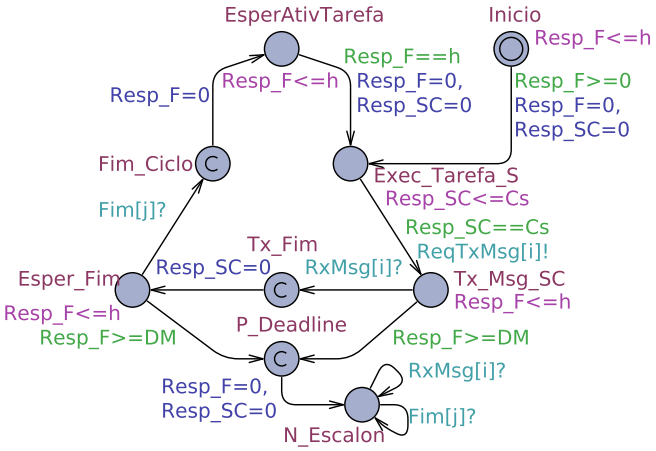
(d) Arbitragem.

Figura 5.4: Conjunto de Autômatos Temporizados para o modelo de NCS CAN na arquitetura S-CA.

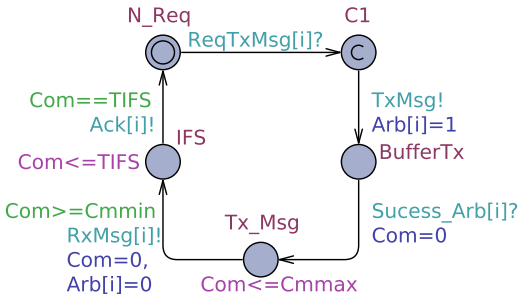
A descrição dos autômatos são semelhantes aos descritos no modelo anterior. A principal diferença é que a execução da tarefa de controle é modelada em conjunto com a execução da tarefa de atuação, no autômato tarefa controle e atuação, ao contrário do primeiro modelo, onde a execução da tarefa de controle foi modelada em conjunto com a tarefa aquisição. Além disso, o autômato que modela a tarefa de aquisição, possui o relógio local *Resp_SC* ao invés do relógio *Resp_CA*. Esse relógio é usado para marcar o tempo de computação da tarefa de aquisição Cs e contar o tempo de resposta da mensagem sensor-controlador (m_{sc_j}). Os modelos da rede CAN funcionam de modo idêntico aos apresentados no modelo anterior.

5.2.3 Modelagem da arquitetura S-C-A

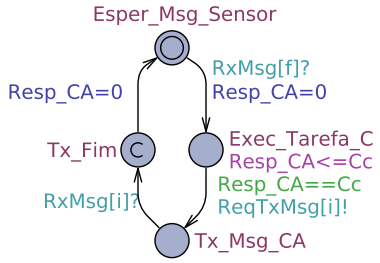
Na arquitetura S-C-A (Figura 5.5(c)), o modelo do NCS CAN[3] é constituído por cinco autômatos (Figura 5.5): tarefa aquisição; tarefa controle; tarefa atuação; controlador CAN e arbitragem. Cada malha de controle j possui três automatos: tarefa aquisição, tarefa controle e tarefa atuação. Os autômatos de cada malha são gerados automaticamente pela ferramenta UPPAAL. O código UPPAAL do modelo é apresentado na Seção D.3 do Apêndice D.



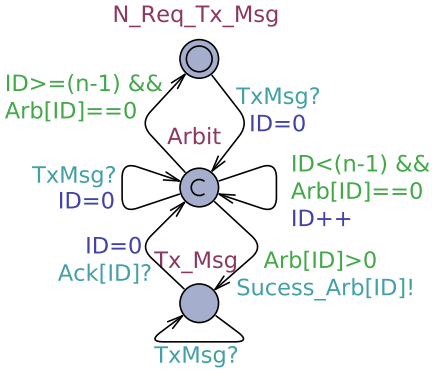
(a) Tarefa aquisição.



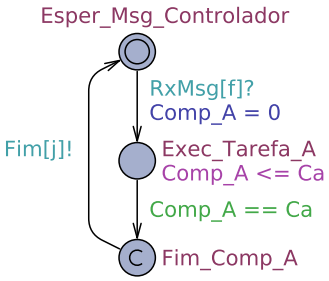
(b) Controlador CAN.



(c) Tarefa controle.



(d) Arbitragem.



(e) Tarefa atuação.

Figura 5.5: Conjunto de Autômatos Temporizados para o modelo de NCS CAN na arquitetura S-C-A.

Os modelos dessa arquitetura são também semelhantes aos descritos nos modelos anteriores. A principal diferença é que a execução das tarefas de aquisição, controle e atuação são modelados separadamente. O autômato que modela a tarefa de aquisição, possui o relógio local *Resp_SC*, que é usado para marcar o tempo de computação da tarefa de aquisição Cs e contar

[3]Uma versão simplificada desse modelo foi desenvolvida por Sá et al. (2010).

o tempo de resposta da mensagem sensor-controlador (m_{sc_j}). O autômato que modela a tarefa de controle, possui o relógio local *Resp_CA*, que é usado para marcar o tempo de computação da tarefa de controle Cc e contar o tempo de resposta da mensagem controlador-atuador (m_{ca_j}). Os modelos da rede CAN funcionam também de modo idêntico aos apresentados nos modelos anteriores.

5.3 Estudos de Caso dos Modelos de NCSs com *Offset* Dinâmico, Relógios e Barramento Ideais

Os estudos de caso apresentados nesta seção consiste de um NCS CAN com cinco malhas de controle, cujos atributos das tarefas e mensagens estão indicados nas Tabelas 5.1 e 5.2. Algumas propriedades lógicas e temporais do sistema foram formuladas e verificadas de acordo com os modelos descritos para as arquiteturas SC_A, S_CA e S_C_A. Em particular, os tempos de resposta fim a fim e os tempos de resposta das mensagens sensor-controlador e/ou controlador-atuador no melhor e pior caso serão determinados para cada malha de controle do sistema.

Tabela 5.1: Período, deadline e tempos de computação das tarefas das cinco malhas de controle (M_j) em ms.

M_j	h_j	D_{M_j}	C_{s_j}	C_{c_j}	C_{a_j}
0	8	8	0,520	2,551	0,481
1	10	10	0,520	2,352	0,481
2	18	18	0,520	2,352	0,481
3	48	48	0,520	2,551	0,481
4	71	71	0,520	2,352	0,481

Tabela 5.2: Tempos de transmissão das mensagens das cinco malhas de controle (M_j).

M_j	$C^{min}_{m_{sc_j}}/C^{max}_{m_{sc_j}}$	$C^{min}_{m_{ca_j}}/C^{max}_{m_{ca_j}}$
0	76*τ_{bit}/92*τ_{bit}	76*τ_{bit}/92*τ_{bit}
1	76*τ_{bit}/92*τ_{bit}	76*τ_{bit}/92*τ_{bit}
2	76*τ_{bit}/92*τ_{bit}	76*τ_{bit}/92*τ_{bit}
3	76*τ_{bit}/92*τ_{bit}	76*τ_{bit}/92*τ_{bit}
4	76*τ_{bit}/92*τ_{bit}	76*τ_{bit}/92*τ_{bit}

5.3.1 Verificação simbólica

As propriedades a serem verificadas são descritas na linguagem de especificação no formalismo da ferramenta UPPAAL, onde: S_j, C_j e SC_j equivalem, respectivamente, aos autômatos tarefa aquisição, tarefa controle e tarefa aquisição e controle de uma malha j; $N\_Escalon$, Fim_Ciclo e Tx_Fim são lugares e $Resp\_F$, $Resp\_SC$ e $Resp\_CA$ relógios locais equivalentes aos seus autômatos.

1. O sistema é livre de *deadlock*?
   A [] not deadlock
2. O *deadline* fim a fim da malha j é satisfeito?
   A [] not S_j*.N_Escalon*
3. Qual o tempo de resposta fim a fim no pior caso da malha j?
   $sup\{S_j.Fim\_Ciclo\} : S_j.Resp\_F$
4. Qual o tempo de resposta fim a fim no melhor caso da malha j?
   $inf\{S_j.Fim\_Ciclo\} : S_j.Resp\_F$
5. Qual o tempo de resposta no pior caso da mensagem sensor-controlador da malha j?
   $sup\{S_j.Tx\_Fim\} : S_j.Resp\_SC$
6. Qual o tempo de resposta no melhor caso da mensagem sensor-controlador da malha j?
   $inf\{S_j.Tx\_Fim\} : S_j.Resp\_SC$
7. Qual o tempo de resposta no pior caso da mensagem controlador-atuador da malha j?
   $sup\{SC_j.Tx\_Fim\} : SC_j.Resp\_CA$
8. Qual o tempo de resposta no melhor caso da mensagem controlador-atuador da malha j?
   $inf\{SC_j.Tx\_Fim\} : SC_j.Resp\_CA$
9. Qual o tempo de resposta no pior caso da mensagem controlador-atuador da malha j?
   $sup\{C_j.Tx\_Fim\} : C_j.Resp\_CA$
10. Qual o tempo de resposta no melhor caso da mensagem controlador-atuador da malha j?
    $inf\{C_j.Tx\_Fim\} : C_j.Resp\_CA$

A propriedade 1 é usada para verificar se existe *deadlock* no modelo e a propriedade 2 para verificar se o *deadline* fim a fim de cada malha será satisfeito. Isso ocorrerá, se e somente se, o lugar *N_Escalon* nunca for alcançado para todo caminho. As propriedades 3 e 4 são usadas para determinar, respectivamente, os tempos de resposta fim a fim no pior e melhor caso das malhas

j em qualquer um dos três modelos. As propriedades 5 e 6 são usadas para determinar, respectivamente, os tempos de resposta da mensagem sensor-controlador no pior e melhor caso das malhas j nos modelos das arquiteturas S-CA e S-C-A. As propriedades 7 e 8 são usadas para determinar, respectivamente, os tempos de resposta da mensagem controlador-atuador no pior e melhor caso das malhas j no modelo da arquitetura SC-A. As propriedades 9 e 10 são usadas para determinar, respectivamente, os tempos de resposta da mensagem controlador-atuador no pior e melhor caso das malhas j no modelo da arquitetura S-C-A. Os resultados dos tempos de resposta das mensagens sensor-controlador e controlador-atuador, no pior caso, respectivamente, para os modelos S-CA e SC-A, serão comparados com os resultados obtidos via técnica analítica de Davis et al. (2007) (Ver seção 3.4.3 no Capítulo 3) adaptada para NCSs CAN. Enquanto que, os resultados dos tempos de resposta fim a fim, no pior caso, para o modelo S-C-A, serão comparados com os resultados obtidos via técnica analítica de Henderson, Kendall e Robson (2001) corrigida[4] com o trabalho de Davis et al. (2007). Programas em linguagem C foram desenvolvidos para determinar os tempos de resposta via as técnicas analíticas de Davis et al. (2007) e Henderson, Kendall e Robson (2001).

Inicialmente, consideramos o NCS apenas com três malhas de controle (0, 1 e 2). A taxa de transmissão foi igual a 500 Kbits/s. Para todos os três modelos, foi verificada a ausência de *deadlock* e a propriedade 2 foi satisfeita para todas as malhas de controle do sistema. Portanto, os requisitos de tempo do sistema foram satisfeitos. O tempo gasto para verificação dessas propriedades, em cada modelo, foi de aproximadamente 20 s.

Para o modelo da arquitetura SC_A (Figura 5.3), as propriedades 3, 4, 7 e 8 foram verificadas (tempo de verificação igual a 20 s) e os resultados são apresentados na Tabela 5.3. Para o modelo da arquitetura S_CA (Figura 5.4), as propriedades 3, 4, 5 e 6 foram verificadas (tempo de verificação igual a 20 s) e os resultados são apresentados na Tabela 5.4. Finalmente, para o modelo da arquitetura S_C_A (Figura 5.5) as propriedades de 3, 4, 5, 6, 9 e 10 foram verificadas (tempo de verificação igual a 14 min) e os resultados são apresentados na Tabela 5.5.

Tabela 5.3: Tempos de resposta das malhas de controle (M_j) para o NCS CAN na arquitetura SC-A em ms.

	CMC				**Analítica**
M_j	$R_{M_j}^{max}$	$R_{M_j}^{min}$	$R_{m_{ca_j}}^{max}$	$R_{m_{ca_j}}^{min}$	$R_{M_j}^{max}$
0	3,926	3,704	3,445	3,223	3,926
1	3,917	3,505	3,505	3,024	3,917
2	3,917	3,505	3,436	3,024	3,917

[4]A análise de Henderson, Kendall e Robson (2001) é baseada no trabalho incompleta de Tindell e Burns (1994b), Tindell e Burns (1994a) e Tindell, Burns e Wellings (1995).

Tabela 5.4: Tempos de resposta das malhas de controle (M_j) para o NCS CAN na arquitetura S-CA em ms.

	CMC				**Analítica**
M_j	$R_{M_j}^{max}$	$R_{M_j}^{min}$	$R_{m_{sc_j}}^{max}$	$R_{m_{sc_j}}^{min}$	$R_{M_j}^{max}$
0	3,926	3,704	0,894	0,672	3,926
1	3,917	3,505	1,084	0,672	3,917
2	3,917	3,505	1,084	0,672	3,917

Tabela 5.5: Tempos de resposta das malhas de controle (M_j) para o NCS CAN na arquitetura S-C-A em ms.

	CMC						**Analítica**	
M_j	$R_{M_j}^{max}$	$R_{M_j}^{min}$	$R_{m_{sc_j}}^{max}$	$R_{m_{sc_j}}^{min}$	$R_{m_{ca_j}}^{max}$	$R_{m_{ca_j}}^{min}$	$R_{M_j}^{max}$	$R_{M_j}^{min}$
0	4,300	3,856	0,894	0,672	2,925	2,703	4,300	3,856
1	4,291	3,657	1,084	0,672	2,916	2,504	4,481	3,657
2	4,291	3,657	1,084	0,672	2,916	2,504	4,481	3,657

Veja que os resultados obtidos nas Tabelas 5.3 e 5.4 são idênticos para os tempos de resposta fim a fim no pior e melhor caso. Esses resultados (apenas pior caso) são também idênticos aos obtidos por meio da análise de Davis et al. (2007). Observe na Tabela 5.3, que os tempos de resposta fim a fim no pior caso para cada malha j ($R_{M_j}^{max}$) equivale ao tempo de resposta no pior caso da mensagem controlador-atuador ($R_{m_{ca_j}}^{max}$) mais o tempo de computação da tarefa de atuação ($C_{a_j}^{max}$). Por exemplo, para a malha $j = 0$, $R_{M_j}^{max} = R_{m_{ca_j}}^{max} + C_{a_j}^{max} = 3{,}445 + 0{,}481 = 3{,}926\ \mu$s. Uma análise semelhante pode ser feita para os tempos de resposta fim a fim no melhor caso e para os resultados apresentados na Tabela 5.4.

Para os resultados obtidos a partir do modelo da arquitetura S_C_A (ver Tabela 5.5), observe que o tempo de resposta fim a fim no pior caso para a malha 0 equivale a soma dos tempos de resposta no pior caso das mensagens e do tempo de computação da tarefa de atuação ($R_{M_0}^{max}$ = 0,894 + 2,925 + 0,481 = 4,300 ms). Esse resultado é compatível com a análise de escalonamento holístico (HENDERSON; KENDALL; ROBSON, 2001). O tempo de resposta $R_{m_{sc_1}}^{max} = 0{,}894\ \mu s$ equivale a soma do tempo de computação da tarefa de aquisição (0,520 ms), atraso bloqueio no pior caso mais o intervalo entre mensagens (0,184 + 0,006 = 0,190 ms) e tempo de transmissão da mensagem no pior caso (0,184 ms). O valor de $R_{m_{ca_1}}^{max} = 2{,}925$ ms pode ser obtido de forma semelhante, ou seja, a soma do tempo de computação da tarefa de controle (2,551 ms), atraso de bloqueio no pior caso mais o intervalo entre mensagens (0,184 + 0,006 = 0,190 ms) e o tempo de transmissão da mensagem no pior caso (0,184 ms). O resultado dos tempos de resposta da malha 1 indicam, aparentemente, que os cenários de pior caso para as mensagens m_{sc_1} e m_{ca_1} ocorrem no mesmo ciclo de operação da malha. Entretanto, observamos que os tempos de

resposta fim a fim das malhas 2 e 3 não correspondem a soma dos tempos de resposta no pior caso das mensagens e do tempo de computação da tarefa de atuação. Nas malhas de controle 1 e 2, a soma desses tempos (1,084 + 2,916 + 0,481 = 4,481 ms) é 0,190 ms maior que o tempo de resposta fim a fim obtido via a técnica de verificação de modelos ($R_{M_3}^{max}$ = 4,291 ms). Portanto, para NCSs com a arquitetura apresentada nessa seção, a análise de escalonamento holístico atual resulta em tempos de resposta fim a fim pessimistas. No cenário de melhor caso, observamos que os tempos de resposta das mensagens m_{sc_j} ($R_{m_{sc_j}}^{min}$) são iguais aos obtidos usando a análise dos tempos de resposta no melhor caso das mensagens CAN de Henderson, Kendall e Robson (2001), ou seja, a soma do tempo de computação da tarefa de aquisição (0,520 ms) e do tempo de transmissão no melhor caso (0,152 ms). Esse é um cenário onde a mensagem não sofre bloqueio e interferência de outras mensagens. Um resultado semelhante é obtido para as mensagens m_{ca_j}.

Não foi possível verificar as propriedades para mais de três malhas de controle, devido uma explosão do espaço de estados. Para resolver esse problema foi utilizada a verificação estatística do modelo. Na seção seguinte apresenta-se os resultados obtidos utilizando essa abordagem.

5.3.2 Verificação estatística

Nesta seção apresenta-se a verificação estatística do modelo para a arquitetura S-C-A para o NCS com as cinco malhas de controle apresentadas na Tabela 5.1. As seguintes propriedades serão verificadas para determinar a probabilidade dos tempos de resposta fim a fim (propriedade 1), tempos de resposta da mensagem sensor-controlador (propriedade 2) e tempos de resposta da mensagem controlador-atuador (propriedade 3) satisfazerem seus deadlines. A verificação dessas propriedades será realizada com os parâmetros estatísticos $\epsilon = 0{,}01$ e $\alpha = 0{,}01$.

1. *Pr[Sj.Resp_F <= DMj] (<> Sj.Fim_Ciclo)*
2. *Pr[Sj.Resp_SC <= DMj] (<> Sj.Fim_Ciclo)*
3. *Pr[Sj.Resp_CA <= DMj] (<> Cj.Fim_Ciclo)*

A ferramenta retorna um limite inferior e superior da probabilidade acumulada da propriedade verificada com confidência de 99%. Os gráficos das funções de distribuição acumulada dos tempos de resposta fim a fim, dos tempos de resposta da mensagem sensor-controlador e dos tempos de resposta da mensagem controlador-atuador das cinco malhas de controle são apresentados nas Figuras 5.6 a 5.20. O tempo gasto na verificação foi de 15 s. Na malha de controle 0, a probabilidade do tempo de resposta fim a fim ser igual a 4070 μs é de quase 100%. Isso significa que a probabilidade de ocorrer atrasos de controle maiores do que 4070 μs é muito pequena. Pode-se também observar que no melhor caso, o atraso de controle é igual a aproximadamente 3860 μs. Então, pode-se afirmar que, nas condições especificadas, o atraso

de controle para a malha 0 varia de 0 a 110 μs com confidência de 99%. Os tempos de resposta das mensagens sensor-controlador e controlador-atuador podem ser analisados de modo semelhante.

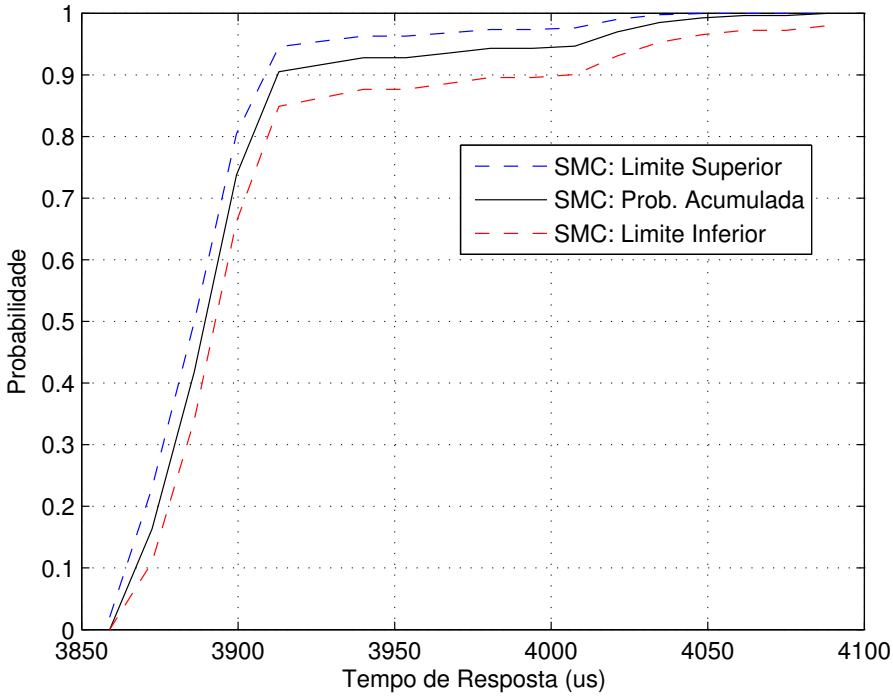


Figura 5.6: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 0.

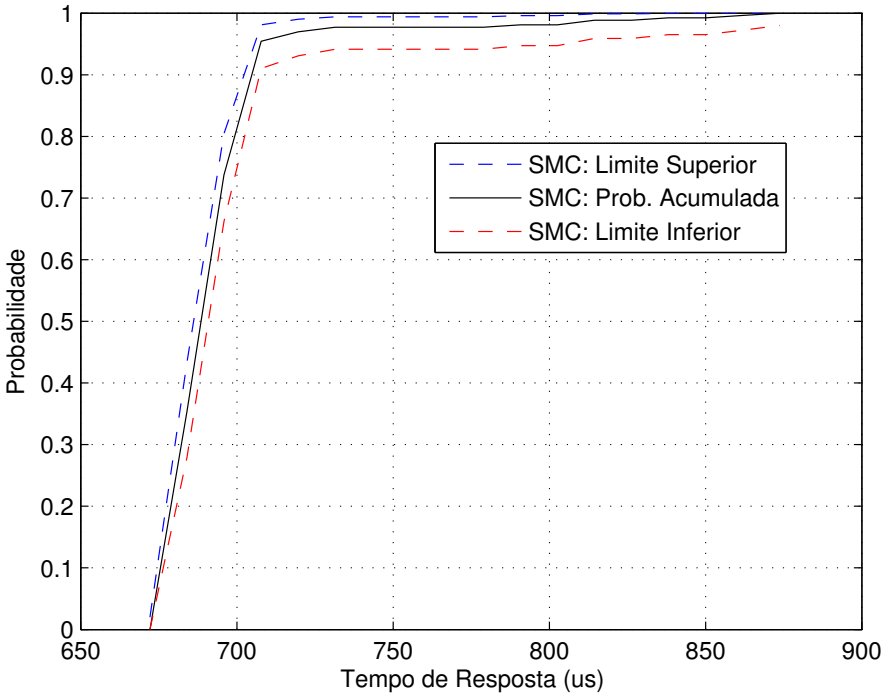


Figura 5.7: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 0.

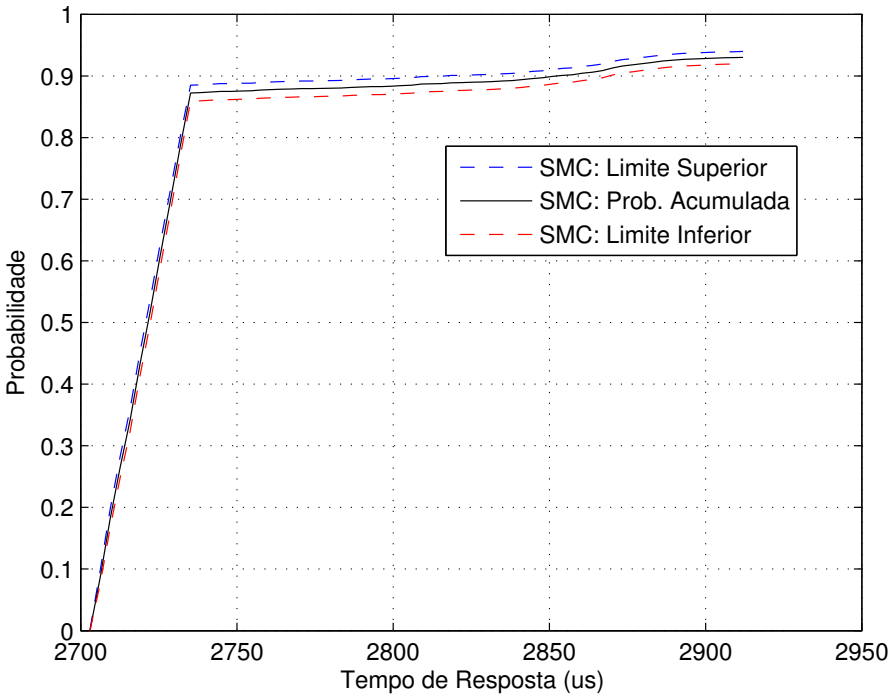


Figura 5.8: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 0.

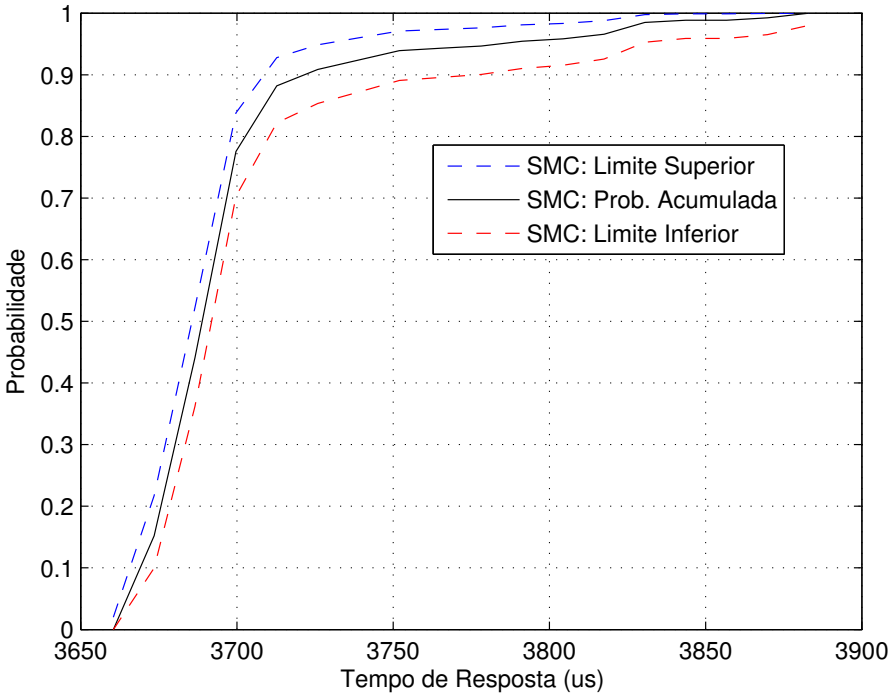


Figura 5.9: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 1.

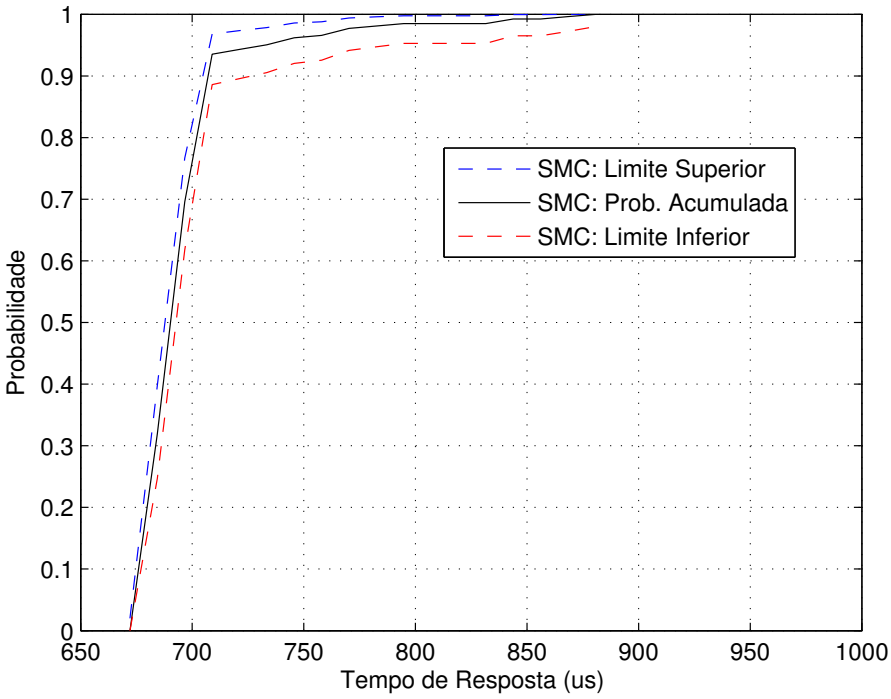


Figura 5.10: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 1.

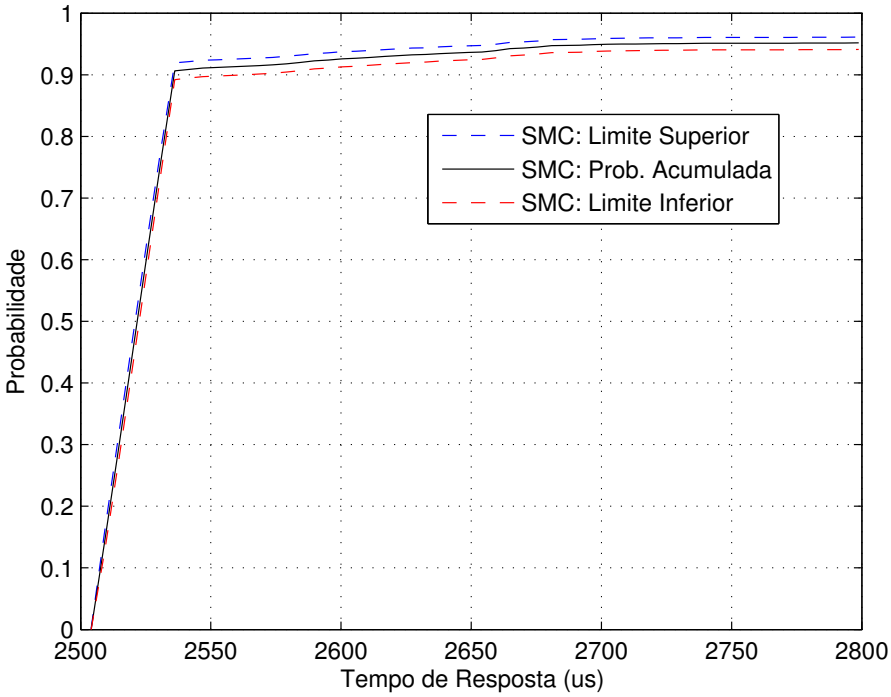


Figura 5.11: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 1.

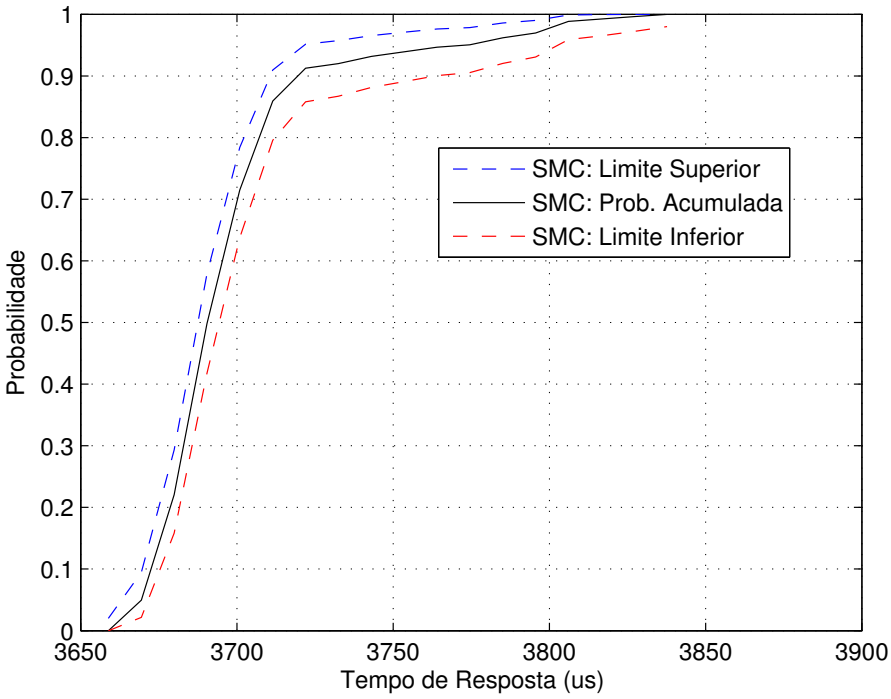


Figura 5.12: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 2.

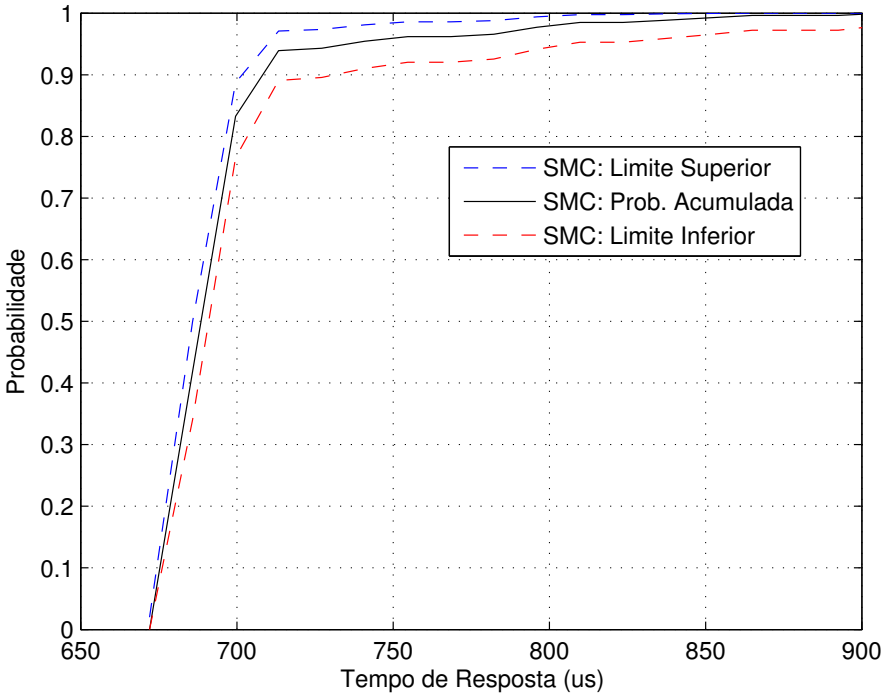


Figura 5.13: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 2.

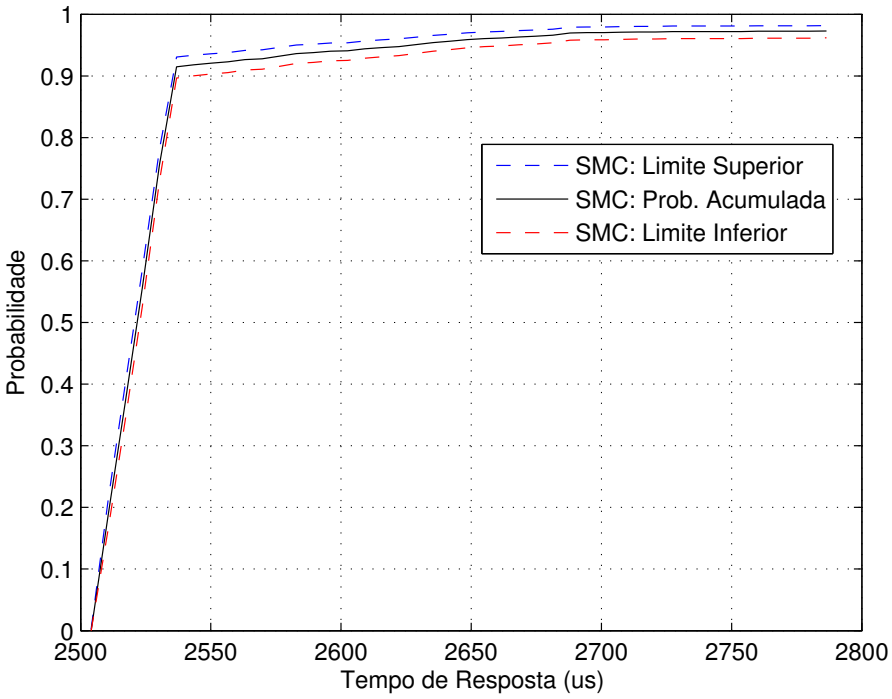


Figura 5.14: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 2.

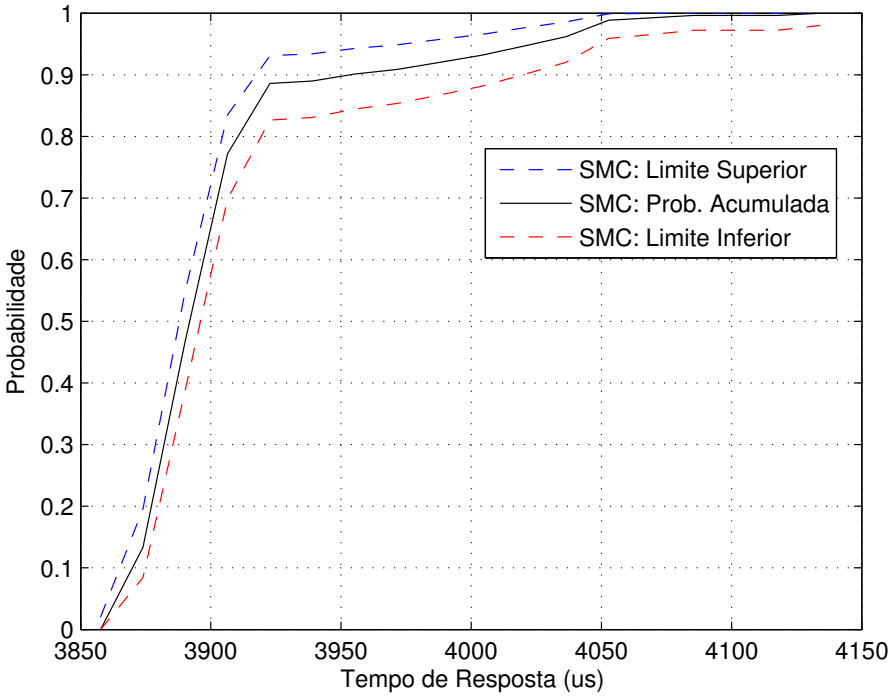


Figura 5.15: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 3.

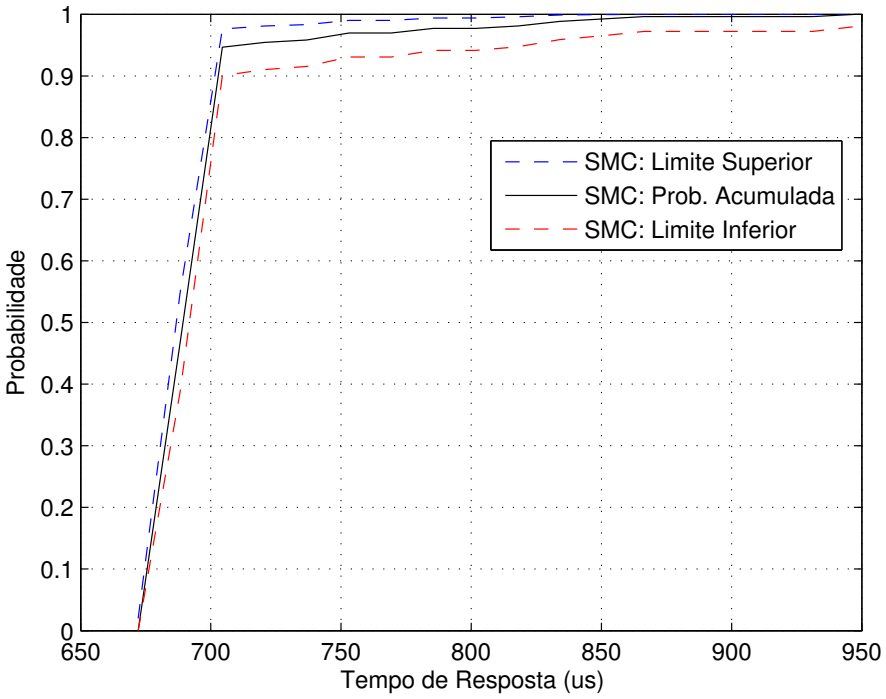


Figura 5.16: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 3.

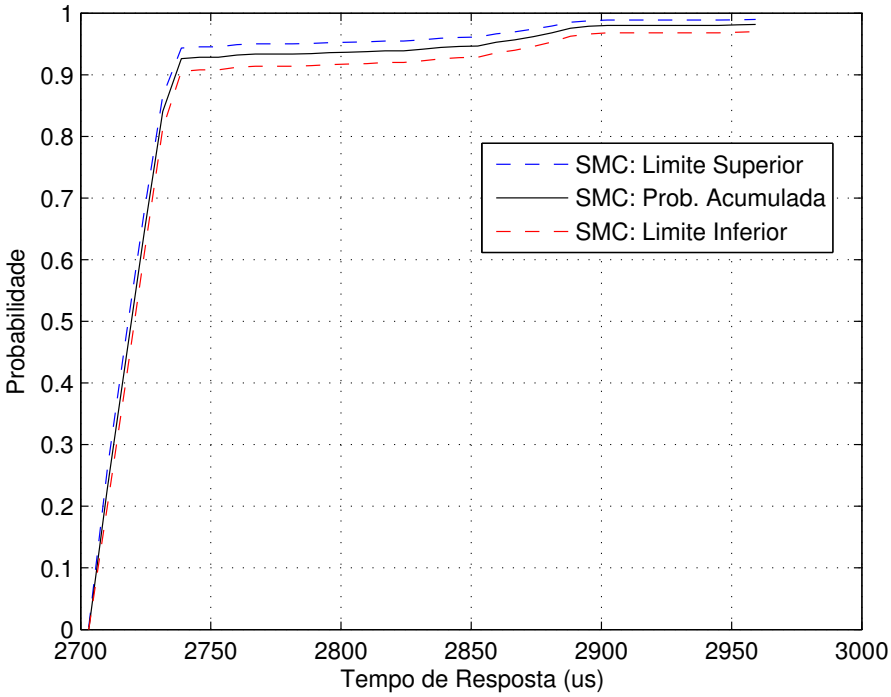


Figura 5.17: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 3.

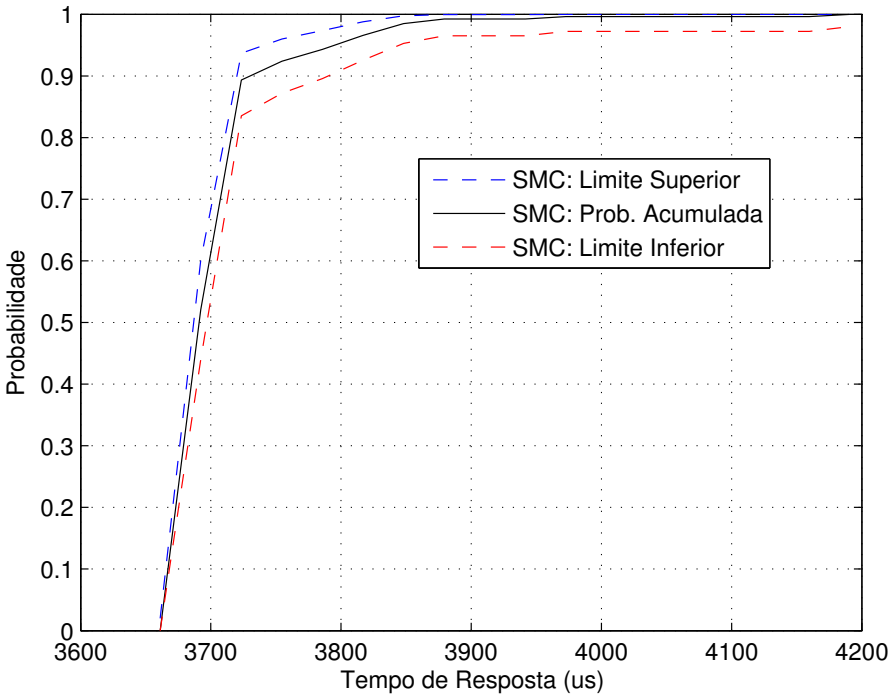


Figura 5.18: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 4.

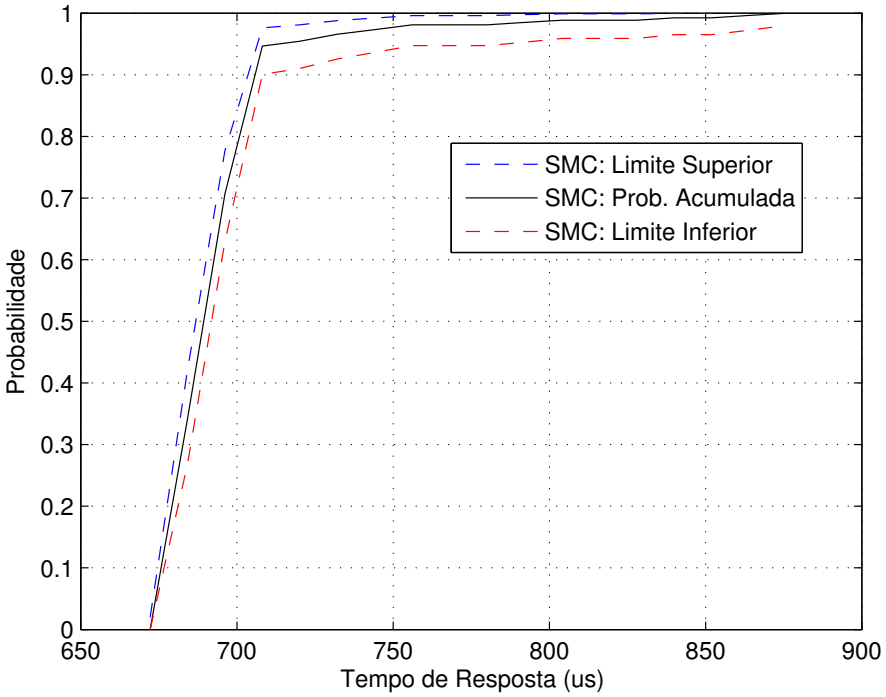


Figura 5.19: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 4.

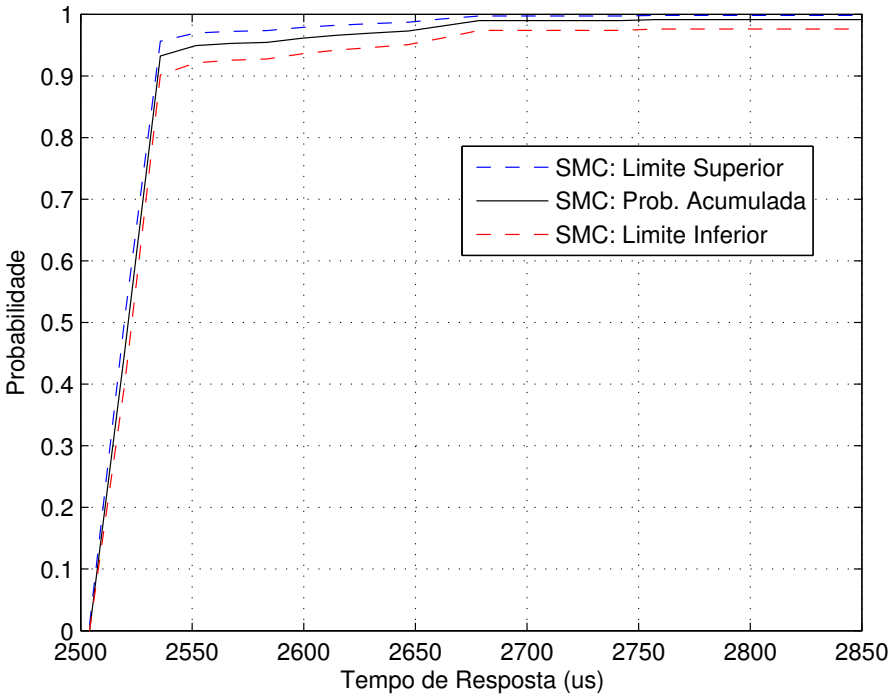


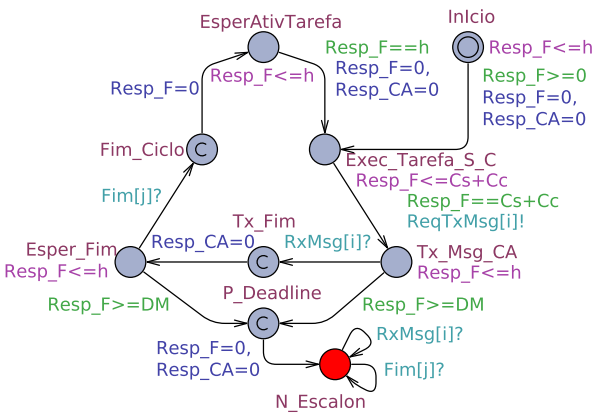
Figura 5.20: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 4.

5.4 Modelos formais de NCSs CAN com *offset* dinâmico, relógios ideais e barramento real

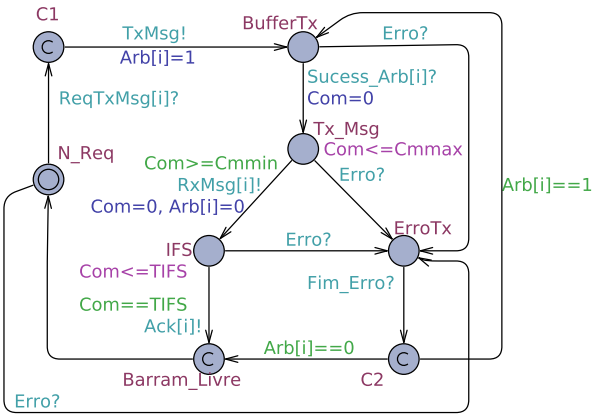
Nesta seção, os três modelos de NCSs CAN desenvolvidos na seção 5.2 serão estendidos para considerar erros no barramento.

5.4.1 Modelagem das arquiteturas SC-A, S-CA e S-C-A

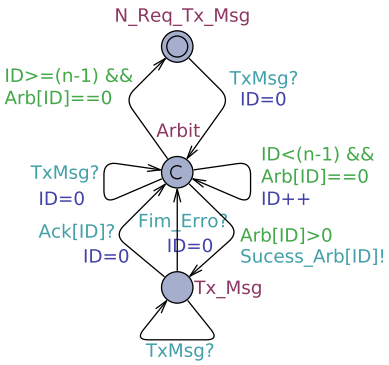
Os modelos são apresentados nas Figuras 5.21, 5.22 e 5.23. O funcionamento dos autômatos tarefa aquisição e controle e tarefa atuação, da arquitetura SC-A, são idênticos aos apresentados na seção 5.2.1. Enquanto que, o funcionamento dos autômatos tarefa aquisição e tarefa controle e atuação, da arquitetura S-CA, são idênticos aos apresentados na seção 5.2.2 e o funcionamento dos autômatos tarefa aquisição, tarefa controle e tarefa atuação, da arquitetura S-C-A, são idênticos aos apresentados na seção 5.2.3. O funcionamento dos autômatos restantes, controlador CAN, arbitragem, interferência e erros, são idênticos aos apresentados na seção 4.5 do Capítulo 4, exceto que no autômato controlador CAN, a transição temporizada do lugar *Tx_Msg* para o lugar *IFS* ocorre no intervalo de tempo [Cmmin, Cmmax], ao invés do instante de tempo em que o relógio *Com* = *Cm*. Isso significa que o tempo de transmissão da mensagem é variante, mas limitado no tempo. Essa variação foi modelada devido ao fato de que os dados transmitidos são variantes em valor (não em tamanho) e, possivelmente, haverá na sequência de *bytes* no campo de dados mais de cinco *bits* com mesmo nível lógico. Então, automaticamente, o controlador CAN inserirá *bits* para não violar o mecanismo de inserção de *bits*. Detalhes sobre o mecanismo de inserção de bits estão descritos na seção A.1 do Apêndice A.



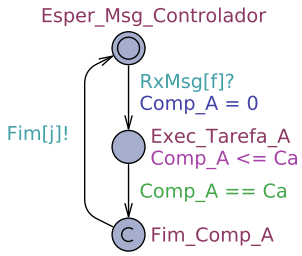
(a) Tarefa aquisição e controle.



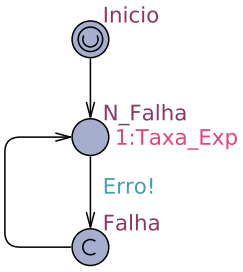
(b) Controlador CAN.



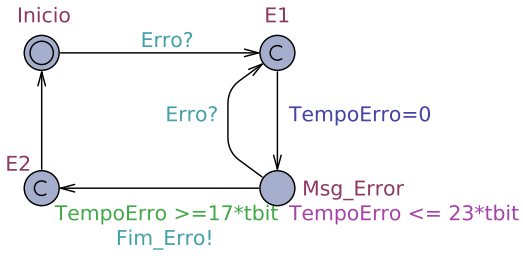
(c) Arbitragem.



(d) Tarefa atuação.

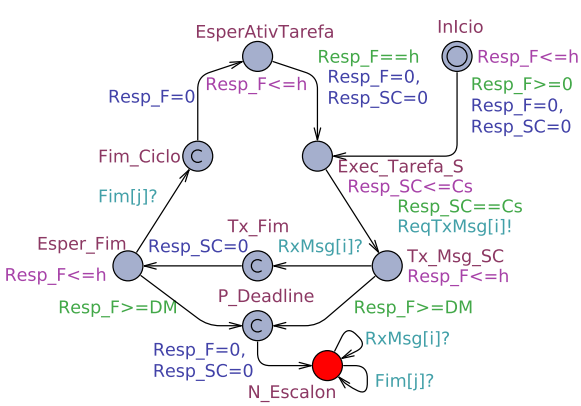


(e) Interferência.

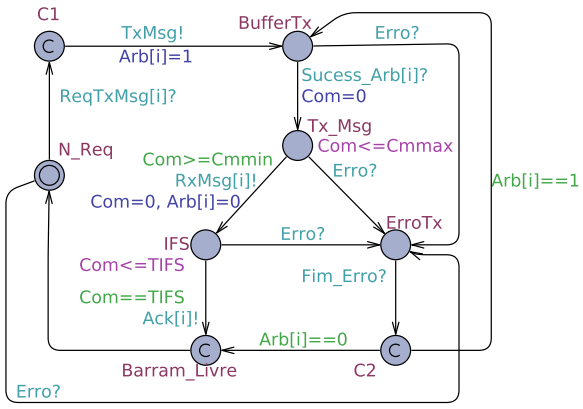


(f) Erros.

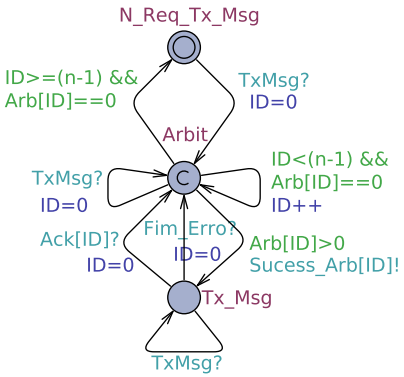
Figura 5.21: Conjunto de Autômatos Temporizados para o modelo de NCS CAN na arquitetura SC-A considerando erros no barramento.



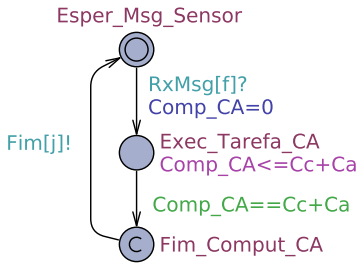
(a) Tarefa aquisição.



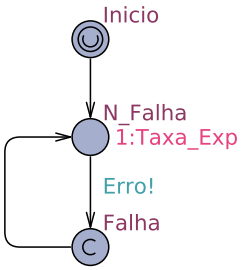
(b) Controlador CAN.



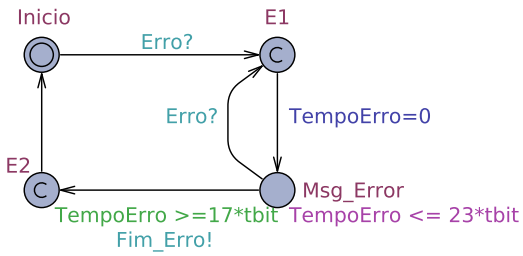
(c) Arbitragem.



(d) Tarefa controle e atuação.

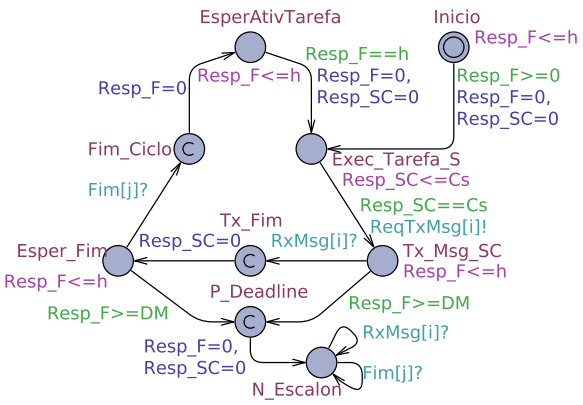


(e) Interferência.

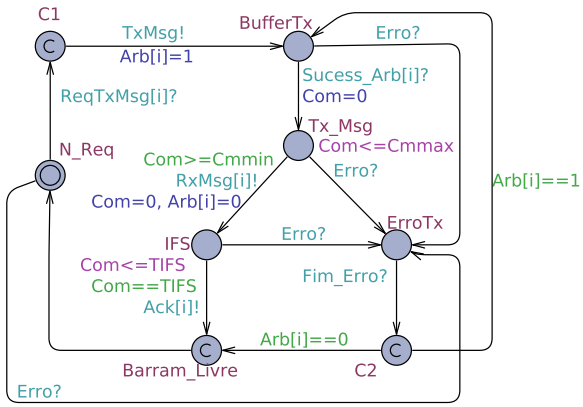


(f) Erros.

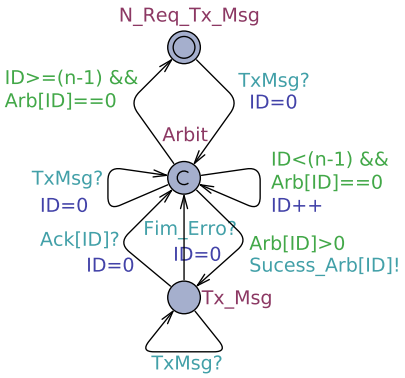
Figura 5.22: Conjunto de Autômatos Temporizados para o modelo de NCS CAN na arquitetura S-CA.



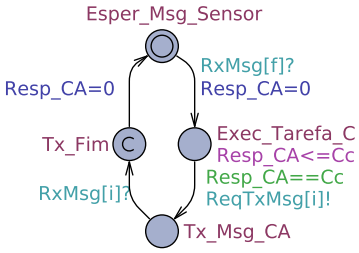
(a) Tarefa aquisição.



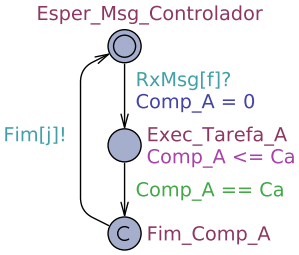
(b) Controlador CAN.



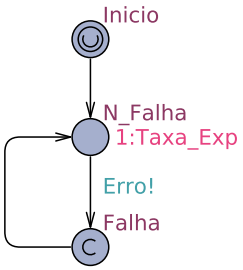
(c) Arbitragem.



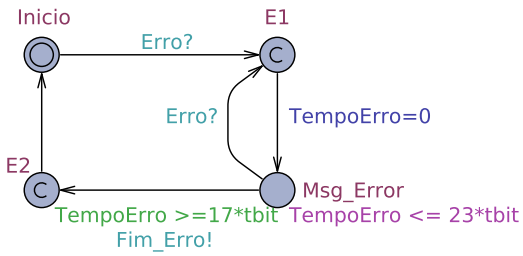
(d) Tarefa controle.



(e) Tarefa atuação.



(f) Interferência.



(g) Erros.

Figura 5.23: Conjunto de Autômatos Temporizados para o modelo de NCS CAN na arquitetura S-C-A.

5.5 Estudo de caso do modelo na arquitetura S-C-A com *off-set* dinâmico, relógios ideais e barramento real

Nesta seção apresenta-se a verificação estatística do modelo para a arquitetura S-C-A para o NCS com as cinco malhas de controle apresentadas na Tabela 5.1. As mesmas propriedades verificadas na seção 5.3.2 serão utilizadas para determinar a probabilidade dos tempos de resposta fim a fim e a probabilidade dos tempos de resposta das mensagens sensor-controlador e controlador-atuador satisfazerem seus *deadlines*. A verificação dessas propriedades também será realizada com os parâmetros estatísticos $\epsilon = 0{,}01$ e $\alpha = 0{,}01$.

Os gráficos das funções de distribuição acumulada (com os limites superiores e inferiores) dos tempos de resposta fim a fim e dos tempos de resposta da mensagem sensor-controlador e dos tempos de resposta da mensagem controlador-atuador das cinco malhas de controle sem taxas de erros (barramento ideal) e para uma taxa de 30 erros/s são apresentados nas Figuras 5.6 a 5.20. O tempo gasto na verificação foi de 15 s.

Foi verificado que as curvas da função de distribuição acumulada para o caso com erros no barramento estão sempre abaixo das curvas da função de distribuição acumulada para o caso sem erros no barramento, na faixa de probabilidade de 0 a pelo menos 95%. Como esperado, isso significa que os erros no barramento tendem a aumentar os tempos de resposta das mensagens e, consequentemente, do tempo de resposta fim a fim. Então, a partir de 95% de probabilidade, as curvas tenderam a se tornarem iguais. Observou-se que esse comportamento ocorreu devido a dois fatores: (1) o baixo fator de utilização da rede (em torno de 11,61%), que resulta em um alto número de ocorrências próximas ao tempo de resposta no melhor caso e (2) a ocorrência rara dos cenários de pior caso. Os gráficos para os tempos de resposta com erro no barramento se estendeu ao longo da faixa de probabilidade 99 a 100%. Por exemplo, veja na Figura 5.24 que a probabilidade de satisfazer o *deadline* é igual a 90% para um aumento de menos de 50 μs, ou seja, em torno de 3915 μs, com relação ao tempo de resposta fim a fim no melhor caso (em torno de 3865 μs). Resultados semelhantes podem ser vistos para os demais gráficos. Observe agora na malha de controle 4 (Figura 5.36) que, a partir de 3900 até 4350 μs, a probabilidade de satisfazer o *deadline* está entre 99 e 100%. Do mesmo modo, pode-se verificar resultados semelhantes para as demais malhas de controle.

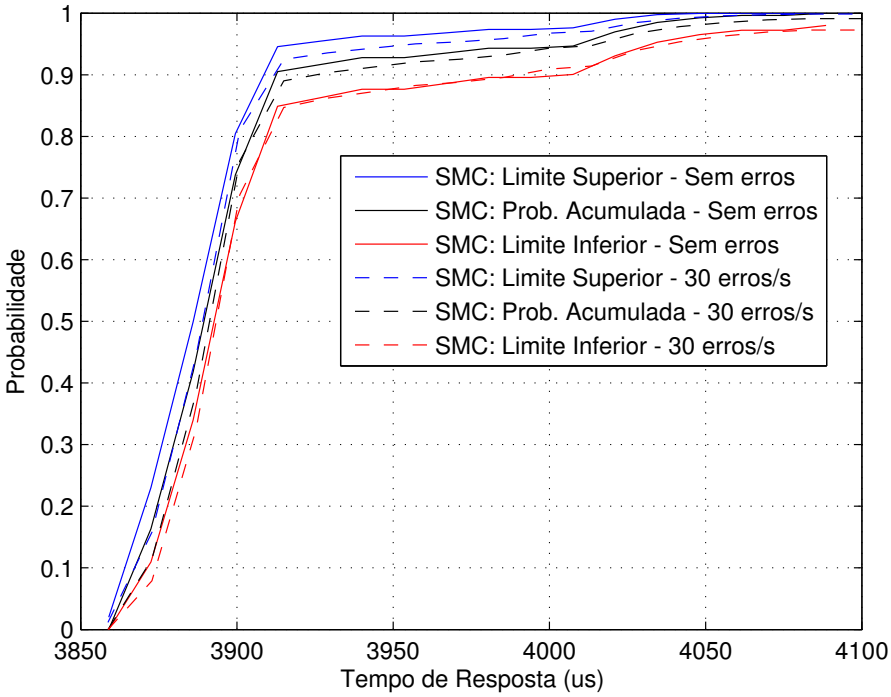


Figura 5.24: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 0 sem erros e com taxa de 30 erros/s no barramento.

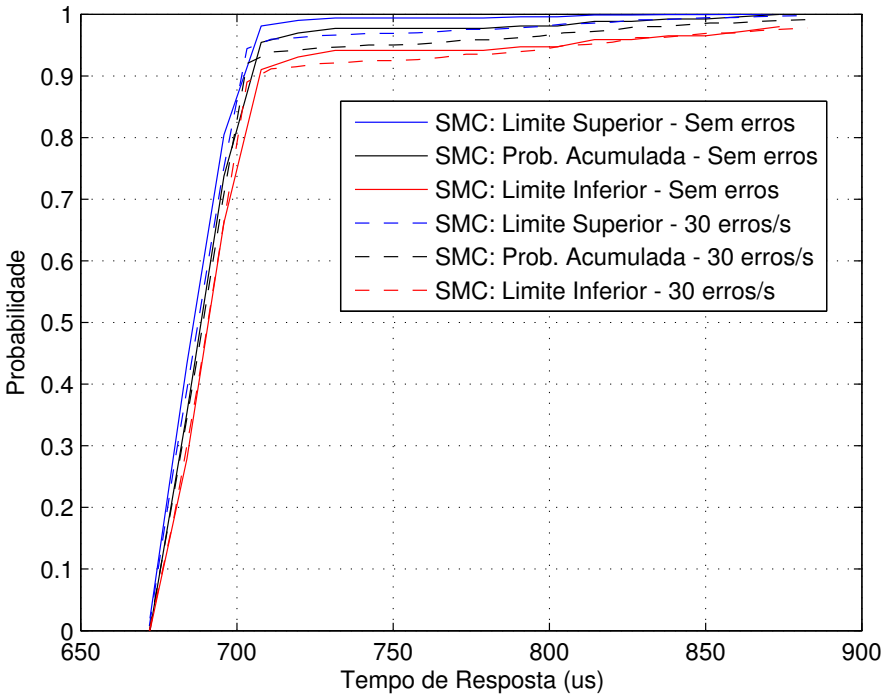


Figura 5.25: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 0 sem erros e com taxa de 30 erros/s no barramento.

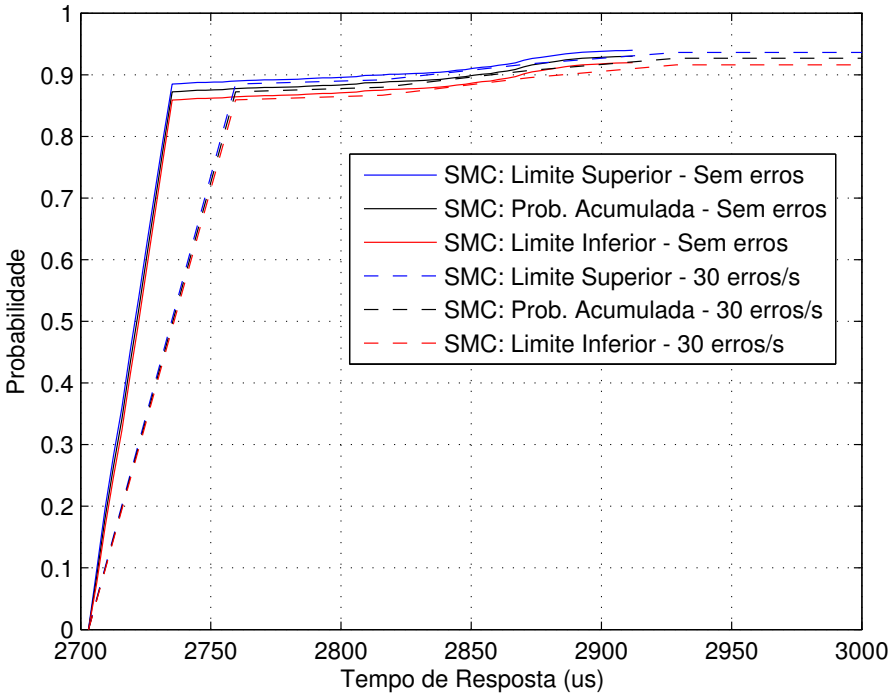


Figura 5.26: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 0 sem erros e com taxa de 30 erros/s no barramento.

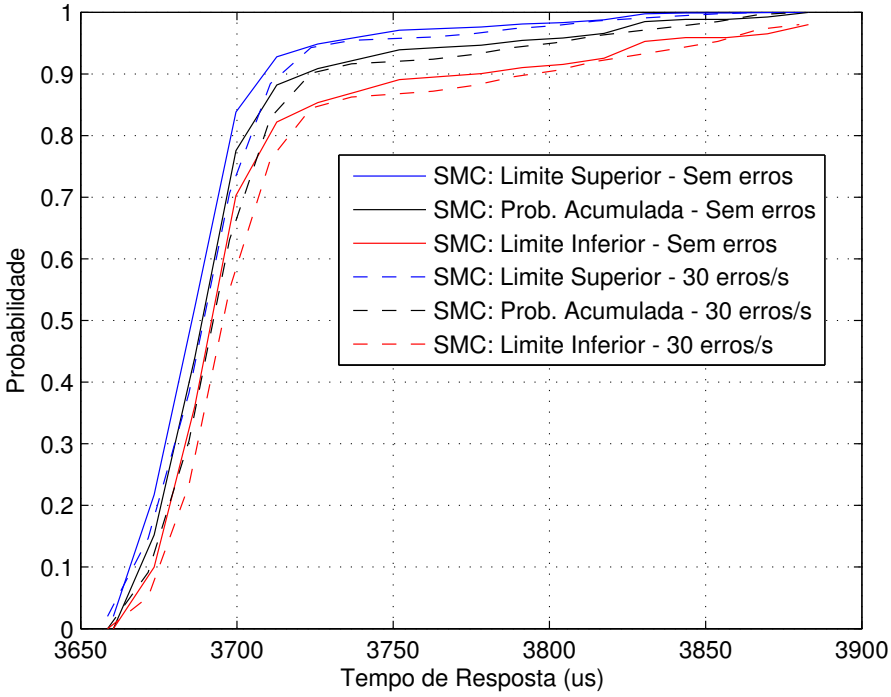


Figura 5.27: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 1 sem erros e com taxa de 30 erros/s no barramento.

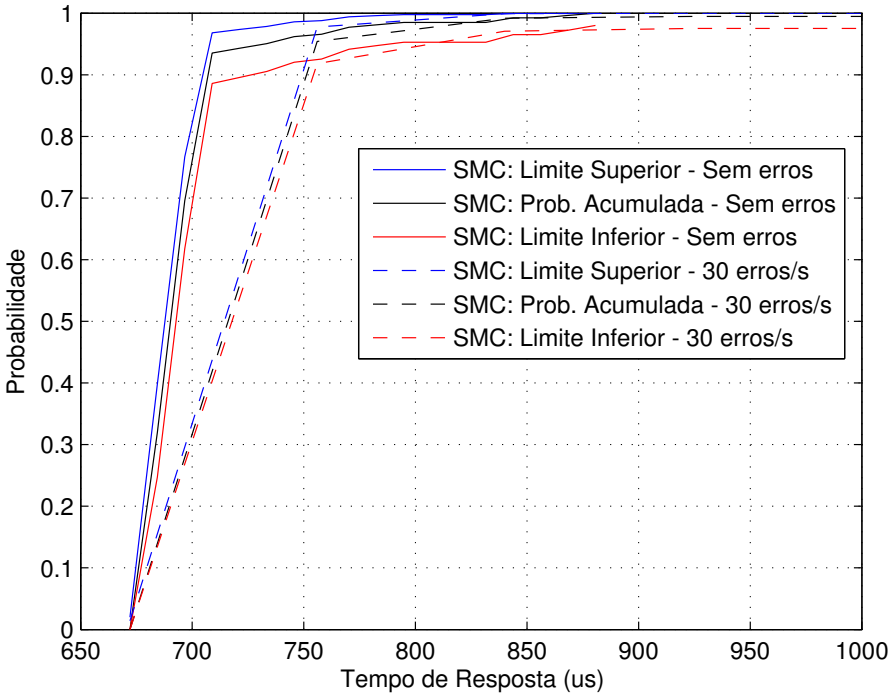


Figura 5.28: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 1 sem erros e com taxa de 30 erros/s no barramento.

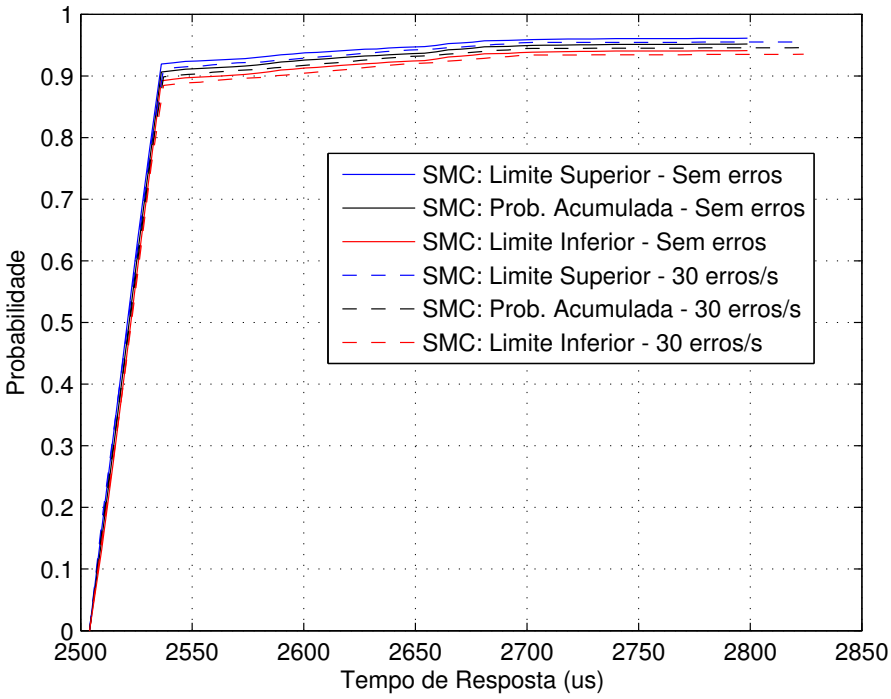


Figura 5.29: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 1 sem erros e com taxa de 30 erros/s no barramento.

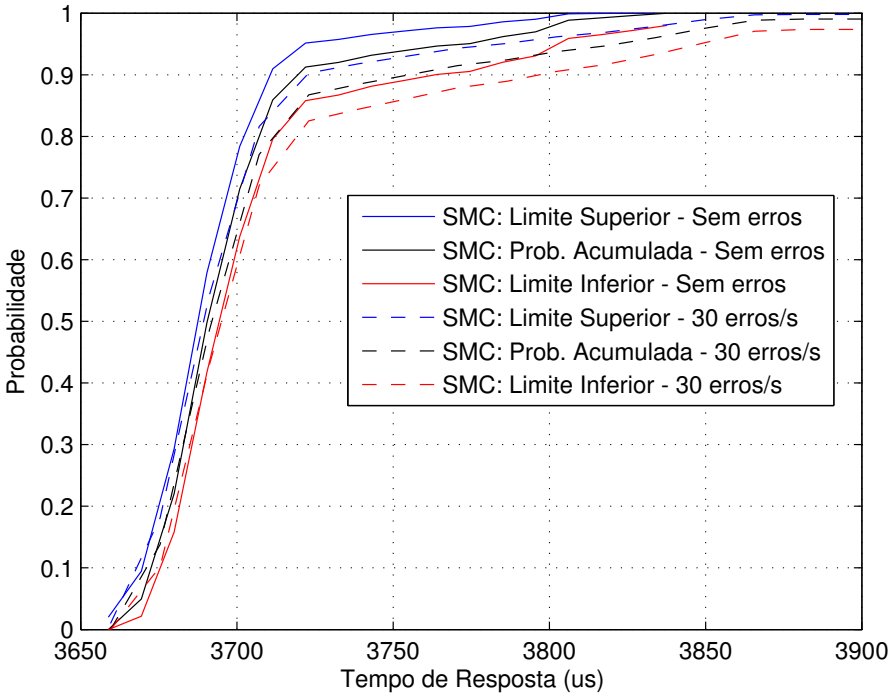


Figura 5.30: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 2 sem erros e com taxa de 30 erros/s no barramento.

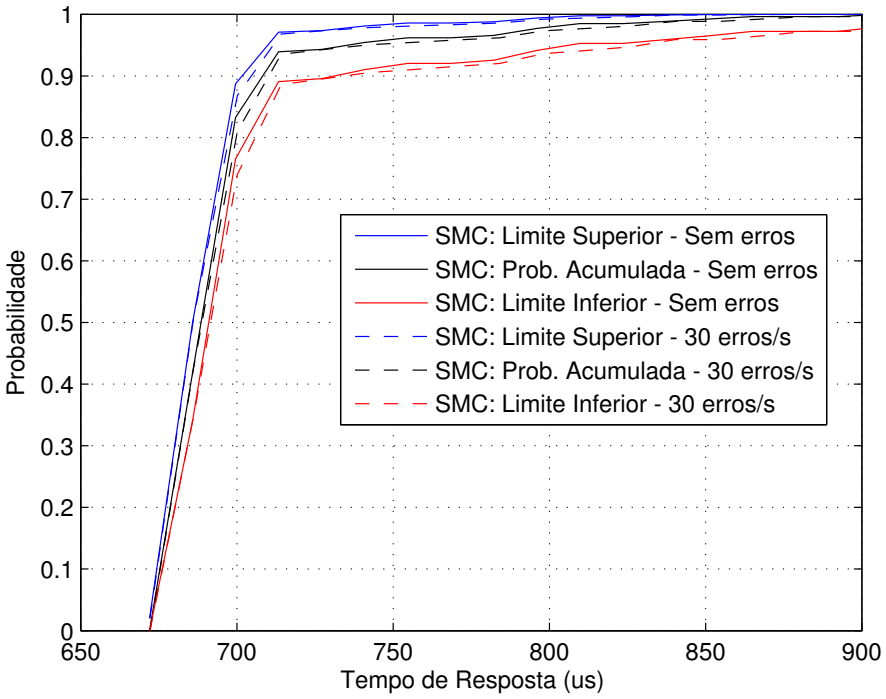


Figura 5.31: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 2 sem erros e com taxa de 30 erros/s no barramento.

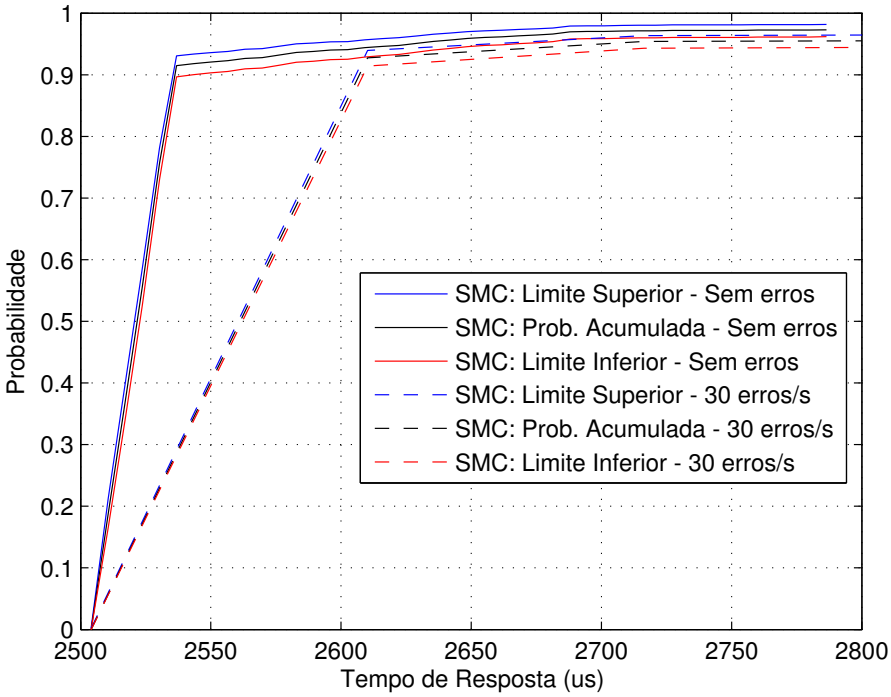


Figura 5.32: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 2 sem erros e com taxa de 30 erros/s no barramento.

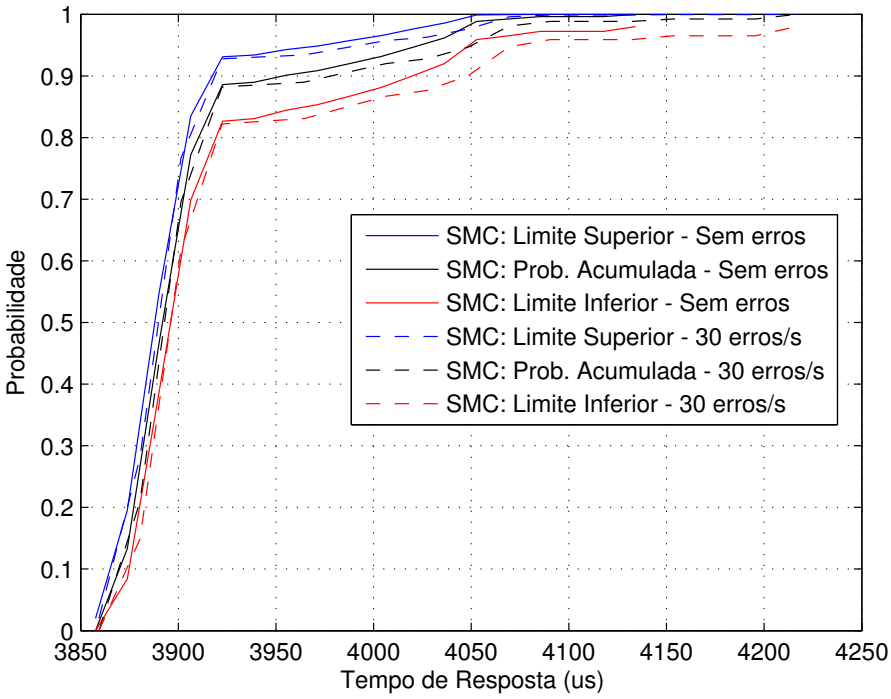


Figura 5.33: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 3 sem erros e com taxa de 30 erros/s no barramento.

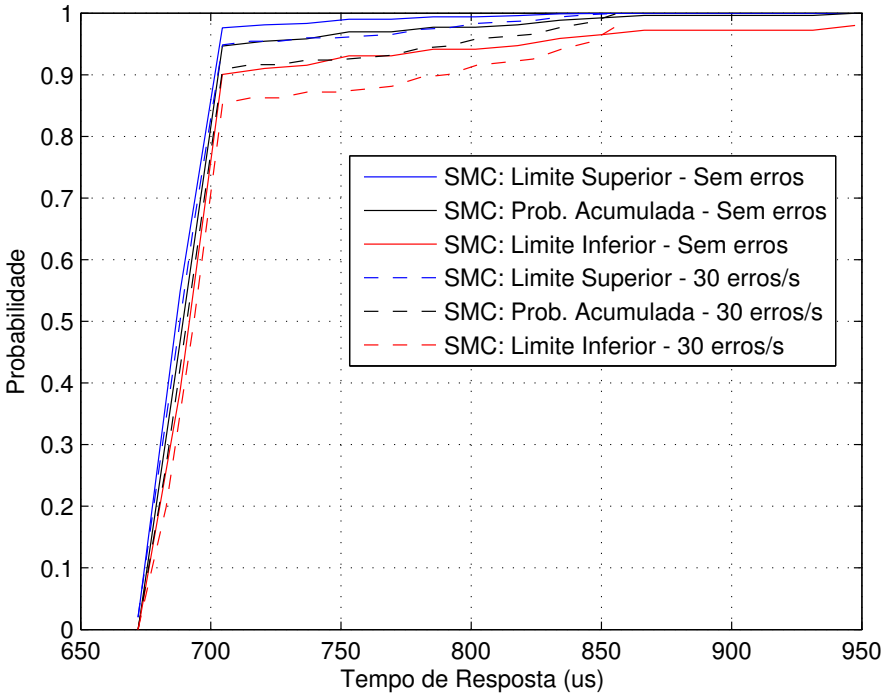


Figura 5.34: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 3 sem erros e com taxa de 30 erros/s no barramento.

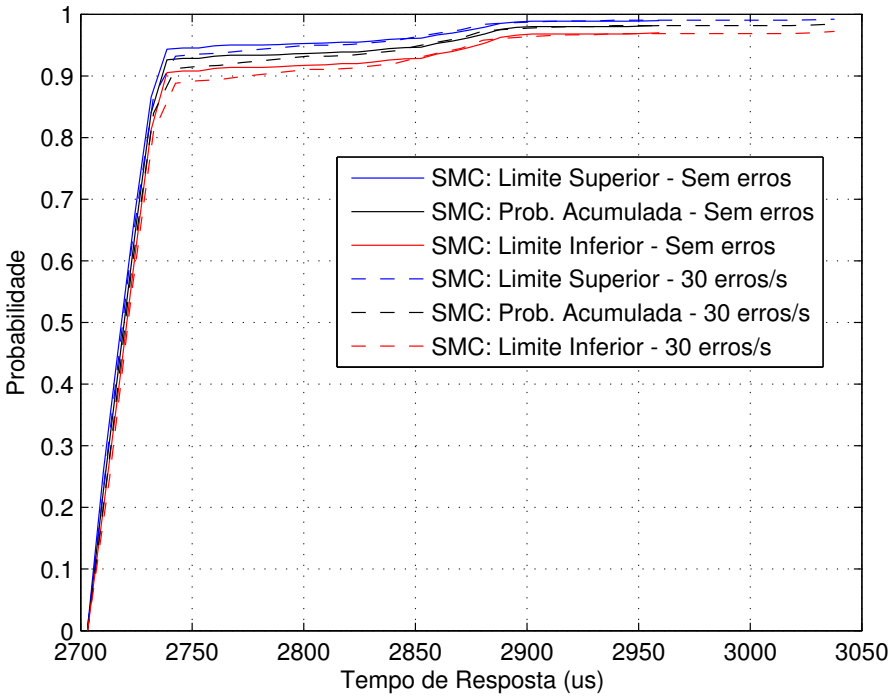


Figura 5.35: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 3 sem erros e com taxa de 30 erros/s no barramento.

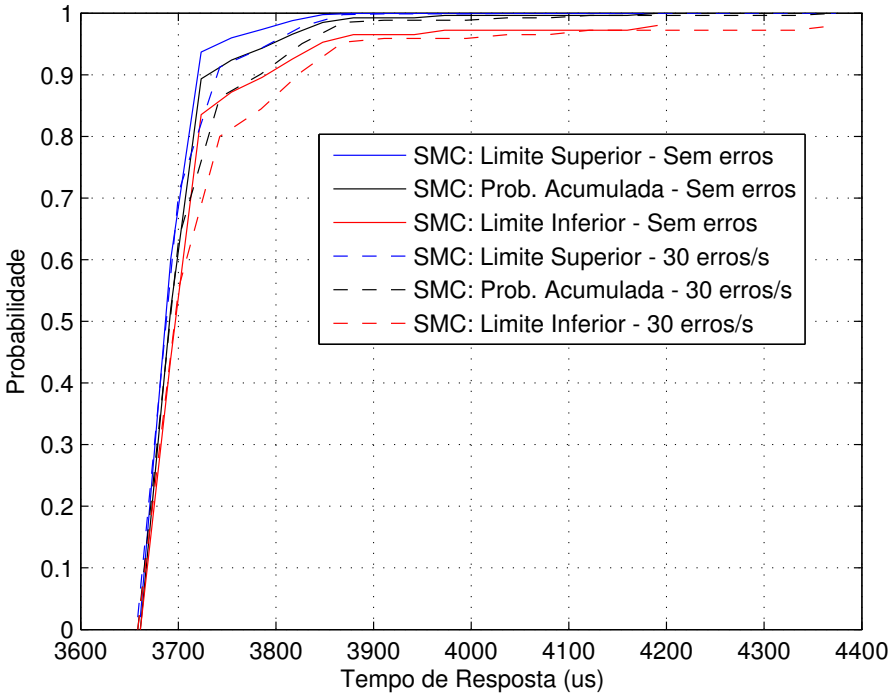


Figura 5.36: Gráfico da função de distribuição acumulada do tempo de resposta fim a fim da malha de controle 4 sem erros e com taxa de 30 erros/s no barramento.

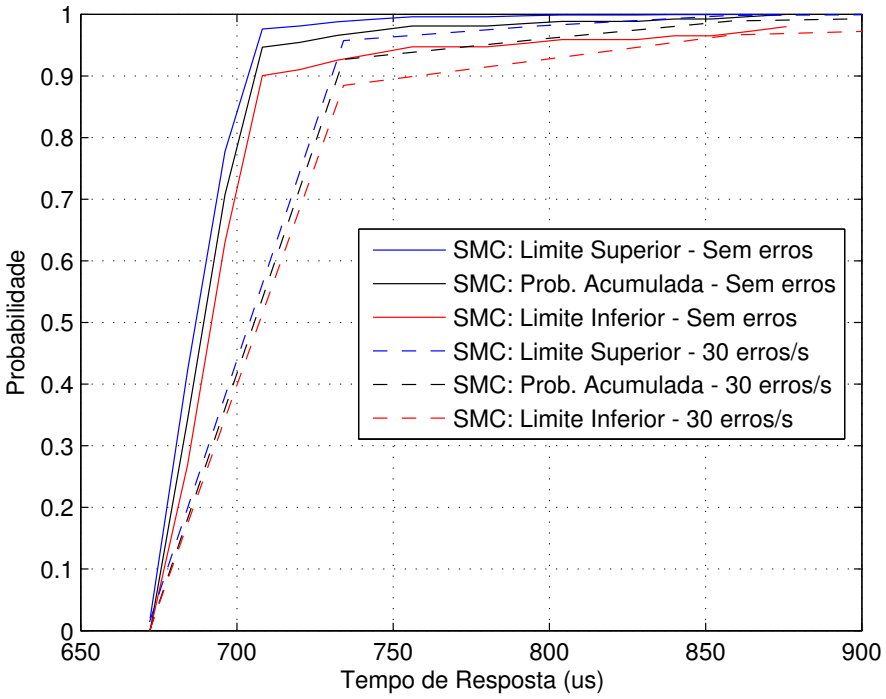


Figura 5.37: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem sensor-controlador da malha de controle 4 sem erros e com taxa de 30 erros/s no barramento.

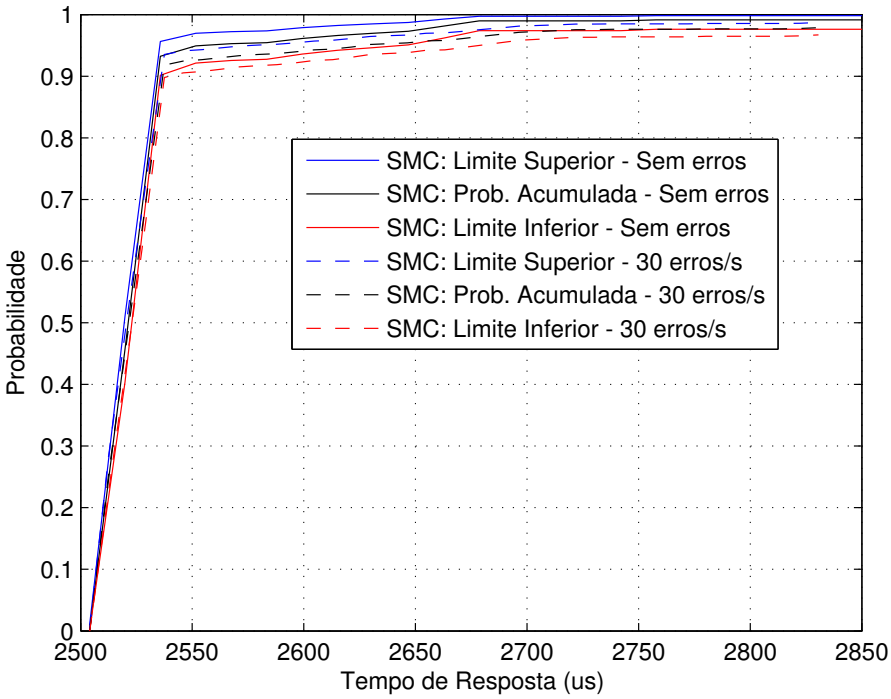


Figura 5.38: Gráfico da função de distribuição acumulada do tempo de resposta da mensagem controlador-atuador da malha de controle 4 sem erros e com taxa de 30 erros/s no barramento.

5.6 Conclusões

Neste Capítulo apresentou-se modelos em autômatos temporizados para três das quatro principais arquiteturas de NCSs CAN considerando offsets dinâmicos, relógios ideais e barramento sem (ideal) e com (real) erros. Assim foram gerados seis modelos: dois para a arquitetura SC-A, dois para a arquitetura S-CA e dois para a arquitetura S-C-A. Um estudo de caso com cinco malhas de controle foi realizado. A partir desses modelos, utilizou-se as técnicas de verificação de modelos simbólica e estatística para determinar o comportamento temporal das malhas de controle.

Inicialmente, considerou-se o caso com barramento sem erros e aplicou-se a verificação de modelos simbólica clássica. Dez propriedades foram verificadas para cada malha. A verificação simbólica conseguiu apenas avaliar as propriedades para um conjunto com até três malhas de controle. Então, os tempos de resposta fim a fim, os tempos de resposta da mensagem sensor-controlador e da mensagem controlador-atuador, ambas no melhor e pior caso, foram determinadas. Os resultados para os NCSs nas arquiteturas SC-A e S-CA foram comparados com os resultados obtidos via a análise de pior caso de Davis et al. (2007). Foi verificado que os tempos de resposta no pior caso são idênticos. Os resultados dos tempos de resposta fim a fim para o NCS na arquitetura S-C-A foram comparados com os resultados obtidos via a análise de Henderson, Kendall e Robson (2001). Como esperado, foi demonstrado analiticamente que os

tempos de resposta fim a fim no pior caso obtidos via a análise de Henderson, Kendall e Robson (2001) são pessimistas, ou seja, maiores do que acontecem na prática. Enquanto que os tempos de resposta fim a fim no melhor caso, usando ambas as técnicas, foram idênticos. Para mais de três malhas de controle, ocorreu o problema da explosão do espaço de estados. Então, para resolver esse problema, foi utilizado a verificação estatística, que determinou a probabilidade do deadline das cinco malhas serem satisfeitas. Além disso, verificou-se a probabilidade das mensagens satisfarem seus *deadlines* (utilizamos o *deadline* igual ao período da malha). A ferramenta retornou gráficos das funções de distribuição acumulada com 99% de confidência. A partir desses gráficos, verificou-se mais uma vez que o cenário de pior caso ocorre raramente. Foi possível determinar as variações nos atrasos fim a fim.

Por fim, considerou-se o barramento com erros. O mesmo estudo de caso com cinco malhas foi utilizado. Então, verificou-se apenas para a arquitetura S-C-A a probabilidade dos deadlines serem satisfeitos para uma taxa de transmissão de 500 Kbits/s e uma taxa de 30 erros/s no barramento. Os resultados foram comparados para o caso onde o barramento é livre de erros. Foi verificado que as curvas da função de distribuição acumulada para o caso com erros no barramento estão sempre abaixo das curvas da função de distribuição acumulada para o caso sem erros no barramento, na faixa de probabilidade de 0 a pelo menos 95%. Como esperado, isso significa que os erros no barramento tendem a aumentar os tempos de resposta das mensagens e, consequentemente, do tempo de resposta fim a fim. Então, a partir de 95% de probabilidade, as curvas tenderam a se tornarem iguais. Observou-se que esse comportamento ocorreu devido a dois fatores: (1) o baixo fator de utilização da rede (em torno de 11,61%), que resulta em um alto número de ocorrências próximas ao tempo de resposta no melhor caso e (2) a ocorrência rara dos cenários de pior caso.

Capítulo 6

Projeto e simulação em tempo real de NCSs CAN

Neste Capítulo apresenta-se um procedimento para projetar NCS CAN na arquitetura S-C-A usando os conceitos da margem de jitter e margem de fase aparente, e controladores PIDs. No procedimento de projeto utilizamos o modelo NCS CAN apresentado no Capítulo 5 para determinar os tempos de resposta fim a fim das malhas de controle considerando a ausência e a presença de erros no barramento. Em seguida, apresenta-se uma plataforma experimental para simulação em tempo real de NCSs CAN usando a técnica *hardware in the loop*. Um estudo de caso de um NCS CAN com cinco malhas de controle será apresentado e simulações serão realizadas usando a plataforma experimental.

6.1 Procedimento para projeto de NCSs CAN com atrasos variantes

Considera-se a seguintes suposições no procedimento para projetar NCSs CAN com atrasos variantes usando o conceito da margem de *jitter*:

- Os processos são lineares, estáveis, invariantes no tempo, com uma única entrada e uma única saída e são controlados por controladores PID;
- Os sensores são guiados a tempo, e os controladores e atuadores são guiados a eventos;
- Os tempos de computação das tarefas (aquisição, controle e atuação) e os tempos de transmissão das mensagens (sensor-controlador e controlador-atuador) são conhecidos.

O procedimento de projeto consiste de nove passos:

1. Defina a função de transferência de cada processo j no tempo contínuo;

2. Projete o controlador PID de cada processo j no tempo contínuo e defina as funções de transferência desses controladores;

3. Defina o período de amostragem (nominal) inicial h_j para cada controlador j usando a regra $\omega_{b_j} h_j \in [0{,}2,\ 0{,}6]$, onde ω_{b_j} é a largura de banda do sistema contínuo em malha fechada (ASTROM; WITTENMARK, 1997);

4. Discretize o controlador PID (método backward Euler[1]) usando os períodos de amostragem definidos no passo 3;

5. Determine os tempos de resposta fim a fim no pior caso ($R_{M_j}^{max}$) e melhor caso ($R_{M_j}^{min}$) para cada malha de controle j considerando uma determinada taxa de transmissão, taxa de erro no barramento e o período da malha de controle $h_{M_j} = h_j$;

   - Se em pelo menos uma malha de controle $R_{M_j}^{max} > h_{M_j}$, aumente a taxa de transmissão da rede e/ou reduza a taxa de erro no barramento, e refaça o passo 5. Caso essa condição ainda seja satisfeita, aumente o período de amostragem da respectiva malha de controle j até a condição não ser mais satisfeita;

6. Calcule a margem de *jitter* e a margem de fase aparente de cada malha de controle j considerando um atraso constante $L_j = R_{M_j}^{min}$ e o atraso variante (*jitter*) $J_j = R_{M_j}^{max} - R_{M_j}^{min}$ (definido no passo 3).

7. Para cada malha de controle j, calcule a degradação de desempenho relativa Fd_{M_j} (Equação 3.27) e o seu valor médio $F\bar{d}_{M_j}$;

8. Para cada malha de controle j, ajuste o período h_j de acordo com a equação $h_j = h_j + kh_j(Fd_{M_j} - F\bar{d}_{M_j})/F\bar{d}_{M_j}$, onde $k < 1$ é um parâmetro de ganho;

9. Repita a partir do passo 3 até que não seja mais possível obter alguma melhoria. Um possível critério de parada é quando a soma das diferenças dos desempenhos, $\sum |Fd_{M_j} - F\bar{d}_{M_j}|$ parar de decrescer.

Na seção seguinte apresenta-se a plataforma experimental proposta para simular e testar em tempo real NCSs CAN na arquitetura S-C-A[2]. A plataforma é baseada na técnica *hardware in the loop* (HIL). Essa técnica possibilita simular partes do sistema que são complexas de serem inseridas fisicamente nos testes, e interagir com partes reais do sistema, permitindo experimentá-lo em um ambiente virtual antes de testá-lo com todas as partes reais.

[1] Esse método garantirá que o sistema discreto é estável se o sistema contínuo for estável.

[2] As outras arquiteturas de NCSs CAN podem ser facilmente adaptadas.

6.2 Plataforma experimental para simulação de NCSs CAN

A plataforma proposta (Figura 6.1) consiste dos componentes de uma malha de controle (nó sensor, nó controlador e nó atuador), do simulador HIL e dos nós supervisor e injetor de falhas. Os nós são conectados via um barramento CAN. A comunicação entre os nós sensor e atuador de cada malha de controle e o simulador HIL é realizada por meio da interface serial RS232.

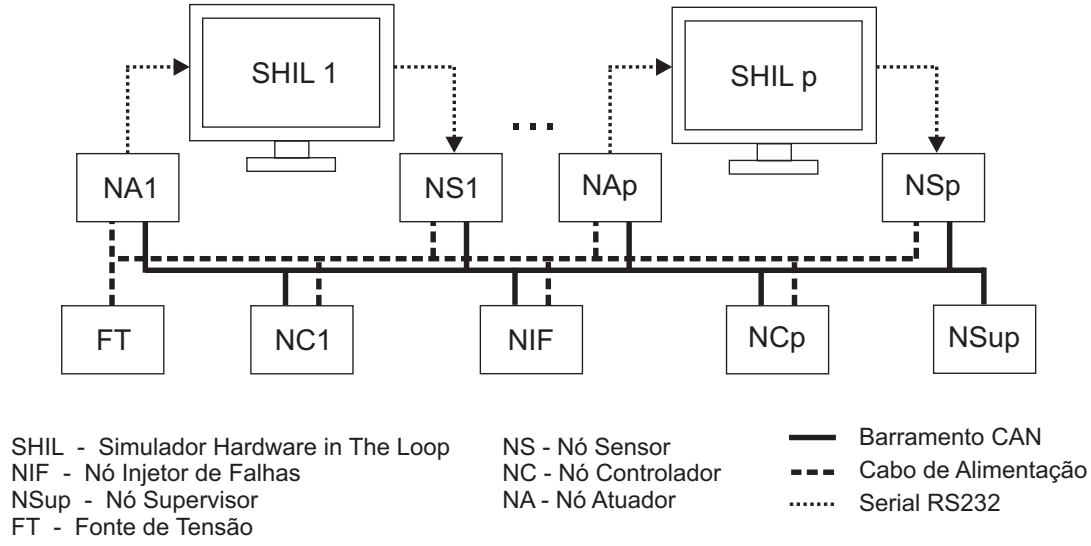


Figura 6.1: Representação da plataforma experimental para simulação e teste em tempo real de NCSs CAN na arquitetura S-C-A.

Os nós sensor, controlador e atuador são constituídos pelo microcontrolador PIC16F876A, controlador CAN MCP2515 e *transceiver* CAN MCP2551. O nó supervisor é formado pelo dispositivo CAN *leaf professional* conectado via USB a um computador pessoal. Esses tipos de nós são idênticos, respectivamente, aos nós CAN 0 a 7 e nó 8 apresentado na seção 4.4.1 do Capítulo 4 (ver Figura 4.21). O nó injetor de falhas é constituído pela placa Stellaris LM4F120 baseada em um microcontrolador ARM Cortex-M4F, e um *transceiver* CAN MCP2551. Esse nó é idêntico ao apresentado na seção 4.6.1 do Capítulo 4 (ver Figura 4.40). O software desses nós foi desenvolvido em linguagem C.

O simulador HIL é composto por um computador pessoal com sistema operacional em tempo real Linux/RTAI (*Real-Time Aplication Interface*). Esse sistema foi desenvolvido para simular em tempo real o processo físico da malha de controle, armazenar dados e plotar gráficos. O software do simulador HIL consiste de três programas. O primeiro programa foi desenvolvido usando as linguagens assembly e C, e é executado como módulo RTAI. Basicamente, esse programa realiza as seguinte atividades: simulação do processo físico usando o método numérico Runge Kutta de quarta ordem (BOYCE; DIPRIMA, 2010, p. 358); comunicação serial RS232 a um baudrate de 115200 com o nó sensor (envio do sinal de saída do processo) e atuador (recepção do sinal de controle) e armazenamento de dados (sinais de entrada e saída do processo,

tempo real da simulação e tempos de resposta fim a fim) na memória compartilhada. O segundo programa, desenvolvido em linguagem C e executado como programa Linux no espaço do usuário, tem a função de retirar os dados da memória compartilhada e salvá-los em arquivos de texto. O terceiro programa é um script Gnuplot que lê os dados nos arquivos de texto e plota os gráficos dinamicamente.

Na seção seguinte será utilizado o procedimento de projeto apresentado na seção 6.1 para projetar um NCS CAN com cinco malhas de controle. Ao longo da seção, será realizado algumas comparações dos resultados obtidos com a plataforma experimental, com os obtidos via Matlab.

6.3 Projeto de NCSs CAN

Considere um sistema de controle via rede CAN com cinco malhas de controle. O processo físico da malha 0 é um motor DC e o processo da malha 1 é um circuito RC com constante de tempo RC = 0,1 s. Os processos das malhas 2 e 4 são, respectivamente, sistemas de primeira ordem com atrasos iguais a 0,05 s e 0,2 s (FOLPD - *First Order Lag Plus Delay*). O processo da malha 3 é um sistema de segunda ordem com atraso igual a 0,05 s (SOSPD - *Second Order System Plus Delay*). As funções de transferência e as equações diferenciais desses processos[3] no tempo contínuo são representados pelas Equações 6.1 a 6.5, sendo: $u_i(t)$ e $y_i(t)$, respectivamente, os sinais de entrada e saída do processo i, e $x_1(t)$, $x_2(t)$ e $x_3(t)$ as variáveis de estado. As condições iniciais das equações diferenciais dos processos são nulas. Os ruídos de estado e de medição foram desconsiderados.

$$\left.\begin{aligned} P_0(s) &= \frac{2}{(s+9{,}997)(s+2{,}003)} \\ \dot{y_0}(t) &= 10y_0(t) + x_1 \\ \dot{x_1}(t) &= 0{,}02y_0(t) + 2x_1 + 2u_0(t) \end{aligned}\right\} \quad \text{Motor DC} \tag{6.1}$$

$$\left.\begin{aligned} P_1(s) &= \frac{10}{(s+10)} \\ \dot{y_1}(t) &= -10y_1(t) + 10u_1(t) \end{aligned}\right\} \quad \text{Circuito RC} \tag{6.2}$$

[3]Os processos foram definidos a partir da consideração de que a maioria dos processos industriais podem ser representados por modelos de primeira e segunda ordem, sem e com atrasos.

$$\left.\begin{aligned} P_2(s) &= \frac{10}{(s+1)}e^{-0{,}05s} \approx \frac{-10(s-40)}{(s+1)(s+40)} \\ y_2(t) &= -1{,}25x_1(t) + 6{,}25x_2(t) \\ \dot{x}_1(t) &= -41x_1(t) - 5x_2(t) + 8u_2(t) \\ \dot{x}_2(t) &= 8x_1(t) \end{aligned}\right\} \quad \text{FOLPD1} \tag{6.3}$$

$$\left.\begin{aligned} P_3(s) &= \frac{3}{2{,}85s^2 + 3{,}95s + 1}e^{-0{,}05s} \approx \frac{-0{,}075s + 3}{0{,}07125s^3 + 2{,}949s^2 + 3{,}975s + 1} \\ y_3(t) &= -0{,}0658x_2(t) + 2{,}632x_3(t) \\ \dot{x}_1(t) &= -41{,}39x_1(t) - 6{,}974x_2(t) - 1{,}754x_3(t) + 2u_3(t) \\ \dot{x}_2(t) &= 8x_1(t) \\ \dot{x}_3(t) &= x_2(t) \end{aligned}\right\} \quad \text{SOSPD} \tag{6.4}$$

$$\left.\begin{aligned} P_4(s) &= \frac{1}{(s+1)}e^{-0{,}2s} \approx \frac{-(s-10)}{(s+1)(s+10)} \\ y_4(t) &= -0{,}5x_1(t) + 1{,}25x_2(t) \\ \dot{x}_1(t) &= -11x_1(t) - 2{,}5x_2(t) + 2u_4(t) \\ \dot{x}_2(t) &= 4x_1(t) \end{aligned}\right\} \quad \text{FOLPD2} \tag{6.5}$$

As funções de transferência dos controladores PIDs no tempo contínuo são apresentados nas Equações 6.6 a 6.10. Todos os processos com atraso foram linearizados pela aproximação de Padé de primeira ordem $\left(e^{-\tau s} = \frac{-(\tau/2)s+1}{(\tau/2)s+1}\right)$. Os controladores PIDs dos processos motor DC, circuito RC e SOSPD foram sintonizados por tentativa e erro, com o intuito de obter o menor tempo de subida e acomodação, e o menor *overshoot*. Enquanto que os controladores PIDs dos processos FOLPD foram projetados de acordo com as regras de sintonia de Vitecková apresentadas em O'Dwyer (2006, pg. 44), que tem como especificação de desempenho *overshoot* igual a 0%.

$$C_0(s) = \frac{13(s+7{,}355)(s+4{,}183)}{s} \qquad K_p = 150,\ K_i = 400 \text{ e } K_d = 13 \tag{6.6}$$

$$C_1(s) = \frac{2(s+10)}{s} \qquad K_p = 2,\ K_i = 20 \text{ e } K_d = 0 \tag{6.7}$$

$$C_2(s) = \frac{0{,}68(s+1)}{s} \qquad K_p = 0{,}68,\ K_i = 0{,}68 \text{ e } K_d = 0 \tag{6.8}$$

$$C_3(s) = \frac{2{,}5(s^2+2{,}4s+1{,}6)}{s} \qquad K_p = 6,\ K_i = 4 \text{ e } K_d = 2{,}5 \tag{6.9}$$

$$C_4(s) = \frac{1{,}7(s+1)}{s} \qquad K_p = 1{,}7,\ K_i = 1{,}7 \text{ e } K_d = 0 \tag{6.10}$$

A resposta ao degrau unitário de cada um dos processos, simulados no tempo contínuo em um programa computacional e no sistema HIL (com período de amostragem T=100 μs) são apresentados nas Figuras 6.2 a 6.6. Observou-se que as curvas simuladas no tempo contínuo e no sistema HIL no tempo discreto são praticamente idênticas. Nos processos com atraso, a aproximação de Padé alterou os atrasos nos processos. Por exemplo, no processo FOLPD1, o atraso de 0,05 s foi reduzido, para 0,04 s. Devido o sinal de saída dos processos com atraso se tornar negativo com a aproximação de Padé, será necessário colocar na implementação do sistema em malha fechada um saturador no sinal de saída. A resposta ao degrau unitário para esses processos com o sinal de saída saturado é também apresentado nas Figuras 6.4 a 6.6. Os sinais de entrada e saída dos processos estão limitados, respectivamente, nas seguintes faixas: 0 a 12 V e 0 a 1,1985 rad/s (Motor DC); 0 a 5 V e 0 a 5 V (Circuito RC); 0 a 5 e 0 a 5 (FOLPD1); 0 a 12 e 0 a 36 (SOSPD) e 0 a 5 e 0 a 10 (FOLPD2). Não há unidades específicas de medida nos sinais de entrada e saída dos processos FOLPD1, FOLPD2 e SOSPD, pois representam processos físicos fictícios de primeira ou segunda ordem com atraso.

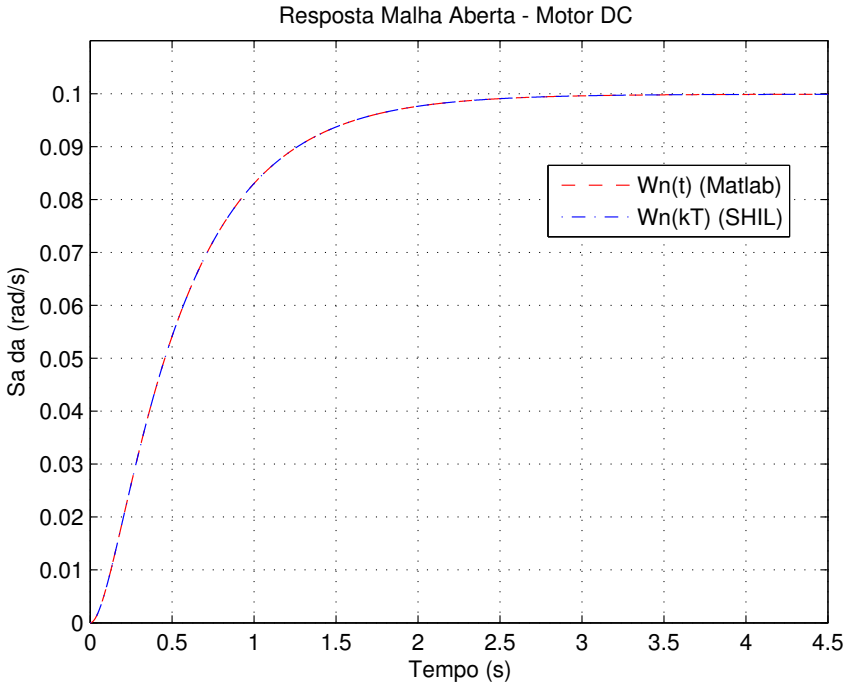


Figura 6.2: Gráfico da resposta ao degrau unitário em malha aberta do processo motor DC simulado no tempo contínuo e no SHIL (tempo discreto com período de 100 μs).

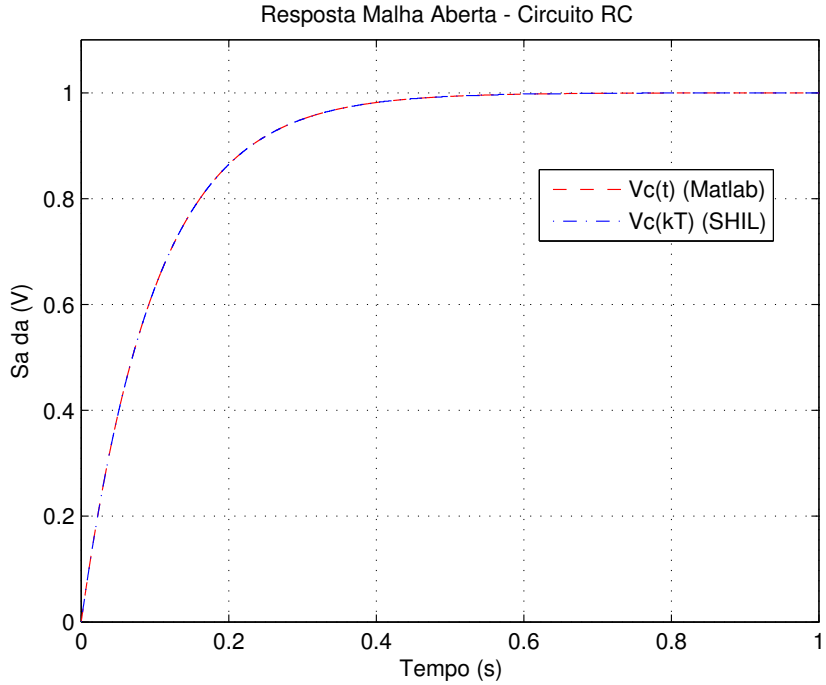


Figura 6.3: Gráfico da resposta ao degrau unitário em malha aberta do processo circuito RC simulado no tempo contínuo e no SHIL (tempo discreto com período de 100 μs).

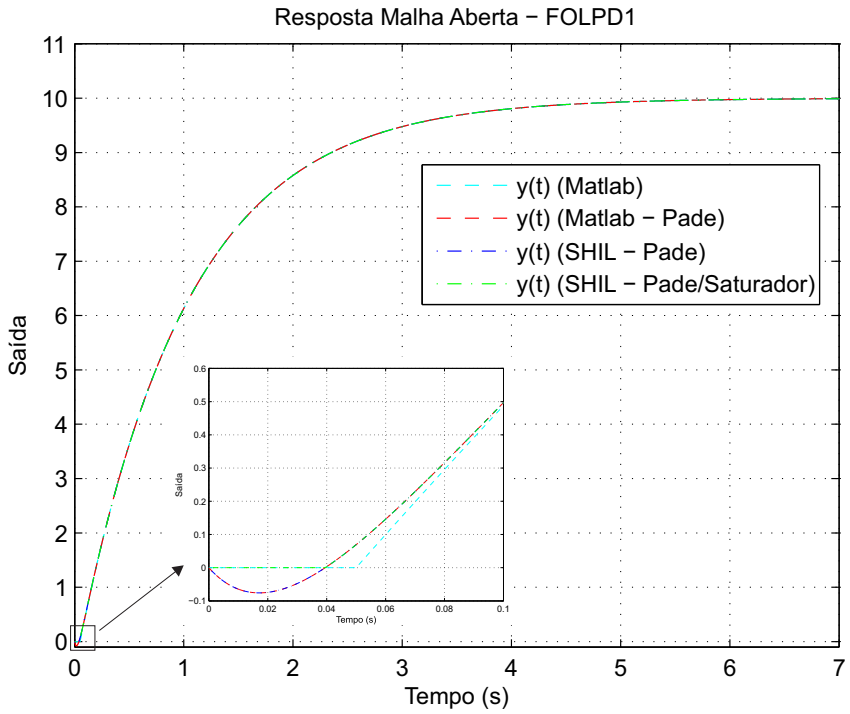


Figura 6.4: Gráfico da resposta ao degrau unitário em malha aberta do processo FOLPD1 simulado no tempo contínuo e no SHIL (tempo discreto com período de 100 μs).

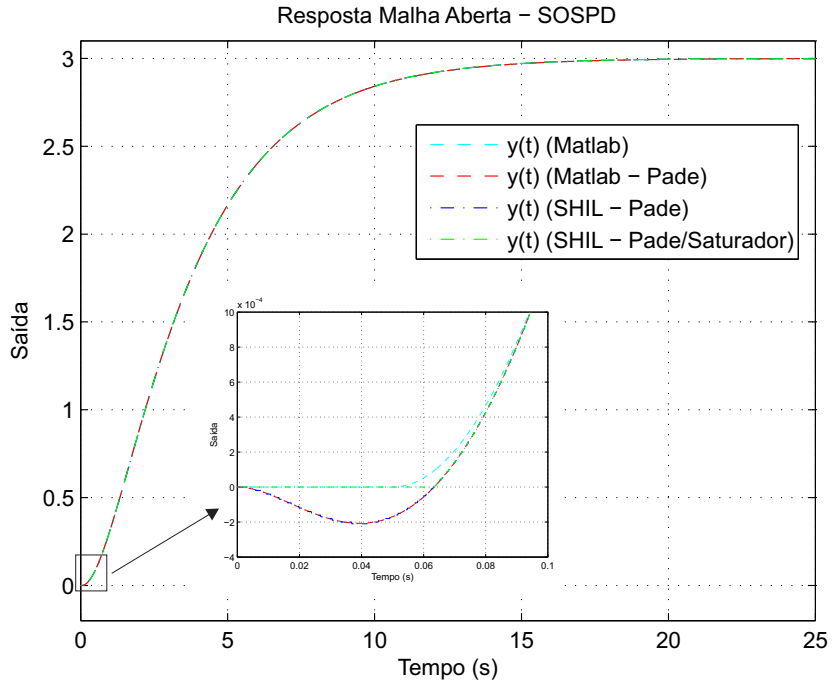


Figura 6.5: Gráfico da resposta ao degrau unitário em malha aberta do processo SOSPD simulado no tempo contínuo e no SHIL (tempo discreto com período de 100 μs).

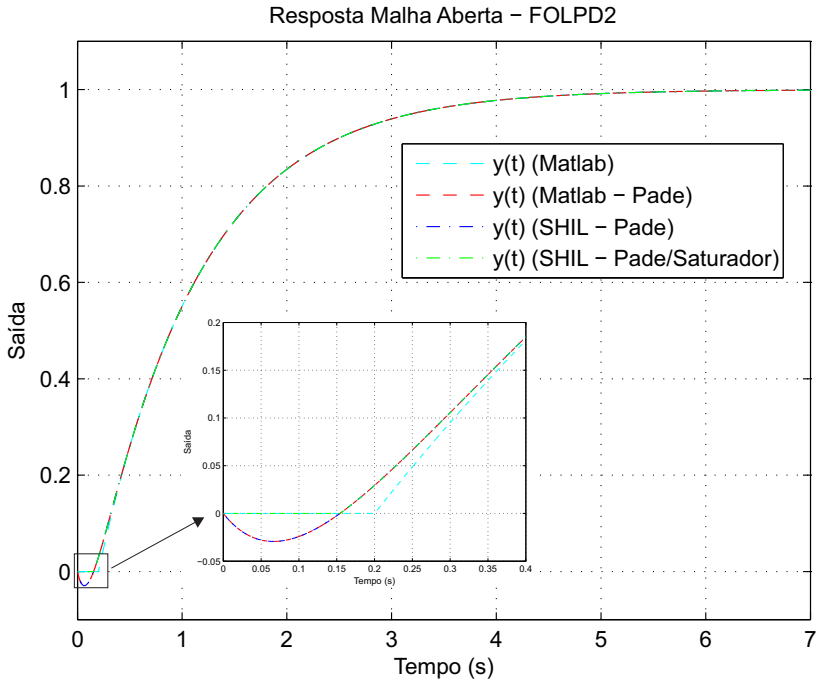


Figura 6.6: Gráfico da resposta ao degrau unitário em malha aberta do processo FOLPD2 simulado no tempo contínuo e no SHIL (tempo discreto com período de 100 μs).

A margem de fase (φ_m) e a respectiva frequência (ω_c) e a margem de atraso (L_m) dos modelos em malha aberta (processo + controlador), e a largura de banda (ω_b) do sistema em malha fechada são apresentadas na Tabela 6.1.

Tabela 6.1: Margem de fase e a respectiva frequência, e margem de atraso dos modelos (processo + controlador) em malha aberta, e a largura de banda do sistema em malha fechada.

Malha j	φ_m (graus)	ω_c (rad/s)	L_m (s)	ω_b (rad/s)
0	90,49	25,44	0,0621	25,16
1	90,00	20,00	0,0785	19,95
2	70,70	6,80	0,1815	11,16
3	62,44	2,85	0,3819	4,15
4	70,70	1,70	0,7259	2,79

Assumiu-se para a determinação dos tempos de resposta das malhas de controle, os atributos apresentados nas Tabelas 6.2 e 6.3, onde τ_{bit} é o tempo de transmissão de um bit. Para a determinação do período de amostragem h_{M_j} foi considerado a regra $\omega_{b_j} h_j \in [0{,}2,\ 0{,}6]$. Desse modo, foi obtido uma faixa de períodos de amostragem para cada malha. Os tempos de computação das tarefas e os tempos de transmissão das mensagens foram obtidos experimentalmente considerando o hardware da plataforma experimental apresentado na seção 6.1. Os conversores A/D e D/A são representados por variáveis no módulo RTAI executado no simulador HIL, cujos

valores são enviados pela interface digital serial RS232. Os valores dos sinais de entrada e saída foram limitados no intervalo de 0 a um limite superior (dependendo da malha), e normalizados na faixa de 0 a 9999. Então, 4 bytes são transmitidos via interface serial do simulador HIL para o nó sensor (sinal de saída) e do nó atuador para o simulador HIL (sinal de controle). Cada mensagem (sensor-controlador ou controlador-atuador) possui também 4 bytes no campo de dados contendo o sinal de saída do processo ou o sinal de controle. As variáveis dos controladores estão na representação ponto flutuante. A princípio, considerou-se que o *deadline* (D_{M_j}) é igual ao período (h_j)[4].

Tabela 6.2: Período, deadline e tempos de computação das tarefas das malhas em ms.

Malha j	h_j	D_{M_j}	C_{s_j}	C_{c_j}	C_{a_j}
0	[8,0, 23,0]	[8,0, 23,0]	0,520	2,551	0,481
1	[10,0, 30,0]	[10,0, 30,0]	0,520	2,352	0,481
2	[18,0, 53,0]	[18,0, 53,0]	0,520	2,352	0,481
3	[48,0, 144,0]	[48,0, 144,0]	0,520	2,551	0,481
4	[71,0, 214,0]	[71,0, 214,0]	0,520	2,352	0,481

Tabela 6.3: Tempos de transmissão das mensagens das malhas de controle.

Malha j	$C^{min}_{m_{sc_j}}/C^{max}_{m_{sc_j}}$	$C^{min}_{m_{ca_j}}/C^{max}_{m_{ca_j}}$
0	$76\tau_{bit}$/$92\tau_{bit}$	$76\tau_{bit}$/$92\tau_{bit}$
1	$76\tau_{bit}$/$92\tau_{bit}$	$76\tau_{bit}$/$92\tau_{bit}$
2	$76\tau_{bit}$/$92\tau_{bit}$	$76\tau_{bit}$/$92\tau_{bit}$
3	$76\tau_{bit}$/$92\tau_{bit}$	$76\tau_{bit}$/$92\tau_{bit}$
4	$76\tau_{bit}$/$92\tau_{bit}$	$76\tau_{bit}$/$92\tau_{bit}$

Os controladores PIDs apresentados na Equação 6.6 foram discretizados usando os períodos de amostragem (H_{min} e H_{max}) e o método backward Euler (Esse método garantirá que o sistema discreto é estável se o sistema contínuo for estável.). As funções de transferência no tempo discreto são apresentadas, em função do período de amostragem h, na Equação 6.11.

[4]O modelo desenvolvido em autômatos temporizados considera que o *deadline* deve ser menor ou igual ao período

$$C_0(z) = 150 + \frac{400hz}{z-1} + \frac{13(z-1)}{hz}$$

$$C_1(z) = 2 + \frac{20hz}{z-1}$$

$$C_2(z) = 0{,}68 + \frac{0{,}68hz}{z-1} \tag{6.11}$$

$$C_3(z) = 6 + \frac{4hz}{z-1} + \frac{2{,}5(z-1)}{hz}$$

$$C_4(z) = 1{,}7 + \frac{1{,}7hz}{z-1}$$

Os tempos de resposta fim a fim obtidos via verificação de modelos estatísticos, a partir dos modelos desenvolvidos no Capítulo 5, retornam probabilidades de satisfazer determinados tempos de resposta. Então, foi considerado como tempo de resposta fim a fim no pior caso de cada malha, o tempo cuja probabilidade foi igual a pelo menos 95%. Esses tempos de resposta são apresentados na Tabela 6.4, para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8$ μs) e 500 kbits/s ($\tau_{bit} = 2$ μs), sem erros no barramento (caso ideal) e com uma taxa de 30 erros/s.

Tabela 6.4: Tempos de resposta fim a fim no pior caso das malhas de controle ($R_{M_j}^{max}$) em μs para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8$ μs) e 500 kbits/s ($\tau_{bit} = 2$ μs), considerando os períodos mínimo (H_{min}) e máximo (H_{max}), sem erros no barramento (s/ erro) e com uma taxa de 30 erros/s (c/ erro).

Malha j	$R_{M_j}^{max}$ (μs)							
	125 Kbits/s				500 Kbits/s			
	H_{min}		H_{max}		H_{min}		H_{max}	
	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro
0	5946	7756	5962	7863	4088	4168	4065	4153
1	5601	7860	5574	7907	3882	3880	3836	3888
2	6105	8540	6083	8522	3837	4024	3851	4055
3	6480	8988	6431	8696	4134	4216	4170	4214
4	6232	11010	6203	10825	4190	4373	4143	4369

Observou-se para a taxa de transmissão de 500 kbits/s que os tempos de resposta fim a fim das malhas de controle com os períodos H_{min} e H_{max} e mesma taxa de erro, são quase iguais (compare $R_{M_j}^{max}$ de H_{min} sem erro com H_{max} sem erro, e $R_{M_j}^{max}$ de H_{min} com erro com H_{max} com erro). Isso ocorreu devido ao baixo fator de utilização da rede, que é no caso ideal (sem

erros) igual a 3,95% (para H_{max} e 500 kbits/s) e 11,61% (para H_{min} e 500 kbits/s)[5]. Nessa taxa de transmissão e sem erros, a maior diferença entre os tempos de resposta foi igual a 4190 - 4143 = 47 μs (malha de controle 4). Considerando a taxa de erros, a maior diferença foi igual a 4055 - 4024 = 31 μs (malha de controle 2). Observou-se também que o impacto dos erros no barramento nos tempos de resposta foi baixo. Por exemplo, para a malha de controle 4, cujas mensagens são de menor prioridade, e H_{max}, o tempo de resposta aumentou 226 μs com a presença de erros no barramento.

Os tempos de resposta para a taxa de transmissão de 125 kbits/s, comparando H_{min} e H_{max}, foram também semelhantes. Entretanto, a variação foi um pouco maior. Para o caso sem erro, a maior diferença entre os tempos de resposta foi igual a 49 μs (malha de controle 3). Para o caso com erro, a maior diferença nos tempos de resposta foi igual a 292 μs (malha de controle 3). O impacto dos erros no barramento nos tempos de resposta foi considerável. Para a malha de controle 4, cujas mensagens são de menor prioridade, a diferença entre os tempos de resposta foi maior do que 4000 μs. Esses resultados são justificados pelo aumento do fator de utilização, que é igual a 15,79% (para H_{max} e 125 kbits/s) e 46,44% (para H_{min} e 125 kbits/s).

Os tempos de resposta fim a fim no melhor caso são apresentados na Tabela 6.5. Considerou-se o melhor caso dos tempos de computação das tarefas e dos tempos de transmissão das mensagens.

Tabela 6.5: Tempos de resposta fim a fim no melhor caso das malhas de controle ($R_{M_j}^{min}$) em μs para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8\ \mu$s) e 500 kbits/s ($\tau_{bit} = 2\ \mu$s), considerando os períodos mínimo (H_{min}) e máximo (H_{max}), sem erros no barramento (s/ erro) e com uma taxa de 30 erros/s (c/ erro).

Malha j	$R_{M_j}^{min}$ (μs)							
	125 Kbits/s				500 Kbits/s			
	H_{min}		H_{max}		H_{min}		H_{max}	
	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro
0	4768	4768	4768	4768	3856	3856	3856	3856
1	4569	4569	4569	4569	3657	3657	3657	3657
2	4569	4569	4569	4569	3657	3657	3657	3657
3	4768	4768	4768	4768	3856	3856	3856	3856
4	4569	4569	4569	4569	3657	3657	3657	3657

O jitter ($J_j = R_{M_j}^{max} - R_{M_j}^{min}$) de cada malha de controle são apresentados na Tabela 6.6.

Considerando os modelos do processo (tempo contínuo) e controladores PIDs discretos, e

[5]Considerou-se no cálculo do fator de utilização que não há relações de precedência entre as mensagens. Isso torna o fator de utilização maior. Por outro lado, não foi considerado a taxa de erros, a qual aumenta o fator de utilização da rede.

Tabela 6.6: Jitter das malhas de controle (J_j) em ms para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8$ μs) e 500 kbits/s ($\tau_{bit} = 2$ μs), considerando os períodos mínimo (H_{min}) e máximo (H_{max}), sem erros no barramento (s/ erro) e com uma taxa de 30 erros/s (c/ erro).

Malha j	J_j (ms)							
	125 Kbits/s				500 Kbits/s			
	H_{min}		H_{max}		H_{min}		H_{max}	
	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro
0	1,178	2,988	1,194	3,095	0,232	0,312	0,209	0,297
1	1,032	3,291	1,005	3,338	0,225	0,223	0,179	0,231
2	1,536	3,971	1,514	3,953	0,180	0,367	0,194	0,398
3	1,712	4,220	1,663	3,928	0,278	0,360	0,314	0,358
4	1,663	6,441	1,634	6,256	0,533	0,716	0,486	0,712

os tempos de resposta fim a fim no melhor e pior caso apresentados nas Tabelas 6.4 e 6.5, foram obtidos as margens de jitter (Tabela 6.7) e as margens de fase aparente (Tabela 6.8) para cada malha de controle, respectivamente, de acordo com as definições 4 e 6 apresentadas no Capítulo 3. Foi observado, comparando J_j (Tabela 6.6) e J_{m_j} (Tabela 6.7) que, para todas as malhas de controle, a condição $J_j < J_{m_j}$ é satisfeita. Isso significa que, de acordo com os parâmetros do sistema, todas as malhas de controle são estáveis.

De acordo com o esperado, a margem de fase aparente, para ambas as taxas de transmissão, reduziu com o aumento do período de amostragem. Na presença de erros no barramento, observou-se que para a taxa de transmissão de 500 kbits/s, a margem de fase aparente não foi afetada, exceto para a malha de controle 0, onde houve uma pequena redução, de 42,03 graus para 40,76 graus. Para a taxa de transmissão de 125 kbits/s, a presença de erros no barramento reduziu de modo considerável a margem de fase aparente, especialmente para H_{max}, onde a margem de fase aparente foi reduzida para todas as malhas. Esses resultados podem também ser observados pela degradação de desempenho relativo apresentados na Tabela 6.9. A malha de controle 0, cujo processo é um motor DC, foi a malha que obteve o menor desempenho, embora seja a malha com as mensagens de maior prioridade. Isso é justificado pelo fato de o processo ter uma dinâmica mais rápida que os demais processos.

Tabela 6.7: Margem de jitter das malhas de controle (J_{m_j}) em ms para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8\ \mu$s) e 500 kbits/s ($\tau_{bit} = 2\ \mu$s), considerando os períodos mínimo (H_{min}) e máximo (H_{max}), sem erros no barramento (s/ erro) e com uma taxa de 30 erros/s (c/ erro).

Malha j	J_{m_j} (ms)							
	125 Kbits/s				500 Kbits/s			
	H_{min}		H_{max}		H_{min}		H_{max}	
	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro
0	28,1	28,1	13,5	13,5	28,8	28,8	14,6	14,6
1	40,5	40,5	30,6	30,6	41,1	41,1	30,6	30,6
2	99,6	99,6	82,9	82,9	100,3	100,3	83,7	83,7
3	266,4	266,4	162,2	162,2	267,2	267,2	162,8	162,8
4	396,3	396,3	313,5	313,5	397,0	397,0	314,2	314,2

Tabela 6.8: Margem de fase aparente das malhas de controle ($\hat{\varphi}_{m_j}$) em graus para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8\ \mu$s) e 500 kbits/s ($\tau_{bit} = 2\ \mu$s), considerando os períodos mínimo (H_{min}) e máximo (H_{max}), sem erros no barramento (s/ erro) e com uma taxa de 30 erros/s (c/ erro).

Malha j	$\hat{\varphi}_{m_j}$ (graus)							
	125 Kbits/s				500 Kbits/s			
	H_{min}		H_{max}		H_{min}		H_{max}	
	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro
0	63,09	57,87	31,86	23,04	73,23	73,23	42,03	40,76
1	79,61	66,82	59,44	48,79	80,70	80,70	68,04	68,04
2	65,32	65,32	50,71	47,08	65,68	65,68	58,72	58,72
3	54,93	54,93	39,52	33,65	55,09	55,09	39,69	39,69
4	67,11	67,11	60,83	52,11	67,21	67,21	60,93	60,93

Tabela 6.9: Degradação de desempenho relativo das malhas de controle (Fd_{M_j}) para as taxas de transmissão de 125 kbits/s ($\tau_{bit} = 8\ \mu$s) e 500 kbits/s ($\tau_{bit} = 2\ \mu$s), considerando os períodos mínimo (H_{min}) e máximo (H_{max}), sem erros no barramento (s/ erro) e com uma taxa de 30 erros/s (c/ erro).

Malha j	Fd_{M_j}							
	125 Kbits/s				500 Kbits/s			
	H_{min}		H_{max}		H_{min}		H_{max}	
	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro	S/ Erro	C/ Erro
0	0,697	0,639	0,352	0,254	0,809	0,809	0,464	0,450
1	0,884	0,742	0,660	0,542	0,896	0,896	0,756	0,756
2	0,924	0,924	0,717	0,666	0,929	0,929	0,830	0,830
3	0,879	0,879	0,633	0,539	0,882	0,882	0,635	0,635
4	0,949	0,949	0,860	0,737	0,950	0,950	0,862	0,862

6.4 Simulação em tempo real de NCSs CAN

Nesta seção apresenta-se os resultados da simulação do NCS CAN projetado na seção anterior usando a plataforma experimental descrita na seção 6.1. Uma foto dessa plataforma é apresentada na Figura 6.7.

Figura 6.7: Foto da plataforma experimental para simulação de NCSs CAN.

Foram realizadas duas simulações com as cinco malhas de controle: (1) NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e (2) NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento. Para cada malha foi determinado a resposta ao degrau unitário

e os resultados foram comparados com a resposta ao degrau para o sistema no tempo contínuo. Os gráficos com o sinal de saída e sinal de controle de cada malha são apresentados nas Figuras 6.8 a 6.17. Não há unidades específicas de medida nos sinais de saída dos processos FOLPD1, FOLPD2 e SOSPD, pois representam processos físicos fictícios de primeira ou segunda ordem com atraso.

Na malha de controle 0, cujo processo é um motor DC, pode-se observar que a resposta ao degrau unitário (Fig. 6.8) nas duas simulações do NCS CAN são semelhantes. Nota-se uma degradação do desempenho ocasionada pelas mudanças dos parâmetros da rede, ou seja, para o caso onde a taxa de transmissão diminuiu (de 500 para 125 kbits/s) e a taxa de erros aumentou (de 0 para 30 erros/s). Para reduzir a complexidade da simulação, especificamente na transmissão dos dados via serial, os sinais de controle foram limitados em 12 V. Isso explica o sinal de controle saturado no regime transitório. As curvas dos sinais de controle são apresentados na Figura 6.9. Pode-se observar que o sinal de controle no tempo contínuo é saturado em 12 V (durante o transitório) e, em seguida, atinge abruptamente o valor de aproximadamente 10 V, no regime permanente, de acordo com as curvas dos sinais de controle para as duas simulações do NCS CAN.

Na malha de controle 1 (circuito RC), observou-se na simulação (2) que houve um atraso na resposta ao degrau unitário (ver curva vermelha na Fig. 6.10) ocasionado pelo aumento da variação do atraso de controle, provavelmente ocasionado por erros no barramento. Comparando as curvas das duas simulações, pode-se notar uma degradação do desempenho ocasionado pela redução da taxa de transmissão e aumento da taxa de erros no barramento. Observou-se também que o sinal de controle para o caso da simulação 2 (ver Figura 6.11), onde detectou-se uma maior degradação de desempenho, atingiu durante o transitório um valor duas vezes maior que o sinal de controle obtido a partir da simulação 1.

Na malha de controle 2 e 4, onde os processos são sistemas de primeira ordem com atraso, pode-se notar que a redução da taxa de transmissão e aumento da taxa de erro gerou um aumento no atraso do sistema e no *overshoot* da resposta ao degrau unitário (ver curva vermelha nas Figs. 6.12 e 6.16, respectivamente, para as malhas 2 e 4). Observou-se, a partir das Figuras 6.13 e 6.17 que os sinais de controle foram semelhantes nas três simulações. Mas, nota-se um maior gasto de energia para as simulações no NCS CAN.

Foi também verificado na malha de controle 3, cujo processo é um sistema de segunda ordem com atraso, resultados semelhantes aos obtidos nas malhas 2 e 4. Observou-se que houve um aumento no atraso e *overshoot* da resposta ao degrau unitário na simulação (2). Entretanto, o tempo de acomodação em ambas as simulações são semelhantes. Notou-se, a partir das curvas do sinal de controle (Figura 6.15), que a rede provocou uma maior gasto de energia na malha 4 durante o regime transitório.

Em geral, foi observado em todas as malhas de controle, conforme os resultados apresenta-

dos na Tabela 6.9, que houve uma degradação de desempenho na simulação (2), ou seja, quando a taxa de transmissão foi reduzida de 500 para 125 kbits/s, na presença de erros no barramento e com aumento do período de amostragem da malha.

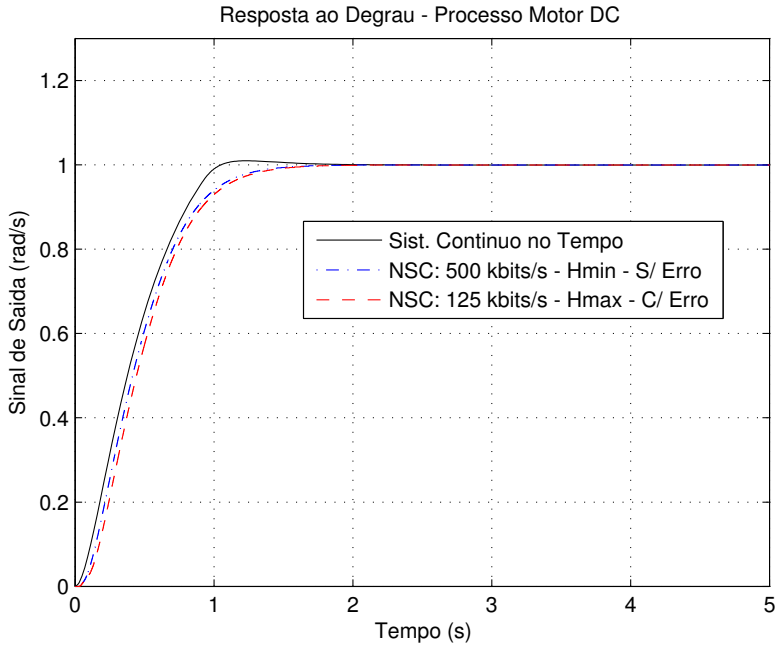


Figura 6.8: Sinal de saída da resposta ao degrau unitário da malha de controle 0 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

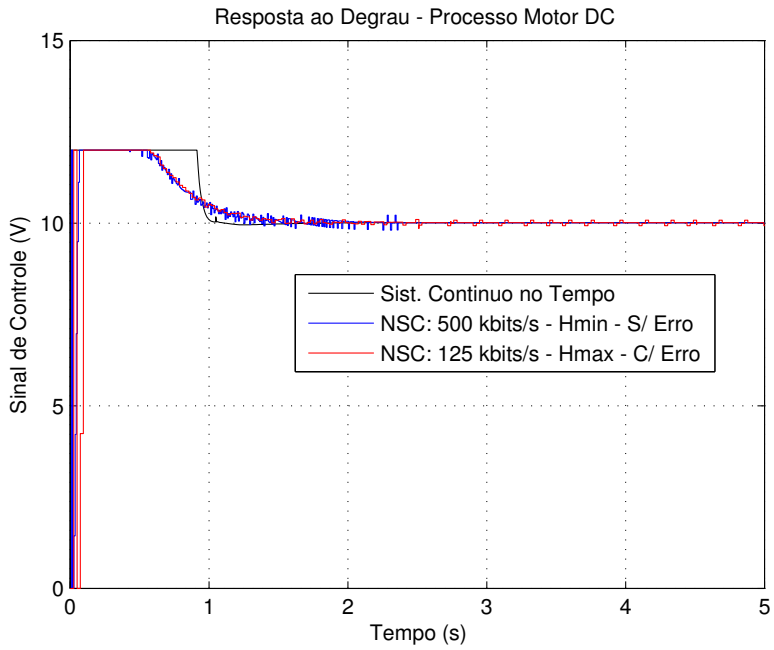


Figura 6.9: Sinal de controle da resposta ao degrau unitário da malha de controle 0 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

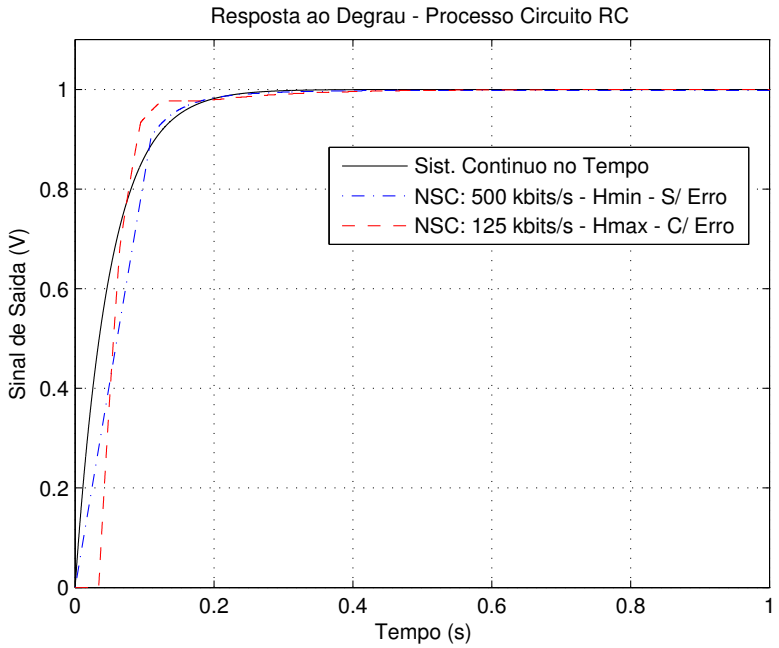


Figura 6.10: Sinal de saída da resposta ao degrau unitário da malha de controle 1 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

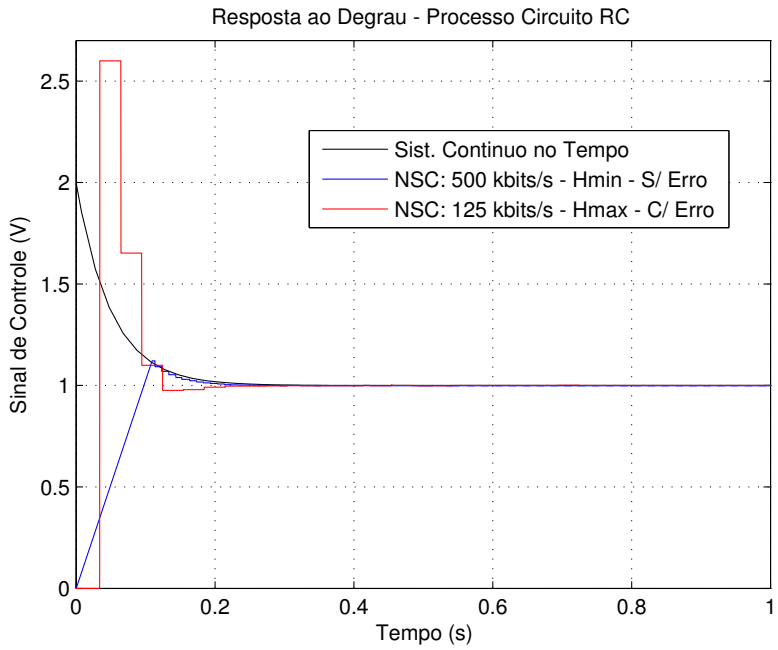


Figura 6.11: Sinal de controle da resposta ao degrau unitário da malha de controle 1 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

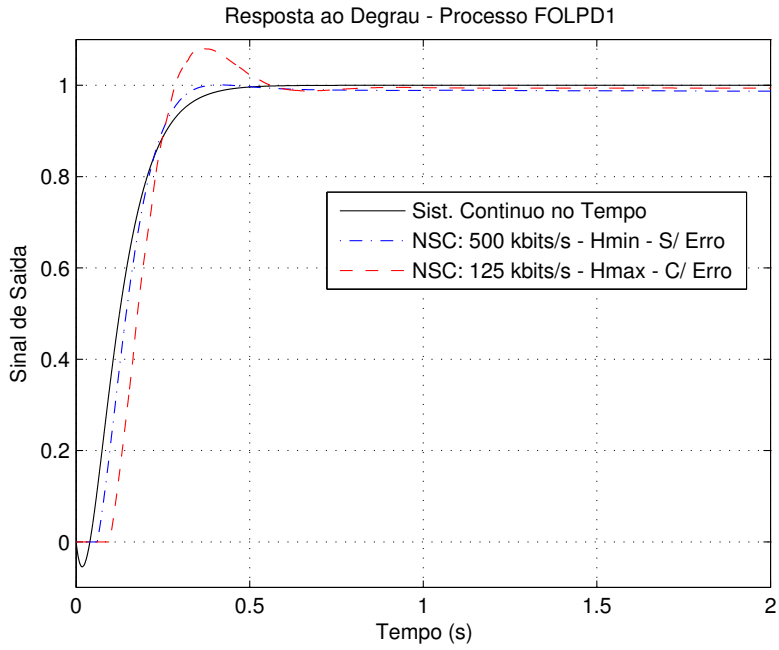


Figura 6.12: Sinal de saída da resposta ao degrau unitário da malha de controle 2 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

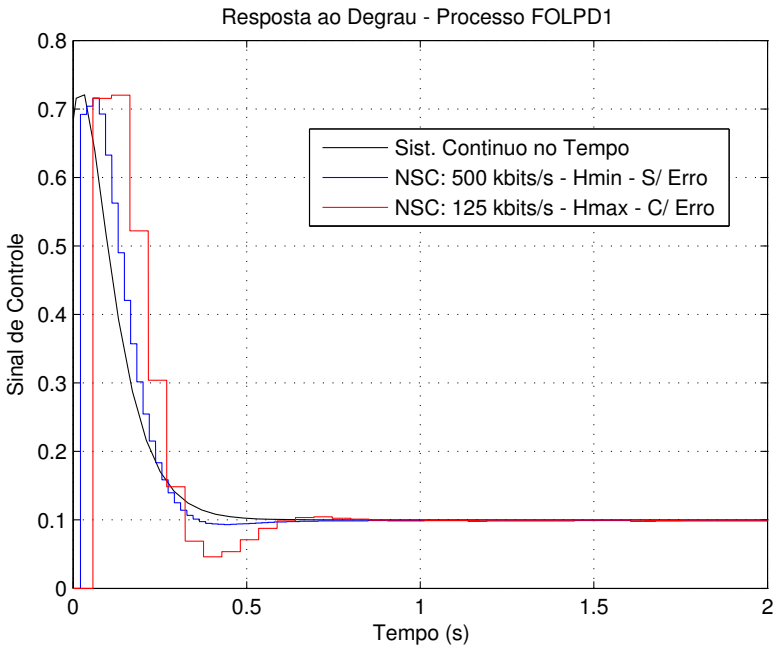


Figura 6.13: Sinal de controle da resposta ao degrau unitário da malha de controle 2 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

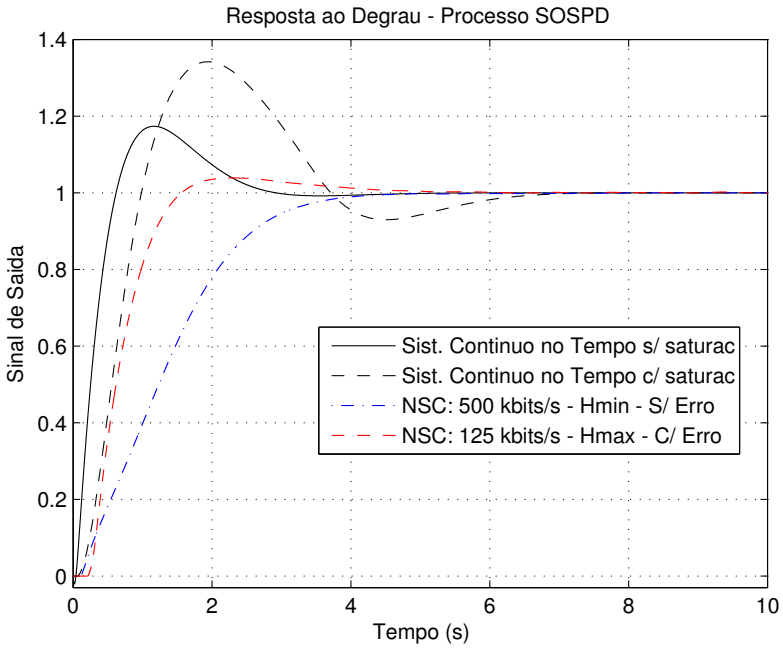


Figura 6.14: Sinal de saída da resposta ao degrau unitário da malha de controle 3 no tempo contínuo (sem e com saturação), em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

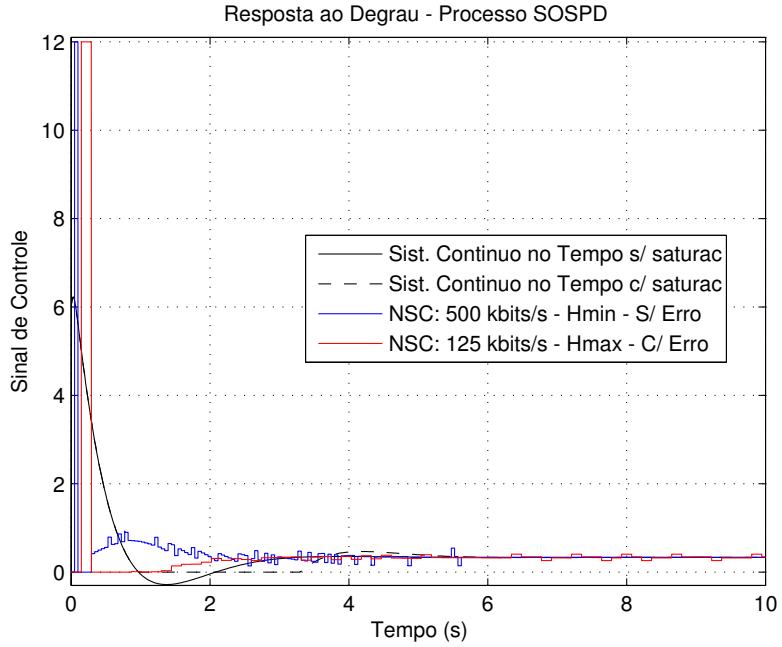


Figura 6.15: Sinal de controle da resposta ao degrau unitário da malha de controle 3 no tempo contínuo (sem e com saturação), em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

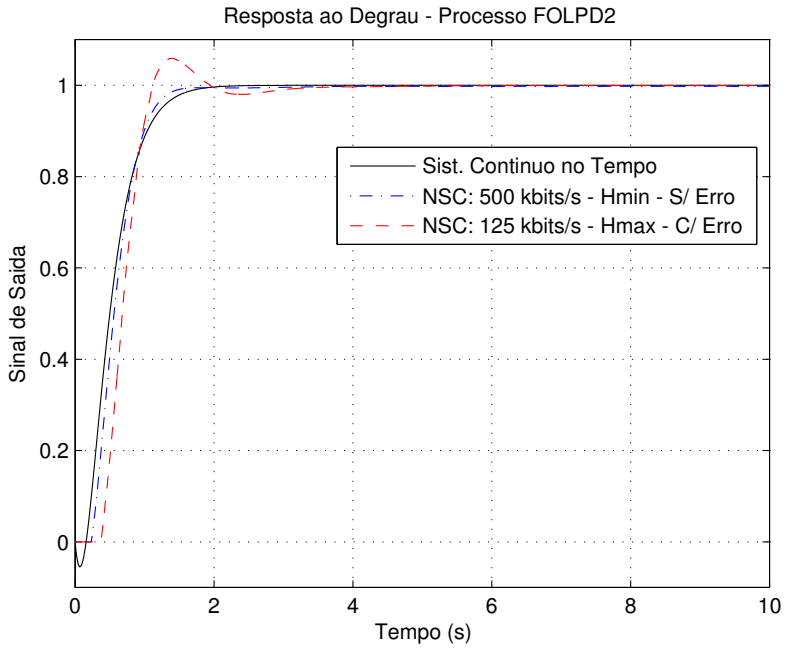


Figura 6.16: Sinal de saída da resposta ao degrau unitário da malha de controle 4 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

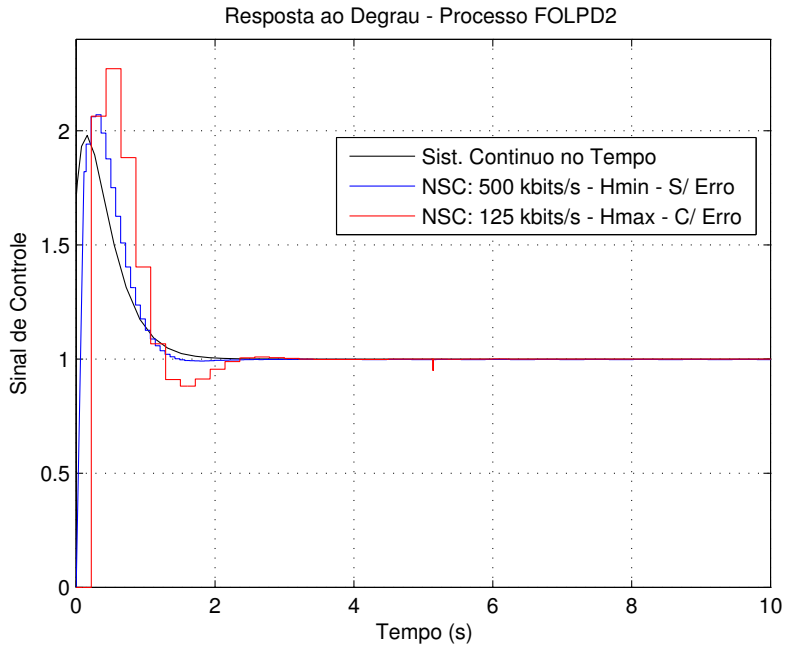


Figura 6.17: Sinal de controle da resposta ao degrau unitário da malha de controle 4 no tempo contínuo, em um NCS CAN com taxa de transmissão igual a 500 kbits/s, malhas com período H_{min} e sem erros no barramento, e em um NCS CAN com taxa de transmissão igual a 125 kbits/s, malhas com período H_{max} e com uma taxa de 30 erros/s no barramento.

6.5 Comentários sobre o procedimento de projeto

Considerando o procedimento de projeto apresentado neste Capítulo, têm-se os seguintes parâmetros que influenciam o desempenho da malha de controle em um NCS CAN: período de amostragem, taxa de transmissão, taxa de erros, prioridade das mensagens das malhas; tipo de arquitetura do NCS e das malhas, e quantidade de malhas de controle. Todos os parâmetros citados influenciam diretamente no atraso de controle das malhas do NCS.

Para o engenheiro de controle, o passo 3, ou seja, definição do período de amostragem, é fundamental no desempenho da malha e na determinação dos tempos de resposta e, consequentemente, no cálculo da margem de *jitter*. Do ponto de vista do controle, um período de amostragem pequeno a princípio implica de modo positivo no desempenho da malha. Entretanto, isso resulta no aumento do fator de utilização da rede, que pode gerar um aumento nos atrasos de controle, na variabilidade dos atrasos e, consequentemente, uma redução da margem de *jitter*. Isso pode ser visualizado no gráfico apresentado na Figura 6.18, que representa a mudança da margem de *jitter* em função do período de amostragem para o processo motor DC apresentado na seção 6.3. Observe que a margem de jitter é reduzida pela metade (de 28 ms para 14 ms) quando o período aumenta três vezes (de 8 ms para 24 ms).

Como diretriz para o projetista do controle, sugere-se projetar o controlador para uma faixa

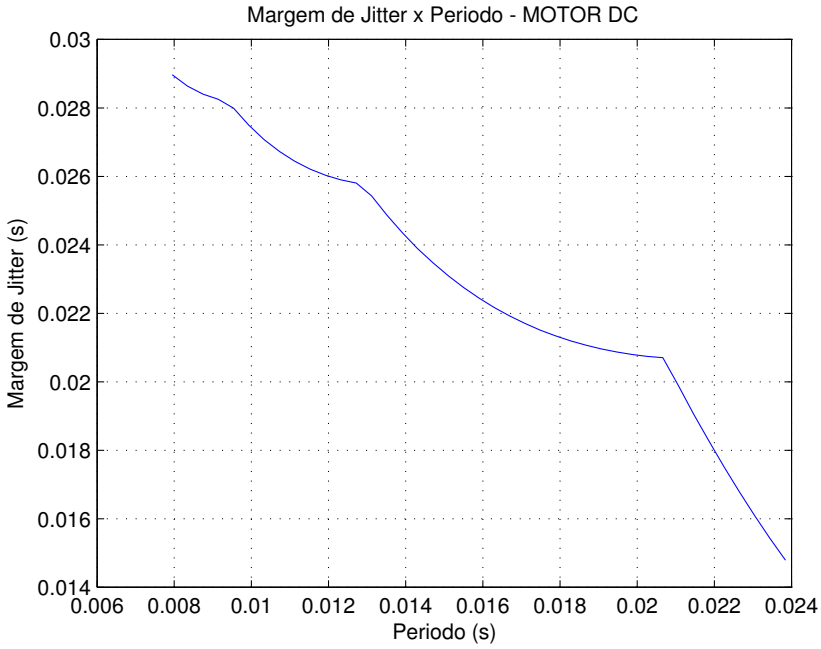


Figura 6.18: Gráfico da margem de *jitter* em função do período de amostragem para um motor DC e controlador PID.

de possíveis valores do período de amostragem, conforme foi apresentado na seção 6.3. Então, a partir desses períodos, o projetista da rede poderá usar esse parâmetro para definir a menor degradação de desempenho para as malhas, ajustando os parâmetros da rede que influenciam nos atrasos de controle tais como, taxa de transmissão, taxa de erros, prioridade das mensagens e arquitetura do NCS. De acordo com os resultados, observou-se como a redução da taxa de transmissão e o aumento da taxa de erro influencia negativamente no desemepnho da rede. Portanto, para ambientes onde a taxa de erro é alta, pode-se aumentar a taxa de transmissão para compensar a degradação do desempenho das malhas. Especificamente para rede CAN, essa diretriz é uma boa solução do ponto de vista econômico, uma vez que o aumento da taxa de transmissão não implica em um aumento nos custos do hardware.

6.6 Conclusões

Neste Capítulo apresentou-se um procedimento para projetar NCSs CAN e uma plataforma para simulação em tempo real de NCSs CAN. No procedimento, foi utilizado os modelos sem erros e com erros da arquitetura S-C-A para determinar os tempos de resposta fim a fim das malhas de controle para diferentes taxas de transmissão e diferentes taxas de erros no barramento. A consideração de erros no barramento foi um item diferencial com relação aos outros procedimentos ((CERVIN et al., 2004), (SANTOS; VASQUEZ, 2005) e (PEREZ; MORENO; MONTEZ, 2006)). Um estudo de caso com cinco malhas de controle foi realizado e, observou-se o impacto das taxas

de transmissão e, principalmente, de erros no barramento no desempenho das malhas de controle. Finalmente, apresentou-se uma plataforma experimental baseada na técnica de simulação *hardware in the loop*, contendo como partes reais a rede CAN e um injetor de falhas aleatórias. Então, essa plataforma foi utilizada para simular o NCS CAN com cinco malhas de controle projetado de acordo com o procedimento proposto. Verificou-se que os resultados são compatíveis com os esperados no projeto. Especialmente, observou-se uma tendência de degradação do desempenho para baixas de taxas de transmissão e a ocorrência de erros no barramento.

Capítulo 7

Conclusões

Neste trabalho foi abordado a modelagem, projeto e simulação de NCSs CAN considerando a integração das teorias relacionadas, de modo a verificar e mitigar os efeitos causados pela arquitetura em rede no desempenho das malhas de controle com controladores PIDs variantes no tempo. Para isso, quatro objetivos especificos apresentados na introdução deste trabalho foram definidos e cumpridos.

Correspondendo ao primeiro objetivo desta tese, foram desenvolvidos quatro de oito possíveis modelos para rede CAN utilizando autômatos temporizados e autômatos temporizados com custo, com o propósito de determinar o comportamento temporal de mensagens CAN periódicas por meio das abordagens de verificação de modelos clássica e estatística. Esses resultados são apresentados no Capítulo 4 e as conclusões de cada modelo são apresentadas a seguir.

Na seção 4.2 do Capítulo 4, foi desenvolvido um modelo para rede CAN com relógios ideais, onde os *offsets* das mensagens são dinâmicos e o barramento é livre de erros. Esse modelo foi uma adaptação mais compacta e passível de ser estendida para considerar outros comportamentos, do modelo desenvolvido por Krakora (2008) e Waszniowsky, Krakora e Hanzalek (2009). Um estudo de caso para uma rede CAN com oito mensagens foi realizado. Algumas propriedades lógicas e temporais foram verificadas. Utilizando a verificação de modelos clássica, foi possível determinar os tempos de resposta no melhor e pior caso para no máximo um subconjunto com quatro mensagens. Esses resultados foram comparados com os resultados obtidos via a técnica analítica desenvolvida por Davis et al. (2007). Foi verificado que os resultados são idênticos. A verificação de modelos clássica não conseguiu determinar os tempos de resposta para mais de quatro mensagens CAN, devido a uma explosão do espaço de estados do modelo. Então, como solução desse problema, foi aplicada a verificação de modelos estatística para o conjunto completo de mensagens. Então, foram obtidos as funções de distribuição acumulada das mensagens. Foi constatado que os tempos de resposta no pior caso ocorrem raramente. De fato, conforme a análise de Davis et al. (2007), o cenário de pior caso ocorre a partir de uma combinação muito específica de eventos. Então, foi verificado que a probabilidade de ocorrên-

cia desse cenário é muito baixa. Esse é um resultado importante, pois possibilita projetar redes CAN, do ponto de vista temporal, analisando as probabilidades dos deadlines serem satisfeitos, e não apenas analisando o cenário raro do pior caso.

Esse modelo foi validado utilizando uma plataforma experimental constituída por nove nós CAN. Uma estratégia foi desenvolvida para derivar as funções de distribuição acumulada das mensagens de cada nó, e esses resultados foram comparados com as funções de distribuição acumulada obtidas via verificação de modelos estatísticos. Então, aplicou-se o teste de aderência Kolmogorov-Smirnov para duas amostras com nível de confidência de 99% e verificou-se que as duas funções de distribuição acumulada pertencem a mesma distribuição de probabilidade. Portanto, o modelo foi validado com sucesso.

Na seção 4.3, foi desenvolvido um modelo para rede CAN com relógios físicos (reais), onde os *offsets* das mensagens são dinâmicos e o barramento é livre de erros. Foi apresentado um método para modelar as variações nos relógios dos nós CAN. A partir desse modelos, usou-se a verificação de modelos clássica para determinar os tempos de resposta no melhor e pior caso e a verificação de modelos estatísticos para determinar as distribuições dos tempos de resposta para cada mensagem considerando diferentes taxas de variação dos relógios. Verificou-se, conforme esperado (ver (MONOT et al., 2012)), que as variações nos relógios não influenciam de modo significante nos tempos de resposta e nas distribuições dos tempos de resposta das mensagens. Do ponto de vista da modelagem, esse resultado é importante, uma vez que nos possibilita desconsiderar as variações nos relógios, reduzindo assim a complexidade e o espaço de estados do modelo. Entretanto, essa desconsideração só é válido para modelos onde os *offsets* das mensagens são dinâmicos.

No terceiro modelo da rede CAN, apresentado na seção 4.5, considerou-se relógios ideais, *offsets* dinâmicos e a presença de erros no barramento. Nesse modelo, assumiu-se suposições realísticas que, normalmente, as abordagens analíticas não consideram, e que podem gerar resultados pessimistas e incorretos. No modelo, considera-se que os erros no barramento ocorrem aleatoriamente seguindo uma distribuição exponencial com média λ. A partir desse modelo, foi realizado um estudo de caso para uma rede CAN com oito mensagens, taxa de transmissão de 500 kbits/s e taxa de erros igual a 30 erros/s. Esse modelo foi validado com sucesso de modo semelhante ao primeiro modelo. Para isso, utilizou-se a mesma plataforma experimental em conjunto com um injetor de falhas aleatórias. Uma estratégia para injeção de falhas no barramento CAN foi desenvolvida considerando as especificações do protocolo da rede CAN. Para a injeção aleatória de falhas foi desenvolvido um gerador de variáveis aleatórias baseado no método da transformada inversa e no gerador de L'Ecuyer. Então, esse injetor foi implementado em um microcontrolador.

O quarto modelo da rede CAN foi apresentado na seção 4.7. Esse modelo considera uma rede CAN com relógios ideais, mensagens com *offsets* estáticos e livres de erros de transmissão.

Verificou-se que os tempos de resposta no pior caso são reduzidos significantemente quando as mensagens são escalonadas com *offsets* estáticos usando uma simples regra de atribuição de *offsets*. Para modelar corretamente a rede CAN, demonstrou-se a necessidade de usar a capacidade da ferramenta UPPAAL em atribuir prioridades aos autômatos, afim de evitar inversões de prioridade no acesso ao meio pelas mensagens. Usando a capacidade de simulação do UPPAAL, foi determinado a partir dos cenários que conduzem a inversão de prioridades, uma política de atribuição de prioridades aos autômatos. Para validar essa política e, consequentemente, o comportamento do modelo, determinou-se graficamente os tempos de resposta no melhor e pior caso para um conjunto de mensagens e comparou-se com os resultados obtidos sem e com a política de atribuição de prioridades aos autômatos. Observou-se também que a atribuição de prioridades aos autômatos reduziu o espaço de estados do modelo, de modo a eliminar estados que na prática não ocorrem. Essa redução do espaço de estados possibilitou a verificação, usando a abordagem clássica, de um conjunto com 16 mensagens CAN (*benchmark* SAE). Por outro lado, o uso de prioridades nos autômatos impossibilita (na versão atual da ferramenta UPPAAL) verificar modelos via a abordagem estatística.

Em seguida, no Capítulo 5, os modelos para rede CAN com relógios ideais, *offsets* dinâmicos e barramento sem e com erros, ou seja, os modelos 1 e 3, foram estendidos para modelar três das quatro principais arquiteturas de NCSs CAN, cumprindo desse modo, o segundo objetivo específico desta tese. Assim foram gerados seis modelos: dois para a arquitetura SC-A, dois para a arquitetura S-CA e dois para a arquitetura S-C-A. Um estudo de caso com cinco malhas de controle foi realizado.

Inicialmente, considerou-se o caso com barramento sem erros e aplicou-se a verificação de modelos simbólica clássica. Dez propriedades foram verificadas para cada malha. A verificação simbólica conseguiu apenas avaliar as propriedades para um conjunto com até três malhas de controle. Então, os tempos de resposta fim a fim, os tempos de resposta da mensagem sensor-controlador e da mensagem controlador-atuador, ambas no melhor e pior caso, foram determinadas. Os resultados para os NCSs nas arquiteturas SC-A e S-CA foram comparados com os resultados obtidos via a análise de pior caso de Davis et al. (2007). Foi verificado que os tempos de resposta no pior caso são idênticos. Os resultados dos tempos de resposta fim a fim para o NCS na arquitetura S-C-A foram comparados com os resultados obtidos via a análise de Henderson, Kendall e Robson (2001). Como esperado, foi demonstrado analiticamente que os tempos de resposta fim a fim no pior caso obtidos via a análise de Henderson, Kendall e Robson (2001) são pessimistas, ou seja, maiores do que acontecem na prática. Enquanto que os tempos de resposta fim a fim no melhor caso, usando ambas as técnicas, foram idênticos. Para mais de três malhas de controle, ocorreu o problema da explosão do espaço de estados. Então, para resolver esse problema, foi utilizado a verificação estatística.

A verificação estatística foi utilizada para verificar a probabilidade do deadline das cinco

malhas serem satisfeitos. Além disso, verificou-se a probabilidade das mensagens satisfarem seus *deadlines* (utilizamos o *deadline* igual ao período da malha). A ferramenta retornou gráficos das funções de distribuição acumulada com 99% de confidência. A partir desses gráficos, verificou-se mais uma vez que o cenário de pior caso ocorre raramente. Foi possível determinar as variações nos atrasos fim a fim. Esse foi um resultado importante, pois possibilitou utilizar essa técnica no projeto de NCSs CAN com o conceito da margem de *jitter* apresentado no Capítulo 6.

Em seguida, considerou-se o barramento com erros. O mesmo estudo de caso com cinco malhas foi utilizado. Então, verificou-se apenas para a arquitetura S-C-A a probabilidade dos deadlines serem satisfeitos para uma taxa de transmissão de 500 Kbits/s e uma taxa de 30 erros/s no barramento. Os resultados foram comparados para o caso onde o barramento é livre de erros. Foi verificado que as curvas da função de distribuição acumulada para o caso com erros no barramento estão sempre abaixo das curvas da função de distribuição acumulada para o caso sem erros no barramento, na faixa de probabilidade de 0 a pelo menos 95%. Como esperado, isso significa que os erros no barramento tendem a aumentar os tempos de resposta das mensagens e, consequentemente, do tempo de resposta fim a fim. Então, a partir de 95% de probabilidade, as curvas tenderam a se tornarem iguais. Observou-se que esse comportamento ocorreu devido a dois fatores: (1) o baixo fator de utilização da rede (em torno de 11,61%), que resulta em um alto número de ocorrências próximas ao tempo de resposta no melhor caso e (2) a ocorrência rara dos cenários de pior caso. Os gráficos para os tempos de resposta com erro no barramento se estendeu ao longo da faixa de probabilidade 99 a 100%.

No Capítulo 6, de acordo com o terceiro objetivo específico, foi apresentado um procedimento baseado nos trabalhos de Cervin et al. (2004), Santos e Vasquez (2005) e Perez, Moreno e Montez (2006) para projetar NCSs CAN usando controladores PIDs e o conceito da margem de *jitter*. No procedimento, foi utilizado os modelos sem erros e com erros da arquitetura S-C-A para determinar os tempos de resposta fim a fim das malhas de controle para diferentes taxas de transmissão e diferentes taxas de erros no barramento. A consideração de erros no barramento foi um item diferencial com relação aos outros procedimentos. Um estudo de caso com cinco malhas de controle foi realizado e, observou-se o impacto das taxas de transmissão e, principalmente, de erros no barramento no desempenho das malhas de controle.

Finalmente, apresentou-se uma plataforma experimental baseada na técnica de simulação *hardware in the loop*, contendo como partes reais a rede CAN e um injetor de falhas aleatórias. Então, essa plataforma foi utilizada para simular o NCS CAN com cinco malhas de controle projetado de acordo com o procedimento proposto. Verificou-se que os resultados são compatíveis com os esperados no projeto. Especialmente, observou-se uma tendência de degradação do desempenho para baixas de taxas de transmissão e a ocorrência de erros no barramento. No entanto, observa-se que para os parâmetros da rede, o fenômeno da amostragem vazia não

ocorrerá e, consequentemente, o fenômeno da rejeição de mensagens. A partir do procedimento proposto, foi observado como os parâmetros da rede podem contribuir para o melhoramento do desempenho das malhas de controle. Especificamente para a rede CAN, o aumento da taxa de transmissão pode ser usado para compensar outros atrasos gerados, por exemplo, pela ocorrência de erros no barramento devido a interferências eletromagnéticas provindas do ambiente da aplicação, sem nenhum aumento no custo do hardware. Sendo assim, o quarto e último objetivo da tese foi cumprido.

7.1 Trabalhos Futuros

A partir desta tese, os seguintes trabalhos podem ser desenvolvidos:

- Estender os modelos da rede CAN para transmitir e receber mais de uma mensagem por nó, implementado *buffers* de transmissão e recepção com políticas de prioridade semelhantes aos implementados nos controladores CAN encontrados no mercado, e escalonadores de tarefas. Esse modelo possibilitará considerar *deadlines* maiores que os períodos das mensagens;
- Desenvolver um modelo da rede CAN com *offsets* estáticos e relógios reais, para verificar o impacto da variação dos relógios nos tempos de resposta;
- Desenvolver algoritmos para atribuição ótima de *offsets* estáticos, com o objetivo de reduzir o tempos de resposta no pior caso das mensagens;
- Estender os modelos NCSs CAN para considerar *deadlines* maiores que os períodos das malhas de controle;
- Estender os modelos NCSs CAN considerando *offsets* estáticos;
- Desenvolver regras de sintonia para controladores PIDs com atrasos variantes utilizando os conceitos da margem de *jitter* e margem de fase aparente;
- Desenvolver um procedimento de projeto de NCSs CAN para considerar atrasos de controle maiores que os períodos de amostragem das malhas;
- Projetar NCSs CAN com sensores guiados a eventos.

Referências Bibliográficas

AG, B.; MOTOROLA; ELMOS; INFINEON; EC, S.; PROZESSAUTOMATISIERUNG, S. T. F.; IXXAT. *Byteflight Specification*. http://www.byteflight.com, 2001.

ALBERT, A. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded World*, p. 235–252, 2004.

ALLIANCE, Z. Zigbee specification. 2006. Disponível em: <www.zigbee.org>.

ALMEIDA, L. M. P. de. *Flexibility and Timeliness in Fieldbus-Based Real-Time Systems*. Tese (Doutorado) — University of Aveiro, Portugal, 1999.

ALMEIDA, L. M. P. de; FONSECA, J. A.; FONSECA, P. Flexible time-triggered communication on a controller area network. *In Proc. of The 19th IEEE Real-Time Systems Symposium*, 1998.

ALUR, R.; DILL, D. L. Automata for modeling real-time systems. *In Proc. of The 17th International colloquium on Automata, languages and programming*, v. 443 of LNCS, p. 322–335, 1990.

ALUR, R.; DILL, D. L. A theory of timed automata. *Journal of Theoretical Computer Science*, v. 126, n. 2, p. 183–235, 1994.

ALUR, R.; TORRE, S. L.; MADHUSUDAN, P. Perturbed timed automata. *In Proc. of the 8th International Workshop, Hybrid Systems: Computation and Control*, Porto, Portugal, n. 3414 in LNCS, p. 70–85, March 2005.

ASTROM, K. J.; HAGGLUND, T. *PID Controllers: Theory, Design and Tuning*. 2nd. ed. USA: Instrument Society of America, 1995.

ASTROM, K. J.; HAGGLUND, T. Revisiting the ziegler-nichols step response method for pid control. *Journal of Process Control*, v. 14, p. 635–650, 2004.

ASTROM, K. J.; HAGGLUND, T. *Advanced PID Control*. USA: ISA-The Instrumentation, Systems and Automation Society, 2006.

ASTROM, K. J.; WITTENMARK, B. *Computer-Controlled Systems: Theory and Design*. Third. USA: Prentice Hall, 1997.

AUDSLEY, N.; BURNS, A.; RICHARDSON, M.; TINDELL, K. W.; WELLINGS, A. J. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, v. 8, n. 5, p. 284–292, 1993.

AUDSLEY, N. C. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. *YCS164, Dept. of Computer Science, University of York*, 1991.

AZIMI-SADJADI, B. Stability of networked control systems in the presence of packet losses. *In Proc. of the IEEE Conference on Decision and Control*, p. 676–681, December 2003.

BARONTI, P.; PILLAI, P.; CHOOK, V. W. C.; CHESSA, S.; GOTTA, A.; HU, Y. F. Wireless sensor network: a survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, v. 30, n. 7, p. 1655–1695, May 2007.

BARUAH, S. K.; ROSIER, L. E.; HOWELL, R. R. Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor. *Real-time Systems*, v. 2, n. 4, p. 301–324, 1990.

BATE, I.; BURNS, A. Investigation of the pessimism in distributed systems timing analysis. *In Proc. of The 10th Euromicro Workshop on Real-Time Systems*, Berlin, Germany, p. 107–114, June 1998.

BAUER, H.; SCHARBARG, J.-L.; FRABOUL, C. Applying and optimizing trajectory approach for performance evaluation of afdx avionics network. *In Proc. of the 14th International Conference on Emerging Technologies and Factory Automation*, Mallorca, Spain, 2009.

BAUER, H.; SCHARBARG, J.-L.; FRABOUL, C. Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach. *IEEE transactions on industrial informatics*, v. 6, n. 4, p. 521–533, 2010.

BAUER, H.; SCHARBARG, J.-L.; FRABOUL, C. Worst-case end-to-end delay analysis of an avionics afdx network. *In Proc. of The Design, Automation and Test in Europe Conference and Exhibition*, p. 1220–1224, 2010.

BENGTSSON, J.; YI, W. Timed automata: Semantics, algorithms and tools. *Lecture Notes on Concurrency and Petri Nets, LNCS 3098, Springer-Verlag*, 2004.

BERHMANN, G.; DAVID, A.; LARSEN, K. G. A tutorial on uppaal, formal methods for the design of real-time systems. *In Proc. of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, p. 200–236, 2004.

BERTHOMIEU, B.; DIAZ, M. Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*, v. 17, n. 3, p. 259–273, 1991.

BHAMBHANI, V.; CHEN, Y.; XUE, D. Optimal fractional order proportional integral controller for varying time-delay systems. *In Proc. of the 17th IFAC World Congress*, COEX, Korea, South, July 2008.

BJöRKBOM, M. *Wireless control systems simulation ans network adaptive control*. Tese (Doutorado) — Helsinki University of Technology Control Engineering, Finland, 2010.

BJöRKBOM, M.; JOHANSSON, M. Networked pid control: Tuning and outage compensation. *In Proc. of The 36th Annual Conference on IEEE Industrial Electronics Society - IECON 2010*, p. 168–173, November 2010.

BOSCH, R. *CAN Specification Version 2.0*. Germany, 1991.

BOUDEC, J.-Y. L.; THIRAN, P. *Network calculus: A theory of deter ministic queuing systems for the internet*. 4th. ed. USA: Online Version of the Book Springer Verlag - LNCS 2050, 2012.

BOYCE, W. E.; DIPRIMA, R. C. *Equações Diferenciais Elementares e Problemas de Valores de Contorno*. Nona. USA: LTC, 2010.

BRANICKY, M. S.; PHILLIPS, S. R.; ZHANG, W. Stability of networked control systems: Explicit analysis of delay. *Proceedings of the American control conference*, v. 4, p. 2352–2357, 2000.

BRAUN, C.; HAVET, L.; NAVET, N. Netcarbench: a benchmark for techniques and tools used in the design of automotive communication systems. *In Proc. of the 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, November 2007.

BROSTER, I. *Flexibility in Dependable Real-Time Communication*. Tese (Doutorado) — University of York, England, August 2003.

BROSTER, I.; BURNS, A.; RODRIGUES-NAVAS, G. Probabilistic analysis of can with faults. *In Proc. of the 23rd IEEE real-time systems symposium*, p. 269–278, 2002.

BULYCHEV, P.; DAVID, A.; LARSEN, K. G.; LEGAY, A.; LI, G.; POULSEN, D. B.; STAINER, A. Monitor-based statistical model checking for weighted metric temporal logic. *In Proc. of LPAR-18*, p. 168–182, 2012.

BULYCHEV, P.; DAVID, A.; LARSEN, K. G.; MIKUCIONIS, M.; POULSEN, D. B.; LEGAY, A.; WANG, Z. Uppaal-smc: Statistical model checking for priced timed automata. *In Proc. of The 10th Workshop on Quantitative Aspects of Programming Languages and Systems*, p. 1–16, 2012.

BULYCHEV, P.; DAVID, A.; LARSEN, K. G.; LEGAY, A.; LI, G.; POULSEN, D. B. Rewrite-based statistical model checking of wmtl. *Lecture Notes in Computer Science*, v. 7687, p. 260–275, 2013.

BUTTAZZO, G. C. *Hard Real-Time Computing Systems*. 2nd. ed. USA: Springer-Verlag, 2005.

CARNEVALE, D.; TEEL, A. R.; NëSIC, D. Further results on stability of networked control systems: a lyapunov approach. *Proceedings of the american control conference*, July 2007.

CARNEVALE, D.; TEEL, A. R.; NëSIC, D. A lyapunov proof of an improved maximum allowable transfer interval for networked control systems. *IEEE transactions on Automatic Control*, v. 52, n. 5, p. 892–897, May 2007.

CERVIN, A.; HENRIKSSON, D.; LINCOLN, B.; EKER, J.; ARZéN, K.-E. How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime. *IEEE Control Systems Magazine*, v. 23, n. 3, p. 16–30, June 2003.

CERVIN, A.; LINCOLN, B.; EKER, J.; ARZEN, K. E.; BUTTAZZO, G. The jitter margin and its application in the design of real-time control systems. *In Proc. of The 10th International Conference on Real-Time and Embedded Computing Systems and Applications*, Gothenburg, Sweden, 2004.

CHEN, Y.; KURACHI, R.; ZENG, G.; TAKADA, H. Schedulability comparison for can message with offset: Priority queue versus fifo queue. *In Proc. of the 19th International Conference on Real-Time and Network Systems*, September 2011.

CRUZ, R. L. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, v. 37, n. 1, p. 114–131, 1991.

CRUZ, R. L. A calculus for network delay, part ii: Network analysis. *IEEE Transactions on Information Theory*, v. 37, n. 1, p. 132–141, 1991.

CUENCA, A.; GARCíA, P.; ALBERTOS, P.; SALT, J. A non-uniform predictor-observer for a networked control system. *International Journal of Control, Automation, and Systems*, v. 9, n. 6, p. 1194–1202, 2011.

DAVID, A.; LARSEN, K. G.; LEGAY, A.; MIKUCIONIS, M.; POULSEN, D. B.; VLIET, J. V.; WANG, Z. Statistical model checking for networks of priced timed automata. *In Proc. of Formal Modeling and Analysis of Timed Systems*, p. 80–96, 2011.

DAVID, A.; LARSEN, K. G.; LEGAY, A.; MIKUCIONIS, M.; WANG, Z. Time for statistical model checking of real-time systems. *In Proc. of the 23rd international conference on computer aided verification*, 2011.

DAVID, A.; LARSEN, K. G.; LEGAY, A.; MIKUCIONIS, M. Statistical model checking in uppaal: Lets practice. *In: Electronic Proceedings in Theoretical Computer Science*, 2014.

DAVID, A.; LARSEN, K. G.; MIKUCIONIS, M.; POULSEN, D. B.; LEGAY, A.; SEDWARDS, S.; DU, D. Statistical model checking for stochastic hybrid systems. *In Proc. of the 1st International Workshop on Hybrid Systems and Biology*, v. 92, p. 122–136, 2012.

DAVIS, R. I.; BURNS, A.; BRIL, R. J.; LUKKIEN, J. J. Controller area network schedulability analysis: refuted, revisited and revised. *Real-Time Systems*, v. 35, n. 3, p. 239–272, 2007.

DEVICENET. Devicenet an overview. Outubro 2011. Disponível em: <www.cancia.org/devicenet>.

DU, L.; XU, G. Worst case response time analysis for can messages with offsets. *In Proc of The IEEE International Conference on Vehicular Electronics and Safety*, p. 41–45, November 2009.

DUNN, W. L.; SHULTIS, J. K. *Exploring Monte Carlo Methods*. Second. USA: Elsevier, 2012.

ERIKSSON, L. *PID Controller Design and Tunning in Networked Control Systems*. Tese (Doutorado) — Helsinki University of Technology, Finland, October 2008.

ERIKSSON, L.; JOHANSSON, M. Pid controller tuning rules for varying time-delay systems. *In Proc. of the american control conference*, New York, USA, p. 619–625, July 2007.

ERIKSSON, L.; JOHANSSON, M. Simple pid tuning rules for varying time-delay systems. *In Proc. of the 46th IEEE conference on decision and control*, New Orleans, USA, p. 1801–1807, December 2007.

ERIKSSON, L.; KOIVO, H. N. Comparison of low-complexity controllers in varying time-delay systems. *Journal of control, measurement and systems integration*, v. 1, n. 2, p. 111–119, 2008.

ERIKSSON, L.; OKSANEN, T. Pid controller tuning for integrating processes: analysis and new design approach. *In Proc. fourth International Symposium on Mechatronics and its Applications*, Sharjah, UAE, March 2007.

ERIKSSON, L.; OKSANEN, T.; MIKKOLA, K. Pid controller tuning rules for integrating processes with varying time-delays. *Journal of the Franklin Institute*, Gothenburg, Sweden, v. 346, n. 5, p. 470–487, June 2009.

ETSCHBERGER, K. *Controller Area Network: Basics, Protocols, Chips and Application*. First. Germany: IXXAT Press, 2001.

EYISI, E.; BAI, J.; RILEY, D.; WENGA, J.; YANA, W.; XUEA, Y.; KOUTSOUKOS, X.; SZTIPANOVITS, J. Ncswt: An integrated modeling and simulation tool for networked control systems. *Simulation Modelling Practice and Theory*, v. 27, p. 90–111, June 2012.

FERSMAN, E.; MOKRUSHIN, L.; PETTERSSON, P.; YI, W. Schedulability analysis of fixed-priority systems using timed automata. *Theoretical Computer Science*, v. 354, p. 301–317, 2006.

FLEXRAY. *FlexRay Communications Systems Protocol Specification Version 2.1*. USA, 2005.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. *Feedback Control of Dynamic Systems*. 4th. ed. USA: Prentice Hall, 2002.

GAO, H.; MENG, X.; CHEN, T.; LAM, J. Stabilization of networked control systems via dynamic output-feedback controllers. *Society for Industrial and Applied Mathematics*, v. 48, n. 5, p. 3643–3658, 2010.

GARCíA-RIVERA, M.; BARREIRO, A. Analysis of networked control systems with drops and variable delays. *Automatica*, v. 43, n. 12, p. 2054–2059, 2007.

GARDEY, G.; LIME, D.; MAGNIN, M.; ROUX, O. Roméo: A tool for analyzing time petri nets. *In Proc. of The 17th International Conference on Computer Aided Verification, volume 3576 of Lecture Notes in Computer Science*, p. 418–423, July 2005.

GEORGE, L.; RIVIERRE, N.; SPURI, M. *Preemptive and Non-Preemptive Real-Time Uniprocessor Scheduling*. France, 1996.

GONZáLEZ, G. R.-N. *Design and formal verification of a fault-tolerant clock synchronization subsystem for the controller area network*. Tese (Doutorado) — Universitat de les Illes Balears, Palma, June 2010.

GRAHAM, S.; KUMAR, P. R. The convergence of control, communication, and computation. *Lecture Notes in Computer Science: Personal Wireless Communications*, v. 2775, p. 458–475, 2003.

GRENIER, M.; HAVET, L.; NAVET, N. Pushing the limits of can-scheduling frames with offsets provides a major performance boost. *In Proc. of The 4th European Congress Embedded Real Time Software*, January 2008.

GRESSER, K. An event model for deadline verification of hard real-time systems. *In Proc. of The 5th Euromicro Workshop on Real-Time Systems*, Oulu, Finland, p. 118–123, 1993.

GUTIéRREZ, J. C. P.; GARCIA, J. J. G.; HARBOUR, M. G. On the schedulability analysis for distributed hard real-time systems. *In Proc. of The 9th Euromicro Workshop on Real-Time Systems*, Toledo, Spain, p. 136–143, June 1997.

GUTIéRREZ, J. C. P.; GARCIA, J. J. G.; HARBOUR, M. G. Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems. *In Proc. of The 10th Euromicro Workshop on Real-Time Systems*, Berlin, Germany, p. 35–44, June 1998.

HALEVI, Y.; RAY, A. Integrated communication and control systems: Part i - analysis. *Jornal of Dynamic Systems, Measurement, and Control*, v. 110, p. 367–373, 1988.

HARBOUR, M. G.; KLEIN, M. H.; LEHOCZKY, J. P. Fixed priority scheduling of periodic task with varying execution priority. *In Proc. of The 12th IEEE Real-Time Systems Symposium*, p. 116–128, 1991.

HENDERSON, W.; KENDALL, D.; ROBSON, A. Accouting for clock frequency variation in the analysis of distributed factory control systems. *In Proc. of The Factory Communication Systems*, p. 51–58, 2000.

HENDERSON, W.; KENDALL, D.; ROBSON, A. Improving the accuracy of scheduling analysis applied to distributed systems computing minimal response times and reducing jitter. *Real-Time Systems*, v. 20, n. 1, p. 5–25, 2001.

HENIA, R.; RACU, R.; ERNST, R. Improved output jitter calculation for compositional performance analysis of distributed systems. *In Proc of The Parallel and Distributed Processing Symposium*, p. 1–8, 2007.

HENZINGER, T. A.; NICOLLIN, X.; SIFAKIS, J.; YOVINE, S. Symbolic model checking for real-time systems. *Journal of Information and Computation*, v. 111, n. 2, p. 193–244, 1994.

HESPANHA, J. P.; NAGHSHTABRIZI, P.; XU, Y. A survey of recent results in networked control systems. *In Proc. of The IEEE*, v. 95, n. 1, p. 138–162, January 2007.

HETEL, L.; DAAFOUZ, J.; IUNG, C. Analysis and control of lti and switched systems in digital loops via an event-based modeling. *International Journal of Control*, 2008.

HOLZMANN, G. J. The modelo checker spin. *IEEE Transactions on Software Engineering*, v. 23, n. 5, p. 279–295, May 1997.

HU, N.; LV, T.; HUANG, N. Applying trajectory approach for computing worst-case end-to-end delays on an afdx network. *Advanced in control enginnering and information science*, v. 15, p. 2555–2560, 2011.

HU, S.; YAN, W.-Y. Stability robustness of networked control systems with respect to packet loss. *Automatica*, v. 43, n. 7, p. 1243–1248, 2007.

IEEE. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Especifications*. USA, 1999.

IEEE. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. New York, USA, September 2006.

INSTRUMENTS, T. *Stellaris LM4F120 Launchpad Evaluation board - User Manual*. Spmu289a. USA, 2012.

ISA100. Wireless systems for automation. Março 2012. Disponível em: <www.isa.org/isa100>.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 1: Data link layer and physical signalling*. ISO 11898-1, 1999.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 2: High-speed medium access unit and medium-dependent interface*. ISO 11898-2, 1999.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 3: Low-speed, fault-tolerant, medium-dependent interface*. ISO 11898-3, 1999.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 4: Time triggered communication*. ISO 11898-4, 2004.

JAIN, R. *The Art of Computer Systems Performance Analysis*. First. USA: Willey, 1991.

JOSEPH, M.; PANDYA, P. Finding response times in a real-time system. *BCS Computer Journal*, v. 29, n. 5, p. 390–395, 1986.

JUNG, E.-H.; LEE, H.-H.; SUB, Y.-S. Lmi-based output feedback control of networked control systems. *In Proc. of the Factory Communication Systems*, p. 311–314, September 2004.

KAO, C. Y.; LINCOLN, B. Simple stability criteria for systems with time-varying delays. *Automatica*, v. 40, n. 8, p. 1429–1434, August 2004.

KIM, D. S.; LEE, Y. S.; KWON, W. H.; PARK, H. S. Maximum allowable delay bounds of networked control systems. *Control Engineering Practice*, v. 11, p. 1301–1313, 2003.

KIM, Y. H.; PARK, H. S.; KWON, W. H. Stability and a scheduling method for network-based control systems. *The 22nd annual conference of the IEEE industrial electronics society*, v. 2, p. 934–939, 1996.

KIM, Y. H.; PARK, H. S.; KWON, W. H. A scheduling method for network-based control systems. *Proceedings of the 1998 American control conference*, v. 2, p. 718–722, 1998.

KOHTAMAKI, T.; POHJOLA, M.; BRAND, J.; ERIKSSON, L. M. Piccsim toolchain - design, simulation and automatic implementation of wireless networked control systems. *In Proc. of the IEEE International Conference on Networking, Sensing and Control*, Okayama, Japan, p. 49–54, March 2009.

KOPETZ, H.; BAUER, G. The time-triggered architecture. *In Proc. of The IEEE*, v. 91, n. 1, p. 112–126, 2003.

KRAKORA, J. *Using Timed Automata for Verification, Testing and Scheduling in Distributed Systems*. Tese (Doutorado) — Czech Technical University in Prague, Prague, 2008.

KRIESEL, W. R.; MADELUNG, O. W. *AS Interface - The Actuator-Sensor-Interface for Automation*. Germany: Carl Hanser Verlag, 1999.

KRTOLICA, R.; ÖZGüNER, Ü.; CHAN, H.; GöKTAS, H.; WINKELMAN, J.; LI-UBAKKA, M. Stability of linear feedback systems with random communication delays. *International Journal of control*, v. 59, n. 4, p. 925–953, 1994.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: uma Nova Abordagem*. 1st. ed. Brasil: Addison Wesley, 2003.

KVASER. Kvaser - advanced can solutions. Disponível em: www.kvaser.com. September 2014.

LARSEN, K. G.; PETTERSSON, P.; YI, W. Model-checking for real-time systems. *In Proc. of the 10th international conference on fundamentals of computation theory*, v. 965, p. 62–88, 1995.

LARSEN, K. G.; PETTERSSON, P.; YI, W. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, v. 1, n. 1-2, p. 134–152, October 1997.

LEDIN, J. A. Hardware-in-the-loop simulation. *Embedded Systems Programming*, v. 12, n. 2, p. 42–60, 1999.

LEEN, G.; HEFFERNAN, D. Ttcan: A new time-triggered controller area network. *Microprocessors and Microsystems*, v. 26, n. 2, p. 77–94, 2002.

LEHOCZKY, J. P. Fixed priority scheduling of periodic task sets with arbitrary deadlines. *Proceedings 11th IEEE Real-Time Systems Symposium*, p. 201–209, 1990.

LEUNG, J. Y. T.; WHITEHEAD, J. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, v. 2, n. 4, p. 237–250, 1982.

LI, S.; WANG, Z.; SUN, Y. Observer-based compensator design for networked control systems with long time delays. *In Proc. of the IEEE Industrial Electronics Society*, Busan, Korea, p. 678–683, November 2004.

LIAN, F. L. *Analysis, Design, Modelling and Control of Networked Control Systems*. Tese (Doutorado) — University of Michigan, 2001.

LIMA, G. M. de A. *Fault Tolerance in Fixed-Priority Hard Real-Time Distributed Systems*. Tese (Doutorado) — University of York, England, 2003.

LIME, D.; ROUX, O. H. Formal verification of real-time systems with preemptive scheduling. *Real-Time Systems*, v. 41, p. 118–151, 2009.

LINCOLN, B.; BERNHARDSSON, B. Optimal control over networks with long random delays. *In Proc. of the 14th International Symposium on Mathematical Theory of Networks and Systems*, January 2000.

LIOU, L.-W.; RAY, A. Integrated communication and control systems: Part iii - nonidentical sensor and controller sampling. *Jornal of Dynamic Systems, Measurement, and Control*, v. 112, p. 357–364, 1990.

LIU, C. L.; LAYLAND, J. W. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of The ACM*, v. 20, n. 1, p. 46–61, 1973.

LUCK, R.; RAY, A. An observer-based compensator for distributed delays. *Automatica*, v. 26, n. 5, p. 903–908, 1990.

MA, C.; CHEN, S.; LIU, W. Maximum allowable delay bound of networked control systems with multi-step delay. *Simulation modelling practive and theory*, v. 15, p. 513–520, 2007.

MARTIN, S.; MINET, P. Improving the analysis of distributed non-preemptive fd/dp* scheduling with the trajectory approach. *Telecomunication systems*, v. 30, p. 49–79, 2005.

MARTIN, S.; MINET, P.; GEORGE, L. End-to-end response time with fixed priority scheduling: trajectory approach versus holistic approach. *International Journal of Communication Systems*, v. 18, p. 37–56, 2005.

MEDLEJ, S.; MARTIN, S.; COTTIN, J.-M. Identifying sources of pessimism in the trajectory approach with fifo scheduling. Toulouse, France, In Proc. of the embedded real-time software and systems, 2012.

MERLIN, P. M. *A study of the recoverability of computing systems*. Tese (Doutorado) — Dep of Information and Computer Science, University of California, Irvine, CA, USA, 1974.

METZNER, A.; STIERAND, I. Analyzing mixed event triggered/time triggered systems. *In Proc. of The 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, p. 235–252, 2006.

MICROCHIP. *PIC16F87XA Data Sheet*. Ds39582a. USA, 2001.

MICROCHIP. *High-Speed CAN Transceiver*. Ds21667d. USA, 2003.

MICROCHIP. *Stand-Alone CAN Cotroller with SPI Interface*. Ds21801d. USA, 2005.

MIRKIN, L. Some remarks on the use of time-varying delay to model sample-and-hold circuits. *IEEE Transactions Automatic Control*, v. 52, n. 6, p. 1109–1112, 2007.

MITTER, S. K. System science: The convergence of communication, computation and control. *In Proc. of The 2002 Conference on Control Applications*, Glasgow, Scotland, September 2002.

MODBUS. The modbus organization. Março 2012. Disponível em: <www.modbus.org>.

MOHALIK, S.; CHOKSHI, D. B.; DIXIT, M. G.; RAJEEV, A. C.; RAMESH, S. Scalable model-checking for precise end-to-end latency computation. *In Proc. of The IEEE Conference on Computer Aided Control System Design*, Hyderabad, India, p. 19–24, August 2013.

MOHALIK, S.; RAJEEV, A. C.; DIXIT, M. G.; RAMESH, S.; SUMAN, P. V.; PANDYA, P. K.; JIANG, S. Model checking based analysis of end-to-end latency in embedded, real-time systems with clock drifts. *In Proc. of the 45th annual Design Automation Conference*, p. 296–299, 2008.

MONOT, A.; NAVET, N.; BAVOUX, B. Impact of clock drifts on can frame response time distributions. *In proc. of the 16th IEEE International Conference on Emerging Technologies and Factory Automation*, Toulouse, France, September 2011.

MONOT, A.; NAVET, N.; BAVOUX, B.; MAXIM, C. Fine-grained simulation in the design of automotive communication systems. *Embedded Real-Time Software and Systems*, Toulouse, France, February 2012.

MONTESTRUQUE, L. A.; ANTSAKLIS, P. J. Stability of model-based networked control systems with time-varying transmission times. *IEEE Transactions on Automatic Control*, v. 49, n. 9, p. 1562–1572, 2004.

MURRAY, R. M.; ASTROM, K. J.; BOYD, S. P.; BROCKETT, R. W.; STEIN, G. Control in an information rich world. *Control Systems, IEEE*, v. 23, n. 2, p. 20–33, April 2003.

NAGHSHTABRIZI, P.; HESPANHA, J. Designing observer-based controller for network control system. *In Proc. 44th Conference Decision and Control*, p. 848–853, December 2005.

NAGHSHTABRIZI, P.; HESPANHA, J. P. Stability of network control systems with variable sampling and delays. *In Proc. of the Forty-Fourth Annual Allerton Conference on Communication, Control, and Computing*, September 2006.

NAGHSHTABRIZI, P.; HESPANHA, J. P.; TEEL, A. R. Stability of infinite dimensional impulsive systems with application to network control systems. *In Proc. of the American Control Conference*, New York, USA, p. 899 – 4904, July 2007.

NAVET, N.; SONG, Y.-Q.; SIMONOT, F. Worst-case deadline failure probability in real-time applications distributes over controller area network. *Jornal of Systems Architecture*, v. 46, p. 607–617, 2000.

NIEMINEN, J.; BJöRKBOM, M.; JANTTI, R.; ERIKSSON, L. Multichannel communications in wireless automation: Interdependencies between communication and control parameters. *International Journal of Distributed Sensor Networks*, v. 2012, p. 1–14, December 2011.

NILSSON, J. *Real Time Control Systems with Delay*. Tese (Doutorado) — Lund Institute of Technology, Sweden, 1998.

NOVAK, J.; FRIED, A.; VACEK, M. Can generator and error injector. *In Proc. 9th International Conf. Electronics, Circuits and Systems, IEEE press*, p. 967–970, September 2002.

NëSIC, D.; TELL, A. R. Input-output stability properties of networked control systems. *IEEE transactions on automatic control*, v. 49, n. 10, p. 1650–1667, October 2004.

ODVA. Controlnet. Março 2012. Disponível em: <www.controlnet.org>.

O'DWYER, A. *Handbook of PI and PID Controller Tuning Rules*. 2nd. ed. London: Imperial College Press, 2006.

OGATA, K. *Engenharia de Controle Moderno*. 3rd. ed. Brasil: LTC, 2000.

PALENCIA, J. C.; HARBOUR, M. G. Schedulability analysis for tasks with static and dynamic offsets. *In Proc. of The 19th IEEE Real-time Systems Symposium*, Madrid, Spain, p. 26–37, December 1998.

PAN, I.; DAS, S.; GUPTA, A. Handling packet dropouts and random delays for unstable delayed processes in ncs by optimal tuning of $pi^{\lambda}d^{\mu}$ controllers with evolutionary algorithms. *ISA transactions*, v. 50, p. 557–572, 2011.

PAN, Y.-J.; MARQUEZ, H. J.; CHEN, T. Stabilization of remote control systems with unknown time varying delays by lmi techniques. *International Journal of Control*, v. 79, n. 7, p. 752–763, July 2006.

PARK, H. S.; KIM, Y. H.; KIM, D. S.; KWON, W. H. A scheduling method for network based control system. *IEEE Transaction on Control System Technology*, v. 10, p. 318–330, 2002.

PEREZ, D. A.; MORENO, U. F.; MONTEZ, C. B. Codesign of can networked control systems with remote controllers using jitter margin. *In Proc. of The 32nd Annual Conference of The IEEE Industrial Eletronics - IECON*, Paris, France, p. 252–257, November 2006.

PFEIFFER, O.; AYRE, A.; KEYDEL, C. *Embedded Networking eith CAN and CANopen*. RTC Books: ISBN 0-929392-78-7, 2003.

PHAT, V. N. Robust stability and stabilizability of uncertain linear hybrid systems with state delays. *IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 52, n. 2, p. 94–98, 2005.

PILTZ, S.; BJöRKBOM, M.; ERIKSSON, L. M.; KOIVO, H. N. Step adaptative controller for networked mimo control systems. *In Proc. of the IEEE International Conference on Networking, Sensing and Control, ICNSC 2010*, Chicago, USA, p. 464–469, April 2010.

POHJOLA, M. Adaptative jitter margin pid controller. *The 4th IEEE Conference on Automation Science and Engineering*, USA, p. 534–539, August 2008.

POP, T.; ELES, P.; PENG, Z. Holistic scheduling and analysis of mixed time/event-triggered distributed embedded systems. *In Proc. of The 10th International Symposium on Hardware/software codesign*, Colorado, USA, p. 187–192, May 2002.

PROENZA, J.; MIRO-JULIA, J. Majorcan: a modification to the controller area network to achieve atomic broadcast. *In IEEE International Workshop on Group Communication and Computations*, 2000.

PROFIBUS. *PROFIBUS - Descrição Técnica*. Brasil, 2000.

PUNNEKKAT, S.; HANSSON, H.; NORSTRöM, C. Response time analysis under errors for can. *Proceedings 6th real-time technology and applications symposium. IEEE Computer Society Press*, p. 258–265, 2000.

RAJEEV, A. C.; MOHALIK, S.; DIXIT, M. G.; CHOKSHI, D. B.; RAMESH, S. Schedulability and end-to-end latency in distributed ecu networks: formal modeling and precise estimation. *In Proc. of the 10th ACM International Conference on Embedded Software*, p. 129–138, October 2010.

RAY, A.; HALEVI, Y. Integrated communication and control systems: Part ii - design considerations. *Jornal of Dynamic Systems, Measurement, and Control*, v. 110, p. 374–381, 1988.

REDELL, O. *Response Time Analysis for Implementation of Distributed Control Systems*. Tese (Doutorado) — Royal Institute of Technology, Stockholm, 2003.

REDELL, O.; SANFRIDSON, M. Exact best-case response time analysis of fixed priority scheduled tasks. *In Proc. of The 14th Euromicro Conference on Real-Time Systems*, Washington, DC, USA, 2002.

REORDA, M. S.; VIOLANTE, M. On-line analysis and perturbation of can networks. *In Proc. of The 19th IEEE International Symposium on Detect and Fault Tolarance in VLSI Systems*, p. 424–432, 2004.

RICHTER, K. *Compositional Scheduling Analysis Using Standard Event Models*. Tese (Doutorado) — Technical University of Braunschweig, 2004.

RIMéN, M.; CHRISTMANSSON, J. Testing can-based safety-critical systems using fault injection. *In Proc. of The 6th International CAN Conference*, 1999.

RITCHER, K.; ZIEGENBEIN, D.; JERSAK, M.; ERNST, R. Model composition for scheduling analysis in platform design. *In Proc. of The 39th Design Automation Conference*, 2002.

RODRIGUEZ-NAVAS, G.; JIMENEZ, K.; PROENZA, J. An architecture for physical injection of complex fault scenarios in can network. *In Proc. IEEE Conf. Emerging Technologies and Factory Automation, IEEE Press*, p. 125–128, 2003.

RODRIGUEZ-NAVAS, G.; PROENZA, J. Using timed automata for modeling distributed systems with clocks: challenges and solutions. *IEEE Transactions on Software Engineering*, v. 39, n. 6, p. 857–868, June 2013.

ROX, J.; ERNST, R. Compositional performance analysis with improved analysis techniques for obtaining viable end-to-end latencies in distributed embedded systems. *International Journal on Software Tools for Technology Transfer*, v. 15, n. 3, p. 171–187, 2013.

RTAW. Rtaw-sim: Controller area network simulation. Maio 2012. Available at https://www.realtimeatwork.com/software.

RUFINO, J.; VERíSSIMO, P.; ARROZ, G.; ALMEIDA, C.; RODRIGUES, L. Fault-tolerant broadcast in can. *In digest of papers, the 28th IEEE International Symposium on Fault-Tolerant Computing*, 1998.

SANTOS, M. M.; VASQUEZ, F. Aplicando margem de jitter no projeto de sistemas de controle via redes. *Simposio Brasileiro de Redes de Computadores*, Fortaleza, Brasil, 2005.

SANTOS, T. L. M. *Uma Contribuição ao Desenvolvimento de Sistemas de Controle Via Redes Usando a Margem de Jitter*. Dissertação — Universidade Federal de Santa Catarina - UFSC, Brasil, Fevereiro 2008.

SANTOS, T. L. M.; MORENO, U. F.; MONTEZ, C. B. Determination of a sampling period upper limit that guarantees closed loop stability in controller area network. *In Proc. of The 3rd Workshop on Networked Control Systems: Tolerant to Faults*, Nancy, France, 2007.

SANTOS, T. L. M.; MORENO, U. F.; MONTEZ, C. B.; PEREZ, D. A. Using the pole placement approach in a codesign procedure for networked control systems. *In Proc. of The 3rd IFAC Syposium on System Structure and Control*, Foz do Iguaçu, Brazil, 2007.

SEILER, P.; SENGUPTA, R. Analysis of communication losses in vehicle control problems. *In Proc. of the American Control Conference*, Arlington, USA, p. 1491–1496, June 2001.

SEILER, P.; SENGUPTA, R. An h_∞ approach to networked control. *IEEE Transactions on Automatic Control*, v. 50, n. 3, p. 356–364, 2005.

SEN, K.; VISWANATHAN, M.; AGHA, G. Statistical model checking of black-box probabilistic systems. *In: CAV, LNCS 3114*, p. 202–215, 2004.

SEN, K.; VISWANATHAN, M.; AGHA, G. A. Vesta: A statistical model-checker and analyzer for probabilistic systems. *In Proc. of The 2nd International Conference on The Quantitative Evaluation of Systems*, p. 251–252, September 2005.

SHOUSONG, H.; QIXIN, Z. Stochastic optimal control and analysis of stability of networked control systems with long delay. *Automatica*, v. 39, p. 1877–1884, 2003.

SONG, J.; HAN, S.; MOK, A. K.; CHEN, D.; LUCAS, M.; NIXON, M.; PRAT, W. Wirelesshart: Applying wireless technology in real-time industrial process control. *In Proc. IEEE Real-Time and Embedded Technology and Applications Symposium*, 2008.

SONG, J.; MOK, A. K.; CHEN, D.; NIXON, M.; BLEVINS, T.; WOJSZNIS, W. Improving pid control with unreliable communications. *In ISA EXPO*, 2006.

SPRUNT, B.; SHA, L.; LEHOCZKY, J. Aperiodic tasks scheduling for hard real-time systems. *The Journal of Real-Time Systems*, v. 1, p. 27–60, 1989.

SPURI, M. *Holistic Analysis for Deadline Scheduled Real-Time Distributed Systems*. France, April 1996.

SUN, J.; LIU, J. *bounding the end-to-end times for tasks in a distributed real-time systems using the direct synchronization protocol*. USA, March 1996.

SZAKALY, A. *Response Time Analysis with Offsets for CAN*. Dissertação — Department of Computer Engineering, Chalmers University of Technology, Sweden, November 2003.

Sá, A.; MACêDO, R.; SANTOS, T.; MORENO, U. Um procedimento para avaliação de redes ethernet comutada baseada em uma métrica de qualidade de controle. *In Proc. of the XVII Congresso Brasileiro de Automática*, Juiz de Fora, Brasil, Setembro 2008.

Sá, J. S.; LIMA, A. M. N.; PERKUSIC, A.; NETO, J. S. R. Análise dos tempos de resposta fim a fim de sistemas de controle via rede baseado na técnica de verificação de modelos. *XVIII Congresso Brasileiro de Automática - CBA*, Bonito, Brasil, 2010.

Sá, J. S.; LIMA, A. M. N.; PERKUSIC, A.; NETO, J. S. R. Análise probabilística dos tempos de resposta de uma rede can utilizando a técnica de verificação de modelos estatísticos. *XIX Congresso Brasileiro de Automática - CBA*, Campina Grande, Brasil, 2012.

Sá, J. S.; LIMA, A. M. N.; PERKUSIC, A.; NETO, J. S. R. Modelagem e verificação formal dos tempos de resposta de mensagens can com *Offsets* estáticos. *XI Simpósio Brasileiro de Automação inteligente - SBAI*, Fortaleza, Brasil, 2013.

Sá, J. S.; LIMA, A. M. N.; PERKUSIC, A.; NETO, J. S. R. A random fault injector for can network. *IEEE Latin America Transaction (no prelo)*, 2016.

TANENBAUM, A. S. *Computer Networks*. 4th edn. ed. Upper Saddler River, NJ 07458, USA: Prentice Hall, 2003.

TANENBAUM, A. S.; WOODHULL, A. S. *Operating systems: design and implementation*. 2nd. ed. New Jersey, USA: Prentice Hall, 1997.

TEJADO, I.; VINAGRE, B. M.; CHEN, Y. Fractional gain and order scheduling controller for networked control systems with variable delay. aplication to a smart wheel. *The 4th IFAC Workshop on Fractional Differentiation and its Application*, Badojoz, Spain, October 2010.

THIELE, L.; CHAKRABORTY, S.; NAEDELE, M. Realtime calculus for scheduling hard real-time systems. *In Proc. of The International Symposium on Circuits and Systems*, Geneva, Switzerland, 2000.

TINDELL, K.; BURNS, A. *Guaranteed message latencies for distributed safety-critical hard real-time control networks*. UK, 1994.

TINDELL, K. W.; BURNS, A. Guaranteeing message latencies on controller area network (can). *In Proc. of the 1st International CAN Conference*, p. 1–11, 1994.

TINDELL, K. W.; BURNS, A.; WELLINGS, A. J. An extendible approach for analysing fixed priority hard real-time systems. *Journal of Real-Time Systems*, v. 6, n. 2, p. 133–152, 1994.

TINDELL, K. W.; BURNS, A.; WELLINGS, A. J. Calculating controller area network (can) message response times. *Controle Engineering Practice*, v. 3, n. 8, p. 1163–1169, 1995.

TINDELL, K. W.; CLARK, J. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessors and Microprogramming Euromicro Journal*, v. 40, p. 117–134, April 1994.

TIPSUWAN, Y.; CHOW, M.-Y. Control methodologies in networked control systems. *Control Engineering Practice*, v. 11, p. 1099–1111, 2003.

TOVAR, E.; PINHO, L. M.; ALMEIDA, L. Position paper on time and event-triggered communication services in the context of e-manufacturing. *IEEE Workshop on Large Scale Real-Time and Embedded Systems*, 2002.

TRAN, E. *Multi-Bit Error Vulnerabilities in the Controller Area Network Protocol*. Pittsburgh, PA, May 1999.

TRAN, H.-D.; GUAN, Z.-H.; DANG, X.-K.; CHENG, X.-M.; YUAN, F.-S. A normalized pid controller in networked control systems with varying time delays. *ISA transactions*, v. 52, p. 592–599, 2013.

TTAGROUP. *Time-Triggered Protocol TTP/C High-Level Specification Document Protocol Version 1.1. Specification edition 1.4.3*. http://www.ttagroup.org, 2003.

UNGAN, V. *Networked PID controllers for wireless systems*. Dissertação — KTH Eletrical Engineering, Stockholm, November 2010.

UNRUH, J.; MATHONY, H.-J.; KAISER, K.-H. Error detection analysis of automotive communication protocols. *SAE Technical Paper 900699*, 1990.

WALSH, G. C.; HONG, Y. Scheduling of networked control systems. *IEEE Control Systems Magazine*, v. 21, p. 57–65, 2001.

WALSH, G. C.; YE, H.; BUSHNELL, L. Stability analysis of networked control systems. *In Proc. of The American Control Conference*, San Diego, California, p. 2876–2880, June 1999.

WALSH, G. C.; YE, H.; BUSHNELL, L. G. Stability analysis of networked control systems. *IEEE Transactions on Control Systems Technology*, v. 10, n. 3, p. 438–446, May 2002.

WASZNIOWSKY, L.; HANZALEK, Z. Formal verification of multitasking applications based on timed automata model. *Real-Time Systems*, v. 38, n. 1, p. 39–65, 2008.

WASZNIOWSKY, L.; KRAKORA, J.; HANZALEK, Z. Case study on distributed and fault tolerant system modeling based on timed automata. *Journal of Systems and Software*, v. 82, n. 1, p. 1678–1694, 2009.

WEBERMANN, H.; BLOCK, A. Can error injection, a simple but versatile approach. *In Proc. of The 13th International CAN Conference*, p. 12–14, 2012.

WILLIG, A.; MATHEUS, K.; WOLISZ, A. Wireless technology in industrial networks. *In Proc. of The IEEE*, v. 93, n. 6, p. 1130–1151, 2005.

WIRELESSHART. Hart communication foundation. 2011. Disponível em: <www.hartcomm.org>.

WITTENMARK, B.; NILSSON, J.; TöRNGREN, M. Timing problems in real-time control systems. *In Proc. of The American Control Conference*, p. 2000–2004, 1995.

WORLDFIP. Worldfip the efective fieldbus. Março 2012. Disponível em: <www.worldfip.org>.

XIE, G.; WANG, L. Stabilization of networked control systems with time-varying network-induced delay. *In Proc. of the 43rd IEEE Conference on Decision and Control*, Bahamas, p. 3551–3556, December 2004.

XIONG, J.; LAM, J. Stabilization of linear systems over networks with bounded packet loss. *Automatica*, v. 43, n. 1, p. 80–87, 2007.

XU, J.; PARNAS, D. L. On satisfying timing constraints in a hard-real-time systems. *IEEE Transactions on Software Engineering*, v. 19, n. 1, p. 70–84, 1993.

YANG, T. C. Networked control system: a brief survey. *Control Theory Application*, v. 153, p. 403–412, 2006.

YANG, Y.; WANG, Y. J. Modeling and control for ncs with time-varying long delays. *In Proc. of The 4th International Conference on Machine Learning and Cybernetics*, Guangzhou, p. 1407–1411, August 2005.

YANG, Y.; XU, D.; TAN, M.; DAI, X. Stochastic stability analysis and control of networked control systems with randomly varying long time-delays. *In Proc. of the 5th World Congress on Intelligent Control and Automation*, Hangzhou, China, p. 1391–1395, June 2004.

YEPEZ, J.; MARTI, P.; FUERTES, J. M. Control loop performance analysis over networked control systems. *In Proc. of The 28th Annual Conference of The IEEE Industrial Electronics Society (IECON02)*, v. 4, p. 2880–288, 2002.

YOMSI, P. M.; BERTRAND, D.; NAVET, N.; DAVIS, R. I. Controller area network (can): Response time analysis with offsets. *In Proc of The 9th IEEE International Workshop on Factory Communication System*, Germany, May 2012.

YOU, K.-Y.; XIE, L.-H. Survey of recent progress in networked control systems. *Acta Automatica Sinica*, v. 39, n. 2, p. 101–117, February 2013.

YOUNES, H. L. S. *Verification and Planning for Stochastic Processes with Asynchronous Events*. Tese (Doutorado) — Carnegie Mellon, 2005.

YOUNES, H. L. S. Ymer: A statistical model checker. *In Proc. of The 17th International Conference on Computer Aided Verification*, p. 429–433, 2005.

YU, M.; WANG, L.; CHU, T.; XIE, G. An lmi approach to networked control systems with data packet dropout and transmission delays. *In Proc. of the 43rd IEEE Conference on Decision and Control*, Bahamas, p. 3545–3550, December 2004.

YU, M.; WANG, L.; CHU, T.; XIE, G. Stabilization of networked control systems with data packet dropout and network delays via switching system approach. *In Proc. of The 43rd IEEE Conference on Decision and Control*, p. 3539–3544, 2004.

YU, M.; WANG, L.; CHU, T.; XIE, G. Sampled-data stabilization of networked control systems with nonlinearity. *IEE Proc.-Control Theory Appl.*, v. 152, n. 6, p. 609–614, 2005.

YUE, D.; HAN, Q.-L.; PENG, C. State feedback controller design for networked control systems. *IEEE Transaction on Circuits and Systems*, v. 51, n. 11, p. 640–644, November 2004.

ZAMPIERI, S. Trends in networked control systems. *In Proc. of the 17th World Congress IFAC*, Seoul, Korea, p. 2886–2894, July 2008.

ZENG, H.; NATALE, M. D.; GIUSTO, P.; SANGIOVANNI-VINCENTELLI, A. L. Stochastic analysis of can-based real-time automotive systems. *IEEE Trans. Industrial Informatics*, v. 5, p. 388–401, 2009.

ZHANG, W. *Stability Analysis of Networked Control Systems*. Tese (Doutorado) — Case Western Reserve University, USA, 2001.

ZHANG, W.; BRANICKY, M. S.; PHILLIPS, S. R. Stability of networked control systems. *IEEE Control Systems Magazine*, v. 21, p. 84–99, 2001.

Apêndice A

Rede CAN

A.1 Codificação das Mensagens CAN

As mensagens são constituídas por uma sequência de *bits*, que são representadas fisicamente pela codificação NRZ (*Non-Return to Zero*). Nessa codificação, o nível do sinal permanece constante durante o tempo de transmissão de um *bit*, chamado de tempo de *bit*. Os *transceivers* tratam esses *bits* de acordo com o mecanismo *wired-AND*, onde o nível lógico '0' é dominante e o nível lógico '1' é recessivo. Na codificação NRZ, é possível que uma seqüência longa de *bits* com mesmo nível lógico seja transmitida durante um período de tempo, causando a inexistência de bordas de transição, que são utilizadas no processo de sincronização durante a recepção das mensagens. Para evitar a ausência de bordas de transição na seqüência de *bits* de uma mensagem, CAN utiliza o mecanismo de inserção de *bits*. Esse mecanismo insere automaticamente após uma seqüência de cinco *bits* com mesma polaridade, um *bit* com polaridade oposta. Desse modo, garante-se uma transição de borda no máximo a cada cinco *bits*. Após a recepção da mensagem, os *bits* inseridos são retirados automaticamente pelo controlador do nó receptor.

A.2 Mensagem de Dados

A mensagem de dados (Figura A.1) é formada por cinco campos (arbitragem, controle, dados, CRC (*Cyclic Redundancy Code*) e ACK (*Acknowledgement*)) delimitados por um *bit* dominante (SOF - *Start Of Frame*) no início da mensagem e por sete *bits* recessivos (EOF - *End Of Frame*) no fim da mensagem. O *bit* SOF até o penúltimo *bit* do campo de CRC são submetidos ao mecanismo de inserção de *bits*. As informações são transmitidas no campo de dados e podem variar de 0 a 8 *bytes*. A quantidade de *bytes* de dados é especificada no campo de controle por valores inteiros variando de 0 a 8, ou seja, uma mensagem pode conter 0, 1, 2,... ou 8 *bytes* no campo de dados. Cada mensagem possui um identificador único, atribuído pelo projetista do sistema, localizado no campo de arbitragem, que pode ter 11 *bits* no formato padrão ou 29 *bits*

no formato estendido. Apenas mensagens com formato padrão serão consideradas. Portanto, é possível ter no sistema até 2048 (2^{11}) diferentes mensagens. O identificador possui duas funções: caracterizar o conteúdo da mensagem de modo que sejam filtradas/identificadas no nó receptor, e determinar o nível de prioridade da mensagem quanto o acesso ao barramento.

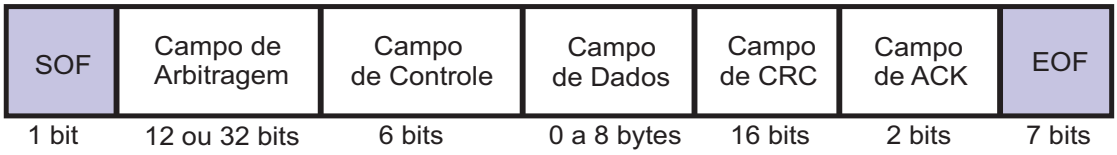


Figura A.1: Esquema de uma mensagem de dados.

O tempo gasto para um nó transmitir fisicamente uma mensagem m_i, denotado por C_i^m, onde $i \in \mathbb{Z}^{*+} = 1,2,3,...$ indica o número da mensagem, é determinado em função do comprimento da mensagem (em *bits*) e do tempo de transmissão de um *bit* (τ_{bit} - tempo de *bit*). O comprimento da mensagem depende do número de *bytes* de dados (b_i^m) e da quantidade de *bits* adicionados pelo mecanismo de inserção de *bits* (Ver seção A.1). Devido ao mecanismo de inserção de *bits*, mensagens com o mesmo número de *bytes* de dados podem ter comprimentos diferentes. Portanto, C_{m_i} é variável de forma limitada. O tempo de transmissão no melhor caso ($C_{m_i}^{min}$) (com nenhum *bit* inserido) e o tempo de transmissão no pior caso ($C_{m_i}^{max}$) (com o valor máximo de *bits* inseridos) são derivados, respectivamente, pelas Equações A.1 e A.2.

$$C_{m_i}^{min} = (44 + 8b_{m_i})\,\tau_{bit} \tag{A.1}$$

$$C_{m_i}^{max} = \left(44 + 8b_{m_i} + \left\lfloor \frac{34 + 8b_{m_i} - 1}{4} \right\rfloor\right)\tau_{bit} \tag{A.2}$$

O termo $\left\lfloor \frac{34+8b_{m_i}-1}{4} \right\rfloor$, correspondente a função "chão"(*floor function*), retorna o maior inteiro menor ou igual a $\frac{34+8b_{m_i}-1}{4}$. Esse termo equivale aos *bits* inseridos pelo mecanismo de inserção de *bits* considerando o pior caso. A Equação A.2 pode ser simplificada, resultando na Equação A.3.

$$C_{m_i}^{max} = (52 + 10b_{m_i})\tau_{bit} \tag{A.3}$$

A.3 Mensagem de Erro

Para garantir a integridade das informações transmitidas em qualquer um dos quatro tipos de mensagens, o protocolo da rede CAN implementa vários mecanismos para detecção de erro. A detecção de um erro é sinalizada por uma mensagem de erro, que é constituída pelos campos

flag de erro e delimitador de erro, como representado na Figura A.2. O campo *flag* de erro possui no mínimo 6 e no máximo 12 *bits* dominantes, conforme explicado a seguir. Enquanto que o delimitador de erro é formado por 8 *bits* recessivos.

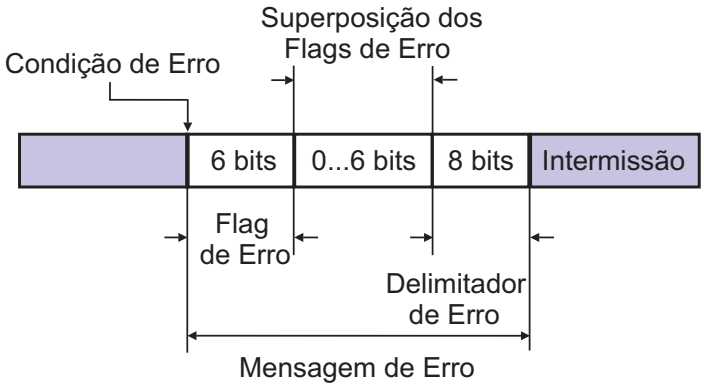


Figura A.2: Esquema de uma mensagem de erro ativo.

Um nó inicia a transmissão de um *flag* de erro, assim que detectar uma condição de erro, exceto no caso de um erro de CRC, onde a transmissão é iniciada no próximo *bit* após o campo ACK, de modo a não atrapalhar a função de reconhecimento da mensagem.

Existem dois tipos de *flag* de erro: o ativo, que é transmitido por nós ativos e consiste de seis *bits* dominantes (níveis lógico '0'), e o passivo, que é transmitido por nós passivos e consiste de seis *bits* recessivos (níveis lógico '1'). A definição de nó ativo e passivo está descrita na seção A.5.

Quando um ou mais nós ativos, transmissor e/ou receptores (possivelmente todos os nós da rede), detectam um erro na mensagem, esse erro é sinalizado pelo envio de um *flag* de erro ativo, que sobrescreve a mensagem com erro. Visto que o *flag* de erro viola a regra de inserção de *bits*, os outros nós da rede (ativos ou passivos) que ainda não detectaram o erro, enviam os seus respectivos *flags* de erro no máximo no final do sexto *bit* do primeiro *flag* de erro. Consequentemente, dependendo do instante em que os nós detectaram a condição de erro, é possível ter uma seqüência entre 6 e 12 *bits* no campo *flag* de erro.

Se um nó passivo e transmissor da atual mensagem detectar um erro, um *flag* de erro passivo será enviado por esse nó. No entanto, os receptores apenas detectarão o erro, se o *flag* de erro estiver sendo transmitido no lugar de um campo da mensagem codificado pelo método da inserção de *bits*, exceto no campo de arbitração e nos últimos seis *bits* do campo CRC. *Flags* de erro passivo enviados por nós receptores não sobrescrevem mensagens com erro. Portanto, não são capazes de sinalizar um erro no barramento.

Após a transmissão do seu *flag* de erro, os nós transmitem *bits* recessivos e monitoram o barramento até reconhecer um *bit* recessivo. Quando um *bit* recessivo for detectado, os sete *bits* recessivos restantes são transmitidos, finalizando o campo delimitador de erro. É importante enfatizar que até uma mensagem de erro está sujeita a ocorrência de um erro.

O tempo gasto para os nós transmitirem uma mensagem de erro, denotado por C_{e_m}, possui um intervalo de tempo variável. A variabilidade depende do instante em que os nós detectaram o erro. Os tempos de transmissão no melhor caso ($C_{e_m}^{min}$) e pior caso ($C_{e_m}^{max}$) são determinados, respectivamente, pelas Equações A.4 e A.5.

$$C_{e_m}^{min} = 14\tau_{bit} \tag{A.4}$$

$$C_{e_m}^{max} = 20\tau_{bit} \tag{A.5}$$

A.4 Intermissão

Mensagens de dados e de requisição de dados são separadas de mensagens precedentes (qualquer um dos tipos) por um intervalo de tempo chamado de intermissão. Após a intermissão, o barramento é considerado livre. O nível lógico lido do barramento no estado livre é sempre 1 (recessivo). O tempo de duração do intervalo entre mensagens depende da classificação atual do nó (Ver seção A.5), que pode ser ativo ou passivo. Para um nó ativo, o tempo de duração é de $3\tau_{bit}$. Para um nó passívo, o tempo dura $11\tau_{bit}$.

A.5 Classificação dos Nós CAN

A classificação dos nós é baseada no mecanismo de confinamento de falhas implementado no controlador CAN. Esse mecanismo consiste de dois contadores, TEC (*Transmit Error Counter*) e REC (*Receive Error Counter*), que são utilizados para classificar o nó em ativo, passivo ou desconectado, e uma lógica para incrementar e decrementar os contadores. Inicialmente, o nó é classificado como ativo. Se $REC \geq 127$ ou $TEC \geq 127$, o nó passa para o estado passivo. Caso $TEC > 255$, o nó é desconectado da rede. Um dos modos de retornar um nó desconectado para o estado ativo é resetá-lo. Para mais detalhes ver Bosch (1991) ou Etschberger (2001).

A.6 Processo de Arbitragem CAN

Considere uma rede CAN com três nós. Os nós 1, 2 e 3 transmitem, respectivamente, as mensagens de dados com identificadores 11001011101 (em decimal 1629), 11001111010 (em decimal 1658) e 11001011001 (em decimal 1625). A mensagem do nó 3 possui a maior prioridade (menor valor em decimal) e a mensagem do nó 2 a menor prioridade (maior valor em decimal). Um exemplo do processo de arbitragem para essa rede está representado na Figura A.3. Assim que o barramento torna-se livre (nível lógico '1'), os nós enviam o *bit* SOF e em seguida os *bits*

do campo de arbitragem. Durante a fase de arbitragem, todos os nós que estão transmitindo também monitoram o atual nível do barramento (’0’ ou 1’) e comparam com o *bit* transmitido. Se um nó enviar um ’0’ e monitorar um ’0’, ele continua no processo de arbitragem e envia o próximo *bit* do seu identificador. Caso envie um ’1’ e monitore um ’0’, esse nó para de transmitir e torna-se apenas um receptor da mensagem, como pode ser observado no *bit* 5 (Nó 2) e no *bit* 2 (Nó 1) do identificador. Nesse exemplo, o nó 3 vence o processo de arbitragem, visto que está transmitindo a mensagem com maior prioridade.

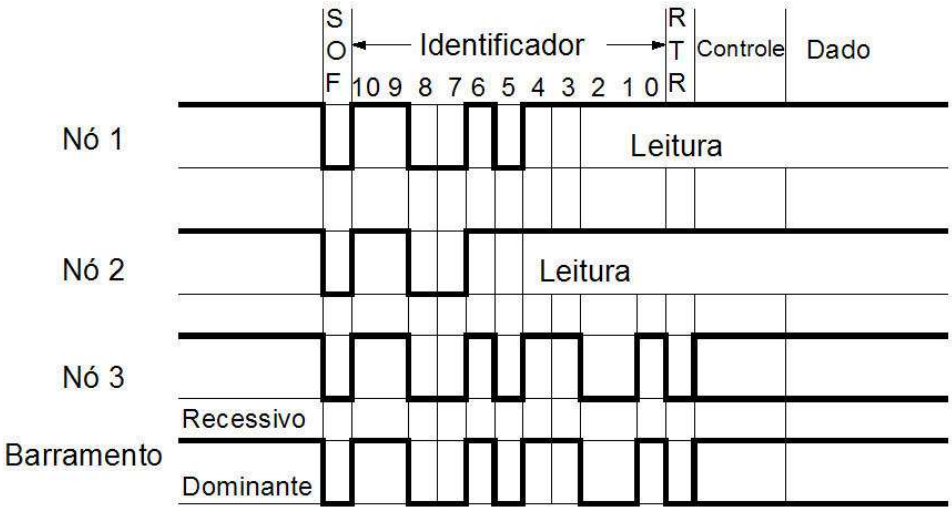


Figura A.3: Representação do processo de arbitragem em uma rede CAN.

Apêndice B

Ferramenta UPPAAL

B.1 Introdução

UPPAAL é uma ferramenta para modelagem, simulação e verificação de modelos baseados nas teorias de autômatos temporizados (TAs), autômatos temporizados com custo (PTAs) e autômatos híbridos estocásticos. UPPAAL possui dois métodos para verificação de modelos: verificação de modelos simbólicos (baseada em zonas) clássica (CMC) e a verificação de modelos estatística (SMC). Apresentaremos a seguir, as características do UPPAAL levando em consideração apenas os dois primeiros tipos de autômatos, que serão usados neste trabalho.

Um autômato, também chamado de processo, no UPPAAL é constituído por lugares (representados por círculos) e bordas ou transições (representadas por arcos). Existem quatro tipos de lugares:

- Inicial - Representado por círculos duplos. O autômato é iniciado nesse tipo de lugar;
- Normal - Representado por um círculo;
- Urgente - Representado por um círculo com a letra U. O tempo não progride quando o sistema está nesse tipo de lugar;
- *Committed* - Representado por um círculo com a letra C. Assim, como lugares urgentes, o tempo não progride nesse tipo de lugar. Além disso, se o sistema estiver em um lugar *committed*, a próxima mudança de lugar deverá envolver uma transição a partir de um dos lugares *committed*. Esse tipo de lugar é útil para criar seqüências atômicas e para codificar sincronização entre mais do que dois componentes.

Cada lugar pode ser rotulado por um nome e uma invariante de estado. A invariante é uma condição de tempo que deve ser satisfeita quando o sistema estiver nesse lugar (lugar ativo). Por exemplo, um lugar rotulado com uma invariante na forma $x <= 100$, onde x é um relógio,

indica que o sistema pode permanecer nesse lugar somente quando o valor de x for menor ou igual a 100. Uma invariante pode ser também uma expressão booleana que não envolve relógios. Além disso, apenas nos PTAs é possível especificar taxas específicas (não apenas 0 e 1 como no caso de *stop-watches*[1]) para os relógios. As taxas podem ser declaradas junto com as invariantes.

Cada borda pode ser rotulada por um canal de sincronização, guardas e atribuições. Um canal de sincronização é representado pelo seu nome com '!' (emissor) ou '?' (receptor). Por exemplo, o canal *press!* de uma transição envia um sinal que habilita uma transição rotulada pelo canal *press?*. Existem três tipos de canais de sincronização:

- Binário - Sincroniza uma transição com outra transição. É declarada utilizando a palavra−chave *chan*;
- Urgente - É semelhante à binária, mas as transições com esses canais não podem ter restrições de tempo. É declarada utilizando a palavra−chave *urgent chan*;
- *Broadcast* - Sincroniza uma transição com um número arbitrário de transições. É declarada utilizando a palavra−chave *broadcast chan*.

Os canais de sincronização são usados para formar redes de autômatos temporizados (NATs) e redes de autômatos temporizados com custo (NPTAs).

Guardas são expressões de condições que são separadas por vírgulas. Por exemplo, $x < 50$, $A[i] == 0$. Apenas relógios, variáveis inteiras e constantes (*arrays* desses tipos) podem ser referenciadas nos guardas. Atribuições são expressões separadas por vírgula que utilizam o sinal $=$ para atribuir valores a relógios, variáveis inteiras e constantes. Por exemplo, $x = 0, i = i + 1$.

As transições entre os lugares seguem as semânticas descritas nas seções 3.5.1 e 3.6.1 do Capítulo 3. Nos ATs, as transições ocorrem por atraso, de acordo com as restrições dos relógios nas invariantes e guardas, ou por uma ação via um canal de sincronização. Nos PTAs, as transições ocorrem também por atraso ou ação. Entretanto, nas transições por atraso, a partir de lugares com invariantes e bordas com guardas, os atrasos seguem uma distribuição uniforme, onde os limites inferiores são as restrições dos guardas e os limites superiores as restrições das invariantes. Na ausência de invariantes e guardas, os atrasos nas transições podem ser determinados por uma distribuição exponencial com uma taxa específica. A taxa é declarada por uma simples expressão inteira ou por duas expressões inteiras separadas por : tais como $r : q$, onde a taxa é determinada pela razão r/q. Assim, a probabilidade do autômato deixar um lugar especificado com uma taxa λ em um instante t é determinada por $1 - e^{-\lambda t}$, onde $e = 2{,}718281828...$. As transições por ação são idênticas as transições nos ATs. Além disso, é possível ter transições

[1] O valor '0' indica que o relógio está parado, ou seja, taxa igual a '0', e o valor '1' indica que o relógio progride com taxa igual a 1, de acordo com um relógio ideal.

discretas onde as bordas são rotuladas com pesos probabilísticos com desvios para mais de um lugar. Desse modo, é possível modelar transições como eventos probabilísticos. A probabilidade de um particular desvio é determinada como uma razão do seu peso pela soma dos pesos de todos os desvios partindo do mesmo nó de desvio.

UPPAAL permite também definir variáveis inteiras limitadas e compartilhar relógios e variáveis inteiras. Além disso, é possível atribuir prioridade às transições dos diferentes autômatos. A ideia é que em uma determinado instante, uma transição seja habilitada somente se nenhuma transição de maior prioridade estiver habilitada. As prioridades podem ser atribuídas aos canais e/ou autômatos do modelo.

Na verificação clássica, UPPAAL indica se as propriedades foram ou não satisfeitas. Para propriedades não satisfeitas, a ferramenta indica um contra exemplo. Na verificação estatística, a ferramenta verifica probabilidades a partir de medidas estatísticas em termos do tempo ou custo, apresenta com resultados histogramas de frequência, tempo ou custo médio, distribuições de densidade de probabilidade e distribuições de probabilidade acumulada, onde as distribuições são representadas com intervalos de confidência. Todos os dados estatísticos podem ser plotados em gráficos usando recursos da própria ferramenta e salvos em arquivos de texto.

B.2 Linguagem da Especificação de Requisitos Usando CMC

A linguagem da especificação de requisitos é baseada em um subconjunto da lógica CTL (*Computation Tree Logic*) (HENZINGER et al., 1994) e é composta pelos seguintes operadores: A (para todo caminho), E (existe um caminho), [ ] (sempre), <> (futuramente), $\rightarrow$ (conduz a), *and*, *or*, *not*, *imply* e *deadlock*. A gramática BNF (*Backus-Naur Formalism*) para a linguagem de especificação utilizada no verificador do UPPAAL é definida por:

$$\text{Prop} ::= \text{"E<>"}\ p \mid \text{"A<>"}\ p \mid \text{"E[]"}\ p \mid \text{"A[]"}\ p \mid p \rightarrow q$$

Onde p e q são variáveis ou expressões lógicas livres de efeito lateral construídas por operadores. Essas variáveis podem assumir os seguintes valores: o estado de um determinado autômato; uma operação de comparação ($=, != , <, \leq, >, \geq$) de um número real com o valor de um dos relógios do autômato; uma fórmula da lógica proposicional (p *and* q, p *or* q, *not* p e p *imply* q) ou a palavra reservada *deadlock*. Desse modo, as seguintes fórmulas lógicas para a linguagem de especificação do UPPAAL podem ser geradas:

E<> p - Existe pelo menos um caminho onde p futuramente acontece.

A<> p - Para todo caminho, p futuramente acontece.

E[] p - Existe pelo menos um caminho onde p sempre acontece.

A[] p - Para todo caminho, p sempre acontece.

$p \rightarrow q$ - Desde que p seja verdade, q será verdade futuramente.

As propriedades alcançabilidade (*reachability* - expressada pela fórmula $E<>p$), segurança (*safety* - expressada pelas fórmulas $E[\,]p$ e $A[\,]p$) e vivacidade (*lifeness* - expressada pelas fórmulas $A<>p$ e $\rightarrow q$) podem ser determinadas pelo verificador do UPPAAL. Os operadores sup (supremum) e inf (infimum) foram integrados ao UPPAAL para verificar propriedades de vivacidade limitada.

B.3 Linguagem de Especificações de Requisitos Usando SMC

O objetivo do SMC é estimar a probabilidade de valores de expressões (propriedades) estatisticamente. As propriedades são expressadas por uma linguagem de especificação de requisitos baseadas em um subconjunto da lógica $WMTL_{\leq}$ (*Weighted Metric Temporal Logic*) (Ver Bulychev et al. (2012a) e Bulychev et al. (2013)). UPPAAL implementa quatro tipos de propriedades estatísticas: quantitativa, qualitativa, comparativa e estimativa. As propriedades são avaliadas em execuções limitadas por tempo (*time*), onde *time* pode ser substituído por passos (*steps*), por um relógio ou por uma restrição de custo. Essas propriedades são apresentadas a seguir:

- Qualitativa - $Pr[time <= const](<> \mid [\,]\, expr) <= \mid >= p$;
- Quantitativa - $Pr[time <= const](<> \mid [\,]\, expr)$;
- Comparativa - $Pr[time <= const](<> \mid [\,]\, expr) <= \mid >= Pr[time <= const](<> \mid [\,]\, expr)$;
- Estimação do valor - $E[time <= const](min : \mid max : expr)$.

Onde: Pr indica probabilidade; (*const*) é um valor constante; $\mid$ é o operador lógico ou; *expr* são expressões de estado do modelo; p é um número ponto flutuante entre 0 e 1 denotando probabilidade, min e max são, respectivamente, o valor mínimo e máximo sobre uma execução da expressão precedente.

A fórmula qualitativa é utilizada para verificar se a probabilidade de satisfazer alguma propriedade é no máximo ou no mínimo igual a p. Esse teste utiliza um algoritmo estatístico que é uma extensão do teste de hipótese sequencial de Wald. O problema dessa propriedade se reduz a testar a hipótese $H : p = \mathbb{P}_{\mathcal{A}}(<>_{x \leq C}\, \phi) \geq \theta$ contra $K : p < \theta$, sendo x um relógio, C um valor limite, ϕ um estado predicado. Para limitar a probabilidade de gerar erros, o algoritmo utiliza os parâmetros α (probabilidade de negativas falsas) e β (probabilidade de positivas falsas), e testa as hipóteses $H_0 : p \geq p_0$ (positivas falsas) e $H_1 : p \leq p_1$ (negativas falsas), onde $p_0 = \theta + \delta_0$ e $p_1 = \theta - \delta_1$. O parâmetro α é a probabilidade de aceitar H_0 quando H_1 for mantida, e o parâmetro β é a probabilidade de aceitar H_1 quando H_0 for mantida. A fórmula quantitativa estima a probabilidade dentro de um intervalo utilizando um algoritmo baseado na simulação

de Monte Carlo. O algoritmo calcula o número N de execuções necessárias de modo a produzir um intervalo de aproximação $[p - \epsilon, p + \epsilon]$ para $p = Pr(\psi)$ com uma confidência $1 - \alpha$. Os valores de ϵ (Probabilidade de incerteza) e α são escolhidos pelo usuário e N requer o limite de Chernoff-Hoeffding. A fórmula comparativa compara duas probabilidades sem avaliá-las e determina o resultado para todos os limites até *CONST*, se ambos os limites forem iguais. O algoritmo utilizado é baseado também em uma extensão do teste de hipótese sequencial de Wald. A última propriedade estima o valor mínimo ou máximo de uma expressão executando um determinado número de simulações. Detalhes dos algoritmos implementados no UPPAAL podem ser vistos em David et al. (2011a).

Apêndice C

Códigos UPPAAL dos Modelos da Rede CAN

C.1 Código do Modelo da Rede CAN com *Offset* Dinâmico, Relógios e Barramento Ideais

O código UPPAAL do modelo para uma rede CAN com três mensagens e uma taxa de transmissão de 500 Kbits/s ($t_{bit} = 2\ \mu$s) é apresentado na Listagem C.1. Observe que nas declarações do sistema apenas um template do autômato arbitragem é gerado, enquanto que para cada mensagem são gerados um template dos autômatos tarefa e controlador CAN. Os valores dos atributos devem ser definidos nas declarações do sistema.

```
// Declarações Globais do Modelo
    const int n = 3; // Número de mensagens na rede
    int [0,1] Arb[n];
    const int tbit = 2; // Tempo de bit em microsegundos
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Sucess_Arb[n], Ack[n];

// Template do Autômato Tarefa
    // Declarações Locais
    clock Tempo;
    // Parâmetros
    const int i, const int T, const int D, const int C

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock Com;
    // Parâmetros
```

```
    const int i, const int Cm, const int TIFS

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n] ID=0;
// Declarações do Sistema
    TAR0 = Tarefa(i, T, D, C);
    TAR1 = Tarefa(i, T, D, C);
    TAR2 = Tarefa(i, T, D, C);
    CON0 = ControladorCAN (i, Cm, TIFS);
    CON1 = ControladorCAN (i, Cm, TIFS);
    CON2 = ControladorCAN (i, Cm, TIFS);
    A = Arbitragem();
    system TAR0, TAR1, TAR2, CON0, CON1, CON2, A;
```

Listing C.1: Código UPPAAL do modelo em autômatos temporizados de uma rede CAN com relógios ideais.

C.2 Código do Modelo da Rede CAN com *Offset* Dinâmico, Relógios Reais e Barramento Ideal

O código UPPAAL do modelo para uma rede CAN com três mensagens e uma taxa de transmissão de 500 Kbits/s ($t_{bit} = 2\ \mu$s) é apresentado na Listagem C.2. Observe que nas declarações do sistema apenas um template do autômato arbitragem é gerado, enquanto que para cada mensagem são gerados um template dos autômatos tarefa e controlador CAN. Os valores dos atributos devem ser definidos nas declarações do sistema. Todas as constantes e variáveis estão em nanosegundos, devido a ferramenta UPPAAL não aceitar variáveis de ponto flutuante.

```
// Declarações Globais do Modelo
    const int n = 3; // Número de mensagens na rede
    int [0,1] Arb[n];
    const int tbit = 2000; // Tempo de bit em nanosegundos
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Sucess_Arb[n], Ack[n];

// Template do Autômato Tarefa
    // Declarações Locais
    clock Tempo;
    // Parâmetros
    const int i, const int Tmax, const int Tmin, const int Dmax, const int Dmin,
```

```
    const int  Cmax, const int Cmin

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock  Com;
    // Parâmetros
    const int  i ,  const int  Cmmax, const int Cmmin, const int TIFSmax, const int TIFSmin

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n]  ID=0;
// Declarações do Sistema
    TAR0 = Tarefa(i,  Tmmax, Tmmin, Dmmax, Dmmin, Cmax, Cmin);
    TAR1 = Tarefa(i,  Tmmax, Tmmin, Dmax, Dmin, Cmax, Cmin);
    TAR2 = Tarefa(i,  Tmmax, Tmmin, Dmax, Dmin, Cmax, Cmin);
    CON0 = ControladorCAN (i, Cmmax, Cmmin, TIFSmax, TIFSmin);
    CON1 = ControladorCAN (i, Cmmax, Cmmin, TIFSmax, TIFSmin);
    CON2 = ControladorCAN (i, Cmmax, Cmmin, TIFSmax, TIFSmin);
    A = Arbitragem();
    system TAR0, TAR1, TAR2, CON0, CON1, CON2, A;
```

Listing C.2: Código UPPAAL do modelo.

C.3 Código do Modelo da Rede CAN com *Offset* Dinâmico, Relógios ideais e Barramento Real

O código UPPAAL do modelo para uma rede CAN com três mensagens, taxa de transmissão de 500 Kbits/s ($t_{bit} = 2\ \mu s$) e taxa de erro de 30 erros/s é apresentado na Listagem C.3. Observe que nas declarações do sistema que apenas um template de cada um dos autômatos arbitragem, interferência e erros são gerados, enquanto que para cada mensagem são gerados um template dos autômatos tarefa e controlador CAN. Os valores dos atributos devem ser definidos nas declarações do sistema. Todas as constantes e variáveis estão em microsegundos.

```
// Declarações Globais do Modelo
    const int  n = 3;  // Número de mensagens na rede
    int [0,1]   Arb[n];
    const int  tbit   = 2;  // Tempo de bit
    const int  Taxa_Exp = 33333; //Em us/erro
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Sucess_Arb[n], Ack[n], Fim_Erro,
```

```
        Erro;

// Template do Autômato Tarefa
    // Declarações Locais
    clock Tempo;
    // Parâmetros
    const int i, const int Tm, const int Dm, const int C

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock Com;
    // Parâmetros
    const int i, const int Cm const int TIFS

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n] ID=0;

// Template do Autômato Erros
    // Declarações Locais
    clock TempoErro;


// Declarações do Sistema
    TAR0 = Tarefa(i, Tm, Dm, C);
    TAR1 = Tarefa(i, Tm, Dm, C);
    TAR2 = Tarefa(i, Tm, Dm, C);
    CON0 = ControladorCAN (i, Cm, TIFS);
    CON1 = ControladorCAN (i, Cm, TIFS);
    CON2 = ControladorCAN (i, Cm, TIFS);
    A = Arbitragem();
    I = Interferencia ();
    E = Erros();
    system TAR0, TAR1, TAR2, CON0, CON1, CON2, A, I, E;
```

Listing C.3: Código UPPAAL do modelo.

C.4 Código do Modelo da Rede CAN com *Offset* Estático, Relógios e Barramento Ideais

O código UPPAAL do modelo para uma rede CAN com três mensagens e uma taxa de transmissão de 125 Kbits/s ($t_{bit} = 8\ \mu$s) é apresentado na Listagem C.4. Observe que nas declarações do sistema apenas um template do autômato arbitragem é gerado, enquanto que para cada mensagem são gerados um template dos autômatos tarefa e controlador CAN. Os valores dos atributos devem ser definidos nas declarações do sistema.

```
// Declarações Globais do Modelo
    const int  n = 3;  // Número de mensagens na rede
    int [0,1]   Arb[n];
    const int  tbit   = 8;  // Tempo de bit
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Sucess_Arb[n], Ack[n];
// Template do Autômato Tarefa
    // Declarações Locais
    clock  Tempo;
    // Parâmetros
    const int  i ,  const int  Tm, const int Dm, const int C, const int  Om

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock  Com;
    // Parâmetros
    const int  i ,  const int  Cm const int TIFS

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n]  ID=0;
// Declarações do Sistema
    TAR0 = Tarefa(i,  Tm, Dm, Cm, Om);
    TAR1 = Tarefa(i,  Tm, Dm, Cm, Om);
    TAR2 = Tarefa(i,  Tm, Dm, Cm, Om);
    CON0 = ControladorCAN (i, Cm, TIFS);
    CON1 = ControladorCAN (i, Cm, TIFS);
    CON2 = ControladorCAN (i, Cm, TIFS);
    A = Arbitragem();
    system A < CON2 < CON1 < CON0 < TAR2 < TAR1 < TAR0;
```

Listing C.4: Código UPPAAL do modelo.

Apêndice D

Códigos UPPAAL dos Modelos de NCSs CAN

D.1 Código do Modelo do NCS CAN na Arquitetura SC-A com *Offset* Dinâmico, Relógios e Barramento Ideais

O código UPPAAL do modelo de NCS CAN na arquitetura SC-A com *offset* dinâmico, relógios e barramento Ideais para três malhas de controle e taxa de transmissão de 500 Kbits/s ($t_{bit} = 2\ \mu$s) é apresentado na Listagem D.2. Os valores dos atributos devem ser definidos nas declarações do sistema. Todas as constantes e variáveis estão em microsegundos

```
// Declarações Globais do Modelo
    const int n = 6; // Número de mensagens na rede
    const int M = 3; // Três malhas de controle
    int [0,1] Arb[n];
    const int tbit = 2; // Tempo de bit em microsegundos
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Fim[M], Sucess_Arb[n], Ack[n];

// Template do Autômato Sensor Controlador
    // Declarações Locais
    clock Resp_F, Resp_CA;
    // Parâmetros
    const int i, const int h, const int DM, const int Cs, const int Cc, const int j

// Template do Autômato Atuador
    // Declarações Locais
    clock Comp_A;
    // Parâmetros
```

```
    const int f, const int Ca, const int j

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock Com;
    // Parâmetros
    const int i, const int Cmmax, const int Cmmin, const int TIFS

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n] ID=0;
// Declarações do Sistema
    SC1 = SensorControlador(const int i, const int h, const int DM, const int Cs,
    const int Cc, const int j);
    A1 = Atuador(const int f, const int Ca, const int j);
    Tmca1 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
    const int TIFS);
    SC2 = SensorControlador(const int i, const int h, const int DM, const int Cs,
    const int Cc, const int j);
    A2 = Atuador(const int f, const int Ca, const int j);
    Tmca2 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
        const int TIFS);
    SC3 = SensorControlador(const int i, const int h, const int DM, const int Cs,
    const int Cc, const int j);
    A3 = Atuador(const int f, const int Ca, const int j);
    Tmca4 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
        const int TIFS);
    A = Arbitragem();

    system
    SC1, A1, Tmca1,
    SC2, A2, Tmca2,
    SC3, A3, Tmca3,
    A;
```

Listing D.1: Código UPPAAL do modelo em autômatos temporizados de NCS CAN na arquitetura SC-A com *offset* dinâmico, relógios e barramento ideais.

D.2 Código do Modelo do NCS CAN na Arquitetura S-CA com *Offset* Dinâmico, Relógios e Barramento Ideais

O código UPPAAL do modelo de NCS CAN na arquitetura S-CA com *offset* dinâmico, relógios e barramento Ideais para três malhas de controle e taxa de transmissão de 500 Kbits/s ($t_{bit} = 2\ \mu$s) é apresentado na Listagem D.2. Os valores dos atributos devem ser definidos nas declarações do sistema. Todas as constantes e variáveis estão em microsegundos

```
// Declarações Globais do Modelo
    const int n = 6; // Número de mensagens na rede
    const int M = 3; // Três malhas de controle
    int [0,1] Arb[n];
    const int tbit = 2; // Tempo de bit em microsegundos
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Fim[M], Sucess_Arb[n], Ack[n];

// Template do Autômato Sensor
    // Declarações Locais
    clock Resp_F, Resp_SC;
    // Parâmetros
    const int i, const int h, const int DM, const int Cs, const int j

// Template do Autômato Controlador Atuador
    // Declarações Locais
    clock Comp_CA;
    // Parâmetros
    const int f, const int Cc, const int Ca, const int j

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock Com;
    // Parâmetros
    const int i, const int Cmmax, const int Cmmin, const int TIFS

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n] ID=0;
// Declarações do Sistema
    S1 = Sensor(const int i, const int h, const int DM, const int Cs, const int j );
    CA1 = ControladorAtuador(const int f, const int Cc, const int Ca, const int j );
    Tmca1 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
```

```
const int  TIFS);
    S2 = Sensor(const int i,  const int  h,  const int  DM, const int Cs, const int  j );
    CA2 = ControladorAtuador(const int f, const int Cc, const int  Ca, const int  j );
    Tmca2 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
const int  TIFS);
    S2 = Sensor(const int i,  const int  h,  const int  DM, const int Cs, const int  j );
    CA2 = ControladorAtuador(const int f, const int Cc, const int  Ca, const int  j );
    Tmca2 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
const int  TIFS);
    A = Arbitragem();

    system
    S1, CA1, Tmca1,
    S2, CA2, Tmca2,
    S3, CA3, Tmca3,
    A;
```

Listing D.2: Código UPPAAL do modelo em autômatos temporizados de NCS CAN na arquitetura S-CA com *offset* dinâmico, relógios e barramento ideais.

D.3 Código do Modelo do NCS CAN na Arquitetura S-C-A com *Offset* Dinâmico, Relógios e Barramento Ideais

O código UPPAAL do modelo de NCS CAN na arquitetura S-C-A com *offset* dinâmico, relógios e barramento Ideais para três malhas de controle e taxa de transmissão de 500 Kbits/s ($t_{bit} = 2\ \mu$s) é apresentado na Listagem D.3. Os valores dos atributos devem ser definidos nas declarações do sistema. Todas as constantes e variáveis estão em microsegundos

```
// Declarações Globais do Modelo
    const int  n = 6;  // Número de mensagens na rede
    const int  M = 3; // Três  malhas de controle
    int [0,1]   Arb[n];
    const int  tbit   = 2;  // Tempo de bit em microsegundos
    broadcast chan RxMsg[n], ReqTxMsg[n], TxMsg, Fim[M], Sucess_Arb[n], Ack[n];

// Template do Autômato Sensor
    // Declarações Locais
    clock  Resp_F, Resp_SC;
    // Parâmetros
```

```
      const int i, const int h, const int DM, const int Cs, const int j

// Template do Autômato Controlador
    // Declarações Locais
    clock Comp_C;
    // Parâmetros
     const int i, const int f, const int Cc

// Template do Autômato Atuador
    // Declarações Locais
    clock Comp_A;
    // Parâmetros
     const int f, const int Ca, const int j

// Template do Autômato Controlador CAN
    // Declarações Locais
    clock Com;
    // Parâmetros
    const int i, const int Cmmax, const int Cmmin, const int TIFS

// Template do Autômato Arbitragem
    // Declarações Locais
    int [0, n] ID=0;
// Declarações do Sistema
    S1 = Sensor(const int i, const int h, const int DM, const int Cs, const int j);
    C1 = Controlador(const int i, const int f, const int Cc);
    A1 = Atuador(const int f, const int Ca, const int j);
    Tmsc1 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
 const int TIFS);
    Tmca1 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
 const int TIFS);
    S2 = Sensor(const int i, const int h, const int DM, const int Cs, const int j);
    C2 = Controlador(const int i, const int f, const int Cc);
    A2 = Atuador(const int f, const int Ca, const int j);
    Tmsc2 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
const int TIFS);
    Tmca2 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
const int TIFS);
    S3 = Sensor(const int i, const int h, const int DM, const int Cs, const int j);
    C3 = Controlador(const int i, const int f, const int Cc);
```

```
    A3 = Atuador(const int f,  const int  Ca,  const int  j );
    Tmsc3 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
 const int  TIFS);
    Tmca3 = ControladorCAN (const int i, const int Cmmax, const int Cmmin,
const int  TIFS);
    A = Arbitragem();

    system
    S1, C1, A1, Tmsc1, Tmca1,
    S2, C2, A2, Tmsc2, Tmca2,
    S3, C3, A3, Tmsc3, Tmca3,
    A;
```

Listing D.3: Código UPPAAL do modelo em autômatos temporizados de NCS CAN na arquitetura S-C-A com *offset* dinâmico, relógios e barramento ideais.