UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

THIAGO JOSÉ MICHELIN

ANÁLISE DO IMPACTO DA COMUNICAÇÃO VIA REDE FLEXRAY EM SISTEMAS DE CONTROLE

Porto Alegre
2014

THIAGO JOSÉ MICHELIN

ANÁLISE DO IMPACTO DA COMUNICAÇÃO VIA REDE FLEXRAY EM SISTEMAS DE CONTROLE

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Controle e Automação

ORIENTADOR: Prof. Dr. João Manoel Gomes da Silva Jr.

CO-ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre
2014

THIAGO JOSÉ MICHELIN

ANÁLISE DO IMPACTO DA COMUNICAÇÃO VIA REDE FLEXRAY EM SISTEMAS DE CONTROLE

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. João Manoel Gomes da Silva Jr., UFRGS
Doutor pela Université Paul Sabatier – Toulouse, França

Banca Examinadora:

Prof. Dr. Alexandre Sanfelice Bazanella, UFRGS
Doutor pela Universidade Federal de Santa Catarina – Florianópolis, Brasil

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS
Doutor pela Universidade Federal de Minas Gerais – Belo Horizonte, Brasil

Prof. Dr. Jeferson Vieira Flores, PUCRS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Coordenador do PPGEE: _____
Prof. Dr. Arturo Suman Bretas

Porto Alegre, março de 2014.

DEDICATÓRIA

Para que pudesse chegar até aqui, muitos foram os momentos em que tive de me privar da companhia daqueles que sem os quais eu jamais teria alcançado este instante. Por isso, fica aqui o resultado destes últimos dois anos dedicados à minha mãe Inês, meu pai Nestor e minha irmã Natália, que mesmo muito distantes e com as cada vez mais raras visitas, estiveram sempre comigo. À minha namorada Thaís, que sempre me deu muita força e foi quem presenciou mais de perto toda essa trajetória, mesmo com meus repetidos momentos de ausência. Aos meus amigos, tanto aos de longa data, com os quais infelizmente o contato já não é tão frequente, quanto aos novos amigos que fiz em Porto Alegre e em Münster durante esse período.

AGRADECIMENTOS

Inicialmente, meus agradecimentos aos amigos Felipe Figueiredo, Raphael Gallo e a meu primo Charles, pois me acolheram logo em minha chegada a Porto Alegre. Ao meu orientador Prof. Dr. João Manoel Gomes da Silva Jr. e ao meu co-orientador Prof. Dr.-Ing Carlos Eduardo Pereira pela oportunidade de realizar este trabalho, por toda a paciência e disposição em me auxiliar e orientar, mesmo durante o período de intercâmbio, quando a diferença de horários chegou a cinco horas durante alguns meses. Ao Prof. Dr.-Ing. Bernd Hellingrath e aos colegas do grupo ERCIS em Münster, Alemanha, por me receberem e pelo período de cooperação. Aos amigos Anderson Giacomolli, Eduardo Maciel e Alessandro Schildt por toda a ajuda durante os inúmeros trabalhos. Aos amigos Thiago Regal, Eduardo Israel e André Albrecht, por toda a ajuda e amizade durante o período na Alemanha. À empresa Vector Informatik GmbH, a qual tive a oportunidade de visitar sua sede em Stuttgart, pelos equipamentos e ferramentas cedidos e, também, pelo suporte durante o último semestre. Aos funcionários do DELET, por toda a assistência recebida. Ao meu orientador de iniciação científica Milton Kayama, por me incentivar a fazer o mestrado. À CAPES, pela concessão de bolsa e oportunidade de realizar o período sanduíche. Por fim, ao *Ilex paraguariensis* (também conhecido como chimarrão), bebida tradicional do Rio Grande do Sul, a qual aprendi a apreciar e que me ajudou nas inúmeras noites em claro.

RESUMO

A importância das redes de comunicação industriais em modernos sistemas de automação e controle industriais tem aumentado significativamente nos últimos anos, devido aos avanços nas áreas de processadores e softwares embarcados, que permitem o desenvolvimento de dispositivos com elevada capacidade de processamento a custos reduzidos. Estas características também são muito importantes em sistemas automotivos, visto que existe uma tendência para a substituição de sistemas mecânicos e hidráulicos em veículos e o espaço disponível para implementação é bastante reduzido. Esta substituição passa pela elaboração de complexos algoritmos de controle, os quais, quando operam sobre uma rede de comunicação, precisam considerar explicitamente os efeitos do canal de comunicação compartilhado na dinâmica do sistema em malha fechada. Este trabalho apresenta uma análise do impacto da comunicação em rede sobre sistemas de controle. Mais especificamente, analisa-se o comportamento do protocolo Flexray, recentemente desenvolvido por um consórcio de importantes empresas e que incorpora interessantes conceitos para escalonamento de mensagens síncronas e assíncronas. No trabalho foram realizados experimentos com três diferentes tipos de controladores aplicados ao estudo de caso de uma suspensão ativa, onde o sistema tem sua malha fechada sobre a rede FlexRay.

Palavras-chave: Sistemas de controle em rede, atrasos, flexray, time-triggered, suspensão ativa.

ABSTRACT

The importance of communication networks on modern automation systems has increased significantly over the last years, mostly due to advances in embedded microprocessor and software technologies, which enable the development of devices with high processing power at reduced costs. These characteristics are very important for vehicle systems, since there is nowadays a trend to replace mechanical and hydraulic systems, and the space available for implementation is limited. This replacement requires very complex control algorithms, which, when operating on a communication network, have to consider explicitly the effects introduced by the shared communication channel on the closed loop system dynamics. This work presents an analysis of the network communication impact over control systems. More specifically, it is of interest to analyse the behavior of the FlexRay protocol, which has been recently developed by a Consortium of important companies and incorporates interesting concepts of synchronous and asynchronous message scheduling. In this work, some experiments were performed with three controllers, which were developed using different methodologies, applied to the case study of an active suspension system, where the loop is closed over the FlexRay protocol.

Keywords: Networked Control Systems, Delay, FlexRay, Time-Triggered, Active Suspension.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

A/D	Analog-Digital
BM	Bus Minus
BP	Bus Plus
BSS	Byte Start Sequence
CAN	Controller Area Network
CAPL	Communication Access Programming Language
CD	Compel Data
CIP	Common Industrial Protocol
CRC	Cyclic Redundancy Check Code
CSMA/BA	Carrier Sense Multiple Access with Bitwise Arbitration
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
D/A	Digital-Analog
DTS	Dynamic Trailing Sequence
DYN	Dynamic Segment
ECU	Electronic Control Unit
FES	Frame End Sequence
FIBEX	Fieldbus Exchange Format
FSS	Frame Start Sequence
FTDMA	Flexible Time Division Multiple Access
ISO	International Organization for Standardization
LAN	Local Area Network
LAS	Link Active Scheduler
LIN	Local Interconnect Network
LMI	Linear Matrix Inequality
LTI	Linear Time Invariant
MAC	Media Access Control

MEDL	Message Descriptor List
MT	Macrotick
MTS	Media Access Test Symbol
NCS	Networked Control System
NIT	Network Idle Time
NRZ	Non Return to Zero
OSI	Open System Interconnection
RT	Real Time
SS	Static Segment
TDMA	Time Division Multiple Access
TSS	Transmission Start Sequence
TT	Time-Triggered
TTA	Time-Triggered Architecture
TTCAN	Time-Triggered CAN
TTEthernet	Time-Triggered Ethernet
TTP	Time-Triggered Protocol
WCRT	Worst Case Response Time
WUDOP	Wakeup During Operation Pattern

LISTA DE SÍMBOLOS

$C(s)$	Representação do controlador
$G_{wz}(s)$	Função de transferência da entrada w para saída z
M_p	Máximo sobrepasso percentual
$P(s)$	Representação da planta do sistema
T_{bit}	Tempo de bit
T_{delay}	Atraso total em um laço de controle
T_{pos}	Tempo de pós-processamento
T_{pre}	Tempo de pré-processamento
T_{prop}	Tempo de propagação
T_{tx}	Tempo de transmissão no canal de comunicação
T_{wait}	Tempo de espera
b_s	Constante de amortecimento da suspensão
f_a	Força aplicada pelo atuador
k_s	Constante da mola da suspensão
k_t	Constante de mola do modelo do pneu
m_s	Massa do corpo do automóvel
m_{us}	Massa do conjunto suspensão, rodas e pneu
t_d	Tempo de atraso
t_p	Tempo de pico
t_r	Tempo de subida
t_s	Tempo de acomodação
$u(t)$	Entrada de controle
x	Vetor de estados
x_{def}	Deflexão da suspensão
x_r	Desnível provocado pelo trajeto (solo)
x_s	Posição vertical do corpo do veículo

x_{us}	Posição vertical do conjunto da suspensão
$w(t)$	Entrada de perturbação
$y(t)$	Saída de medição
$y_{ss}(t)$	Resposta em regime permanente
$y_{tr}(t)$	Resposta transitória
$z(t)$	Saída de performance
γ_{∞}	Valor da norma H_{∞}
$\tau(t)$	Atraso variante no tempo
τ_k^c	Atraso de processamento do controlador no instante k
τ_k^{ca}	Atraso do controlador para o atuador no instante k
τ_{ee}	End-to-End Delay
τ_k^{sc}	Atraso do sensor para o controlador no instante k

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização do Trabalho

Sistemas com componentes espacialmente distribuídos estão em uso há várias décadas. Exemplos deste tipo de arquitetura incluem sistemas de controle em processos químicos, refinarias, usinas de geração de energia e aeronaves. No passado, os componentes destes sistemas eram interligados por conexões físicas ponto-a-ponto individuais, onde um controlador central era responsável por monitorar os processos e calcular as leis de controle. A diferença para o cenário encontrado atualmente é o meio de comunicação utilizado. Graças aos avanços na tecnologia de microprocessadores, é possível colocar uma unidade com poder de processamento em um dispositivo remoto e, assim, a informação pode ser transmitida de forma confiável (ANTSAKLIS; BAILLIEUL, 2007). As grandes vantagens deste tipo de arquitetura estão na maior flexibilidade oferecida, redução de custos com cabeamento, manutenção e maior confiabilidade das informações transmitidas (WALSH; YE, 2001). Por outro lado, a comunicação por um canal compartilhado introduz efeitos indesejados do ponto de vista de controle. Dentre estes, podem-se mencionar principalmente os atrasos na transmissão de dados, variações nos tempos de amostragem (*jitter*), perdas de pacotes e quantização. Dependendo da magnitude destes efeitos, um sistema pode sofrer severa perda de desempenho ou, até mesmo, tornar-se instável. Desta forma, os efeitos introduzidos pela comunicação devem ser tratados explicitamente pelos algoritmos de controle. Os estudos sobre sistemas de controle em rede situam-se na intersecção entre as teorias de controle e comunicação (HESPANHA; NAGHSHTABRIZI; XU, 2007) e, de acordo com Murray, *et al.* (2003), é identificado como a direção a qual as aplicações de controle seguirão.

A Figura 1 ilustra um exemplo típico deste tipo de arquitetura, onde o laço de controle do sistema formado por controlador, sensor, atuador e processo físico é fechado por meio de uma rede. Como pode-se notar, o meio de comunicação é compartilhado com outras aplicações, as quais podem ou não realizar tarefas de controle.

No setor automotivo, particularmente, as vantagens oferecidas pela comunicação compartilhada são importantes, sobretudo, em relação aos custos e espaço disponível para implementação (ZHANG; GAO; KAYNAK, 2013). As primeiras unidades de controle eletrônico (*ECU – Electronic Control Unit*) operavam de forma independente e isolada. Entretanto, rapidamente, ficou evidente para as fabricantes que a coordenação e colaboração de várias ECUs oferecia a possibilidade de expandir e melhorar as funcionalidades de um veículo. Nesta direção, no início dos anos 1980, a empresa alemã *Robert Bosch GmbH* iniciou um estudo sobre dispositivos coordenados através de uma rede, a fim de controlar diversas funções internas a um veículo. Estes estudos traduziram-se, posteriormente, na elaboração do protocolo CAN (*Controller Area Network*) (BAILLIEUL; ANTSAKLIS,

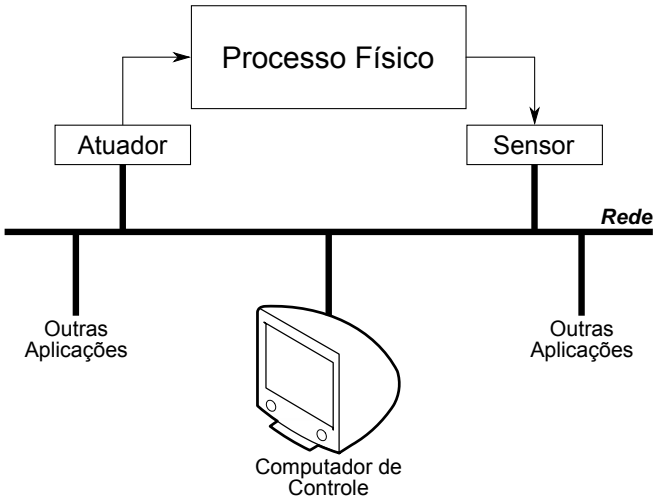

Figura 1: Sistema de controle distribuído com sensor, controlador e atuador. A rede de comunicação é compartilhada com outras aplicações. Baseado em (NILSSON, 1998).

2007). Devido às suas vantagens, como robustez e flexibilidade, o CAN é o protocolo de comunicação mais utilizado para redes automotivas atualmente (PARK; SUNWOO, 2011).

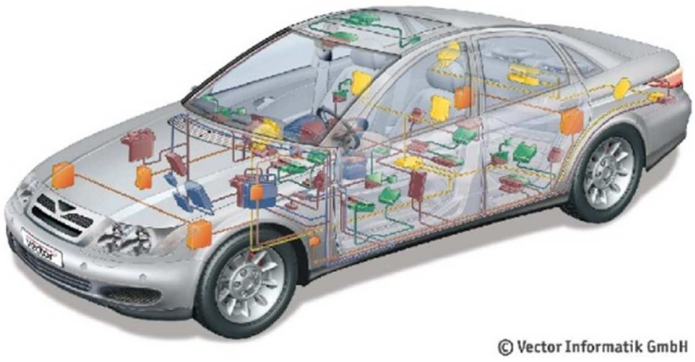

Figura 2: Exemplo de uma rede automotiva com diferentes tipos de barramento (VECTOR INFORMATIK GMBH, 2013a).

Com o aumento do número de sistemas embarcados, a comunicação através das redes automotivas aumentou consideravelmente e ficou evidente que o protocolo CAN não seria suficiente para atender aos requisitos exigidos por aplicações futuras no médio prazo. Devido às suas taxas de transmissão reduzidas, baixas garantias de determinismo e falta de suporte de tolerância à falhas, o protocolo CAN não é indicado para aplicações de controle avançado, em particular aquelas conhecidas por "*X-by-wire*" (PARK; SUNWOO, 2011).

Enquanto aplicações de controle e de assistência ao motorista necessitam de altas taxas de transmissão de dados e de comportamento determinístico, no campo de aplicações de conveniência, as exigências com relação à tráfego de dados e tempo de comunicação não são tão restritivas. Devido aos diversos novos tipos de tráfego de mensagens em uma rede veicular, atualmente, muitos protocolos de comunicação foram desenvolvidos especificamente para cada tipo de aplicação (VECTOR INFORMATIK GMBH, 2014a).

Dentre estes protocolos, no início dos anos 2000, um grupo de montadoras e suas em-

presas fornecedoras fundaram o *FlexRay Consortium*, com o objetivo de desenvolver um sistema de comunicação que pudesse oferecer suporte para as aplicações automobilísticas futuras. Entre os requisitos para o novo padrão, encontravam-se alta taxa de transmissão de dados, tolerância a falhas e determinismo (ELMENREICH, 2008).

O processo de especificação do protocolo FlexRay foi concluído em 2009, ano em que o *FlexRay Consortium* foi desfeito, e as especificações 3.0 do protocolo foram entregues à ISO (*International Organization for Standardization*) para padronização. Muitos autores apontam o protocolo como o padrão que será adotado pela indústria automotiva em aplicações futuras. Este padrão tornou-se tão atraente, que alguns estudos indicam a possibilidade de sua adoção em outros domínios, como na indústria aeronáutica (HELLER; REICHEL, 2009) e em processos de automação industrial (SHAW; JACKMAN, 2008). Em 2006, a BMW iniciou a produção do modelo BMW X5, o qual utiliza o protocolo FlexRay para dar suporte à uma aplicação de controle eletrônico dos amortecedores, de forma a aumentar a segurança e conforto dos passageiros. Outros casos de uso desta tecnologia incluem sua utilização como *backbone* para a interligação de outros tipos de rede.

1.2 Objetivos do Trabalho

A finalidade deste trabalho é avaliar qual é o impacto causado no desempenho de sistemas dinâmicos, quando o laço de controle é fechado através de uma rede FlexRay. Outro ponto importante, é observar quais as necessidades para o correto escalonamento de mensagens do sistema de controle. O estudo de caso escolhido está relacionado ao setor automotivo, o qual é a origem do desenvolvimento deste protocolo.

1.3 Organização do Texto

Este trabalho está organizado da seguinte maneira: o capítulo 2 aborda conceitos teóricos sobre os principais assuntos tratados nesta dissertação: sistemas de controle, redes e protocolos de comunicação e sistemas de controle em rede. O capítulo 3 apresenta uma análise de trabalhos relacionados sobre metodologias e métricas para análise de desempenho de sistemas de controle em rede. O capítulo 4 detalha as principais características do protocolo de comunicação FlexRay, dando maior enfoque à camada física e à política de acesso ao meio. O capítulo 5 descreve a proposta de implementação para avaliação de desempenho do sistema oferecendo suporte a aplicações de controle, onde será apresentado o estudo de caso escolhido e discutidos os objetivos dos experimentos propostos. O capítulo 6 apresenta os passos necessários para a implementação do estudo de caso e os resultados obtidos através dos experimentos. Por fim, o capítulo 7 as considerações finais da dissertação e discussões sobre possíveis caminhos a seguir em trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de Controle

Atualmente, as teorias de controle mais utilizadas são a clássica, a de controle moderno e a de controle robusto (OGATA, 2010). A primeira é fortemente baseada na análise frequêncial de sistemas, avaliando suas condições de estabilidade através de descrições entrada-saída (funções de transferência). A segunda, por sua vez, é baseada na análise temporal e em modelos matemáticos, obtidos através de equações diferenciais, utilizando o conceito de estados para descrever o comportamento de sistemas. Por fim, a terceira foi desenvolvida para a concepção de sistemas de controle que consideram as diferenças entre um sistema físico e seu modelo, incorporando aspectos das análises frequêncial e temporal (OGATA, 2010).

2.1.1 Representação de Sistemas por Função de Transferência

Na teoria de controle, funções de transferência são comumente utilizadas para caracterizar as relações entrada-saída de componentes ou sistemas que podem ser descritos por equações diferenciais lineares e invariantes no tempo (OGATA, 2010).

A função de transferência de um sistema linear e invariante no tempo (LTI – *Linear Time-Invariant*) é definida como a transformada de laplace da resposta impulsiva do sistema ou, equivalentemente, a razão entre a transformada de laplace da saída pela transformada de laplace da entrada, sob a suposição de que todas as condições iniciais são nulas. Considerando o sistema LTI definido pelas equações diferenciais em (1), onde y e x são a saída e a entrada do sistema, respectivamente, a função de transferência é dada por (2) (OGATA, 2010).

$$a_0 \overset{(n)}{y} + a_1 \overset{(n-1)}{y} + \cdots + a_{n-1}\dot{y} + a_n y = b_0 \overset{(m)}{x} + b_1 \overset{(m-1)}{x} + \cdots + b_{m-1}\dot{x} + b_m x, \quad (n \geq m) \tag{1}$$

$$\begin{aligned} \text{Função de Transferência} &= G(s) = \left.\frac{\mathscr{L}[\text{saída}]}{\mathscr{L}[\text{entrada}]}\right|_{\text{condições iniciais nulas}} \\ &= \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1}s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n} \end{aligned} \tag{2}$$

2.1.2 Representação por Variáveis de Estado

Para que um processo possa ser representado por variáveis de estado, é necessário, inicialmente, ter o conhecimento de todos os fenômenos físicos que regem a dinâmica do

sistema. Este procedimento de modelagem resulta em um conjunto de equações diferenciais lineares ou não-lineares, que podem ou não ter parâmetros variantes no tempo.

O estado de um sistema dinâmico é o menor conjunto de variáveis de forma que o conhecimento destas variáveis no instante $t = t_0$, juntamente com o conhecimento da variável de entrada para $t \geq t_0$, caracteriza completamente o comportamento do sistema para qualquer instante de tempo $t \geq t_0$ (OGATA, 2010).

Para sistemas lineares e invariantes no tempo, a representação por variáveis de estado de um processo é dada pelas equações (3) e (4), onde x representa o vetor de estados, u a variável de entrada e y a variável de saída.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3}$$
$$y = Cx(t) + Du(t) \tag{4}$$

Na equação (3), A é chamada de matriz de estados e B de matriz de entrada. Na equação (4), C é a matriz de saída e D a matriz de transmissão direta.

2.1.3 Relação entre Funções Transferência e Variáveis de Estado

A partir da representação de um sistema no espaço de estados, conforme as equações (3) e (4), é sempre possível obter sua função de transferência, conforme equação (5). As transformadas de laplace das equações (3) e (4) são dadas por (6) e (7).

$$G(s) = \frac{Y(s)}{U(s)} \tag{5}$$

$$sX(s) - x(0) = AX(s) + BU(s) \tag{6}$$
$$Y(s) = CX(s) + DU(s) \tag{7}$$

Uma vez que a função de transferência pode ser interpretada como a razão entre a transformada de laplace da saída e a transformada de laplace da entrada quando as condições iniciais são nulas, tem-se que $x(0) = 0$. Isolando o termo $X(s)$ em (6), obtém-se (8).

$$sX(s) - AX(s) = BU(s) \;\Rightarrow\; (sI - A)X(s) = BU(s) \Rightarrow\; X(s) = (sI - A)^{-1}BU(s) \tag{8}$$

Substituindo o resultado de (8) em (7), encontra-se (9).

$$Y(s) = [C(sI - A)^{-1}B + D]U(s) \tag{9}$$

Comparando a equação (9) com a equação (5), observa-se que a função de transferência em função das matrizes A, B, C e D é dada por (10).

$$G(s) \;=\; C(sI - A)^{-1}B + D \tag{10}$$

2.1.4 Métricas de Desempenho de Sistemas de Controle

Sistemas de controle são projetados para realizar tarefas específicas, as quais são normalmente descritas através de determinados critérios de desempenho.

2.1.4.1 Métricas Clássicas

A resposta temporal de um sistema de controle é constituída por duas partes: a resposta transitória ($y_{tr}(t)$) e a resposta em regime permanente ($y_{ss}(t)$), e pode ser escrita conforme (11).

$$y(t) = y_{tr}(t) + y_{ss}(t) \tag{11}$$

Para testar se o sistema atende aos requisitos de desempenho estabelecidos, comumente, utilizam-se sinais de teste de entrada dos tipos degrau, rampa, aceleração, impulso e senóides. As métricas clássicas consideradas para avaliação de desempenho, considerando a resposta ao degrau, são as seguintes (OGATA, 2010):

- Tempo de atraso (t_d): tempo necessário para que a saída do sistema atinja, pela primeira vez, metade de seu valor de regime permanente.
- Tempo de subida (t_r): tempo necessário para a resposta do sistema se elevar de 10% a 90%, 5% a 95%, ou 0% a 100% de seu valor de regime permanente.
- Tempo de pico (t_p): tempo necessário para a resposta atingir o primeiro pico do sobrepasso.
- Máximo sobrepasso (percentual) (M_p): valor máximo alcançado pela resposta do sistema, considerando como referência seu valor final. O máximo sobrepasso é definido por (12).

$$M_p(\%) = \frac{y(t_p) - y(\infty)}{y(\infty)} \tag{12}$$

- Tempo de acomodação (t_s): tempo necessário para que a saída do sistema atinja um estado no qual esta passe a oscilar dentro de uma faixa (percentual) de seu valor final, normalmente de 2% ou 5%, dependendo dos objetivos de controle.

A Figura 3 ilustra uma resposta típica de um sistema à uma entrada do tipo degrau unitário, evidenciando as métricas relacionadas anteriormente.

2.1.4.2 Normas de Sistemas

Uma das maneiras de mensurar se um sistema de controle consegue atingir certas especificações de desempenho é por meio da medição de energia de certos sinais de interesse. Através deste procedimento, é possível ter uma ideia do grau de influência de uma perturbação externa sobre uma variável de desempenho (TROFINO, 2000).

A Figura 4 exibe um modelo geral de um sistema em malha fechada, onde w, u, z e y são vetores que representam as entradas de perturbação, entradas de controle, saídas de desempenho e saídas de medição, respectivamente. $P(s)$ representa a planta e $C(s)$ o controlador, considerando que a estrutura da planta seja representada por uma equação de estados linear e invariante no tempo (13).

$$\begin{cases} \dot{x} &= Ax + B_w w + B_u u \\ z &= C_z x + D_w w + D_u u \\ y &= C_y x + D_y w + D_y u \end{cases} \tag{13}$$

Para este sistema, o operador *perturbação/saída de interesse* (G_{wz}) é dado por (14).

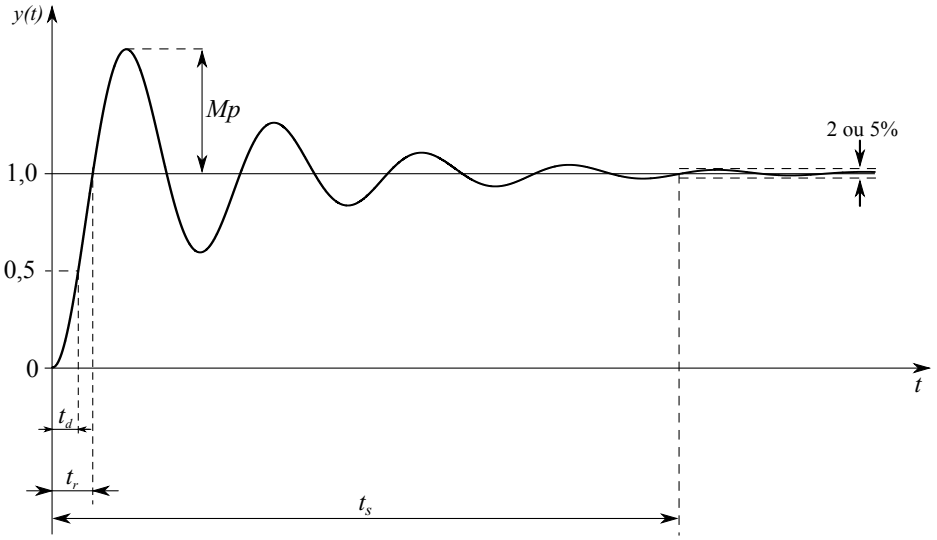

Figura 3: Resposta temporal à uma entrada do tipo degrau unitário. Baseado em (OGATA, 2010)

.

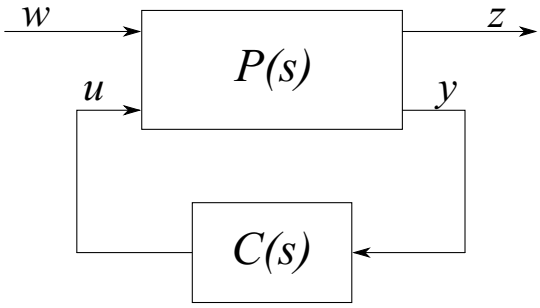

Figura 4: Modelo Geral de Performance

$$G_{wz}(s) = C_z(sI - A)^{-1}B_w + D_w \tag{14}$$

Desta forma, para medir a influência que as perturbações $w(t)$ exercem sobre as saídas de controle desejadas z, deve-se analisar qual o "tamanho" do operador G_{wz}. As duas normas mais comuns e com significado físico para quantificar o tamanho de sistemas são as normas H_2 e H_∞ (TROFINO, 2000).

2.1.4.2.1 Norma H_2

A norma H_2 do operador $G_{wz}(s)$ é definida como sendo a energia da resposta ao impulso, que é a energia do sinal de saída $z(t)$ quando aplicado um impulso de Dirac em $w(t)$. A norma H_2 do operador entrada-saída é definida por (15), onde $g(t)$ é a resposta impulsiva do sistema (13) (TROFINO, 2000).

$$\|G_{wz}(s)\|_2 = \sqrt{\int_0^\infty tr[g(t)'g(t)]dt} \tag{15}$$

2.1.4.2.2 Norma H_∞

A norma H_∞ está relacionada ao maior ganho em energia que pode existir entre uma determinada entrada do sistema e uma de suas saídas, ao longo de todo o espectro de sinais, ou seja, ela quantifica o maior acréscimo de energia que pode ocorrer entre entradas e saídas (TROFINO, 2000).

De uma maneira mais formal, a norma H_∞ do operador G_{wz} é o valor supremo entre os sinais de energia de saída e entrada, para todo w de energia limitada, conforme (16).

$$\|G_{wz}\|_\infty = \sup \frac{\|z\|_2}{\|w\|_2}, \quad \|w\|_2 \neq 0 \tag{16}$$

onde $\|.\|_2$ denota a norma $\mathscr{L}_2$ do sinal, ou seja

$$\|z\|_2 = \sqrt{\int_0^\infty z(t)'z(t)dt} \tag{17}$$

Para sistemas LTI, a norma H_∞ coincide com o máximo ganho da função de transferência $G_{wz}(s)$ para todo o espectro de frequências.

2.2 Redes de Comunicação

Muitos protocolos de comunicação surgiram nas últimas décadas. Os motivos para a introdução de redes de comunicação em sistemas industriais de controle são redução de custos com cabeamento, modularização de sistemas e maior flexibilidade para implementação e posteriores alterações. Os chamados "*Fieldbuses*" são um grupo de protocolos usualmente utilizados para a interconexão de dispositivos de campo (NILSSON, 1998). Atualmente, "*Fieldbuses*" são padronizados e considerados os sistemas de comunicação mais importantes presentes em aplicações comerciais de controle (PEREIRA; NEUMANN, 2009).

Redes de comunicação são elaboradas baseadas em um conjunto de protocolos, divididos em camadas, como o modelo ISO/OSI (*Open System Interconnection*) (TANENBAUM; WETHERALL, 1999), sendo responsabilidade deste conjunto garantir o suporte necessário à aplicação construída sobre o mesmo. Protocolos de comunicação utilizados em automação de processos e controle, normalmente, implementam apenas as primeiras duas camadas do modelo OSI: física e de enlace, deixando as demais para a aplicação.

2.2.1 Paradigmas *Time-Triggered* e *Event-Triggered*

A transmissão de mensagens entre dispositivos de uma rede de comunicação pode ser classificada de acordo com a característica da informação: periódica ou esporádica (XIA; SUN, 2008). Para estas diferentes características, distintas estratégias de acesso ao meio são utilizadas. Com relação às suas políticas de controle de acesso ao meio (*MAC – Media Access Control*), protocolos de comunicação podem ser classificados como *time-triggered* ou *event-triggered*. Cada uma destas abordagens apresenta vantagens e desvantagens, sendo melhor utilizadas em tipos de aplicação específicos.

2.2.1.1 Paradigma Time-Triggered

Em sistemas *time-triggered*, a competição entre componentes de rede pelo uso do meio de transmissão é evitada através de políticas de escalonamento baseadas no método TDMA (*Time Division Multiple Access*). Algumas vantagens deste método consistem em

garantias temporais, baixo *jitter* para transmissão de mensagens e comportamento previsível, simplificando análise e projeto de sistemas distribuídos. Para que as garantias temporais possam ser cumpridas, é necessário que todos os dispositivos da rede estejam sincronizados em uma base de tempo global. Por outro lado, é extremamente inflexível a alterações na topologia da rede ou ao escalonamento de mensagens, visto que, durante a fase de projeto da rede deve-se ter conhecimento sobre as características de todas as mensagens que serão trocadas pelos componentes. Com exceção do caso quando todos os componentes possuem dados para transmitir em todos os instantes, não utiliza completamente o recurso disponível.

2.2.1.2 Paradigma Event-Triggered

Em comunicações *event-triggered*, a requisição de acesso ao meio ocorre sob demanda, ou seja, apenas no momento em que se faz necessário transmitir alguma informação. Neste instante, o dispositivo em questão iniciará uma tentativa de acesso ao meio físico. Esta abordagem provê maior flexibilidade ao sistema, uma vez que a adição ou remoção de um componente da rede, ou mesmo a alteração de seu comportamento, não irá influenciar a maneira como outros componentes tentam acessar a rede. Entretanto, neste caso, ocorre uma disputa pelo acesso ao meio entre dois (ou mais) componentes que necessitem transmitir mensagens em instantes de tempo muito próximos, o que pode causar um aumento considerável no tempo de transmissão da mensagem. Exemplos deste tipo de política são encontrados em protocolos como CAN e Ethernet. O protocolo CAN utiliza como método de acesso ao meio a estratégia CSMA/BA (*Carrier Sense Multiple Access with Bitwise Arbitration*), enquanto o padrão Ethernet utiliza o método CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*).

2.2.2 Protocolos de Comunicação

2.2.2.1 CAN

O protocolo CAN (*Controller Area Network*) (CAN, 1992) foi desenvolvido pela empresa alemã Bosch, no início dos anos 1980, com o intuito inicial de prover comunicação entre ECUs em automóveis. Atualmente, este protocolo é utilizado em diversos outros segmentos, como automação industrial e robótica. É um protocolo baseado na transmissão de mensagens, onde cada uma possui um identificador, o qual também é responsável por indicar a prioridade da mensagem. Através deste procedimento, conflitos durante o acesso ao meio de transmissão são resolvidos. A mensagem com o menor identificador possui maior prioridade sobre as demais. Quando dois dispositivos acessam o barramento ao mesmo tempo, aquele que possui menor prioridade identifica a colisão e interrompe imediatamente sua transmissão, aguardando pelo momento em que o barramento estiver disponível novamente. A camada física do protocolo é definida pelos padrões ISO 11898-2 e ISO 11898-3. A transmissão de bits utiliza o padrão NRZ (*Non-Return to Zero*), e a taxa de transferência de dados pode ser de até 1 $Mbit/s$, quando a maior distância entre dispositivos na rede não superar 40 m (pelo padrão ISO 11898-3). Para distâncias maiores, a taxa de transmissão é reduzida (por exemplo, 125 $kbit/s$ para distâncias maiores que 50 m).

2.2.2.2 TTCAN

O protocolo TTCAN (*Time-Triggered CAN*) (CAN in Automation (CiA), 2013), é uma extensão do protocolo CAN, sendo que seu principal objetivo é evitar o *jitter* de

acesso ao meio através da utilização da técnica de divisão do tempo. O padrão ISO 18898 foi extendido para dar suporte à comunicação *time-triggered*, sob o código ISO 18898-4. TTCAN é baseado em uma comunicação periódica, ditada por mensagens de referência originadas por um dispositivo mestre: o período entre duas mensagens de referência define o ciclo de comunicação. Este ciclo é denominado "Ciclo Básico", sendo composto por algumas janelas de tempo de tipos e tamanhos diferentes. Dentre estas, a chamada janela exclusiva é utilizada para transmissão de mensagens periódicas. Para mensagens esporádicas, ou não periódicas, as janelas de arbitragem são utilizadas. Neste caso, quando múltiplos dispositivos competem pelo acesso à rede, o mecanismo de arbitragem por prioridades padrão do protocolo CAN é utilizado. O último tipo de janela é a chamada janela de tempo livre, a qual é reservada para extensões futuras da rede, sendo possível alterar esta janela para uma do tipo exclusiva ou de arbitragem, de acordo com a necessidade (FÜHRER et al., 2000).

2.2.2.3 Foundation Fieldbus

O protocolo *Foundation Fieldbus* (FIELDBUS FOUNDATION, 2013) é um padrão aberto serial, digital, desenvolvido e administrado pela *Fieldbus Foundation*. O padrão $H1$ foi projetado para controle de processos, substituindo os sinais de comunicação analógicos de $4-20$ *mA* sem necessidade de substituição dos fios, provendo comunicação e, também, possível sinal de alimentação para dispositivos de campo, como sensores e atuadores. Neste padrão, a taxa máxima de transmissão de dados é de $31,25$ $kbit/s$. O comprimento máximo da rede é dependente da taxa de transmissão, tipo de cabo, dimensão dos fios e escolha de alimentação do barramento. O acesso ao meio físico é coordenado por um escalonador chamado *Link Active Scheduler* (LAS). O LAS possui uma lista com os instantes de transmissão de todas as mensagens (de todos os componentes da rede) que precisam ser periodicamente transmitidas. Durante a operação do sistema, quando chega o instante em que um dispositivo deve enviar uma mensagem dentro do ciclo de comunicação, o LAS envia uma mensagem CD (*Compel Data*) para o mesmo. Após o recebimento da mensagem CD, o dispositivo "publica" (*publish*) uma mensagem no barramento. Todo componente configurado para receber essa mensagem é chamado "*subscriber*". Cada dispositivo recebe do LAS, ainda, a chance de transmitir mensagens esporádicas. Para tanto, o LAS envia a um dispositivo uma mensagem "*pass token*", concedendo um período de tempo para transmissão de mensagens assíncronas (SMAR Equipamentos Industriais, 2013).

2.2.2.4 TTP/C

O TTP (*Time-Triggered Protocol*) (TTA-GROUP, 2013) foi desenvolvido originalmente na Universidade de Tecnologia de Viena, na Áustria, no início dos anos 1980. É um padrão baseado na família de protocolos originados da arquitetura "time-triggered" (TTA). O TTP/C tem por objetivo a interconexão de componentes para aplicações críticas de segurança em ambientes automotivos e aeronáuticos (ELMENREICH, 2008). O controle de acesso ao meio do TTP é baseado no TDMA, onde cada componente (nó) da rede tem um período de tempo limitado para enviar dados através do barramento. A comunicação dentro de um *cluster* é baseada em *TDMA rounds*, ou ciclos, que se repetem em um determinado período de tempo. Neste tipo de arquitetura, todas as informações relacionadas à configuração da rede, receptores e transmissores de mensagens, e suas propriedades são definidas em fase de projeto, tornando o comportamento do sistema altamente previsível. Os atributos das mensagens transmitidas pelos componentes da rede são descritas

através de uma estrutura de dados estática, denominada *Message Descriptor List (MEDL)*. Cada canal que compõe o meio físico é protegido por um dispositivo chamado guardião do barramento (*bus guardian*), o qual impede o acesso ao barramento de dispositivos que tentem transmitir mensagens fora de seus respectivos *time-slots*, devido a erros ou a algum defeito do nó em questão (ATAIDE; PEREIRA, 2012). A taxa de transmissão máxima do TTP é de 25 $Mbit/s$, podendo ser dobrada quando diferentes mensagens são enviadas em cada canal. Cada quadro do TTP, pode conter de 2 a 240 $bytes$.

2.2.2.5 LIN

O protocolo LIN (*Local Interconnect Network*) foi desenvolvido para criar um padrão para a comunicação de baixo custo entre dispositivos discretos automotivos, como por exemplo, controle de posição dos bancos de passageiros, sistemas de travamento de portas, sensores de chuva e controle de luzes internas (ATAIDE; PEREIRA, 2012). Redes automotivas modernas utilizam uma combinação entre LIN (para aplicações de baixo custo), CAN (para aplicações do conjunto "*powertrain*") e FlexRay (para aplicações de controle que exigem alta taxa de transferência). O LIN é um protocolo que implementa o paradigma mestre-escravo, onde o mestre realiza o chamado "*polling*" (consulta) aos escravos e, desta forma, controla a comunicação no barramento, visto que, cada escravo só pode acessar o meio físico com autorização do mestre. Utilizando este procedimento, o protocolo não necessita de políticas de arbitragem de acesso ao meio, conseguindo, ainda, garantir tempo de latência fixa para cada mensagem. Seu meio físico consiste de apenas um canal de comunicação, com taxa máxima de transmissão de dados de 20 $Kbit/s$. Um mestre pode controlar até 16 escravos, em uma distância de até 40 m.

2.2.2.6 FlexRay

O protocolo FlexRay foi o escolhido para este trabalho, e será detalhado, posteriormente, no capítulo 4.

2.2.2.7 Ethernet

Ethernet é o padrão mais utilizado para redes LAN (*Local Area Network*). Suas taxas de transmissão variam de 1 $Mbit/s$ a 100 $Gbit/s$, enquanto o meio físico de transmissão compreende cabos coaxiais, par trançado e fibra ótica. Atualmente, maiores taxas de transmissão estão em desenvolvimento. Como método de acesso ao meio, emprega o CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) padronizado sob o código IEEE 802.3 (XIA; SUN, 2008). Com esta política de arbitragem, qualquer dispositivo pode enviar pacotes de dados a qualquer instante, caso o meio de comunicação esteja disponível. Enquanto a transmissão ocorre, o dispositivo "escuta" o barramento com o intuito de detectar uma possível colisão. Em caso de colisão, o dispositivo interrompe a transmissão e aguarda por um tempo aleatório para iniciar a transmissão novamente. A grande vantagem do CSMA/CD é sua facilidade de implementação, o que contribuiu para a popularização do padrão Ethernet. Ao contrário de outros protocolos, Ethernet não possui um mestre de comunicação. Cada mensagem possui campos de dados que contém os endereços (*MAC address*) de seu transmissor e seu receptor. Desta maneira, o número de componentes em uma rede é limitado pela quantidade de endereços disponíveis na rede.

2.3 Sistemas de Controle em Rede

Sistemas de controle em rede (*NCS – Networked Control Systems*) são sistemas cujos componentes estão espacialmente distribuídos. Os primeiros projetos de NCSs consideravam uma conexão física própria para cada sinal entre um dispositivo remoto e um controlador central, responsável por gerenciar toda a rede. A diferença para os projetos atuais está no novo método de comunicação utilizado. Devido aos avanços na tecnologia de microprocessadores, é possível colocar em uma unidade remota um processador de baixo custo, e, assim, a informação pode ser transmitida de maneira confiável através de uma rede compartilhada (ANTSAKLIS; BAILLIEUL, 2007)(ZHANG; BRANICKY; PHILLIPS, 2001). Este novo conceito de arquitetura apresenta algumas vantagens sobre os métodos inicialmente utilizados, como redução dos custos de implementação e manutenção, e maior flexibilidade caso as configurações da rede precisem ser alteradas (ZHANG; GAO; KAYNAK, 2013)(CLOOSTERMAN et al., 2010). A Figura 5, mostra um exemplo genérico de um NCS.

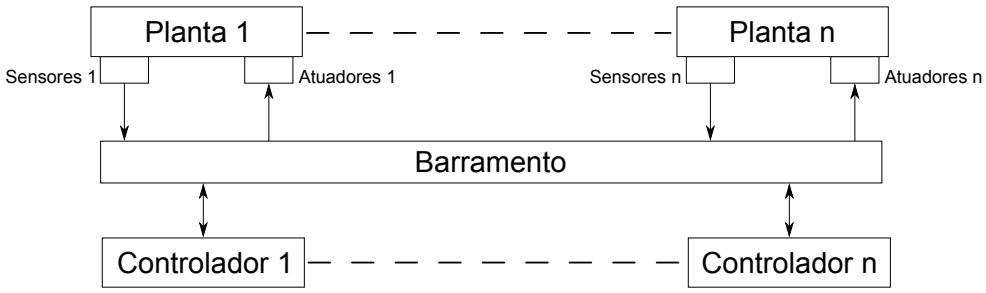

Figura 5: Conceito de um sistema de controle em rede. Baseado em (HESPANHA; NAGHSHTABRIZI; XU, 2007).

Entretanto, quando comparado aos sistemas tradicionais de controle, a introdução de um meio de comunicação compartilhado traz uma série de problemas que podem deteriorar o desempenho ou, no pior caso, instabilizar o sistema (ZHANG; GAO; KAYNAK, 2013)(ZHANG; BRANICKY; PHILLIPS, 2001). Portanto, nesta nova condição, novos métodos de modelagem e análise devem ser utilizados para trabalhar com sistemas deste tipo, uma vez que as imperfeições introduzidas pelo uso da rede devem ser explicitamente consideradas (HESPANHA; NAGHSHTABRIZI; XU, 2007).

A teoria por trás de sistemas de controle em rede encontra-se na intersecção entre as teorias de controle e comunicação. A teoria de controle está concentrada na análise de sistemas dinâmicos que são interligados através de um canal ideal, enquanto a teoria de comunicação concentra-se na transmissão de mensagens através de canais imperfeitos (HESPANHA; NAGHSHTABRIZI; XU, 2007).

Segundo Gupta e Chow (2010), NCSs podem ser classificados de duas maneiras:

- *Controle da rede*: este campo de estudo envolve pesquisas nas áreas de comunicação e redes de forma a torná-los propícios para o uso em sistemas de controle de tempo real. São citados como exemplos: controle de roteamento, redução de congestionamento, comunicação eficiente e protocolos de rede.

- *Controle sobre a rede*: este campo de estudo está mais relacionado às estratégias de controle e projetos de sistemas de controle operando sobre uma rede de comunicação compartilhada, de forma a minimizar os impactos dos efeitos introduzidos pelo uso da rede.

2.3.1 Arquiteturas de Sistemas de Controle em Rede

De acordo com Tipsuwan e Chow (2003), existem basicamente duas configurações principais para um sistema de controle em rede: *Estrutura Direta* e *Estrutura Hierárquica*.

2.3.1.1 Estrutura Direta

Na configuração *estrutura direta*, o conjunto formado pela planta, sensores e atuadores está em uma localização diferente daquela do controlador. Entretando, os dois sistemas se comunicam diretamente através de uma rede, por onde se dá o fechamento do laço de controle. A Figura 6 ilustra um exemplo deste tipo de configuração.

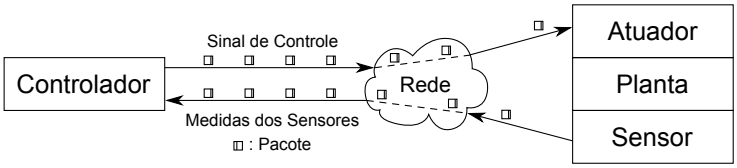

Figura 6: NCS em estrutura direta. Baseado em (TIPSUWAN; CHOW, 2003)

Como pode ser observado através da Figura 6, o controlador recebe os dados com informações da planta através de pacotes enviados pela rede. Neste ponto, o controlador irá processar o pacote recebido, calcular a nova lei de controle, construir um pacote com a informação calculada e, finalmente, enviar de volta à planta a nova lei de controle.

2.3.1.2 Estrutura Indireta

A *estrutura indireta* é mostrada pela Figura 7.

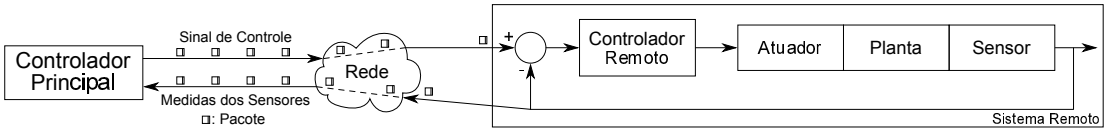

Figura 7: Estrutura indireta. Baseado em (TIPSUWAN; CHOW, 2003).

Neste tipo de estrutura, o controlador principal recebe, através da rede, medições dos sensores da planta remota, para, então, calcular um novo ponto de referência, o qual é enviado novamente através da rede para o sistema remoto. O sistema remoto, de posse dessa nova informação, irá realizar as tarefas em laço fechado localmente para atender às novas especificações enviadas pelo controlador principal. Um dos objetivos deste tipo de arquitetura é procurar reduzir o tráfego de rede, uma vez que a frequência de envio de mensagens dos sensores para o controlador principal é bem reduzida se comparada àquela do laço de controle local.

2.3.2 Descrição dos Principais Efeitos Introduzidos pela Rede

Conforme mencionado anteriormente, a introdução de um meio de comunicação compartilhado na malha de controle introduz diversos efeitos indesejados, os quais podem degradar de forma significativa o desempenho do sistema ou mesmo instabilizá-lo. Entretanto, são efeitos inerentes ao meio de comunicação utilizado e, portanto, devem ser analisados de forma a causar o menor impacto possível no sistema de controle.

2.3.2.1 *Atrasos*

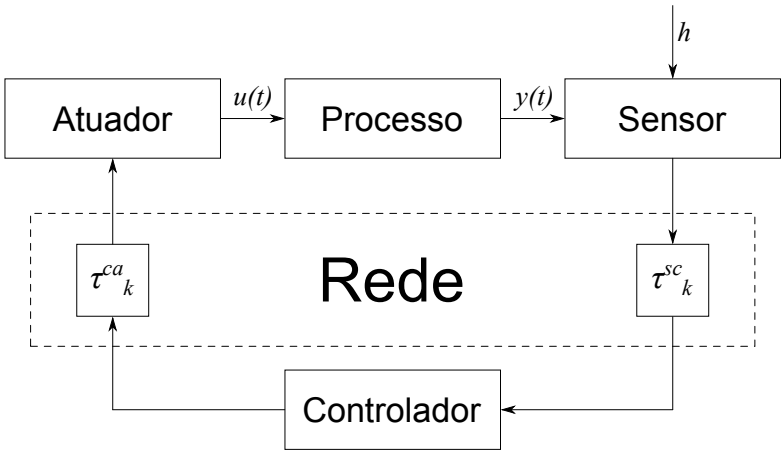

Figura 8: Atrasos. Baseado em (NILSSON, 1998).

Para melhor representar as fontes de atraso, considera-se a situação exibida pela Figura 8. Nesta configuração, sensores e atuadores comunicam-se com o controlador remoto através de uma rede.

Em Nilsson (1998), os atrasos são classificados de acordo com a direção do tráfego das mensagens: do sensor para o controlador, τ_k^{sc}, e do controlador para o atuador, τ_k^{ca}. O índice k é utilizado para indicar uma possível dependência dos atrasos com relação ao tempo. Neste trabalho é considerado também o atraso τ_k^c, devido ao tempo de processamento do controlador. Entretanto, este último pode ser incorporado em algum dos outros dois atrasos devido à rede.

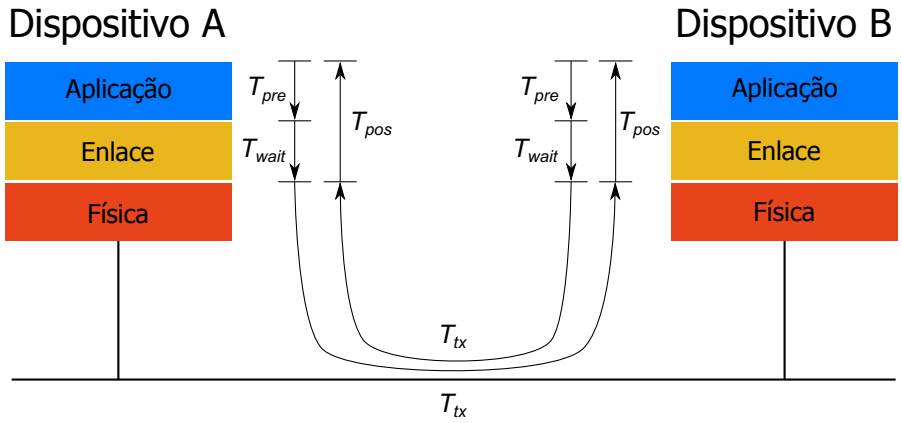

Figura 9: Análise dos atrasos. Baseado em (LIAN; MOYNE; TILBURY, 2002)

Em Lian, *et al.* (2002), o atraso total é dividido em duas partes: atraso devido à rede e atraso devido ao dispositivo. O atraso total presente em uma aplicação (T_{delay}) pode ser obtido através da equação (18). Cada componente do atraso total será descrito nas próximas seções, e são ilustrados pela Figura 9.

$$T_{delay} = T_{pre} + T_{wait} + T_{tx} + T_{post} \tag{18}$$

2.3.2.1.1 Tempo de pré-processamento (T_{pre})

O tempo de pré-processamento é o tempo necessário para o dispositivo transmissor obter dados externos (como, por exemplo, um sensor que obtém dados da planta) e transformá-los em um formato apropriado para envio pela rede (por exemplo, um quadro específico de um protocolo). Obviamente, este tempo depende de cada dispositivo, uma vez que o tempo necessário para execução de todo o procedimento depende do *hardware*. Normalmente, considera-se que este tempo é muito menor que o tempo necessário para a transmissão dos dados pelo barramento.

2.3.2.1.2 Tempo de espera (T_{wait})

O tempo de espera é o tempo necessário para que o dispositivo receba permissão para acessar o barramento e iniciar a transmissão. Este tipo de atraso é bastante dependente da estratégia de acesso ao meio utilizada pelo protocolo de comunicação.

2.3.2.1.3 Tempo de transmissão no canal de comunicação (T_{tx})

O tempo de transmissão depende apenas da taxa de transmissão, do tamanho da mensagem e da distância entre dois dispositivos. O tempo de transmissão pode ser descrito pela equação (19), onde N é a quantidade de bits em uma mensagem, T_{bit} é o tempo de bit e T_{prop} é o tempo de propagação entre dois dispositivos. Normalmente, o tempo de propagação no meio de comunicação é desprezado, devido à velocidade de propagação dos sinais no meio ($2 \cdot 10^8\ m/s$).

$$T_{tx} = N \cdot T_{bit} + T_{prop} \tag{19}$$

2.3.2.1.4 Tempo de pós-processamento (T_{pos})

O tempo de pós-processamento é o tempo necessário pelo receptor para decodificar a mensagem recebida pelo barramento e processar a nova informação disponível.

2.3.2.2 Perda de Pacotes

Perdas de pacote em uma rede podem ocorrer por diversos motivos: erros durante a transmissão de mensagens, falhas em dispositivos e colisão de pacotes (ZHANG; BRANICKY; PHILLIPS, 2001). Até mesmo uma mensagem que sofra algum atraso bastante acentuado pode ser considerada como uma perda de pacote (ZHANG; GAO; KAYNAK, 2013), uma vez que no momento em que a informação chegar ao seu destino, esta pode não ter mais valor algum. Muitos protocolos possuem mecanismos para retransmissão de mensagens perdidas, entretanto, em algumas situações, pode ser melhor desconsiderar o pacote perdido em prol de dados mais novos.

2.3.2.3 Tempo de Amostragem Variante no Tempo

Em NCSs é difícil garantir um período fixo entre amostragens de um sistema. Isto ocorre devido à falta de sincronização entre os relógios dos diversos dispositivos e também devido ao acesso à rede quando considerado um sistema com vários sensores distribuídos (ZHANG; GAO; KAYNAK, 2013).

2.3.2.4 Quantização

Erros devido à quantização estão presentes em qualquer sistema digital. Em sistemas em rede, o problema de quantização pode ser acentuado dependendo da capacidade do protocolo utilizado. Entretanto, muitos trabalhos desconsideram os efeitos provocados pela quantização, pois avaliam que a maioria dos protocolos utilizados atualmente contam com mensagens que oferecem uma quantidade de bits suficientemente grande para que este efeito possa ser desprezado. Por outro lado, em alguns cenários, este tipo de erro ainda causa muitos problemas, podendo levar um sistema à instabilidade (ZHANG; GAO; KAYNAK, 2013).

2.3.3 Efeitos dos Atrasos em Malhas de Controle

Atrasos em sistemas de controle são bastante conhecidos por degradação de desempenho. Da mesma forma, são os efeitos causados pelos atrasos devido à rede em NCSs. Para ilustrar este efeito, foi utilizado o caso de um controlador PI com atrasos no laço de controle, conforme Figura 10, baseado no estudo apresentado em (TIPSUWAN; CHOW, 2003). As funções de transferência do controlador e da planta são descritas por (20) e (21), respectivamente.

$$G_c(s) = \frac{\beta K_P(s + \frac{K_I}{K_P})}{s}, \;\; K_P = 0,1701, \;\; K_I = 0,378 \tag{20}$$

$$G_P(s) = \frac{2029,826}{(s+26,29)(s+2,296)} \tag{21}$$

Na equação (20), K_P é o ganho proporcional e K_I o ganho integral. Neste exemplo, o parâmetro β possui valor unitário, e os atrasos τ_{sc} e τ_{ca} possuem valor $\tau/2$.

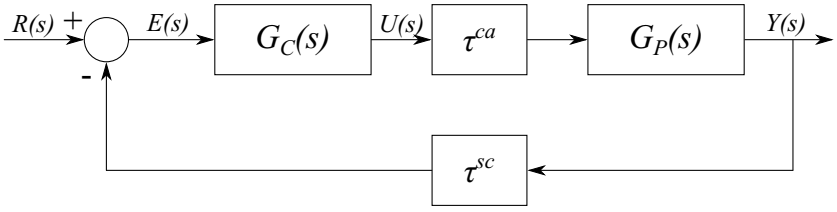

Figura 10: Exemplo de sistema em malha fechada com atrasos. Baseado em (TIPSUWAN; CHOW, 2003).

A resposta do sistema ao degrau unitário é ilustrada pela Figura 11, onde fica clara a degradação de desempenho, refletida através dos maiores sobrepasso e tempo de acomodação à medida que o atraso é incrementado.

Atrasos em um laço de controle podem, ainda, instabilizar o sistema, uma vez que reduzem sua margem de fase. Pode-se perceber pela Figura 11 que quando $\tau = 0,3150$, o sistema apresenta comportamento instável.

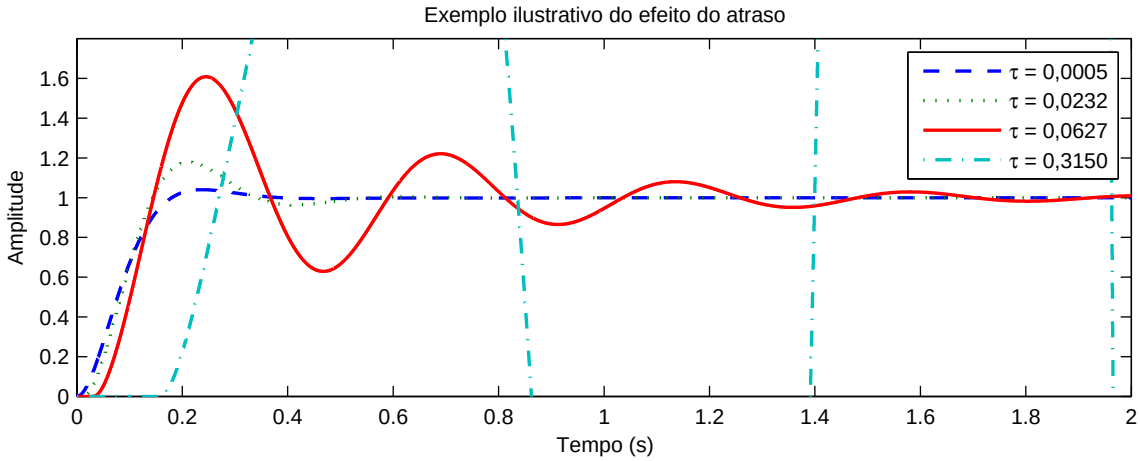

Figura 11: Ilustração da degradação de performance causada pela presença de atrasos no laço de controle. Baseado em (TIPSUWAN; CHOW, 2003).

3 ANÁLISE DO ESTADO DA ARTE

3.1 Introdução

Devido ao grande e crescente interesse na área de sistemas de controle em rede, diversos trabalhos que abordam diferentes características têm sido publicados, em especial nos últimos anos. Apesar de ser um campo multidisciplinar, envolvendo, sobretudo, as áreas de controle e comunicação, geralmente os autores focam suas atenções ou na parte de controle ou na parte de comunicação, não havendo uma integração entre os dois assuntos.

Trabalhos relacionados a sistemas de controle em rede, de um ponto de vista puramente de controle, procuram métodos matemáticos formais para a modelagem dos efeitos introduzidos pela rede. Entre estes, podem ser mencionados: intervalos de amostragem variantes no tempo, atrasos na transmissão de mensagens, perdas de pacotes e quantização. Neste caso, o foco está sobre diferentes metodologias para análises de estabilidade e condições de síntese de controladores, onde os efeitos introduzidos pela comunicação são considerados como atributos inerentes ao sistema. Em trabalhos orientados pelo ponto de vista da comunicação, comumente, são estudadas técnicas de escalonamento e de topologias de rede, de maneira a otimizar o uso do meio de comunicação (considerado como um recurso). Para tanto, são utilizadas diferentes métricas, como, por exemplo, taxa de utilização da banda disponível, pior tempo de resposta (*WCRT - Worst Case Response Time*), tempo de latência e *jitter* na transmissão de mensagens sob diferentes condições de tráfego.

Recentemente, alguns estudos começaram a abordar a questão da distância entre estas duas áreas, em uma tentativa de aproximá-las. Neste cenário, alguns trabalhos procuram estabelecer metodologias para o desenvolvimento conjunto de técnicas de controle e escalonamento, de forma a obter o melhor resultado possível dadas as especificações de desempenho de controle, necessidades temporais de cada sinal e o protocolo utilizado. Nesta situação, normalmente considera-se uma função custo que deve ser minimizada. Trabalhos específicos que relatam a simulação e/ou implementação de sistemas de controle distribuídos, utilizando protocolos específicos, também são bastante encontrados. Entretanto, nesta última categoria, normalmente, são descritas as etapas de modelagem, o conjunto de ferramentas utilizadas para a simulação e/ou implementação e, por fim, os resultados são comparados com a resposta ideal do sistema, na maioria das vezes, sendo esta uma simulação em MATLAB/Simulink. Por fim, poucos trabalhos relatam a análise de sistemas de controle em rede e desempenho de protocolos de comunicação conjuntamente.

Os trabalhos mais relevantes diretamente relacionados ao protocolo FlexRay, até o momento, são estudos que propõem algoritmos e mecanismos de forma a realizar o escalonamento de mensagens nos segmentos estático e dinâmico de forma ótima (conside-

rando principalmente a questão de utilização do barramento).

Tendo em mente o tema deste trabalho, voltado à análise do impacto da comunicação utilizando o protocolo FlexRay sobre sistemas de controle, serão analisados estudos que façam análises de métricas que tragam influência direta ao comportamento de sistemas distribuídos. Por outro lado, trabalhos com este propósito utilizando o protocolo em estudo não são muito numerosos. Desta forma, trabalhos semelhantes que façam uso de outros protocolos de comunicação serão apresentados (com preferência a estudos que possuam simulações ou implementação física do sistema para análise).

Conforme mencionado inicialmente, a área de sistemas de controle em rede possui literatura muito ampla. Apesar de não ser o objetivo "direto" deste trabalho, conforme mencionado no parágrafo anterior, mas com a intenção de apenas ilustrar outros focos de estudo na área, as seções 3.2 e 3.3 fazem menção, muito brevemente, a estudos sobre metodologias de controle para NCSs e metodologias para elaboração de um escalonamento ótimo no protocolo FlexRay, respectivamente.

3.2 Metodologias de Projeto de Controle em Rede

A diversidade de metodologias de projeto e técnicas de controle sobre rede encontradas na literatura é bastante extensa. Prova disto são os trabalhos de Tipsuwan e Chow (2003), Hespanha, Naghshtabrizi e Xu (2007), Gupta e Chow (2010) e Zhang, Gao e Kaynak (2013), os quais se complementam e trazem uma grande variedade de metodologias para modelagem, análise e síntese de controladores para sistemas de controle em rede, onde cada método proposto considera um certo conjunto dos efeitos introduzidos pelo meio de comunicação.

3.2.1 Metodologias de Projeto Conjunto de Controle e Escalonamento

Como exemplos de estudos relacionados ao chamado "*codesign*" (projeto conjunto), pode-se citar Naghshtabrizi e Hespanha (2009) e Samii, *et al.* (2011). Ambos procuram definir uma metodologia para projeto conjunto tanto do controle quanto do escalonamento de mensagens no sistema. O primeiro trabalho é baseado em condições matemáticas formais para a determinação da política de escalonamento baseada nas restrições impostas durante a etapa de síntese do controlador. No segundo trabalho, por sua vez, simulações são utilizadas para caracterização dos atrasos presentes no sistema e, de posse dessa informação, o controlador é sintetizado. Neste caso, são necessárias várias iterações deste laço para que a função custo utilizada seja minimizada. Ambos trabalhos consideram como estudo de caso o segmento dinâmico do protocolo FlexRay.

3.3 Metodologias de Escalonamento no Protocolo FlexRay

Os trabalhos presentes nesta categoria têm foco na elaboração de metodologias para o escalonamento ótimo de mensagens no protocolo FlexRay. Pode-se citar os trabalhos publicados por Schmidt e Schmidt (2009) e Hanzálek, Benes e Waraus (2011), os quais tratam da análise temporal do segmento estático do ciclo de comunicação. Trabalhos relacionados ao segmento dinâmico são encontrados em Hagiescu, *et al.* (2007) e Chokshi e Bhaduri (2010).

3.4 Métricas Temporais de Protocolos de Comunicação

A tese de doutorado de Nilsson (1998) pode ser considerada como um clássico no campo de sistemas de controle sobre rede. Dentre os assuntos abordados, existe um capítulo dedicado exclusivamente para a caracterização dos atrasos τ^{sc} e τ^{ca}, indicados pela Figura 8, presentes em redes CAN (operando com taxa de transmissão de 10 $kbit/s$) e Ethernet. Para a avaliação da rede CAN, são utilizados quatro microcontroladores (os quais contam com um procedimento para sincronização de relógios para que a base de tempo seja mantida uniforme) conectados ao mesmo barramento, conforme ilustrado pela Figura 12. Dois deles se comunicam utilizando um esquema mestre-escravo (dispositivos D_1 e D_2), enquanto os outros dois (D_3 e D_4) são utilizados apenas para alterar a condição de tráfego na rede. O mestre envia mensagens periódicas ao escravo que, logo após o recebimento, envia uma resposta ao mestre. Os instantes de transmissão e recebimento das mensagens são armazenados pelos microcontroladores. Cada dispositivo é conectado a um computador central (utilizado apenas para recolhimento e análise de dados), através de comunicação serial e as análises temporais são realizadas após o término do experimento. Foram consideradas quatro condições de tráfego na rede: (i) sem carga, (ii) uma mensagem periódica entre D_3 e D_4, (iii) várias mensagens periódicas entre D_3 e D_4 e (iv) mensagens randômicas entre D_3 e D_4. Através de histogramas e gráficos com a evolução temporal dos atrasos na comunicação, observou-se que os atrasos em uma rede CAN são variáveis no tempo e que dependem da quantidade de tráfego no barramento e, também, da distribuição de prioridades entre as mensagens. Os experimentos foram realizados com o objetivo de confirmar experimentalmente uma modelagem para os atrasos, para posterior síntese de controladores que utilizem tal modelo.

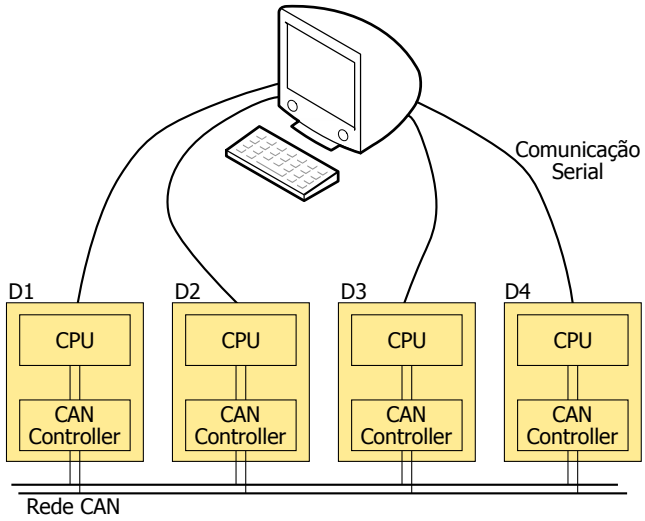

Figura 12: Configuração física utilizada em (NILSSON, 1998).

Em Lian, Moyne e Tilbury (2001) foi realizada uma comparação de três protocolos: Ethernet, ControlNet e DeviceNet. De acordo com os autores, quatro fatores afetam a disponibilidade e utilização da banda de uma rede: taxas de amostragem nas quais vários dispositivos transmitem informações através da rede, o número de elementos que necessitam de transmissão síncrona, o tamanho da mensagem e o protocolo MAC utilizado para arbitragem de acesso ao meio. De um ponto de vista de controle, os protocolos escolhidos devem garantir um tempo máximo de transmissão. Neste contexto, são realizadas simulações (utilizando um *framework* desenvolvido em MATLAB) com o objetivo de avaliar,

principalmente, o tempo médio de atraso na transmissão das mensagens em cada protocolo. Para tanto, são considerados dois cenários diferentes. No primeiro é simulada uma rede composta por dez componentes, e a transmissão das mensagens é cíclica. São elaboradas três situações que demarcam o início da transmissão: (i) todos os nós transmitem logo no início da simulação (o que causa grande competição por acesso ao meio entre todos os componentes), (ii) transmissão das mensagens em algum instante inicial aleatório e, por último, é realizado um (iii) escalonamento para o envio de mensagens. O segundo cenário avalia novamente os três instantes para início da transmissão de mensagens, entretanto, é considerado um cenário de avaliação conhecido por *SAE Benchmark* (TINDELL; BURNS; WELLINGS, 1995), no qual uma rede é composta por 53 nós.

Em Tian, Bai e Chen (2008), inicialmente é realizada uma avaliação quantitativa, através de equações matemáticas sobre os atrasos de transmissão e sobre o tempo de espera para acesso ao meio quando o protocolo FlexRay é utilizado. Para avaliação das equações obtidas (as quais são baseadas no tempo de duração de um bit e no formato dos quadros) é elaborado um cenário para simulação, utilizando o software CANoe.FlexRay, de um veículo híbrido. A rede elaborada para o estudo de caso possui 5 componentes, taxa de transmissão 10 $Mbit/s$ com mensagens distribuídas entre os segmentos estático e dinâmico. Os resultados da simulação realizada reforçam as características de cada um dos segmentos.

Em Steinbach, Korf e Schmidt (2010) é realizada uma comparação entre os protocolos TTEthernet e FlexRay, para uso em aplicações críticas automotivas. Para análise, são considerados quadros com o mesmo tamanho para o segmento de dados nos dois protocolos, mesmo período do ciclo de comunicação e mesmo comprimento de cabos entre componentes. São realizadas análises do tempo de latência e *jitter* da transmissão de mensagens síncronas em relação ao tamanhos do segmento de dados (variando de 1 a 264 *bytes*). Os resultados indicam vantagem para o protocolo TTEthernet quando o tamanho do segmento de dados aumenta. Por outro lado, a transmissão de pacotes em uma rede Ethernet depende da quantidade de *switches* no caminho da mensagem, fazendo com que, dependendo da topologia da rede, a latência na transmissão de mensagens seja maior.

Em Bartols, *et al.* (2011) é proposto um analisador de rede para o protocolo TTEthernet (SAE INTERNATIONAL, 2013), baseado em uma plataforma formada por um sistema embarcado com processador Intel ATOM, utilizando Linux com um *patch* Real Time (RT) como sistema operacional. Para representação da rede TTEthernet, é utilizado um *switch*. O objetivo do trabalho é avaliar o tempo de latência fim-a-fim (*end-to-end delay*) durante a transmissão de mensagens. Com o intuito de reduzir ao máximo erros de medição causados por diferentes bases de tempo (fato presente em sistemas distribuídos, fazendo-se necessária a sincronização de relógios), são utilizadas duas portas no mesmo sistema: uma para o envio de mensagens e outra para o recebimento. Para que outras tarefas gerenciadas pelo sistema operacional não afetem o procedimento de forma significativa, as medições são realizadas muito próximas ao nível de hardware, através do uso de *drivers* modificados de interfaces de rede. Os instantes de tempo de transmissão e recebimento são armazenados no próprio quadro Ethernet, para posterior avaliação. A fim de analisar o tempo de latência na transmissão, um componente envia 100 mensagens de medição separadas por um intervalo de 3 *ms* (tempo utilizado pelo ciclo TTEthernet). Para avaliar os resultados obtidos pelo analisador desenvolvido, são utilizados modelos matemáticos dos tempos de transmissão e delay de cada mensagem e um analisador de tráfego comercial. Os resultados obtidos com as medições dos tempos de latência utili-

zando um *switch* com capacidade *time-triggered* são comparados com um *switch* Ethernet convencional, onde fica claro a garantia temporal oferecida pelo sistema TT em relação ao sistema convencional.

O objetivo do trabalho proposto por Lim, Weckemann e Herrscher (2011) é avaliar as possibilidades de uso do padrão Ethernet em redes automotivas, sem implementar nenhuma alteração do padrão amplamente adotado para a construção de LANs (*Local Area Network*). Com este fim, é elaborado um estudo de caso, onde o modelo de rede considerado utiliza uma topologia com duas estrelas interligadas. Cada estrela é composta por um *switch* (elemento central), câmeras para aplicações de auxílio ao motorista, ECUs que realizam aplicações de controle e unidades de processamento distribuídas que utilizam dados com informações secundárias. Para a elaboração do quadro Ethernet utilizado no modelo de simulação, são analisadas as diferentes características (como tamanho do pacote, periodicidade de mensagens e taxas de transmissão necessárias) do tráfego existente atualmente em aplicações automotivas: dados de aplicação de controle, *streaming* de dados das câmeras que oferecem suporte ao motorista, *streaming* de áudio e vídeo (*infotainment*) e dados das aplicações secundárias. O modelo proposto é elaborado no *framework* OMNet++. Como métricas de desempenho, são utilizados o atraso de fim-a-fim e o tempo entre o recebimento de duas mensagens consecutivas. Como resultado, é observado que tanto a latência na transmissão dos quadros com informações de controle quanto aqueles utilizados pelas câmeras não atendem as especificações de projeto. Dos resultados obtidos, conclui-se que não é possível utilizar o padrão Ethernet em redes automotivas para aplicações que necessitam de garantias temporais sem que exista algum método de priorização de mensagens.

3.5 Implementação e Análise de Sistemas de Controle sobre Rede

O trabalho apresentado por Albert (2004), compara, inicialmente, as características de protocolos baseados nos paradigmas *event-triggered* e *time-triggered*, considerando, também, o ponto de vista das necessidades de controle. Neste trabalho, foi elaborada uma pequena plataforma de testes, constituída por: (i) um circuito formado por amplificadores operacionais, o qual simula um sistema contínuo de segunda ordem, (ii) um conversor A/D, que amostra a planta, e um conversor D/A, que realiza a saída do atuador, (iii) um microcontrolador utilizado para simular a função de sensores e atuador e (iv) um segundo microcontrolador responsável pelo algoritmo de controle. Ambos microcontroladores se comunicam através do protocolo TTCAN. Inicialmente, são descritas as principais fontes de atraso que contribuem significativamente para o tempo de latência total, denominado aqui τ_{ee} (ou seja, o intervalo entre o instante em que a planta é amostrada e o instante em que o atuador realiza a lei de controle). Os resultados do experimento ilustram a resposta do sistema quando um degrau unitário é aplicado à sua entrada. Mesmo com diferentes taxas de amostragem (1 e 20 *ms*), o sistema se mantém estável. De acordo com o autor, como no caso de uma comunicação *time-triggered* o período de latência é garantido, controladores que compensem este atraso fixo podem ser facilmente implementados. Para evidenciar a importância do determinismo neste tipo de aplicação, foram realizados testes em situações onde as ECUs não estão sincronizadas, e os resultados comparados às situações de sincronismo entre os dispositivos, tendo como resultado uma grande perda de performance por parte do sistema não sincronizado.

A dissertação de mestrado de Neto (2007), traz um estudo sobre a utilização do protocolo *Foundation Fieldbus* em um sistema de controle sobre rede. Neste trabalho, inici-

almente, foi desenvolvido um protótipo de circuito eletrônico com o objetivo de simular um sistema de segunda ordem. A escolha da dinâmica apresentada pelo circuito levou em consideração as características de tempo de resposta dos equipamentos de *hardware* disponíveis. O arranjo físico (além do circuito mencionado) considera o uso da planta *Foundation Fieldbus*, presente no laboratório LASCAR da UFRGS, para o controle de nível de um sistema de vazão de líquidos. Foram realizados dois experimentos com o objetivo de avaliar o comportamento dinâmico do sistema (quando o laço de controle é fechado através da rede) e a distribuição temporal dos períodos de envio de mensagens de controle através da rede, procurando caracterizar o determinismo da comunicação. O primeiro experimento considera apenas o tráfego gerado pela aplicação de controle, enquanto que o segundo utiliza a aplicação de controle de nível da planta para a geração de mais tráfego no barramento. Como resultado, pode-se observar a degradação de desempenho nos dois testes e, através da análise da distribuição temporal dos períodos das mensagens de controle, que houve variação no período de transmissão destas mensagens. Os resultados do segundo experimento, inclusive, apresentam degradação de desempenho maior, fato decorrido do aumento do tráfego no barramento.

O impacto da comunicação causado por uma rede CAN em um sistema de direção por fios ("*Steering-by-wire"*) é analisado em Li, Wang e Liao (2008). No trabalho, é utilizado um *framework* baseado em MATLAB/Simulink para análise dos efeitos causados pela comunicação, entre eles: (i) variação da periodicidade das mensagens de controle, (ii) variação do atraso na transmissão das mensagens e, por fim, (iii) erros encontrados durante a transmissão, como perdas de pacotes. Os testes realizados consistem em comparações entre as curvas das saídas de interesse do sistema em condições ideais (comunicação em um canal ideal, contínuo) e aquelas obtidas quando algum efeito introduzido pela rede é considerado. No geral, o desempenho do sistema é negativamente afetado conforme o tempo total τ_{ee} aumenta.

Em Duan, Deng e Yu (2010), mais um sistema *Steering-by-wire* é implementado, entretanto, desta vez, utilizando o protocolo FlexRay. O sistema foi simulado, inicialmente, utilizando a interface existente entre as ferramentas CANoe.FlexRay e Matlab/Simulink. Posteriormente, o sistema foi implementado em dois dispositivos físicos, enquanto uma interface VN3600 foi utilizada para leitura do barramento. Como resultado, observou-se que as curvas de resposta do sistema à uma entrada senoidal (utilizada para simular o movimento da direção do veículo) segue sua referência. Não foram feitas simulações do barramento com carga.

4 O PROTOCOLO FLEXRAY

4.1 Introdução

O FlexRay é um protocolo de comunicação digital serial desenvolvido pelo FlexRay Consortium, especificamente para aplicações veiculares. O FlexRay Consortium foi fundado no ano 2000 por empresas do setor automotivo e fabricantes de dispositivos eletrônicos. À época de sua criação era constituído por quatro companhias: BMW, Daimler-Chrysler, Philips e Freescale (MAKOWITZ; TEMPLE, 2006). Quando o processo de especificação do protocolo foi iniciado, foram considerados três objetivos principais (MAKOWITZ; TEMPLE, 2006):

- **Alta velocidade:** uma ordem de grandeza maior que o CAN;
- **Determinismo:** para oferecer suporte a aplicações críticas de controle (por exemplo: *brake-by-wire*, *steering-by-wire*);
- **Tolerância a falhas:** para que fosse possível substituir sistemas críticos mecânicos e hidráulicos.

Em 2009, o FlexRay Consortium já contava com mais de 1000 associados, distribuídos em diferentes níveis de associação (membro *core*, associado *premium*, associado e desenvolvedor) (GÖHNER, 2012). Neste mesmo ano, com a publicação da versão 3.0 das especificações, o FlexRay Consortium foi desfeito. Atualmente, as especificações foram transformadas em um conjunto de padrões ISO, sob o código 17458. A última versão publicada (considerando o período em que este trabalho foi escrito) data de 2013 e é composta por cinco partes (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2013):

- **ISO 17458-1:2013**: *General information and use case definition*
- **ISO 17458-2:2013**: *Data Link Layer Specification*
- **ISO 17458-3:2013**: *Data Link Layer conformance test specification*
- **ISO 17458-4:2013**: *Electrical Physical Layer Specification*
- **ISO 17458-5:2013**: *Electrical Physical Layer conformance test specification*

As especificações do protocolo FlexRay cobrem os dois primeiros níveis do modelo lógico OSI (GÖHNER, 2012), conforme ilustrado pela Figura 13.

Este capítulo traz uma descrição sobre este protocolo. As informações aqui contidas foram obtidas em (FLEXRAY CONSORTIUM, 2012b), (FLEXRAY CONSORTIUM, 2012a), (FLEXRAY CONSORTIUM, 2012c) e (VECTOR INFORMATIK GMBH, 2013a).

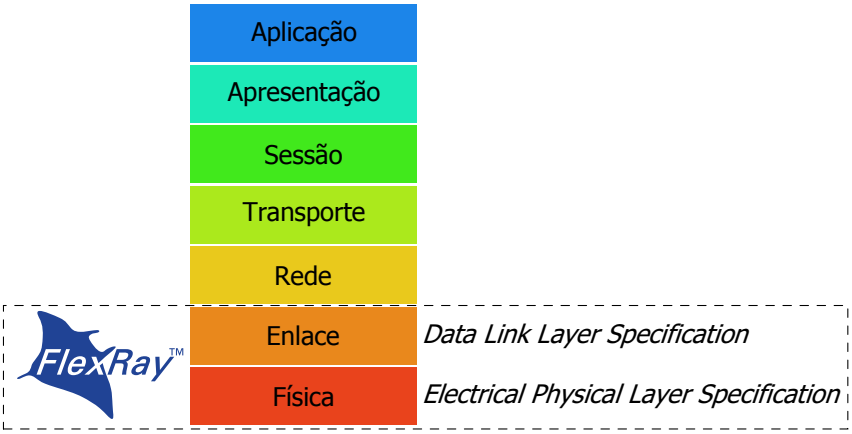

Figura 13: Modelo OSI e as especificações do FlexRay. Baseado em (GÖHNER, 2012).

4.2 Camada Física

4.2.1 Conexão entre dispositivos

A transmissão de sinais em um *cluster* FlexRay ocorre através da diferença de tensão entre os terminais BP (*Bus Plus*) e BM (*Bus Minus*) do *transceiver*. A Figura 14 ilustra um exemplo de conexão entre dois dispositivos. Esta estrutura pode ser extendida com a adição de outros dispositivos, alterando a topologia da rede (as quais serão apresentadas em uma seção posterior).

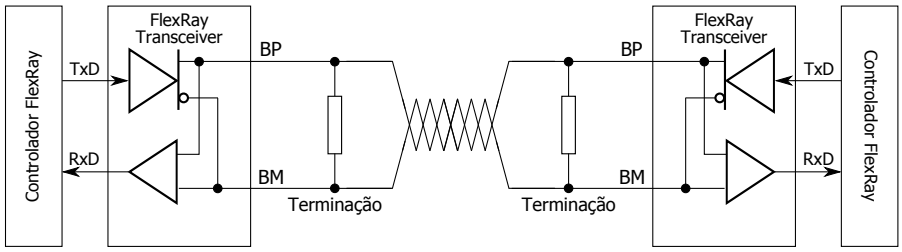

Figura 14: Princípio de comunicação entre dois dispositivos FlexRay. Baseado em (VECTOR INFORMATIK GMBH, 2013a).

4.2.2 Topologias de Rede

Existem várias maneiras de se projetar um *cluster* FlexRay: conexão ponto-a-ponto, barramento, rede em estrela ou em várias combinações híbridas entre estas (FLEXRAY CONSORTIUM, 2012b).

Um *cluster* FlexRay consiste em no máximo dois canais, identificados como canais A e B. Cada componente da rede pode estar conectado a um ou dois canais, sendo que a velocidade de transmissão em cada canal pode ser de até 10 $Mbit/s$. Desta forma, através da utilização de dois canais, pode-se tanto dobrar a banda de transmissão como utilizar outro canal como redundância.

4.2.2.1 Conexão Ponto-a-Ponto

A conexão ponto-a-ponto é a topologia mais simples, e pode ser considerada como a base para a construção de topologias mais complexas. Consiste em apenas dois dispositivos conectados entre si. A Figura 15 ilustra um exemplo deste tipo de conexão.

Figura 15: Conexão ponto-a-ponto. Baseado em (FLEXRAY CONSORTIUM, 2012a).

De acordo com (FLEXRAY CONSORTIUM, 2012a), a extensão máxima dos cabos que interligam as duas ECUs ("*lBus*" na Figura 15) não deve ser maior que 24 metros.

4.2.2.2 Estrela Passiva

A topologia em estrela passiva é obtida quando as ECUs de um cluster são conectadas ao mesmo ponto. A Figura 16 mostra um exemplo deste tipo de topologia.

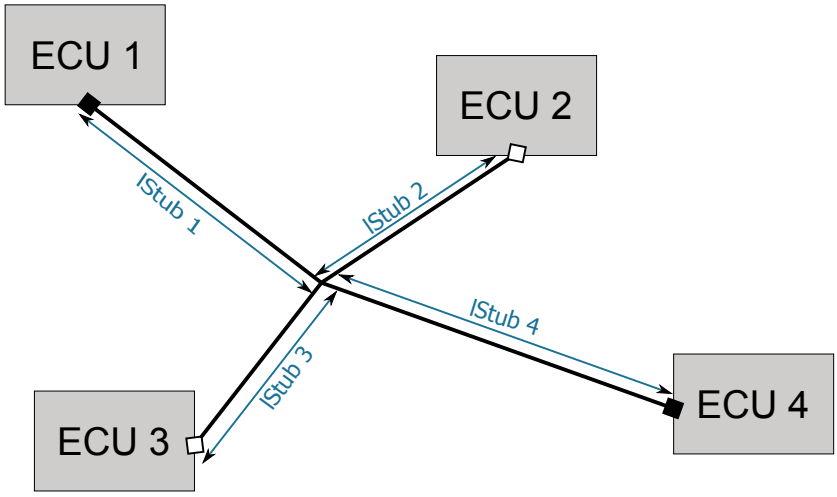

Figura 16: Topologia estrela passiva. Baseado em (FLEXRAY CONSORTIUM, 2012a).

Nesta configuração, a distância máxima entre dois componentes deve ser no máximo 24 metros, e não mais de 22 ECUs devem estar conectadas em forma de estrela passiva. Ainda, os dois nós mais distantes do ponto de conexão devem apresentar terminação dos cabos.

4.2.2.3 Barramento Linear Passivo

A Figura 17 exibe um exemplo de um barramento linear passivo. Neste tipo de topologia, cada ECU é conectada a um barramento principal por "*Stubs*". Quando esta topologia é utilizada, a distância máxima entre quaisquer dois componentes não deve ultrapassar 24 metros quando o sistema opera com taxa de transmissão de 10 $Mbit/s$. Como no caso anterior, um máximo de 22 ECUs pode estar conectado ao mesmo barramento linear passivo (VECTOR INFORMATIK GMBH, 2013a).

4.2.2.4 Estrela Ativa

Uma outra maneira de construir um *cluster* FlexRay é através do uso de um componente ativo (*Active Star*). A Figura 18 mostra um exemplo deste tipo de configuração.

O componente ativo, atua como um repetidor, amplificando e distribuindo o sinal recebido para todos os outros componentes a ele conectados. A distância máxima entre um componente ativo e qualquer ECU FlexRay não deve ser maior que 24 metros.

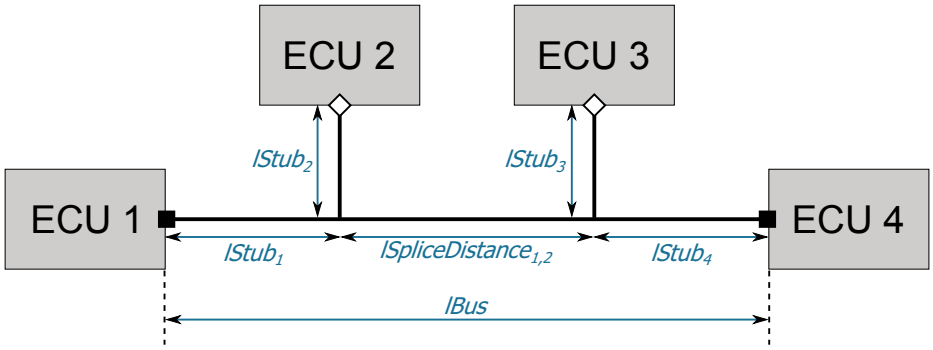

Figura 17: Barramento Passivo. Baseado em (FLEXRAY CONSORTIUM, 2012b).

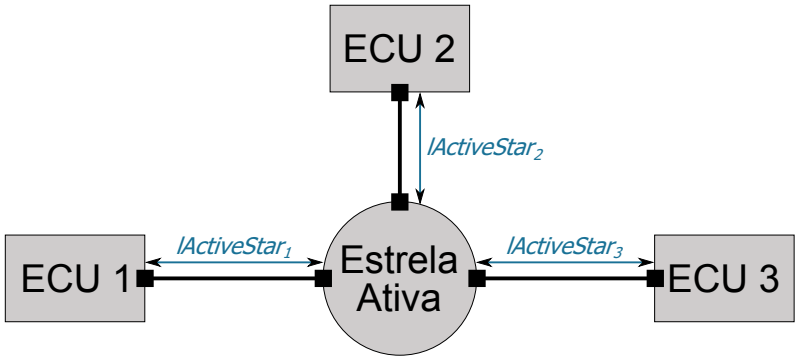

Figura 18: Topologia estrela ativa. Baseado em (FLEXRAY CONSORTIUM, 2012b).

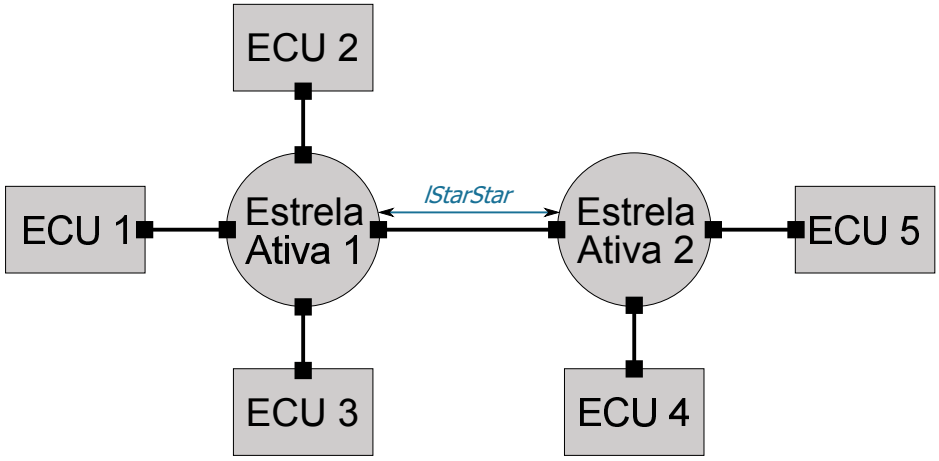

Figura 19: Topologia estrela ativa em cascata. Baseado em (FLEXRAY CONSORTIUM, 2012b).

O uso de componentes do tipo estrela ativa adiciona um atraso na transmissão de sinais, o qual, segundo (FLEXRAY CONSORTIUM, 2012a), não deve ser superior a 450 *ns* para cada estrela ativa.

4.2.2.5 Topologia Híbrida

Além das topologias descritas anteriormente, é possível construir topologias híbridas, as quais combinam topologias do tipo barramento e estrela. Um exemplo deste tipo de combinação é ilustrado pela Figura 20.

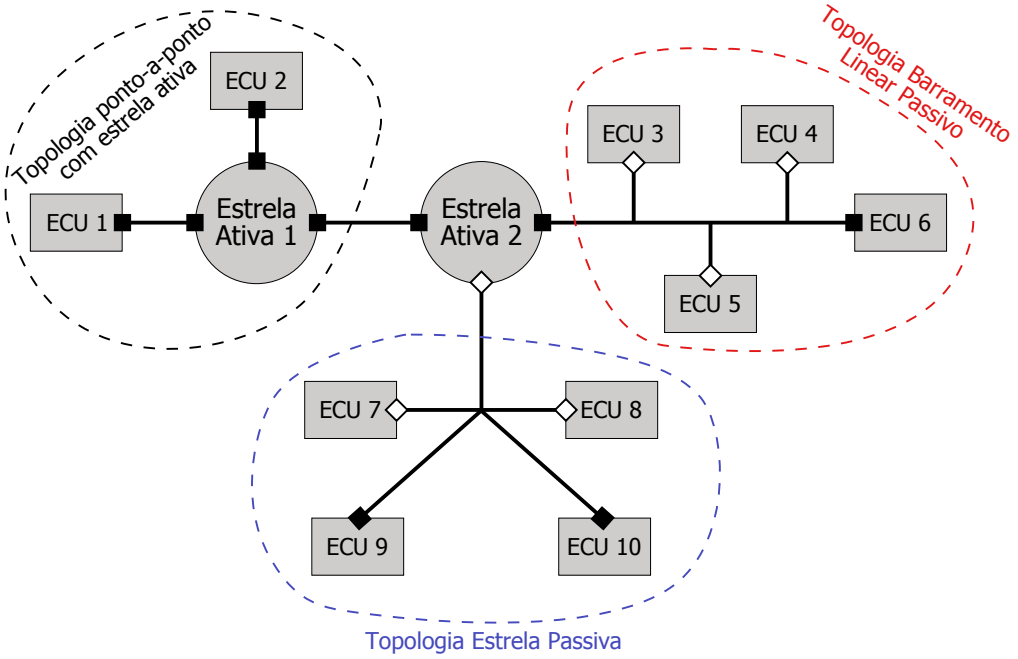

Figura 20: Exemplo de topologia híbrida. Baseado em (FLEXRAY CONSORTIUM, 2012a).

4.2.3 Sinais elétricos

Como mencionado inicialmente, a transmissão de dados em um *cluster* FlexRay ocorre através da diferença de tensão entre os terminais BP e BM dos *transceivers*. Este tipo de abordagem ajuda a proteger o sistema contra interferências eletromagnéticas externas (VECTOR INFORMATIK GMBH, 2013a).

As especificações da camada física do FlexRay definem quatro níveis de tensão (VECTOR INFORMATIK GMBH, 2013a):

- *Idle low power bus level*: modo utilizado quando os *tranceivers* operam em modo de baixa energia. Neste estado, tanto BP quanto BM apresentam tensão no intervalo $[-0,2;0,2]$, sendo que a diferença de tensão é mantida em 0.
- *Idle bus level*: modo onde a diferença de tensão entre os terminais é 0. Neste estado, cada linha apresenta o mesmo nível de tensão, o qual deve estar situado no intervalo $[1,8;3,2]\ V$.
- *Data_1 bus level*: este estado representa nível lógico 1. Nesta situação, BP opera em $3,5\ V$, enquanto BM opera em $1,5\ V$, fazendo com que a diferença seja $2\ V$.
- *Data_0 bus level*: este estado representa nível lógico 0. Nesta situação, BP opera em $1,5\ V$, enquanto BM opera em $3,5\ V$, fazendo com que a diferença seja $-2\ V$.

A Figura 21 exibe os níveis de tensão de operação de um *cluster* FlexRay.

4.3 Controle de Acesso ao Meio (MAC)

4.3.1 Ciclo de Comunicação

O esquema de acesso ao meio do protocolo FlexRay é baseado em um ciclo de comunicação recorrente, onde cada ciclo possui um período de repetição fixo e pré-

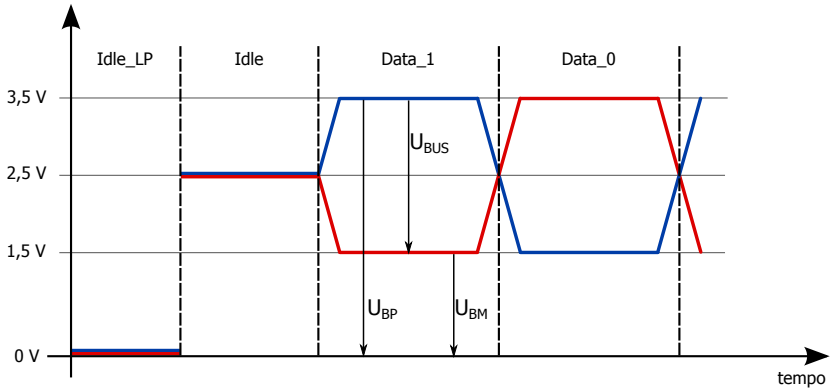

Figura 21: Exemplo de sinais elétricos no barramento. Baseado em (VECTOR INFORMATIK GMBH, 2013a).

determinado. Cada ciclo é divido em quatro segmentos: Segmento Estático (*SS – Static Segment*), Segmento Dinâmico (*DYN – Dynamic Segment*), Janela de Símbolos (*SW – Symbol Window*) e Tempo Inativo de Rede (*NIT – Network Idle Time*). Dentre estes quatro segmentos, todo ciclo de comunicação deve apresentar pelo menos o segmento estático e o tempo inativo de rede. Através deste método de acesso ao meio, o FlexRay é capaz de prover comunicação com garantias temporais (através da utilização do segmento estático), e, também, pode oferecer um método de acesso ao meio para mensagens esporádicas (através do segmento dinâmico, quando este estiver presente).

A Figura 22 ilustra a composição de um ciclo de comunicação a partir das unidades temporais mais básicas que formam a base de tempo da comunicação.

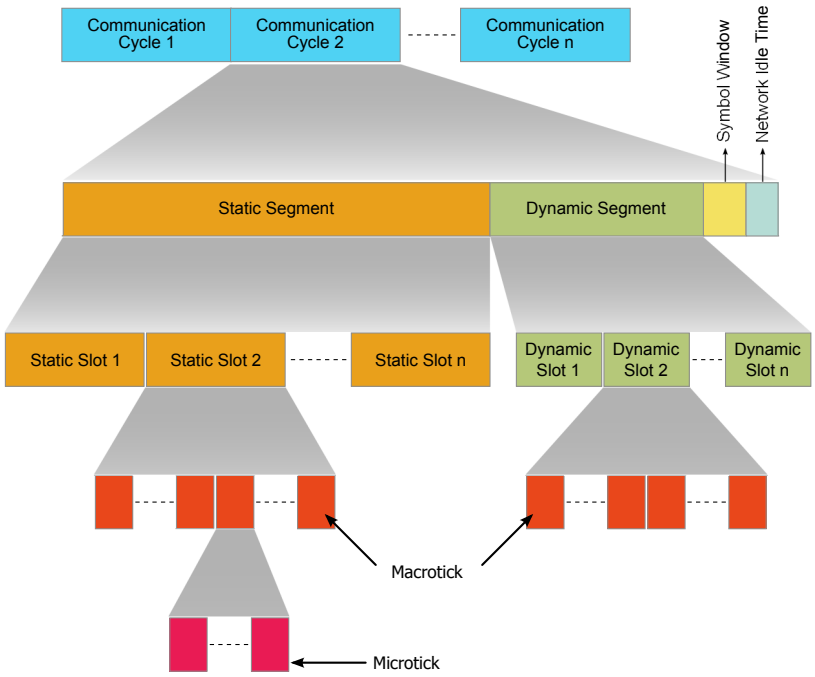

Figura 22: Hierarquia temporal de um ciclo de comunicação FlexRay. Baseado em (FLEXRAY CONSORTIUM, 2012b).

Conforme ilustrado pela Figura 22, no nível mais baixo, encontram-se os *microticks*. Os *microticks* estão relacionados diretamente ao oscilador da ECU e representam a me-

nor unidade de tempo deste componente. Os *microticks* podem ter duração diferente em diferentes dispositivos. Os *macroticks* são formados por um certo número inteiro de *microticks*. Entretanto, a duração de um *macrotick* deve ser, dentro de uma certa tolerância, a mesma entre todos os componentes de um *cluster*.

Cada ciclo de comunicação possui um número fixo e inteiro de *macroticks*, sendo que este número deve ser o mesmo entre todos os dispositivos do *cluster* e, também, de ciclo para ciclo. Os ciclos de comunicação são enumerados de 0 a *gCycleCountMax*, e cada componente da rede deve manter, ainda, um contador de ciclos, denominado *vCycleCounter*, o qual armazena o número do ciclo atual.

4.3.2 Segmento Estático (*SS – Static Segment*)

O segmento estático utiliza o TDMA como método de acesso ao meio. Este segmento possui um número fixo de *slots*, configurável através do parâmetro *gNumberOfStaticSlots*, o qual é uma constante global para um determinado *cluster*. Cada *slot* estático possui a mesma duração, configurada através do parâmetro *gdStaticSlot*. Este último parâmetro especifica quantos *macroticks* são necessários para formar um *slot* do segmento estático e, também, é uma constante global para todo o *cluster*.

A Figura 23 ilustra um exemplo da divisão em slots do segmento estático.

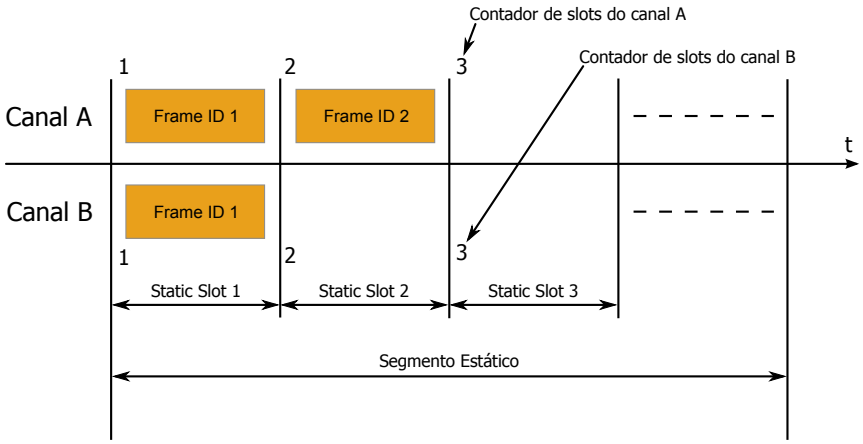

Figura 23: Estrutura do Segmento Estático. Baseado em (FLEXRAY CONSORTIUM, 2012b).

4.3.3 Segmento Dinâmico (*DYN – Dynamic Segment*)

O segmento dinâmico utiliza um procedimento de acesso ao meio mais complexo do que aquele empregado pelo segmento estático. Este método é denominado FTDMA (*Flexible Time Division Multiple Access*). Neste método, o segmento dinâmico é dividido em um número de "*mini-slots*" (configurado pelo parâmetro *gNumberOfMiniSlots*). Quando alguma ECU precisa enviar uma mensagem através deste segmento, o tamanho do *mini-slot* é incrementado de forma a acomodar o pacote a ser transmitido.

Este segmento emprega um esquema de prioridades para a transmissão de quadros. Deste modo, o quadro que possuir o menor ID terá maior prioridade para a transmissão. Os *slots* do segmento dinâmico seguem a contagem iniciada no segmento estático, ou seja, se o segmento estático possui n *slots*, o primeiro *mini-slot* do segmento dinâmico será numerado $n+1$.

A Figura 24 mostra um exemplo de comunicação utilizando o segmento dinâmico.

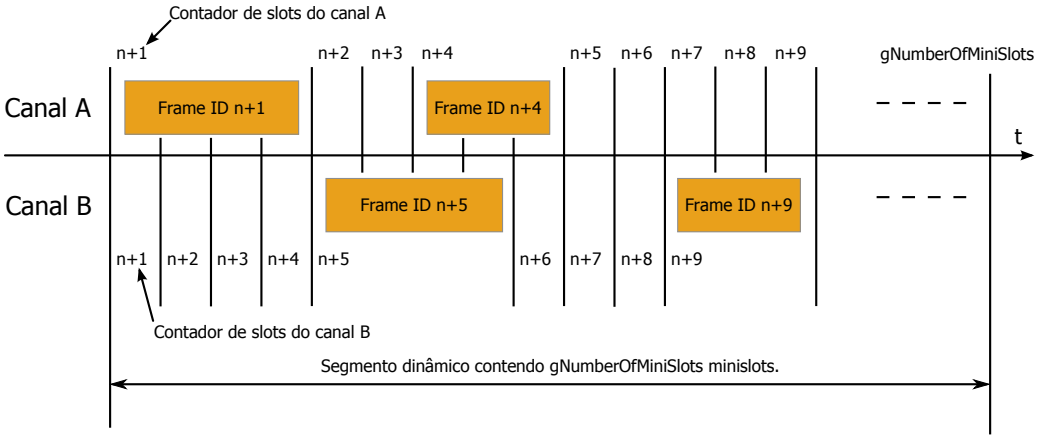

Figura 24: Estrutura do Segmento Dinâmico. Baseado em (FLEXRAY CONSORTIUM, 2012b).

4.3.4 Janela de Símbolos (*SW – Symbol Window*)

A janela de símbolos é utilizada para transmissão de um único símbolo, podendo ser um MTS (*Media Access Test Symbol*) ou WUDOP (*Wakeup During Operation Pattern*). O tamanho deste segmento pode ser definido pela quantidade de *macroticks* que o formam, e definido através da constante *gdSymbolWindow*.

4.3.5 *NIT – Network Idle Time*

O NIT é o tempo que não é utilizado por nenhum dos segmentos descritos anteriormente, e não existe comunicação no *cluster*. Este período é utilizado para que os componentes da rede executem algoritmos para a correção do sincronismo entre seus relógios.

4.4 Formato do Quadro

Um quadro FlexRay é composto por três segmentos: cabeçalho (*header*), carga (*payload*) e cauda (*trailer*). A Figura 25 exibe o esquema lógico de um quadro FlexRay, evidenciando os segmentos que o compõem.

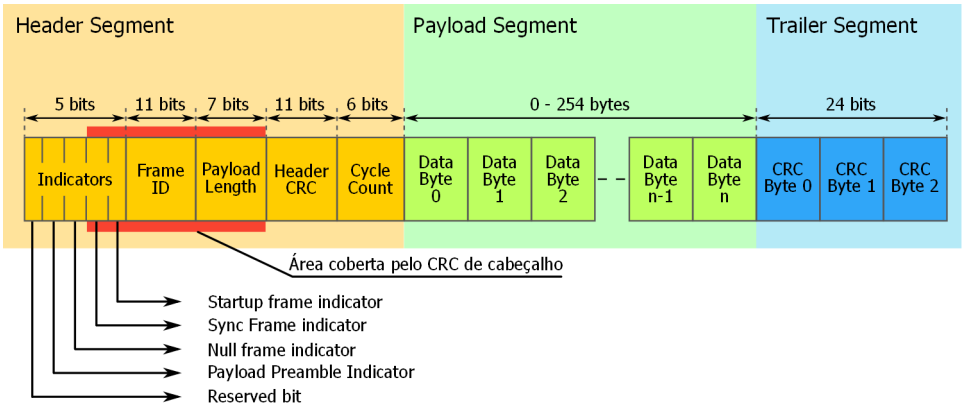

Figura 25: Formato do quadro FlexRay. Baseado em (FLEXRAY CONSORTIUM, 2012b).

4.4.1 Segmento de Cabeçalho (Header Segment)

O cabeçalho de um quadro FlexRay consiste em 5 bytes. A seguir são apresentadas as descrições dos campos deste segmento.

4.4.1.1 Reserved bit – 1 bit

É o primeiro bit do quadro FlexRay. Este bit é reservado para uso futuro. O dispositivo transmissor deve manter este bit em nível lógico 0, e os receptores devem ignorá-lo.

4.4.1.2 Payload Preamble Indicator – 1 bit

Este bit pode ter significados diferentes, dependendo do segmento no qual o quadro será transmitido:

- Caso o quadro seja transmitido no segmento **estático**, este bit indica a presença de um vetor de gerenciamento de rede no início do segmento de carga;
- Caso o quadro seja transmitido no segmento **dinâmico**, este bit indica a presença de um identificador de mensagem no início do segmento de carga.

Quando mantido em nível lógico 0, o segmento de carga não contém nenhuma destas informações. Quando mantido em nível lógico 1, o segmento de carga contém um vetor de gerenciamento de rede ou um identificador de mensagem.

4.4.1.3 Null frame indicator – 1 bit

O bit indicador de quadro nulo indica se o quadro contém dados válidos no segmento de carga:

- Se o indicador de quadro nulo for mantido em nível lógico 0, então, o segmento de dados não contém dados válidos. Todos os bytes no segmento de carga serão mantidos em zero, e o bit *payload preamble indicator* será colocado também em nível lógico zero;
- Se o indicador de quadro nulo for mantido em nível lógico 1, então, o segmento de carga contém dados válidos.

4.4.1.4 Sync Frame Indicator – 1 bit

Indica se o quadro é utilizado para a sincronização dos relógios do cluster FlexRay:

- Se este indicator for mantido em nível lógico 0, então, nenhum dispositivo receptor irá considerar o quadro para sincronização;
- Se este indicador for mantido em nível lógico 1, então, todos os dispositivos receptores utilizarão este quadro para sincronização.

4.4.1.5 Startup frame indicator – 1 bit

Este indicador informa se o quadro é utilizado para a inicialização do cluster FlexRay.

- Se este indicador for mantido em nível lógico 0, então, o quadro em questão não é um quadro de inicialização;
- Se este indicador for mantido em nível lógico 1, então, o quadro em questão é um quadro de inicialização.

4.4.1.6 Frame ID – 11 bits

Esta sequência de bits indica em qual *slot* o quadro deve ser transmitido. Um determinado identificador não pode ser utilizado mais de uma vez em um mesmo canal durante um mesmo ciclo de comunicação. Todo quadro transmitido pelo *cluster* deve ter um identificador a ele associado.

O identificador de um quadro varia de 1 a 2047, sendo que o identificador 0 é considerado inválido.

4.4.1.7 Payload length – 7 bits

Esta sequência de bits é utilizada para indicar o tamanho do segmento de carga. A quantidade de bytes aqui indicada equivale à metade do número de bytes presente no segmento de carga.

4.4.1.8 Header CRC – 11 bits

Este campo contém um código de checagem de redundância cíclica (*Cyclic Redundancy Check Code - CRC*), o qual é computado sobre os campos *sync frame indicator, startup frame indicator, frame ID e payload length*.

O polinômio utilizado para o cálculo do CRC de cabeçalho é dado por (22).

$$x^{11} + x^9 + x^8 + x^7 + x^2 + x^1 \tag{22}$$

O vetor de inicialização do registrador utilizado para gerar o CRC deve ser 0x01A.

4.4.1.9 Cycle Count – 6 bits

Este campo indica o número do ciclo de comunicação de acordo com o contador *vCycleCounter*, de um ponto de vista do transmissor no momento da transmissão de um quadro.

4.4.2 Segmento de Carga (*Payload – 0 a 254 bytes*)

O segmento de carga de um quadro FlexRay pode ser composto por 0 a 254 bytes de dados. Cada byte dentro deste segmento é identificado por um número, que se inicia em zero logo após o término do segmento de cabeçalho.

Para quadros transmitidos no segmento dinâmico, os dois primeiros bytes do segmento de carga podem ser utilizados como *Message ID*, de acordo com a configuração do bit *payload preamble indicator*. Para quadros transmitidos no segmento estático, até os primeiros 12 bytes do segmento de carga podem ser utilizados para transportar um vetor de gerenciamento de rede. A presença ou não deste vetor é também indicada através do bit *payload preamble indicator*.

4.4.2.1 Network Management Vector – NMVector

Para um quadro transmitido no segmento estático, os 12 primeiros bytes do segmento de carga podem ser um vetor de gerenciamento de rede, de acordo com a indicação do bit *payload preamble indicator*. O tamanho do NMVector é definido durante a fase de configuração do protocolo, através da variável *gNetworkManagementVectorLength*.

4.4.3 Segmento de Cauda (*Trailer Segment*)

O segmento de cauda de um quadro FlexRay consiste em apenas um campo: um CRC de 24 bits.

4.4.3.1 CRC de quadro FlexRay (CRC Frame - 24 bits)

O CRC de quadro FlexRay é calculado sobre o segmento de cabeçalho e, também, sobre o segmento de carga. O polinômio utilizado é dado por (23).

$$x^{24}+x^{22}+x^{20}+x^{19}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^{8}+x^{7}+x^{6}+x^{3}+x+1 \quad (23)$$

A ECU transmissora utilizará um vetor de inicialização diferente dependendo do canal no qual o quadro deve ser transmitido: caso o quadro seja transmitido no canal A, o vetor *0xFEDCBA* é utilizado, e, caso o quadro seja transmitido no canal B, o vetor *0xABCDEF* será utilizado.

4.4.4 Codificação do Quadro

Para transmissão de mensagens no meio físico, o protocolo FlexRay utiliza bits adicionais àqueles que compõem a informação. As Figuras 26 e 27 representam a codificação de um quadro transmitido no segmento estático e um quadro transmitido no segmento dinâmico, respectivamente.

4.4.4.1 Transmission Start Sequence – TSS

Esta sequência indica o início de uma transmissão, onde os componentes da rede terão tempo para perceber que um dado canal não está mais livre. O transmissor gera um TSS, o qual consiste de um período contínuo em nível lógico baixo, sendo este período configurado pelo parâmetro *gdTSSTransmitter*.

4.4.4.2 Frame Start Sequence – FSS

A sequência de início de quadro consiste em apenas um bit em nível lógico alto, e ocorre logo após a seguência de bits que marcam o início da transmissão.

4.4.4.3 Byte Start Sequence – BSS

A sequência de início de byte é utilizada para sincronização temporal entre os nós transmissores e receptores. É constituída por um bit em nível lógico alto, seguido de um bit em nível lógico baixo. Cada byte do quadro FlexRay será transmitido num formato "estendido": a sequência de cada byte será constituída por um BSS seguido de 8 bits de dados.

4.4.4.4 Frame End Sequence – FES

A sequência de final de quadro é utilizada para marcar o último byte de um quadro FlexRay. É composto por um bit em nível lógico baixo, seguido por um bit em nível lógico alto.

4.4.4.5 Dynamic Trailing Sequence – DTS

Esta sequência de bits é utilizada apenas por quadros transmitidos no segmento dinâmico e sua principal função é evitar uma detecção equivocada, por parte dos receptores,

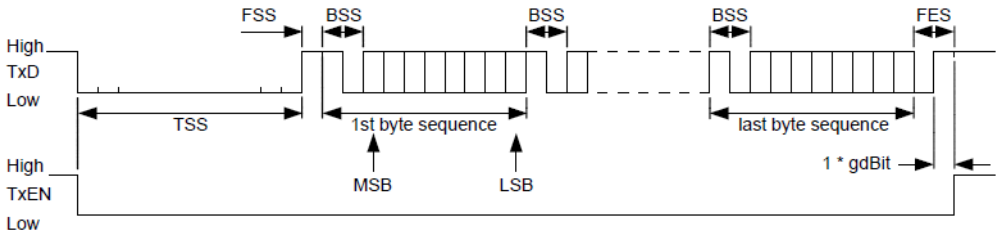

Figura 26: Formatação de um quadro transmitido no segmento estático (FLEXRAY CONSORTIUM, 2012b).

de que um dado canal está desocupado. O DTS é composto por duas partes: um período em nível lógico baixo seguido por um período em nível lógico alto.

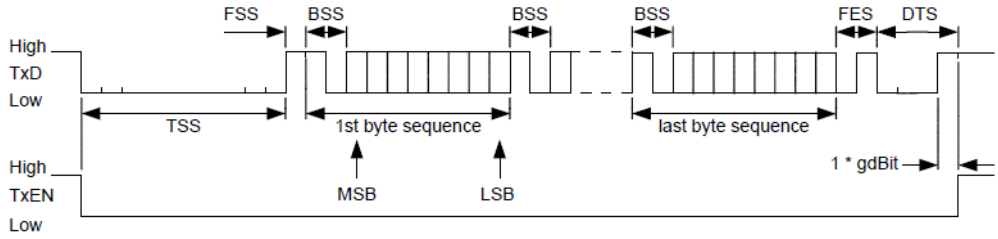

Figura 27: Formatação de um quadro transmitido no segmento dinâmico (FLEXRAY CONSORTIUM, 2012b).

5 INFLUÊNCIA DA COMUNICAÇÃO VIA FLEXRAY EM SISTEMAS DE CONTROLE

5.1 Introdução

O correto escalonamento das mensagens trocadas entre planta e controlador é de fundamental importância em sistemas de controle em rede, uma vez que é responsável direto por garantir um limite de tempo máximo entre os instantes em que a saída da planta é amostrada até a atualização da saída do atuador. O principal responsável por influenciar a duração deste intervalo de tempo é o protocolo da camada MAC utilizado para a arbitragem do acesso ao meio. De forma a ilustrar o atraso τ_{ee}, é considerada a Figura 28. A Figura 28(a) exibe a situação de um sistema de controle em rede com duas ECUs que se comunicam através de um barramento que utiliza comunicação *event-triggered*, enquanto a Figura 28(b) ilustra a sequência de eventos, e os instantes em que estes ocorrem. Nesta situação, a planta é amostrada pelo conjunto de sensores. Durante o período T_1, a *ECU A* irá receber e processar os dados para transmissão. Durante o período T_2, os dados obtidos são transmitidos para a *ECU B*. Após receber a mensagem com os dados, este último componente irá processar a informação durante o período T_3. Durante o período T_4, o barramento está ocupado, fazendo com que a *ECU B* somente possa enviar a mensagem de resposta à *ECU A* durante o período T_5.

De acordo com Albert (2004), o maior problema referente ao atraso presente no laço de controle acontece quando este é variante no tempo. Quando este intervalo é fixo, pode-se elaborar mais facilmente leis de controle que considerem este atraso. Neste contexto, para aplicações críticas de controle, são empregados protocolos de comunicação que sejam capazes de garantir determinismo durante a transmissão de mensagens.

Neste capítulo será apresentado um estudo de caso representativo do segmento automotivo que necessite de garantias temporais e alta taxa de transmissão para a avaliação de desempenho do protocolo FlexRay. Serão desenvolvidos aqui os controladores que serão utilizados para fechamento da malha de controle sobre a rede, para posterior implementação do sistema utilizando equipamentos de hardware que possuem suporte ao protocolo FlexRay.

5.2 Estudo de Caso

O estudo de caso escolhido para a avaliar o desempenho de aplicações de controle em rede foi o sistema de suspensão automotiva. Os principais objetivos de um sistema de suspensão são isolar o chassis do automóvel das perturbações causadas pelas irregularidades do percurso e, ao mesmo tempo, oferecer características para manter o contato

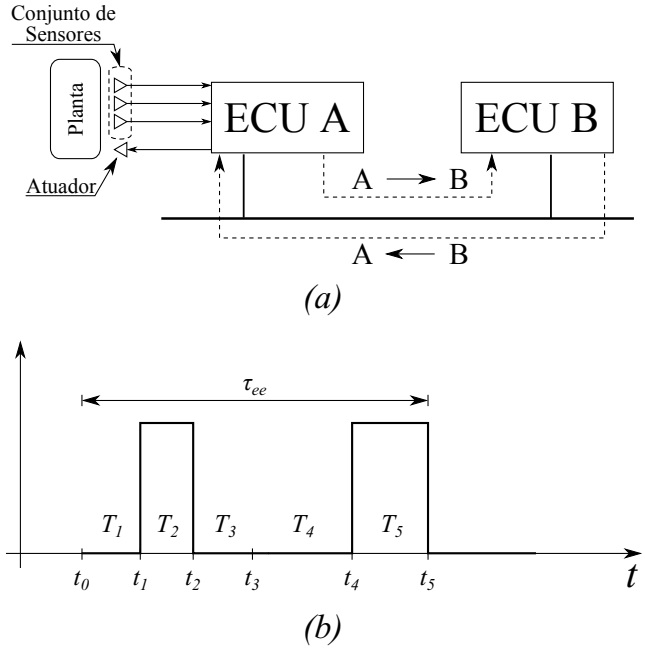

Figura 28: *(a)* Cenário de teste utilizado para exemplo de uma configuração simples de sistema em rede com duas ECUs. *(b)* Evolução temporal dos eventos que ocorrem durante a execução da aplicação de controle e definição do tempo de latência τ_{ee}. Baseado em (ALBERT, 2004).

entre o pneu e o solo (POUSSOT-VASSAL, 2008). A primeira função está relacionada ao conforto oferecido pelo veículo a seus ocupantes, enquanto a segunda está relacionada à segurança. De um ponto de vista de controle, são objetivos contraditórios, sendo assim, o projeto de uma suspensão automotiva é sempre um compromisso entre conforto e desempenho (SANDE et al., 2013).

Para proporcionar maior conforto e segurança, nos últimos anos, fabricantes de automóveis têm desenvolvido sistemas de suspensão ativa (SANDE et al., 2013). Basicamente, este tipo de sistema é utilizado para controlar o movimento de deflexão do sistema de suspensão, através de um atuador externo. No geral, sistemas de suspensão são classificados em três categorias, de acordo com o nível de controle exercido sobre o sistema (DO, 2011):

- **Sistemas Passivos**: encontrados na maioria dos automóveis, são constituídos por uma mola ligada em paralelo a um amortecedor. São capazes apenas de dissipar energia.

- **Sistemas Semi-Ativos**: disponíveis em alguns veículos de categoria mais elevada. Assim como sistemas passivos podem apenas dissipar energia, entretanto, as propriedades de amortecimento podem ser alteradas através de algum mecanismo de controle.

- **Sistemas Ativos**: encontrados em apenas alguns veículos, são compostos por uma mola e um amortecedor ativo. Neste tipo de arranjo, a energia pode ser tanto retirada quanto inserida no sistema.

5.2.1 Modelagem do Estudo de Caso

Para o estudo deste tipo de sistema é muito comum encontrar na literatura trabalhos baseados no modelo conhecido por *Quarter-car model* (Figura 29). Este modelo é formado por conjuntos massa-mola-amortecedor e é utilizado para representar apenas $1/4$ de um veículo. Desta forma, é possível analisar o comportamento vertical do automóvel, sem considerar todas as suas dinâmicas, simplificando a análise.

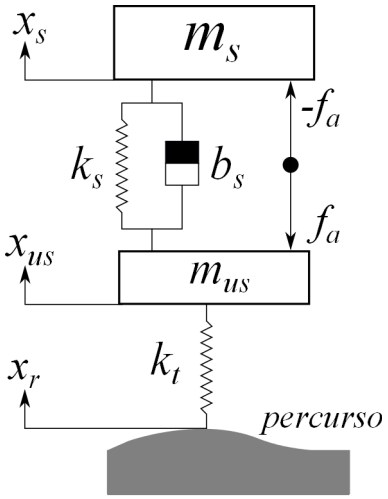

Figura 29: Modelo de suspensão utilizado

A Tabela 1 apresenta uma descrição das variáveis utilizadas pelo modelo.

Tabela 1: Descrição dos parâmetros do sistema de suspensão

Símbolo	Descrição	Unidade
m_s	Massa do corpo do automóvel $(1/4)$	kg
m_{us}	Massa do conjunto suspensão, rodas e pneu	kg
x_s	Posição vertical do corpo do veículo	m
x_{us}	Posição vertical do conjunto da suspensão (considerando o ponto de fixação da roda ao chassis)	m
x_r	Desnível provocado pelo trajeto (solo)	m
x_{def}	Deflexão da suspensão ($x_{def} = x_s - x_{us}$)	m
k_s	Constante da mola da suspensão	N/m
k_t	Constante de mola do modelo do pneu	N/m
b_s	Constante de amortecimento da suspensão	$N \cdot s/m$
f_a	Força aplicada pelo atuador (sinal de controle)	N

A massa não amortecida (em inglês, *unsprung mass*) representa o conjunto formado por pneu, roda, sistema de suspensão e todos os dispositivos auxiliares que formam a "ligação" entre o chasis e o solo (POUSSOT-VASSAL, 2008).

5.2.1.1 Modelagem da Dinâmica do Sistema

Neste trabalho, considera-se um modelo LTI, conforme representado pela Figura 29. A partir da mecânica newtoniana, é possível obter as equações que representam a dinâmica do sistema, dadas por (24) e (25).

$$m_s\ddot{x}_s = -k_s(x_s - x_{us}) - b_s(\dot{x}_s - \dot{x}_{us}) - f_a \tag{24}$$

$$m_{us}\ddot{x}_{us} = k_s(x_s - x_{us}) + b_s(\dot{x}_s - \dot{x}_{us}) + k_t(x_r - x_{us}) + f_a \tag{25}$$

A partir das equações dinâmicas, pode-se obter uma realização no espaço de estados para o sistema. Uma escolha conveniente para o vetor de estados é apresentada por (26). A realização do sistema é dada pela equação (27), onde f_a e $\dot{x}_r$ representam os sinais de controle e perturbação, respectivamente. A descrição das matrizes do sistema dinâmico é dada por (28).

$$x = \begin{bmatrix} x_s - x_{us} \\ \dot{x}_s \\ x_{us} - x_r \\ \dot{x}_{us} \end{bmatrix} \tag{26}$$

$$\dot{x} = Ax + B_u f_a + B_w \dot{x}_r \tag{27}$$

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 & 0 & -1 \\ -\dfrac{k_s}{m_s} & -\dfrac{b_s}{m_s} & 0 & \dfrac{b_s}{m_s} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_s}{m_{us}} & \dfrac{b_s}{m_{us}} & -\dfrac{k_t}{m_{us}} & -\dfrac{b_s}{m_{us}} \end{bmatrix} \\ B_u &= \begin{bmatrix} 0 & \dfrac{1}{m_s} & 0 & -\dfrac{1}{m_{us}} \end{bmatrix}^T \\ B_w &= \begin{bmatrix} 0 & 0 & -1 & 0 \end{bmatrix}^T \end{aligned} \tag{28}$$

Para análise do sistema de suspensão ilustrado pela Figura 29, foram considerados os parâmetros adotados em Do, *et al.* (2011), os quais são exibidos pela Tabela 2.

Tabela 2: Valores adotados para os parâmetros do sistema

Símbolo	Valor	Unidade
m_s	315	kg
m_{us}	37,5	kg
k_s	29500	N/m
k_t	210000	N/m
b_s	1000	$N \cdot s/m$

Substituindo-se os valores dos parâmetros apresentados pela Tabela 2 nas matrizes dadas por (28), obtém-se os valores numéricos dos elementos das matrizes do sistema, apresentados por (29).

$$A = \begin{bmatrix} 0 & 1 & 0 & -1 \\ -93,651 & -3,175 & 0 & 3,175 \\ 0 & 0 & 0 & 1 \\ 786,667 & 26,667 & -5600 & -26,667 \end{bmatrix}, \; B_u = \begin{bmatrix} 0 \\ 0,00317 \\ 0 \\ -0,02667 \end{bmatrix}, \; B_w = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{29}$$

5.2.2 Critérios de Desempenho para Sistemas de Suspensão

Como mencionado anteriormente, os dois principais objetivos de um sistema de suspensão são conforto e controle do veículo, mantendo o contato do pneu com o solo em todos os instantes. Para avaliar o desempenho, quatro funções de transferência são mais comumente utilizadas (POUSSOT-VASSAL, 2008):

- $\ddot{x}_s/\dot{x}_r$: influência da perturbação na aceleração vertical do corpo do automóvel. Esta função de transferência avalia o quesito conforto em frequências na faixa de 4 a 30 Hz.
- $x_s/\dot{x}_r$: influência da perturbação no movimento vertical do corpo do automóvel. Esta função de transferência avalia o quesito conforto em frequências na faixa de 0 a 5 Hz.
- $x_{us}/\dot{x}_r$: influência da perturbação no movimento vertical da roda. Esta função de transferência avalia a condição de contato entre pneu e solo na faixa de frequências de 0 a 20 Hz.
- $x_{def}/\dot{x}_r$: influência da perturbação na deflexão da suspensão. Esta função de transferência representa uma restrição na deflexão do atuador, na faixa de frequências de 0 a 20 Hz.

5.2.3 Análise em Malha Aberta

Utilizando os parâmetros indicados pela Tabela 2, o sistema passivo (quando $f_a = 0$) possui os autovalores conforme indicado pela Tabela 3.

Tabela 3: Parâmetros do sistema em malha aberta

	Autovalor	ζ (%)	ω_n (rad/s)
λ_1 :	$-1,225+9,019i$	13,46	9,102
λ_2 :	$-1,225-9,019i$	13,46	9,102
λ_3 :	$-13,696+78,376i$	17,21	79,564
λ_4 :	$-13,696-78,376i$	17,21	79,564

A Figura 30 exibe a resposta em frequência da função de transferência $x_{def}/\dot{x}_r$ para o sistema em malha aberta.

5.3 Projeto dos Controladores

Com o intuito de observar o efeito causado pelo fechamento do laço de controle sobre uma rede FlexRay, foram elaborados três controladores utilizando diferentes metodologias para síntese:

(1) Alocação de Pólos via realimentação de estados.

(2) Realimentação dinâmica de saída baseada no *framework* H_∞.

(3) Realimentação de estados baseada na metodologia de dados amostrados.

Os controladores foram desenvolvidos tendo como objetivo a rejeição de perturbações provenientes da superfície por onde se locomove o automóvel. Os *scripts* utilizados no desenvolvimento dos controladores encontram-se no anexo A.

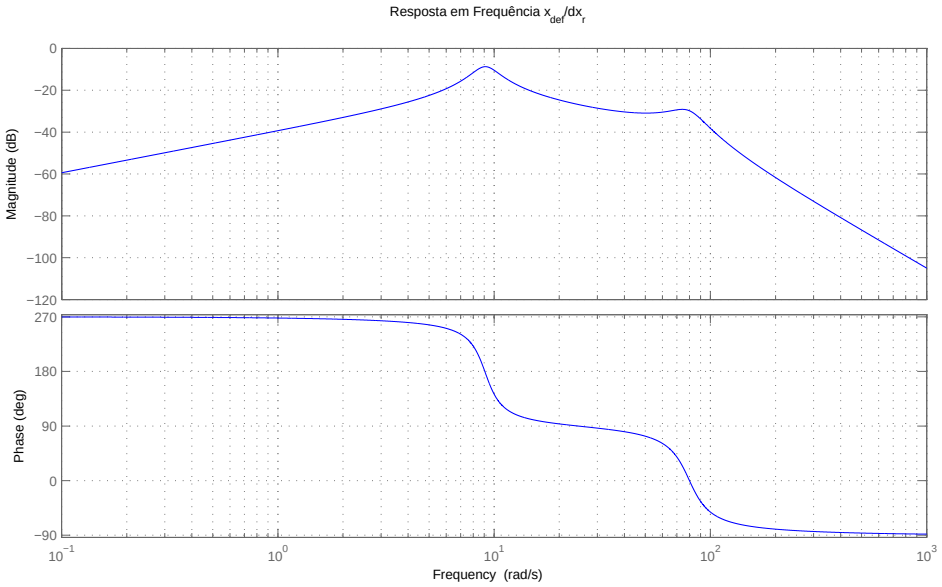

Figura 30: Resposta em frequência para a função de transferência $x_{def}/\dot{x}_r$.

5.3.1 Alocação de Pólos via Realimentação de Estados

O primeiro controlador desenvolvido tem por objetivo a alocação dos pólos do sistema em malha fechada através de uma realimentação de estados, conforme a equação (30), onde K_{pp} é a matriz de ganhos estáticos.

$$f_a = K_{pp}x \tag{30}$$

Neste trabalho, considera-se que todos os estados estão disponíveis para medição. Em muitos casos, esta situação não ocorre, de forma que uma solução é a utilização de um observador para a reconstrução dos estados da planta.

Conforme ilustrado pela Tabela 3, os autovalores $\lambda_{1,2}$ representam os pólos dominantes, uma vez que são muito mais lentos que os autovalores $\lambda_{3,4}$. Desta forma, o objetivo da realimentação de estados é tornar os pólos dominantes mais rápidos e aumentar seu coeficiente de amortecimento. Os pólos $\lambda_{3,4}$ não sofrerão grandes alterações, pois modificações nas posições destes pólos causam um maior esforço de controle sem que seja obtida alteração significativa no comportamento do sistema.

Tabela 4: Parâmetros desejados do sistema em malha fechada

Autovalor desejado	ζ^d (%)	ω_n^d (rad/s)
$\lambda_1^d: \quad -14+14,283i$	70	20
$\lambda_2^d: \quad -14-14,283i$	70	20
$\lambda_3^d: \quad -17+83,283i$	20	85
$\lambda_4^d: \quad -17-83,283i$	20	85

Uma vez definida a localização dos pólos em malha fechada, foi utilizado o método de Moore (MOORE, 1976) para a obtenção da matriz de ganhos K_{pp}. Este método é baseado na equação autovalor-autovetor, e pode ser aplicado quando os autovalores desejados são

diferentes entre si. A equação (31) é utilizada para obter os autovetores associados a um par de autovalores complexos conjugados ($\sigma + \mu i$, $\sigma - \mu i$), onde w_μ e w_σ são direções de entrada a serem arbitradas.

$$\begin{bmatrix} v_\mu \\ v_\sigma \end{bmatrix} = \begin{bmatrix} \mu I - A & -\sigma I \\ \sigma I & \mu I - A \end{bmatrix}^{-1} \begin{bmatrix} B_u & 0 \\ 0 & B_u \end{bmatrix} \begin{bmatrix} w_\mu \\ w_\sigma \end{bmatrix} \tag{31}$$

A matriz de ganhos K_{pp} é obtida através da equação (32), onde V e W são matrizes formadas pelos vetores v_i e w_i, respectivamente, conforme equação (33).

$$K = WV^{-1} \tag{32}$$

$$\begin{array}{rcl} V & = & \begin{bmatrix} v_{\mu_1} & v_{\sigma_1} & v_{\mu_2} & v_{\sigma_2} \end{bmatrix} \\ W & = & \begin{bmatrix} w_{\mu_1} & w_{\sigma_1} & w_{\mu_2} & w_{\sigma_2} \end{bmatrix} \end{array} \tag{33}$$

As posições desejadas (indicadas pelo sobrescrito "*d*") dos pólos do sistema em malha fechada são ilustradas pela Tabela 4. A escolha da localização dos pólos foi resultante de um processo iterativo, realizado por meio de diversas simulações. A partir dos resultados destas simulações, foram escolhidos os parâmetros que apresentaram o melhor desempenho encontrado.

Os ganhos da realimentação de estados obtidos são dados por (34), e o sistema em malha fechada é representado por (35).

$$K_{pp} = \begin{bmatrix} -133062,5 & -11144,375 & -70277,68 & -120,76 \end{bmatrix} \tag{34}$$

$$\dot{x} = (A + B_u K_{pp})x + B_w \dot{x}_r \tag{35}$$

5.3.2 Realimentação Dinâmica de Saída

O segundo controlador projetado trata-se de uma realimentação dinâmica de saída, utilizando o *framework* H_∞. Para qualquer sistema, a síntese de controladores H_∞ é um problema de atenuação, onde o objetivo é encontrar um controlador que estabilize a planta e minimize o impacto das perturbações de entrada nas saídas do sistema de desempenho (POUSSOT-VASSAL, 2008).

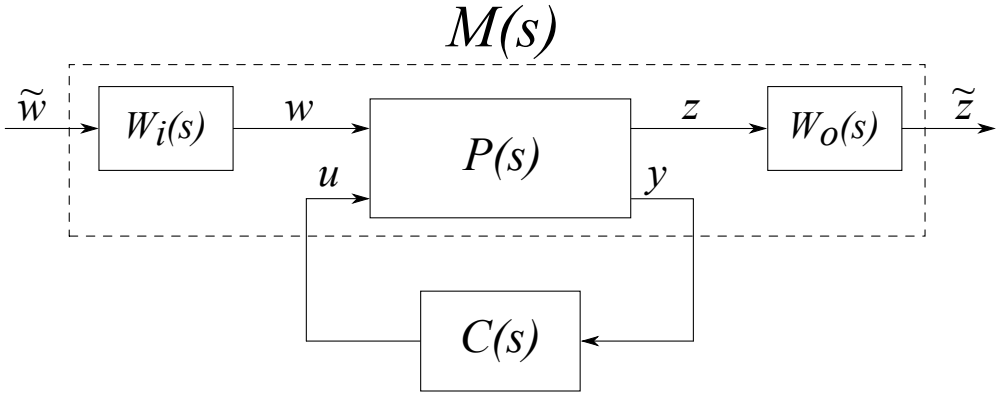

Figura 31: Modelo geral. Baseado em (POUSSOT-VASSAL, 2008).

Este tipo de problema pode ser representado pelo modelo exibido pela Figura 31, onde $P(s)$ representa a planta e $C(s)$ o controlador, enquanto $W_i(s)$ e $W_o(s)$ representam

as funções de ponderação de entrada e saída, respectivamente. A equação (36) apresenta a estrutura da planta $P(s)$, onde w representa o vetor de entradas de perturbação, u o vetor de entradas de controle, z_∞ o vetor de saídas de desempenho e y o vetor de saídas de medição.

$$P(s): \begin{cases} \dot{x} &= Ax + B_w w + B_u u \\ y &= C_y x + D_{yw} w \\ z_\infty &= C_z x + D_{zu} u + D_{zw} w \end{cases} \tag{36}$$

O objetivo da utilização de funções de ponderação é "moldar" a resposta (técnica conhecida como *loop shaping*) de saídas de performance específicas no domínio da frequência (como exemplo, pode ser desejável que o operador G_{wz} apresente baixo ganho em baixas frequências). Com a inclusão das funções de ponderação, considera-se o sistema aumentado $M(s)$, o qual possui estrutura indicada por (37), onde $\bar{x}$ é o vetor de estados do sistema original mais o vetor de estados das funções de ponderação $W_o(s)$, conforme (38).

$$M(s): \begin{cases} \dot{\bar{x}} &= \bar{A}\bar{x} + \bar{B}_u u + \bar{B}_w \tilde{w} \\ y &= \bar{C}_y \bar{x} + \bar{D}_{yw} \tilde{w} \\ \tilde{z}_\infty &= \bar{C}_z \bar{x} + \bar{D}_{zu} u + \bar{D}_{zw} \tilde{w} \end{cases} \tag{37}$$

$$\bar{x} = \begin{bmatrix} x^T & x_o^T \end{bmatrix}^T \tag{38}$$

Como será indicado posteriormente, a função de ponderação de entrada foi adotada como sendo um escalar (w_i), por isso, apenas a função de ponderação de saída apresenta estados no vetor aumentado $\bar{x}$. A equação (39) apresenta a realização da função de ponderação $W_o(s)$.

$$W_o(s): \begin{cases} \dot{x}_o &= A_o x_o + B_o z_\infty \\ \tilde{z}_\infty &= C_o x_o + D_o y_\infty \end{cases} \tag{39}$$

As estruturas das matrizes que compõem o sistema $M(s)$ são apresentadas por (40).

$$\begin{array}{rclrclrcl} \bar{A} &=& \begin{bmatrix} A & 0 \\ B_o C_z & A_o \end{bmatrix} & \bar{B}_u &=& \begin{bmatrix} B_u \\ B_o D_{zu} \end{bmatrix} & \bar{B}_w &=& \begin{bmatrix} B_w w_i \\ B_o D_{zw} w_i \end{bmatrix} \\ \bar{C}_y &=& \begin{bmatrix} C_y & 0 \end{bmatrix} & \bar{C}_z &=& \begin{bmatrix} D_o C_z & C_o \end{bmatrix} & \bar{D}_{yw} &=& \begin{bmatrix} D_{yw} w_i \end{bmatrix} \\ \bar{D}_{zw} &=& \begin{bmatrix} D_o D_{zw} w_i \end{bmatrix} & \bar{D}_{zu} &=& \begin{bmatrix} D_o D_{zu} \end{bmatrix} & & & \end{array} \tag{40}$$

O objetivo é encontrar um controlador $C(s)$, com estrutura exibida por (41), que minimize a norma H_∞ da interconexão entre $M(s)$ e $C(s)$ (ou seja, do sistema em malha fechada, dado por (42)), de forma que a norma H_∞ do operador G_{wz}, com γ_∞ o mínimo possível, satisfaça (43).

$$C(s): \begin{cases} \dot{x}_c &= A_c x_x + B_c y \\ u &= C_c x_c + D_c y \end{cases} \tag{41}$$

$$G_{wz}(s): \begin{cases} \dot{\xi} &= \mathscr{A}\xi + \mathscr{B}\tilde{w} \\ \tilde{z}_\infty &= \mathscr{C}\xi + \mathscr{D}\tilde{w} \end{cases} \tag{42}$$

$$\|G_{wz}\|_\infty = \|\mathscr{C}(sI - \mathscr{A})^{-1}\mathscr{B} + \mathscr{D}\|_\infty < \gamma_\infty \tag{43}$$

Em (43), $\mathscr{A}$, $\mathscr{B}$, $\mathscr{C}$, $\mathscr{D}$ são as matrizes do sistema em malha fechada, dadas por (44), com vetor de estados dado por (45). γ_∞ é um escalar, que representa um limite superior para o valor da norma H_∞ do sistema em malha fechada.

$$\begin{aligned} \mathscr{A} &= \begin{bmatrix} A + B_u D_c C_y & B_u C_c \\ B_c C_y & A_c \end{bmatrix} \\ \mathscr{B} &= \begin{bmatrix} B_w + B_u D_c D_{yw} \\ B_c D_{yw} \end{bmatrix} \\ \mathscr{C} &= \begin{bmatrix} C_z + D_{zu} D_c C_y & D_{zu} C_c \end{bmatrix} \\ \mathscr{D} &= D_{zw} + D_{zu} D_c D_{yw} \end{aligned} \tag{44}$$

$$\xi = \begin{bmatrix} \bar{x}^T & x_c^T \end{bmatrix}^T \tag{45}$$

5.3.2.1 Vetores de Performance e de Medição

O vetor com as saídas de performance escolhidas para o projeto é dado por (46), enquanto o vetor com as saídas de medição é dado por (47).

$$z_\infty = \begin{bmatrix} \ddot{x}_s & x_{def} \end{bmatrix}^T \tag{46}$$

$$y = \begin{bmatrix} x_{def} & \dot{x}_{def} \end{bmatrix}^T \tag{47}$$

5.3.2.2 Funções de Ponderação

O passo fundamental para o projeto de controle H_∞ está na escolha das funções de ponderação, sendo que esta escolha depende fortemente da experiência dos projetistas (DO, 2011).

A função de ponderação de entrada é utilizada para modelar a amplitude do sinal de perturbação $\dot{x}_r$. Neste caso, baseado no trabalho de Do (2011), apenas um ganho estático, representado pelo escalar w_i conforme (48), foi utilizado.

$$W_i = w_i = 0,157 \tag{48}$$

As funções de ponderação de saída utilizadas são dependentes da frequência, uma vez que os objetivos de performance, conforme Seção 5.2.2, também o são. Foram escolhidas duas funções de ponderação, de acordo com a faixa de frequência na qual se deseja atuar.

A função de ponderação para a saída de performance $z_1 = \ddot{x}_s$ é baseada na função utilizada em Do, *et al.* (2011), e representada por (49). A Figura 32 traz o diagrama de Bode da função de ponderação $W_{o_1}(s)$.

$$W_{o_1}(s) = 10 \cdot \frac{0,4901s^2 + 1563s + 360,9}{s^2 + 217,7s + 788,9} \tag{49}$$

Para a segunda saída de performance ($z_2 = x_{def}$) foi utilizado um filtro passa-baixa, conforme equação (50). A Figura 33 exibe o diagrama de Bode da função de ponderação $W_{o_2}(s)$.

$$W_{o_2}(s) = 0,1 \cdot \frac{1}{\frac{1}{2\pi 20}s + 1} \tag{50}$$

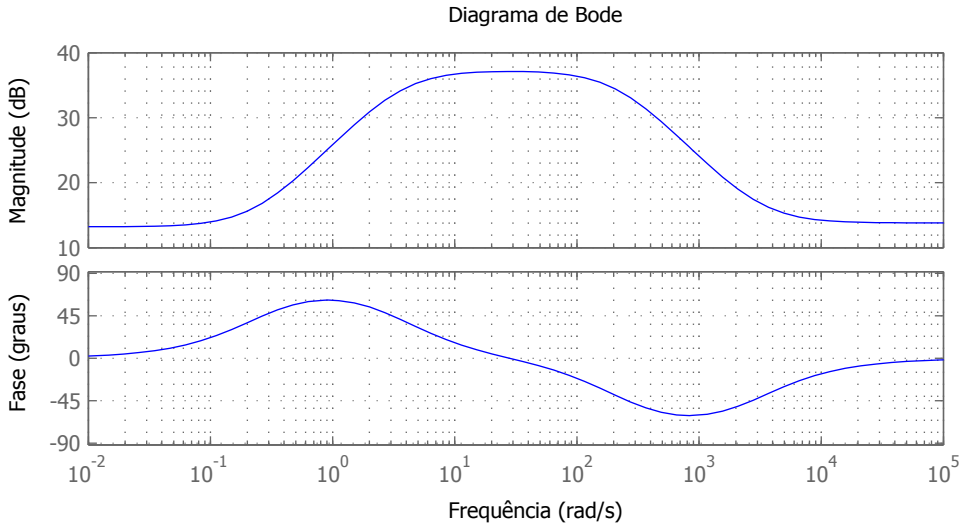

Figura 32: Resposta em frequência da função de ponderação $W_{o_1}(s)$.

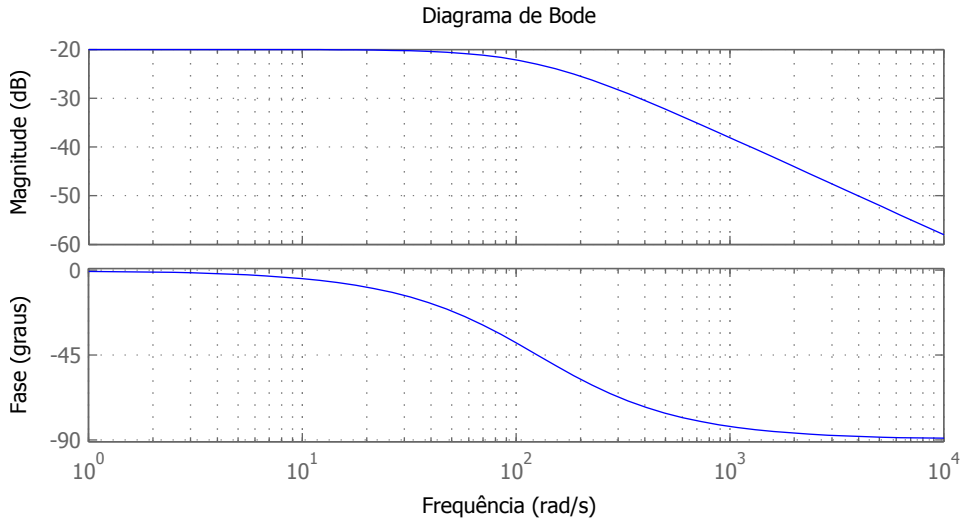

Figura 33: Resposta em frequência para a função de ponderação $W_{o_2}(s)$.

Realizações no espaço de estados para as duas funções de ponderação são apresentadas pela equação (51), enquanto a concatenação destas duas representações para a formação da equação (39) é apresentada por (52).

$$W_{o_1}(s):\begin{cases}\dot{x}_{o_1} &= A_{o_1}x_{o_1}+B_{o_1}z_1\\ \tilde{z}_1 &= C_{o_1}x_{o_1}+D_{o_1}z_1\end{cases},\quad W_{o_2}(s):\begin{cases}\dot{x}_{o_2} &= A_{o_2}x_{o_2}+B_{o_2}z_2\\ \tilde{z}_2 &= C_{o_2}x_{o_2}+D_{o_2}z_2\end{cases} \tag{51}$$

$$W_o(s):\begin{cases}\dot{x}_o &= \begin{bmatrix}A_{o_1} & 0\\ 0 & A_{o_2}\end{bmatrix}\begin{bmatrix}x_{o_1}\\ x_{o_2}\end{bmatrix}+\begin{bmatrix}B_{o_1} & 0\\ 0 & B_{o_2}\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix}\\ \tilde{z}_\infty &= \begin{bmatrix}C_{o_1} & 0\\ 0 & C_{o_2}\end{bmatrix}\begin{bmatrix}x_{o_1}\\ x_{o_2}\end{bmatrix}+\begin{bmatrix}D_{o_1} & 0\\ 0 & D_{o_2}\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix}\end{cases} \tag{52}$$

5.3.2.3 Problema de Otimização

Em Scherer, Gahinet e Chilali (1997), um método para a obtenção de controladores, da forma de (41), através da utilização de otimização baseada em LMIs (*Linear Matrix*

Inequality) é apresentado, e foi o método empregado para a obtenção deste controlador. Neste método, deve-se encontrar a solução do problema de minimização indicado por (53), onde os elementos M_{ij} são descritos por (54). As variáveis do problema são X, Y, $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ e $\tilde{D}$.

$$\min \gamma \; s.a \begin{cases} \begin{bmatrix} X & I \\ I & Y \end{bmatrix} > 0 \\ \begin{bmatrix} M_{11} & * & * & * \\ M_{21} & M_{22} & * & * \\ M_{31} & M_{32} & M_{33} & * \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} < 0 \end{cases} \tag{53}$$

$$\begin{array}{rcl} M_{11} & = & \bar{A}X + X\bar{A}^T + \bar{B}_u\tilde{C} + \tilde{C}^T\bar{B}_u^T \\ M_{21} & = & \tilde{A} + \bar{A}^T + \bar{C}_y^T\tilde{D}^T\bar{B}_u^T \\ M_{22} & = & \bar{A}^TY + Y\bar{A} + \tilde{B}\bar{C}_y + \bar{C}_y^T\tilde{B}^T \\ M_{31} & = & \bar{B}_w^T + \bar{D}_{yw}^T\tilde{D}^T\bar{B}_u^T \\ M_{32} & = & \bar{B}_w^TY + \bar{D}_{yw}^T\tilde{B}^T \\ M_{33} & = & -\gamma_\infty I \\ M_{41} & = & \bar{C}_zX + \bar{D}_{zu}\tilde{C} \\ M_{42} & = & \bar{C}_z + \bar{D}_{zu}\tilde{D}\bar{C}_y \\ M_{43} & = & \bar{D}_{zw} + \bar{D}_{zu}\tilde{D}\bar{D}_{yw} \\ M_{44} & = & -\gamma_\infty I \end{array} \tag{54}$$

O procedimento de reconstrução do controlador é dado por (55), onde as matrizes M e N devem ser definidas de forma que $MN^T = I - XY$.

$$\begin{cases} D_c & = & \tilde{D} \\ C_c & = & (\tilde{C} - D_cC_yX)M^{-T} \\ B_c & = & N^{-1}(\tilde{B} - YB_uD_c) \\ A_c & = & N^{-1}(\tilde{A} - Y\bar{A}X - Y\bar{B}_uD_c\bar{C}_yX - NB_c\bar{C}_yX - Y\bar{B}_uC_cM^T)M^{-T} \end{cases} \tag{55}$$

5.3.2.4 Controlador Sintetizado

Substituindo os valores dos parâmetros dados pela Tabela 2 nas matrizes do sistema e seguindo o procedimento indicado anteriormente, as matrizes do controlador H_∞ obtido, com estrutura dada pela equação (41), são dadas por (56).

$$A_c = \begin{bmatrix} -0,422 & -55,168 & -1,235 & -0,087 & -0,0955 & -9,401\cdot10^{-6} & 6,995\cdot10^{-6} \\ -3374,561 & -3633,361 & -73,466 & -7,319 & -8,168 & -0,001 & 5,5\cdot10^{-4} \\ 78875,93 & 53599,169 & 1341,057 & 46,019 & 59,845 & 0,016 & -2,66\cdot10^{-3} \\ 2,43\cdot10^{8} & 3,43\cdot10^{8} & 1,035\cdot10^{7} & -44353,038 & -57443,11 & -29,22 & 0,031 \\ -1,6\cdot10^{8} & 2,28\cdot10^{8} & 6,86\cdot10^{6} & 29276,4 & 37879,7 & 19,398 & -0,0198 \\ 1,08\cdot10^{11} & 1,57\cdot10^{11} & 4,74\cdot10^{9} & -2\cdot10^{7} & -2,59\cdot10^{7} & -13480,37 & 4,55 \\ 2,17\cdot10^{14} & 3,13\cdot10^{14} & 9,42\cdot10^{12} & -4\cdot10^{10} & -5,2\cdot10^{10} & -2,65\cdot10^{7} & 9264,17 \end{bmatrix}$$

$$B_c = \begin{bmatrix} -1,4\cdot10^{-4} & 2,97\cdot10^{-5} \\ -0,66 & 0,8 \\ 23,73 & 18,39 \\ 1,25\cdot10^{5} & 2,87\cdot10^{5} \\ -8,2\cdot10^{4} & -1,9\cdot10^{5} \\ 5,63\cdot10^{7} & 1,314\cdot10^{8} \\ 1,13\cdot10^{11} & 2,61\cdot10^{11} \end{bmatrix}$$

$$C_c = \begin{bmatrix} -1,18\cdot10^{8} & 1,5\cdot10^{8} & 3,35\cdot10^{6} & 2,37\cdot10^{5} & 2,58\cdot10^{5} & 23,46 & -19,256 \end{bmatrix}$$

$$D_c = \begin{bmatrix} 315,79 & -84,12 \end{bmatrix} \tag{56}$$

5.3.3 Realimentação de Estados Baseada na Metodologia de Dados Amostrados

O terceiro controlador elaborado trata-se de uma realimentação de estados baseada na metodologia de dados amostrados. Nesta metodologia, o sistema é modelado como um sistema contínuo com entrada de controle atrasada, da forma de (57), onde $\tau(t)$ é o atraso da entrada de controle e t_k o instante de amostragem. O sistema em malha fechada é então dado por (58).

$$u(t) = K_{dde}x(t-\tau(t)), \;\; \tau(t) = t - t_k, \;\; \dot{\tau}(t) = 1 \quad para \quad t \in [t_k,\, t_{k+1}) \tag{57}$$

$$\dot{x} = Ax(t) + BK_{dde}x(t-\tau(t)) \tag{58}$$

Em Fridman, Seuret e Richard (2004), condições de estabilidade são encontradas para sistemas com entradas em atraso. A única exigência para o método proposto, é que o intervalo de amostragem não seja maior que uma constante h. Este método obtém condições de estabilidade assintótica mesmo quando a amostragem não é periódica.

Para este controlador, será encontrada uma matriz de ganhos K_{dde} que garanta estabilização exponencial para o sistema, com taxa de decaimento α. A constante h será adotada como $10,5$. Esta escolha foi feita de acordo com período padrão do ciclo de comunicação do protocolo FlexRay, o qual é de 5 ms. Ao adotar $h = 10,5\ ms$, garante-se a estabilidade exponencial para o caso de alguma perda de pacote durante a comunicação. A escolha do funcional de Lyapunov-Krasovskii para a dedução das condições de estabilidade é baseada, em parte, no trabalho apresentado por Gomes da Silva Jr., *et al.* (2011), onde o funcional escolhido é dado por (59).

$$V_\alpha = x'(t)P_1x + \int_{-h}^{0}\int_{t+\theta}^{t} \dot{x}'(s)e^{2\alpha(s-t)}R\dot{x}(s)dsd\theta \tag{59}$$

A partir da solução da LMI apresentada por (60), com variáveis $Q_1 > 0$, $Q_2 > 0$, $\tilde{R} > 0$ e Y, tem-se que o ganho da realimentação de estados dado por (61) garante a estabilidade exponencial do sistema em malha fechada com a lei de controle (57).

$$\begin{bmatrix} 2\alpha Q_1 + AQ_2^T + Q_2A^T - 2he^{-2\alpha h}\tilde{R} & Q_1 + Q_2^T + Q_2A^T\xi & BY + 2he^{-2\alpha h}\tilde{R} \\ * & h^2\tilde{R} - \xi(Q_2^T + Q_2) & \xi BY \\ * & * & -2he^{-2\alpha h}\tilde{R} \end{bmatrix} < 0 \quad (60)$$

$$K_{dde} = Y(Q_2^T)^{-1} \quad (61)$$

A matriz de ganhos K_{dde} com os valores numéricos de seus elementos é dada por (62).

$$K_{dde} = \begin{bmatrix} 2612,114 & -2124,71 & 1017,443 & -4,589 \end{bmatrix} \quad (62)$$

5.4 Análise do Desempenho em Rede usando FlexRay

Como descrito no capítulo 4, a troca de mensagens entre ECUs em um *cluster* FlexRay é baseada em ciclos, compostos por uma parte estática e uma parte dinâmica. O protocolo FlexRay possui mais de 70 parâmetros, distribuídos entre parâmetros globais do *cluster* e específicos para o funcionamento de um componente da rede. A alteração destes parâmetros resulta em modificações tanto no comportamento de ECUs quanto nas características do ciclo de comunicação do protocolo.

5.4.1 Escalonamento de Mensagens no Segmento Estático

De grande importância durante a elaboração da tabela de escalonamento no segmento *time-triggered* é a coordenação das atividades necessárias para a aplicação de controle. Para tanto, se faz necessária uma análise temporal para verificação das necessidades de cada etapa envolvida no processo, de forma que nenhuma atividade perca sua *deadline*, caso que resultaria em um aumento do intervalo τ_{ee}.

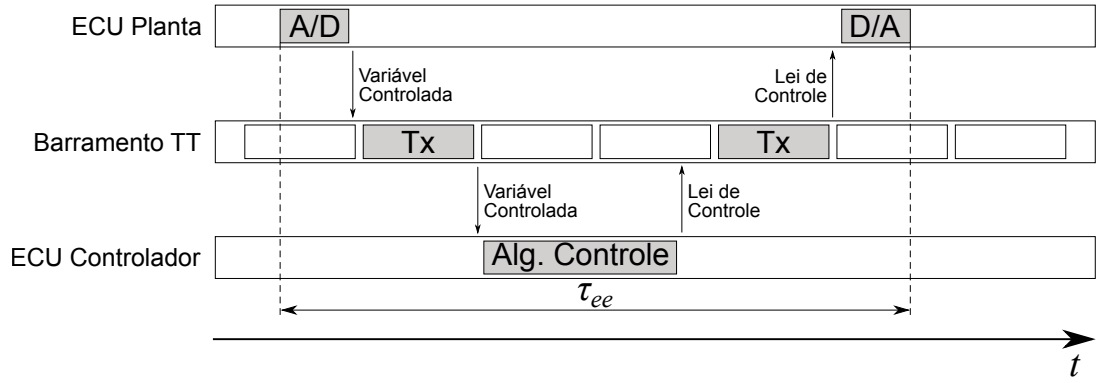

Figura 34: Tempo de latência entre o momento de amostragem do sistema e a atualização da saída do atuador, onde o laço de controle é fechado através de uma rede time-triggered. Baseado em (ALBERT, 2004).

A Figura 34 ilustra a maneira ideal de sincronizar cada etapa de uma aplicação de controle em um sistema *time-triggered*, evidenciando o intervalo τ_{ee}. Nesta figura, o *time-slot* alocado para a transmissão de dados pela "*ECU Planta*" ocorre imediatamente após o processamento dos dados da amostragem realizada pelos sensores. O algoritmo de controle (normalmente *event-triggered*) é acionado imediatamente após o recebimento dos dados e, assim que terminada sua execução, a nova lei de controle é enviada ao atuador

no *time-slot* designado. Obviamente, o tempo necessário para a execução de cada tarefa depende do *hardware* empregado. É importante observar que aqui encontra-se uma possível desvantagem dos protocolos *time-triggered* com relação a protocolos *event-triggered*: caso uma das tarefas perca sua *deadline*, o atuador somente será atualizado no ciclo de comunicação seguinte, fazendo com que τ_{ee} tenha praticamente o dobro de duração.

5.4.2 Escalonamento de Mensagens no Segmento Dinâmico

Conforme mencionado na Seção 4.3.3, no segmento dinâmico, mensagens são transmitidas somente quando necessário, sendo utilizado normalmente por mensagens não-críticas. De maneira análoga ao protocolo CAN, a mensagem que possuir menor identificador terá maior prioridade.

Uma ECU pode transmitir no segmento dinâmico quando seu contador de slots for igual ao identificador da mensagem (frame ID). A quantidade de mini-slots necessários para formar um time-slot neste segmento depende do tamanho da mensagem. Caso um nó necessite transmitir uma mensagem maior que o restante da duração do segmento, este só terá a oportunidade de transmissão no ciclo de comunicação seguinte. A Figura 35 ilustra um exemplo desta situação. Se alguma ECU estiver programada para enviar uma mensagem no time-slot 70, esta ação só poderá ocorrer no ciclo de comunicação seguinte, caso não existam mensagens ocupando o segmento previamente. No exemplo ilustrado, os time-slots 54 e 57 ocupam vários mini-slots, fazendo com que o contador de slots não alcance o time-slot 70.

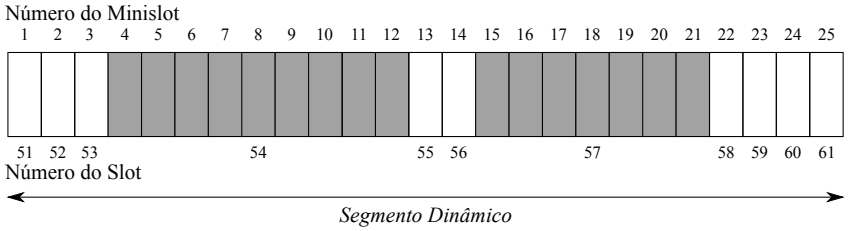

Figura 35: Escalonamento de mensagens no segmento dinâmico do protolo FlexRay. Baseado em (ARNDT, 2009).

5.4.3 Efeito do Número de Mensagens Circulantes no Barramento

Devido à política de acesso ao meio do protocolo FlexRay, a garantia de latência na transmissão de mensagens deve ser mantida mesmo em situações onde ocorra um alto tráfego de mensagens circulando pelo barramento.

5.4.4 Efeito do Aumento do Período de Amostragem

A escolha do período de amostragem do sistema depende basicamente da dinâmica do sistema a ser controlado. Intervalos de tempo muito longos entre instantes de amostragem degradam o desempenho do sistema de controle, uma vez que este será executado por um longo período sem atualização da lei de controle.

Uma situação que pode também ser considerada como uma "subta" alteração no período de amostragem acontece na ocorrência de uma perda de pacote. Vários motivos podem levar a esta situação, entre estes pode-se mencionar: a ocorrência de um erro no barramento ou nas ECUs, ou uma tarefa que não seja concluída a tempo para transmissão de mensagem no ciclo de comunicação atual.

5.4.5 Equipamentos e Softwares Utilizados

Para a realização da parte experimental, foram utilizados dois dispositivos VN8910A (VECTOR INFORMATIK GMBH, 2013b), cada um equipado com um módulo VN8970. O software CANoe.FlexRay (VECTOR INFORMATIK GMBH, 2014b) (juntamente com suas extensões) foi utilizado para a programação dos equipamentos, análise dos dados e, também, para elaboração e alteração da tabela de escalonamento do protocolo FlexRay. Estas são ferramentas desenvolvidas pela empresa alemã *Vector Informatik GmbH*.

5.4.5.1 CANoe.FlexRay

O software CANoe é uma ferramenta utilizada para desenvolvimento, teste e análise de redes automotivas e ECUs. A opção CANoe.FlexRay oferece suporte aos protocolos CAN, LIN, FlexRay e MOST.

Todo o desenvolvimento de sistemas distribuídos através do CANoe é baseado em bases de dados que contém descrições de ECUs, parâmetros de configuração do protocolo de comunicação, descrições completas das mensagens trocadas entre ECUs e tabelas de escalonamento. Desta forma, pode-se dividir o projeto de um sistema distribuído utilizando esta ferramenta da seguinte forma: projeto da base de dados com todas as informações sobre a rede e seus componentes, elaboração da tabela de escalonamento e topologia de rede, programação do comportamento das ECUs e, por fim, simulação e análise do comportamento do *cluster*.

A opção CANoe.FlexRay suporta dois formatos de bases de dados: FIBEX (*Fieldbus Exchange Format*) e DBC (*DBC Communication Database for CAN*). Neste trabalho, a primeira é utilizada para descrição das informações do *cluster* FlexRay. O segundo tipo de base de dados foi desenvolvido para a descrição de sistemas que utilizam o protocolo CAN. Entretanto, no ambiente do CANoe, pode ser utilizada, também, como uma base de dados de suporte, armazenando informações sobre variáveis de ambiente (utilizadas para monitoramento da operação do sistema).

A programação do comportamento das ECUs no CANoe é realizada utilizando a linguagem CAPL (*Communication Access Programming Language*), a qual possui síntaxe muito parecida à da linguagem C/C++. A programação é feita utilizando a ferramenta *CAPL Browser*.

Simulação e análise de dados são realizadas pelo próprio CANoe. É possível, inclusive, gerar arquivos de *log* com os dados da comunicação. A análise de dados pode ser realizada, ainda, na forma gráfica.

5.4.5.2 Vector FIBEX Explorer Pro

Como mencionado na seção anterior, toda a descrição do sistema distribuído é armazenada em uma base de dados FIBEX. Para visualização e modificação da descrição da rede, a ferramenta *Vector FIBEX Explorer Pro* foi utilizada. Através desta ferramenta, é possível alterar rapidamente, por exemplo, a tabela de escalonamento do protocolo e a periodicidade de mensagens. Uma ferramenta particularmente útil incluída neste software, é a verificação de integridade dos parâmetros da comunicação. A cada alteração da base de dados, é recomendável executar o teste de integridade para verificar se a alteração de algum parâmetro não terá influência em outros previamente estabelecidos.

5.4.5.3 CANdb++ Editor

A base de dados DBC foi utilizada para armazenar as descrições das variáveis de ambiente utilizadas durante a operação do sistema. Para visualizar e alterar os dados destas variáveis foi utilizada a ferramenta *CANdb++ Editor*.

5.4.5.4 Sistema VN8900

O conjunto de equipamentos que compõe o sistema VN8900, quando utilizado com o software CANoe, forma uma plataforma de última geração para simulação e análise de ECUs e redes automotivas. Este sistema é composto por uma unidade base, chamada VN8910A, e seus módulos (*plug-ins*) VN8950 e VN8970, sendo que apenas este último oferece suporte para a comunicação via FlexRay.

A unidade base VN8910A possui um processador interno próprio Intel ATOM, cuja frequência de *clock* é de $1,6$ GHz, e conta com 4 GB de memória RAM. O módulo VN8970 oferece a possibilidade de comunicação por quatro protocolos paralelamente: CAN, FlexRay, LIN e J1708. Internamente, o *transceiver* de cada canal é implementado em uma placa separada ao módulo VN8970. A arquitetura deste sistema é ilustrada pela Figura 36. A Figura 37 ilustra uma foto do sistema utilizado com módulo VN8950.

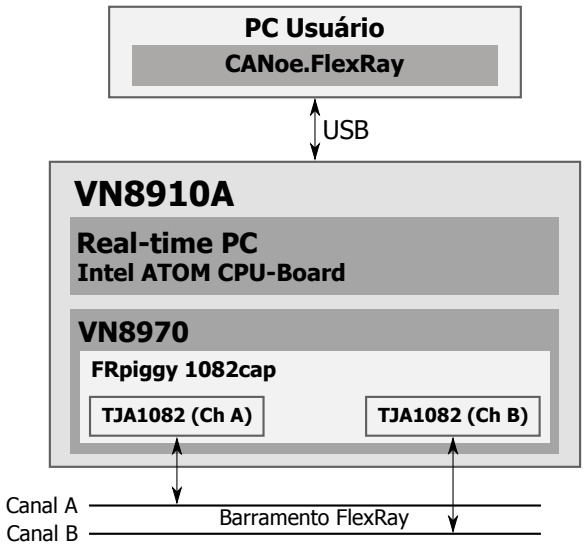

Figura 36: Arquitetura da interface de rede VN8900 e comunicação via USB com o PC do usuário. Baseado em (VECTOR INFORMATIK GMBH, 2013b).

5.4.6 Arranjo Físico

Como arranjo físico, foi considerada uma rede com duas ECUs, interligadas utilizando a topologia ponto-a-ponto, com terminações de 100 Ω nos terminais de ambos dispositivos. A Figura 38 ilustra o arranjo mencionado. Como apenas duas unidades da interface VN8900 estavam disponíveis, esta configuração física não será alterada.

5.5 Definição dos Experimentos e Objetivos

5.5.1 Métricas de Desempenho

A avaliação dos resultados obtidos com os experimentos é realizada através de algumas métricas das áreas de comunicação e controle:

Figura 37: Foto da unidade base VN8910A com módulo VN8950 (VECTOR INFORMATIK GMBH, 2013b).

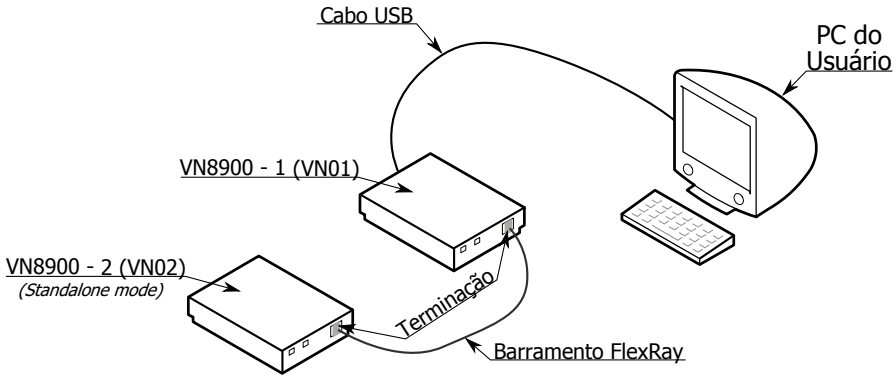

Figura 38: Arranjo físico elaborado para os experimentos.

- *Máximo Sobrepasso (M_p)*
- *Tempo de Acomodação (t_s)*
- *Periodicidade das Mensagens de Controle*

  A periodicidade das mensagens de controle pode ser mensurada através da diferença entre os instantes de transmissão (medidos através do *time-stamp*) da mesma.

- *Atraso de fim-a-fim (τ_{ee})*

  Alguns dos trabalhos analisados no capítulo 3 consideram como atraso de fim-a-fim o intervalo entre os instantes de transmissão e recebimento de uma mensagem, antes que esta atinja a camada de aplicação. Entretanto, aqui será empregada a abordagem apresentada em (ALBERT, 2004), o qual considera o intervalo entre os instantes de amostragem da saída da planta e de atualização da saída do atuador.

5.5.2 Experimento 1

O primeiro experimento tem como objetivo avaliar o desempenho dos controladores elaborados durante a Seção 5.3 quando o laço de controle é fechado através da rede de comunicação. Este experimento será subdividido em três, de forma a avaliar individualmente o comportamento do sistema utilizando cada um dos controladores.

A Figura 39 ilustra a configuração lógica deste experimento. O conjunto sensor-planta-atuador será simulado por uma ECU, enquanto o algoritmo de controle será executado por outro dispositivo. As descrições dos processos de elaboração dos ciclos de

comunicação e das tabelas de escalonamento de mensagens serão abordadas no próximo capítulo.

O sistema será executado em diferentes condições de amostragem da rede. Para a realização deste tipo de procedimento, será empregada a capacidade de multiplexação de *time-slots* do protocolo FlexRay, alterando assim a frequência de transmissão das mensagens da aplicação de controle. Nesta configuração, apenas as mensagens de controle circularão pelo barramento.

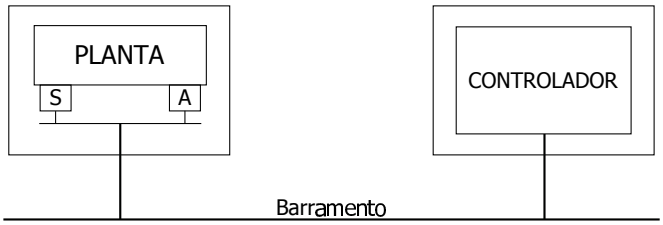

Figura 39: Configuração lógica para o experimento 1. S: sensores. A: atuador.

5.5.3 Experimento 2

O principal objetivo do experimento 2 é observar qual a influência do tráfego de mensagens transmitidas no segmento dinâmico sobre a periodicidade das mensagens transmitidas no segmento estático. Neste estudo de caso, o segmento estático continuará transportando apenas as mensagens da aplicação de controle, enquanto o segmento dinâmico será preenchido gradualmente por mensagens "vazias" (*Dummy Messages*), utilizadas apenas para adicionar tráfego à rede.

Para avaliar esta situação, apenas o controlador 3 (realimentação de estados baseada na técnica de dados amostrados) será aplicado, por apresentar maior robustez a variações nos períodos de amostragem da planta, caso que pode ser representado na ocorrência de alguma perda de pacote durante o processo. A Figura 40 ilustra a configuração lógica para a execução deste experimento. À rede elaborada para o experimento anterior, foram adicionados os componentes TG1 (*Traffic Generator 1*) e TG2 (*Traffic Generator 2*). Ambos são ECUs programadas com o único objetivo de alterar as condições de tráfego no barramento, sendo responsáveis pelo envio das mensagens vazias.

Como resultados, serão obtidas as caracterizações do intervalo τ_{ee} e a periodicidade das mensagens de controle, além da avaliação do impacto causado no desempenho da aplicação de controle.

Figura 40: Configuração lógica utilizada para os experimentos 2 e 3.

5.5.4 Experimento 3

O terceiro experimento pode, de certa forma, ser considerado como uma extensão do segundo. Mais uma vez, o principal objetivo é observar qual o impacto do alto tráfego de mensagens circulantes pelo barramento sobre as características temporais das mensagens da aplicação de controle. O que diferencia este experimento do anterior, é que, neste caso,

tanto o segmento estático quanto o segmento dinâmico se encontrarão em uma situação de alto tráfego de mensagens. Para esta análise, novamente, apenas o controlador 3 será empregado. Neste experimento, serão realizadas, ainda, as caracterizações do intervalo τ_{ee} e periodicidade das mensagens de controle, além da avaliação do impacto causado na aplicação de controle. A configuração lógica empregada é a mesma utilizada no experimento anterior, conforme ilustrado pela Figura 40.

6 IMPLEMENTAÇÃO E RESULTADOS

6.1 Introdução

Este capítulo apresenta, inicialmente, os resultados das simulações do sistema operando sem a rede. São exibidos, também, os passos realizados para a implementação do estudo de caso e, por fim, os resultados obtidos a partir dos experimentos descritos no capítulo anterior.

6.2 Simulações em MATLAB/Simulink

6.2.1 Avaliação dos Controladores no Cenário Ideal

Inicialmente, o sistema em malha fechada utilizando cada um dos controladores elaborados foi avaliado através de simulações em MATLAB/Simulink. A função dada por (63), e representada pela Figura 41, é considerada para um distúrbio em posição (x_r). A derivada da função de perturbação em posição, dada pela equação (64) e exibida pela Figura 42, é utilizada como distúrbio em velocidade ($\dot{x}_r$).

$$x_r = \begin{cases} 0,025 + 0,025\text{sen}(2\pi t - \pi/2), & 0 \leq t \leq 1\ s \\ 0, & t > 1\ s \end{cases} \tag{63}$$

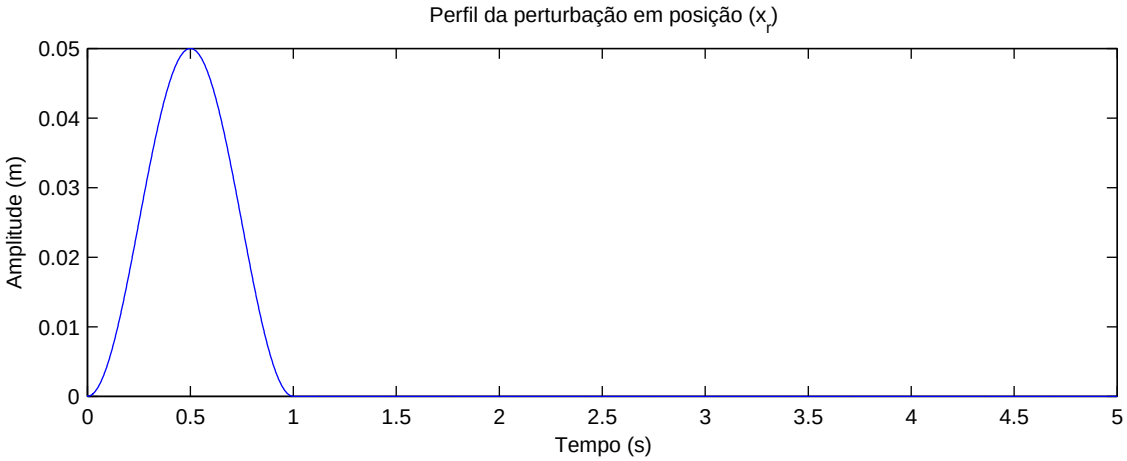

Figura 41: Perfil da perturbação em posição (x_r).

$$\dot{x}_r = \begin{cases} 2\pi \cdot 0,025\cos(2\pi t - \pi/2), & 0 \leq t \leq 1\ s \\ 0, & t > 1\ s \end{cases} \tag{64}$$

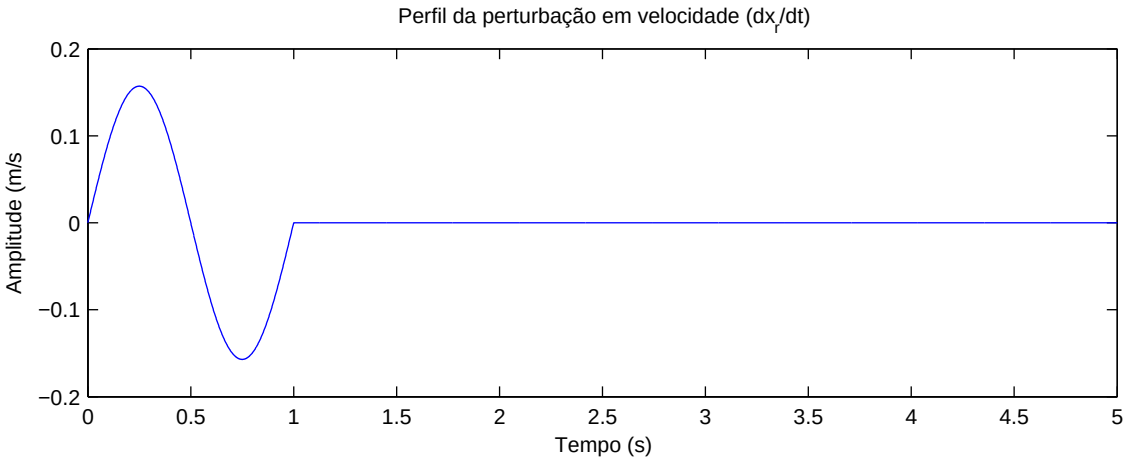

Figura 42: Perfil da perturbação em velocidade ($\dot{x}_r$).

A Figura 43 exibe os resultados obtidos das simulações do sistema contínuo em malha fechada, a partir de cada um dos três controladores calculados, para a função de transferência $x_{def}/\dot{x}_r$. Realizou-se, também, uma comparação com a resposta do sistema em malha aberta. Como pode-se observar, a utilização de cada um dos controladores reduziu a amplitude da deflexão da suspensão quando comparados à resposta do sistema em malha aberta. O tempo de acomodação também é bastante reduzido nas três situações. Para a simulação do sistema utilizando o controlador (3) (dados amostrados), foi considerada a situação com atraso nulo ($\tau(t) = 0$).

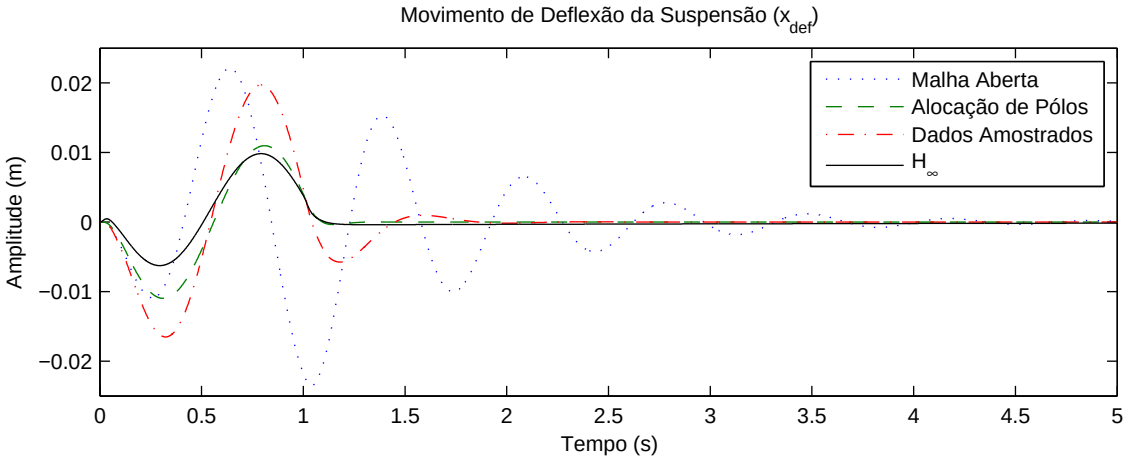

Figura 43: Resposta à perturbação. Avaliação da função de transferência $x_{def}/\dot{x}_r$

6.3 Implementação

Como discutido anteriormente, em protocolos *time-triggered*, a definição dos parâmetros da comunicação é realizada durante a fase de projeto. Desta forma, o primeiro passo para a implementação é a descrição completa do sistema distribuído através da criação de uma base de dados FIBEX.

6.3.1 Definição dos Ciclos de Comunicação

Foram elaborados dois diferentes ciclos de comunicação. O primeiro possui período de 5 *ms*, enquanto o segundo apresenta período de 1 *ms*.

6.3.1.1 Ciclo de Comunicação com Período de 5 *ms*

O primeiro ciclo de comunicação elaborado possui período de 5 *ms*. Os parâmetros deste ciclo são baseados no estudo de caso realizado pela empresa BMW na utilização do protocolo em alguns de seus novos veículos (SCHEDL, 2007) e são fornecidos por padrão quando uma nova base de dados FIBEX é criada utilizando o software Vector FIBEX Explorer Pro. As principais características deste ciclo são listadas pela Tabela 5.

Tabela 5: Principais parâmetros do ciclo de comunicação com período de 5 *ms*

Parâmetro	Valor	Unidade
Período do Ciclo de Comunicação	5000	μs
Tempo de Bit	$0,1$	μs
Macrotick (MT)	$1,375$	μs
Número de slots estáticos por ciclo	91	–
Duração de slot estático	24	MT
Tamanho do Segmento de Dados	8	$WORD$
Número de Mini Slots	289	–
Duração de um Mini Slot	5	MT
NIT	7	MT

A Figura 44 ilustra a representação deste ciclo de comunicação, juntamente com seus principais parâmetros, conforme exibido pelo software Vector FIBEX Explorer Pro.

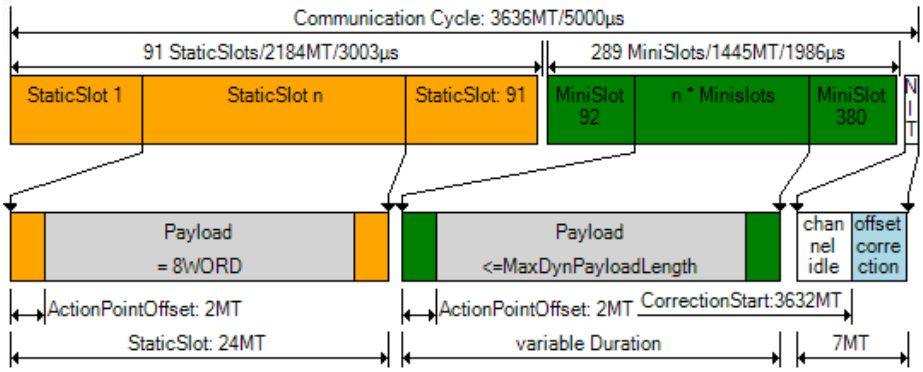

Figura 44: Ciclo de comunicação de período 5 *ms* conforme exibido pelo software Vector FIBEX Explorer Pro.

6.3.1.2 Ciclo de Comunicação com Período de 1 *ms*

O segundo ciclo de comunicação elaborado possui período de 1 *ms*. Neste caso, apenas o segmento estático (com duração total de $0,957$ *ms*) e o tempo inativo de rede estão presentes. As principais características deste ciclo de comunicação são listadas pela Tabela 6.

Tabela 6: Principais parâmetros do ciclo de comunicação de período 1 *ms*

Parâmetro	Valor	Unidade
Período do Ciclo de Comunicação	1000	μs
Tempo de Bit	0,1	μs
Macrotick (MT)	1	μs
Número de slots estáticos por ciclo	29	–
Duração de slot estático	33	*MT*
Tamanho do Segmento de Dados	8	*WORD*
Número de Mini Slots	0	–
NIT	43	MT

A Figura 45 ilustra a representação deste ciclo de comunicação, juntamente com seus principais parâmetros, conforme exibido pelo software Vector FIBEX Explorer Pro.

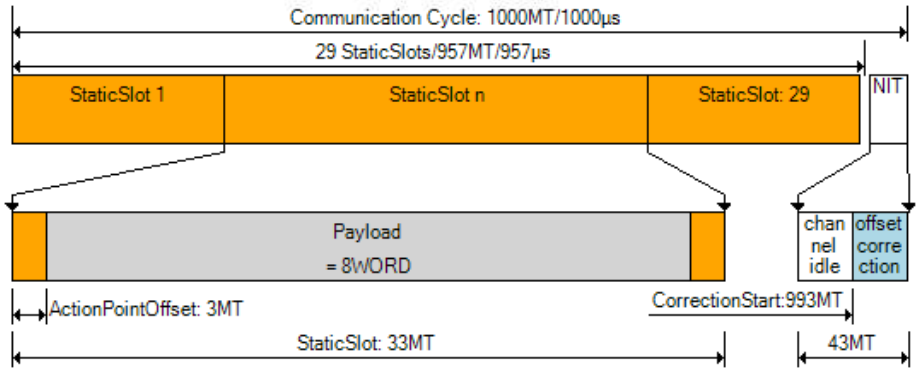

Figura 45: Ciclo de comunicação de período 1 *ms* conforme exibido pelo software Vector FIBEX Explorer Pro.

6.3.2 Definição de ECUs, Sinais e Quadros

6.3.2.1 Electronic Control Units – ECUs

Como mencionado brevemente durante a Seção 5.5, foram utilizadas quatro ECUs lógicas durante os experimentos. O mapeamento de cada ECU lógica para cada interface VN8900 depende do experimento em questão e será abordado posteriormente. A Tabela 7 exibe uma lista com as quatro ECUs utilizadas e suas funções na rede.

Tabela 7: ECUs utilizadas no sistema distribuído

Nome	Função
ECU Suspensão	Simular a planta, sensores e atuador
ECU Controlador	Executar os algoritmos de controle
ECU TG1	Aumentar o tráfego de mensagens na rede
ECU TG2	Aumentar o tráfego de mensagens na rede

A *ECU Suspensão* é responsável por simular a planta, sensores e atuador, além de realizar a coleta de algumas informações para posterior análise de desempenho do *cluster*.

A *ECU Controlador* é responsável por executar os algoritmos de controle. As *ECUs TG1 e TG2* são utilizadas apenas para gerar tráfego na rede. Nenhum processamento lógico é executado em ambas, sendo que sua função se limita a enviar as mensagens "*Dummy*" nos *time-slots* designados. A Figura 46 ilustra a rede FlexRay com as quatro ECUs no ambiente do software CANoe.

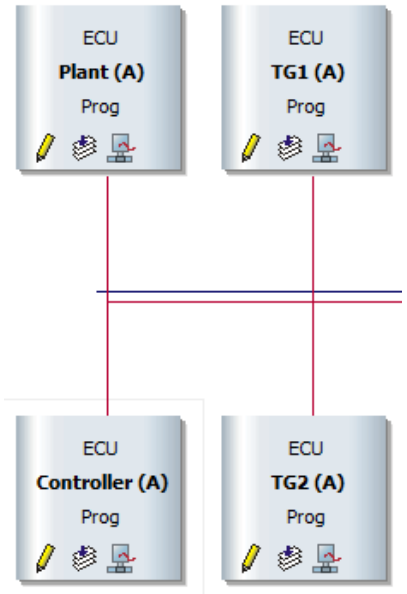

Figura 46: Rede FlexRay com as ECUs elaboradas no ambiente do software CANoe.

6.3.2.2 Sinais

Em sistemas embarcados automotivos, os dados que trafegam pela rede são denominados *sinais*. Dadas as descrições realizadas na Seção 5.5, existem dois tipos de sinais necessários para todos os experimentos: os sinais das aplicações de controle e os sinais utilizados para gerar tráfego na rede. Na primeira categoria, os sinais das aplicações de controle são baseados nas informações que precisam ser transmitidas entre as diferentes ECUs que formam a malha fechada de controle. Como os sinais de *feedback* destas aplicações são os estados da planta, ou um sub-conjunto destes, os sinais que irão trafegar no barramento serão os próprios estados. Na segunda categoria, encontram-se os sinais utilizados apenas para gerar tráfego na rede. Estes foram elaborados apenas para preecher o segmento de dados de um quadro FlexRay. A Tabela 8 exibe a lista completa de todos os sinais elaborados para as aplicações dos experimentos.

6.3.2.3 Quadros

Os quadros representam o conteúdo das mensagens que serão transmitidas através do barramento, e podem ser considerados como uma espécie de "containers" que transportam os sinais das aplicações. No segmento estático do protocolo FlexRay, apenas um quadro é transmitido em cada *time-slot*. Foram elaborados quatro diferentes quadros que poderão ser utilizados por todos os experimentos. As descrições destes se encontram na Tabela 9.

Tabela 8: Sinais do sistema distribuído

Sinal	Descrição	Tipo	Tamanho
$xdef$	Deflexão da suspensão	$float$	32 bits
dxs	Velocidade vertical do corpo do veículo	$float$	32 bits
$xs - xr$	Deflexão do pneu	$float$	32 bits
$dxus$	Velocidade vertical da roda do veículo	$float$	32 bits
cl	Lei de controle	$float$	32 bits
$DummySig1$	Sinal *Dummy* 1	$float$	64 bits
$DummySig2$	Sinal *Dummy* 2	$float$	64 bits
$DummySig3$	Sinal *Dummy* 3	$float$	64 bits
$DummySig4$	Sinal *Dummy* 3	$float$	64 bits

Tabela 9: Quadros FlexRay do sistema distribuído

Quadro	Descrição	Tamanho (Dados)	Origem	Destino
$FrStates$	Quadro para transmissão dos estados	128 bits	ECU Suspensão	ECU Controle
$FrControlLaw$	Quadro para transmissão da nova lei de controle	128 bits	ECU Controle	ECU Suspensão
$FrDummy1$	Quadro utilizado para aumento de tráfego	128 bits	ECU TG1	ECU TG2
$FrDummy2$	Quadro utilizado para aumento de tráfego	128 bits	ECU TG2	ECU TG1

6.3.2.4 Mapeamento dos Sinais aos Quadros FlexRay

O mapeamento dos sinais aos quadros FlexRay ocorre conforme indicado pelas Figuras de 47 a 50. A Figura 47 exibe o mapeamento dos sinais que representam os estados da planta no quadro FlexRay. A Figura 48 mostra o mapeamento do sinal correspondente à nova lei de controle. Neste caso, apenas uma pequena parte do segmento de dados do quadro foi utilizado devido ao tamanho escolhido para este sinal. A Figura 49 exibe o mapeamento do primeiro conjunto de sinais para geração de tráfego na rede. Como pode-se observar, apenas dois sinais podem ser transportados no quadro devido ao tamanho de cada sinal ser maior do que no caso dos sinais da aplicação de controle. Por fim, a Figura 50 ilustra o mapeamento do segundo conjunto de sinais utilizados para gerar tráfego na rede.

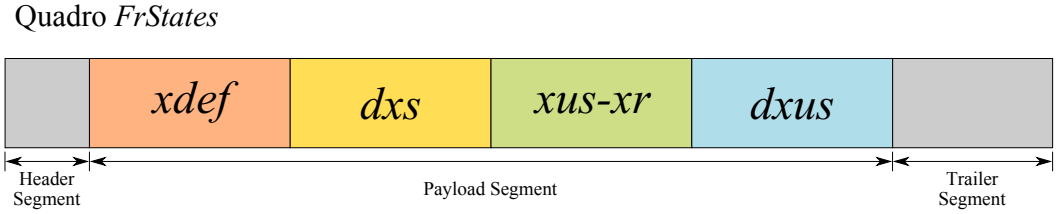

Figura 47: Mapeamento de sinais no quadro $FrStates$.

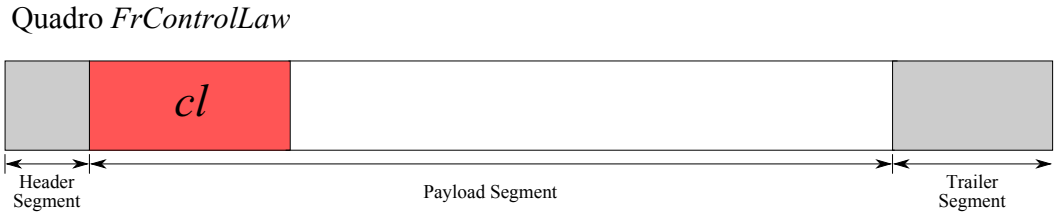

Figura 48: Mapeamento de sinais no quadro *FrControlLaw*.

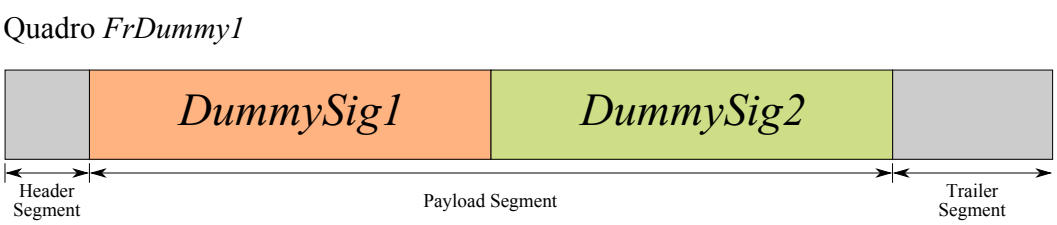

Figura 49: Mapeamento do primeiro conjunto de sinais para gerar tráfego na rede (*DummySig1* e *DummySig2* no quadro *FrDummy*1).

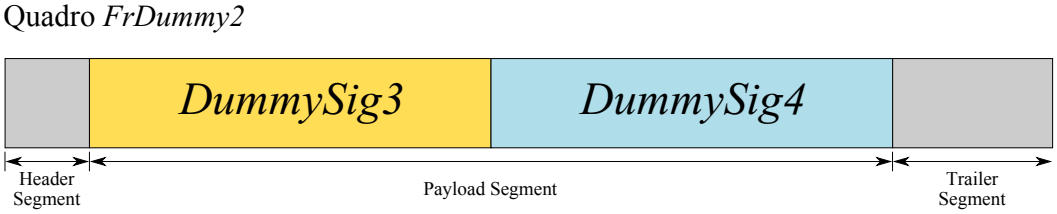

Figura 50: Mapeamento do segundo conjunto de sinais para gerar tráfego na rede (*DummySig3* e *DummySig4* no quadro *FrDummy*2).

6.3.3 Teste de Tempo de Resposta das Interfaces VN8900

Este primeiro teste tem basicamente dois objetivos:

(1) avaliar qual o tempo necessário para que o hardware que simula o conjunto *sensor-planta-atuador* atualize o *buffer* de seu controlador de comunicação.

(2) avaliar qual o tempo necessário para que o hardware que implementa o algoritmo de controle realize o processamento das informações e atualize o *buffer* de seu controlador de comunicação.

Esta avaliação temporal se faz necessária para que o escalonamento das mensagens possa seguir, da melhor maneira possível, o cenário ideal mencionado na Seção 5.4.1, e servirá de base para a implementação e execução dos experimentos descritos no capítulo anterior.

Neste experimento, a configuração lógica é idêntica à configuração física, conforme ilustrado pela Figura 38. Para o dispositivo VN01 será mapeada a ECU Suspensão, enquanto para o dispositivo VN02 será mapeada a ECU Controlador. O comportamento destas duas ECUs será alterado para atender aos requisitos deste teste. Neste cenário, durante o ciclo de comunicação 2, a ECU Suspensão enviará inicialmente à ECU Controlador um conjunto de quatro números de 32 bits cada, em um único *time-slot*, utilizando os sinais definidos para a aplicação de controle e o mapeamento destes no quadro *FrStates*. A ECU Controlador, após recebimento dos dados, irá executar o algoritmo de controle e

enviará o resultado de volta à primeira. Este procedimento é repetido durante o ciclo de comunicação 10. Entretanto, desta vez a ECU Suspensão enviará um conjunto de números diferentes à ECU Controlador.

Os dois conjuntos de números enviados de VN01 a VN02 serão definidos e conhecidos a priori. A avaliação temporal ocorrerá de forma "indireta", pois devido às características das ferramentas utilizadas e da linguagem CAPL, não é possível realizar medições de tempo diretamente. Como solução, a medição será baseada na temporização do ciclo de comunicação.

Este teste fará uso do ciclo de comunicação com período de 1 *ms*. Inicialmente, o escalonamento de mensagens será feito da seguinte forma: (i) o *slot* estático 1 será alocado para a ECU Suspensão e (ii) o *slot* estático 2 será alocado para a ECU Controlador. Caso os equipamentos não consigam transmitir os dados mais atuais em seus *time-slots* designados durante os ciclos de comunicação 2 ou 10, a alocação de *time-slots* será alterada para atender as necessidades de cada dispositivo de hardware. No início de cada ciclo de comunicação, será executado o comando para que a ECU Suspensão atualize seu *buffer* de comunicação. Para avaliar o resultado, observa-se em qual *time-slot* do segmento estático a transmissão realmente ocorre. Procedimento parecido é adotado para a ECU Controlador. Como os dados recebidos por esta última são conhecidos a priori, o resultado do algoritmo de controle também o será. Portanto, basta verificar em qual *time-slot* o resultado correto da lei de controle será transmitido.

A Figura 51 resume a situação. O evento $FrStartCycle$ marca o início de cada ciclo de comunicação, e é utilizado para demarcar o instante em que a ECU Suspensão recebe a ordem para atualizar o *buffer* de seu controlador de comunicação. O objetivo é determinar os *time-slots* "*Slot i*" e "*Slot j*", onde os dispositivos de hardware sejam capazes de responder com as informações corretas.

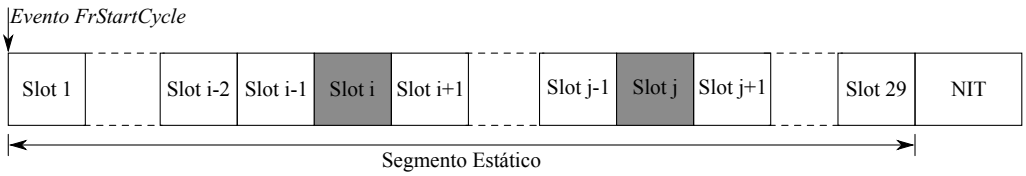

Figura 51: Esquema para teste de tempo de resposta.

6.3.3.1 Resultados dos testes de tempo de resposta

Para os três algoritmos de controle, os resultados foram os mesmos. Para o caso do dispositivo VN01, os resultados corretos apareceram no barramento apenas durante o *time-slot* 14. Considerando os parâmetros exibidos pela Tabela 6, isto equivale a um atraso no intervalo $[396; 429]\ \mu s$.

No caso da interface VN02, os resultados corretos apareceram no barramento apenas durante o *time-slot* 28, o que resulta, também, em um atraso dentro do mesmo intervalo $[396; 429]\ \mu s$

6.3.4 Escalonamento de Mensagens para os Experimentos

O escalonamento das mensagens da aplicação de controle segue os resultados obtidos com o teste de tempo de resposta das interfaces VN8900. Para os experimentos onde o período de amostragem das saídas da planta é menor que 5 *ms*, o ciclo de comunicação

com período de 1 *ms* será utilizado. Este ciclo é mais curto e possui no máximo 29 *time-slots*. Para situações onde o período de amostragem da rede é igual ou superior a 5 *ms*, o ciclo de comunicação com período de 5 *ms* é utilizado. Os *time-slots* designados para cada uma das mensagens, de acordo com o ciclo de comunicação utilizado, são listados pela Tabela 10.

Tabela 10: Escalonamento das mensagens da aplicação de controle

Mensagem	Período do Ciclo de Comunicação	
	1 *ms*	5 *ms*
FrStates	Slot 15	Slot 20
FrControlLaw	Slot 29	Slot 40

6.4 Resultados dos Experimentos

6.4.1 Experimento 1

A Figura 52 ilustra a configuração lógica utilizada neste experimento. Aqui, o objetivo é avaliar o comportamento do sistema em malha fechada utilizando os controladores desenvolvidos anteriormente. Neste arranjo, os sensores são *time-triggered*, realizando amostragens do processo em intervalos constantes. Tanto controlador quanto atuador são sistemas *event-triggered*, executando seus algoritmos apenas quando uma mensagem é recebida. Neste cenário, apenas as mensagens da aplicação de controle circulam pelo barramento.

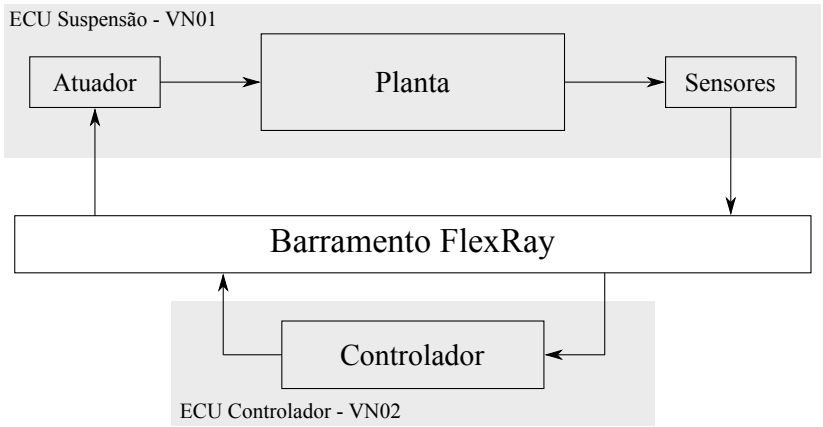

Figura 52: Diagrama da configuração lógica do primeiro experimento.

6.4.1.1 Controlador 1 (Alocação de pólos via realimentação de estados)

A Figura 53 mostra o comportamento da deflexão da suspensão (estado x_{def}) quando o controlador 1 é aplicado sob diferentes períodos de amostragem da rede em uma comparação com o caso contínuo. A degradação de desempenho é praticamente insignificante para períodos de amostragem de até 32 *ms*, conforme evidenciado pelo máximo sobrepasso e pelo tempo de acomodação. Quando selecionado um período de amostragem de 64 *ms*, o sistema torna-se instável. Não foi possível determinar experimentalmente o limite para o

período de amostragem no qual o sistema torna-se instável devido às características das ferramentas utilizadas.

Figura 53: Resposta à perturbação da função de transferência x_{def}/x_r para vários períodos de amostragem da rede quando o controlador 1 é aplicado.

6.4.1.2 Controlador 2 (Realimentação dinâmica de saída)

Apesar de apresentar o melhor desempenho em um cenário ideal, o controlador 2 não pode ser aqui avaliado, uma vez que, mesmo utilizando o ciclo de comunicação de 1 *ms*, o sistema apresentou comportamento instável, conforme ilustrado pela Figura 54.

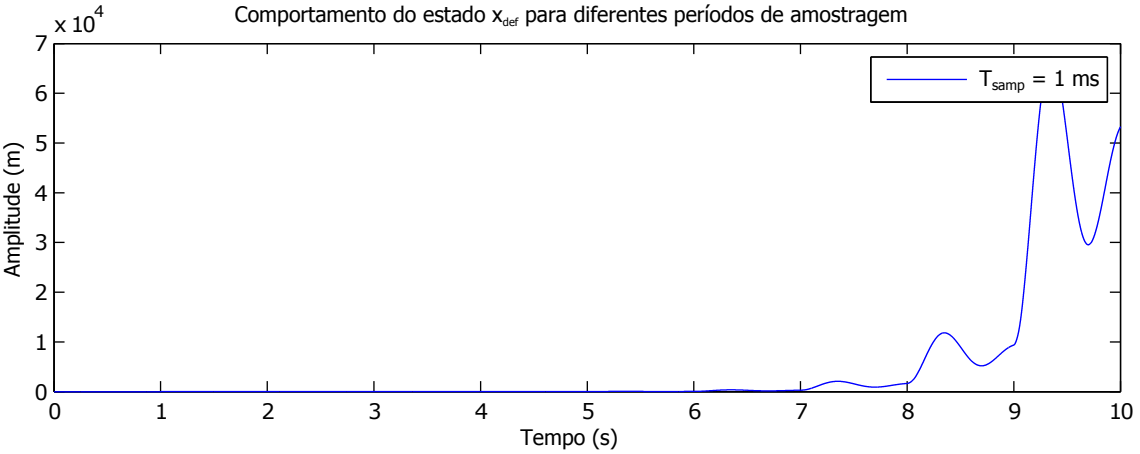

Figura 54: Resposta do estado x_{def} do sistema operando em malha fechada utilizando o controlador 2 com período de amostragem de 1 *ms*.

Um intervalo de tempo menor que 1 *ms* não é possível de se alcançar utilizando as interfaces VN8900 porque, conforme mostrado pelos testes de tempo de resposta, o escalonamento de mensagens foi feito no limite para que os dispositivos conseguissem responder dentro do mesmo ciclo.

6.4.1.3 Controlador 3 (Realimentação de estados baseada na metodologia de dados amostrados)

A análise para o terceiro controlador segue aquela realizada para o primeiro. A Figura 55 exibe uma comparação do comportamento do estado x_{def} entre o caso ideal (simulação

do caso contínuo) e diferentes períodos de amostragem da rede. De forma parecida ao controlador 1, para períodos de amostragem de até 40 *ms* a degradação de desempenho não é muito acentuada. Entretanto, para períodos maiores, percebe-se, claramente, através do sobrepasso, a perda de desempenho. Por outro lado, este controlador apresenta grande robustez a variações no período de amostragem, inclusive, suportando longos períodos sem nenhuma nova informação de controle sem que o sistema se instabilize. Quando o período de amostragem da rede foi estabelecido em 320 *ms*, o sistema se instabilizou.

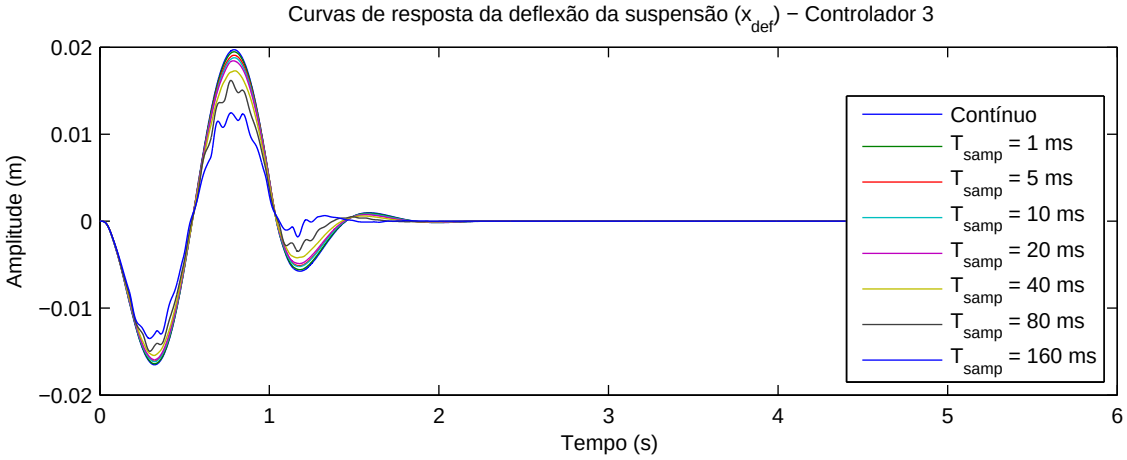

Figura 55: Resposta à perturbação da função de transferência x_{def}/x_r para vários períodos de amostragem da rede quando o controlador 3 é aplicado.

6.4.2 Experimento 2

O experimento 2 contém uma série de avaliações temporais. A medição temporal se dá com base nas capacidades das interfaces VN8900. Dois parâmetros importantes medidos são o instante de amostragem dos estados da planta, t_{ps}, e o instante de atualização do saída do atuador, t_{ua}. O primeiro é medido imediatamente após a atualização das variáveis com os estados da planta. De forma análoga, o segundo é medido imediatamente após a atualização da variável que representa a entrada de controle no sistema. Ambas medições são realizadas na camada de aplicação, através da função CAPL *timeNowNS()*. Esta função retorna o instante da simulação em nanossegundos. Todas as medições são realizadas na interface VN01, uma vez que a interface VN02 trabalha no modo *standalone* (sem contato com o PC do usuário durante o funcionamento do sistema).

Com relação às ECUs TG1 e TG2, a primeira foi implementada na interface VN01, enquanto a segunda foi implementada na interface VN02.

6.4.2.1 Avaliação da Periodicidade das Mensagens de Controle

Para a avaliação da periodicidade das mensagens de controle, a medição de tempo é baseada na leitura do *time-stamp* de cada quadro. Segundo a documentação do CANoe (VECTOR INFORMATIK GMBH, 2013c), o *time-stamp* em mensagens FlexRay é gerado durante o momento da transmissão imediatamente após o término do *Channel Idle Delimiter*. Através da função CAPL *messageTimeNS()* é possível recuperar o *time-stamp* da mensagem em nanossegundos.

O histograma representado pela Figura 56 exibe a distribuição temporal dos períodos das mensagens de controle para cada uma das situações de carregamento do segmento

dinâmico. Como pode-se observar, independente do tráfego na parte dinâmica, não há uma grande alteração no período das mensagens de controle, sendo que a grande maioria (por volta de 60%), possuem período no intervalo [4999995; 5000005] *ns*.

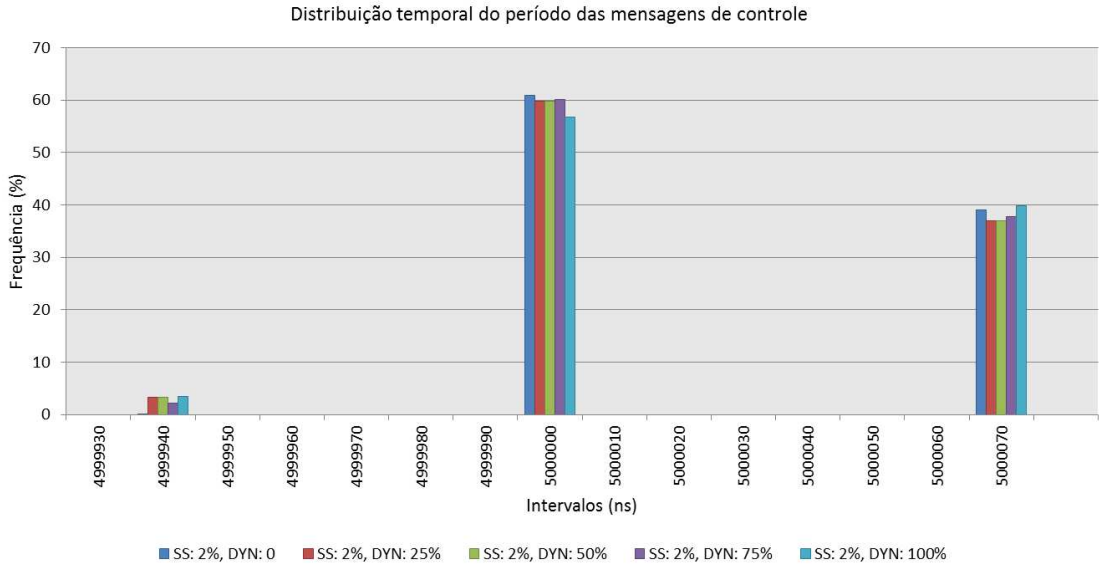

Figura 56: Distribuição temporal dos períodos das mensagens de controle.

A Figura 57 exibe os valores máximos, mínimos e médios dos períodos das mensagens de controle. O objetivo desta análise foi verificar os limites superior e inferior da periodicidade das mensagens circulantes no barramento. Como pode ser observado, independente do tráfego no segmento dinâmico, existe um limite superior para o período da mensagem de controle: 500070 *ns*. A diferença deste limite para o período nominal do ciclo de comunicação é de apenas 70 *ns*, sendo esta diferença menor que o tempo de um bit no barramento.

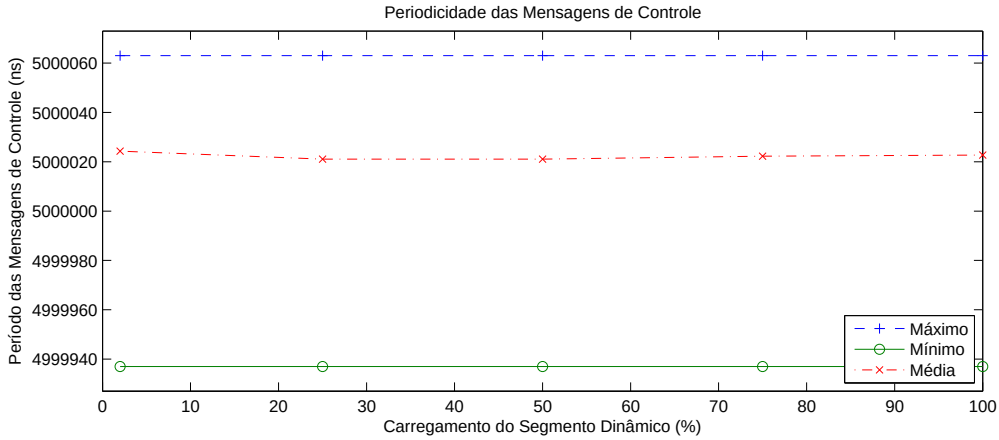

Figura 57: Limites superior e inferior da periodicidade das mensagens de controle.

6.4.2.2 Avaliação Temporal do Atraso de Fim-a-Fim

O atraso de fim-a-fim (τ_{ee}) é obtido através da diferença entre os instantes t_{ua} e t_{ps}. Baseado nas características temporais do ciclo de comunicação com período de 5 *ms* (vide

Tabela 5), este atraso deve estar contido no intervalo $[1,287;\ 1,32]$ *ms* (não considerando atrasos de transmissão e processamento), uma vez que este intervalo representa o tempo decorrido entre o início do ciclo de comunicação e o *time-slot* 40 (sendo este o *time-slot* utilizado pelo controlador para enviar a nova lei de controle à planta).

A Figura 58 exibe a distribuição temporal do atraso de fim-a-fim para as diferentes condições de tráfego no segmento dinâmico. Como pode-se observar entre 60% e 70% das amostras do atraso τ_{ee} estão presentes no intervalo $[1315315;\ 1315325]$ *ns*, indicando que todas as amostras estão dentro do intervalo esperado.

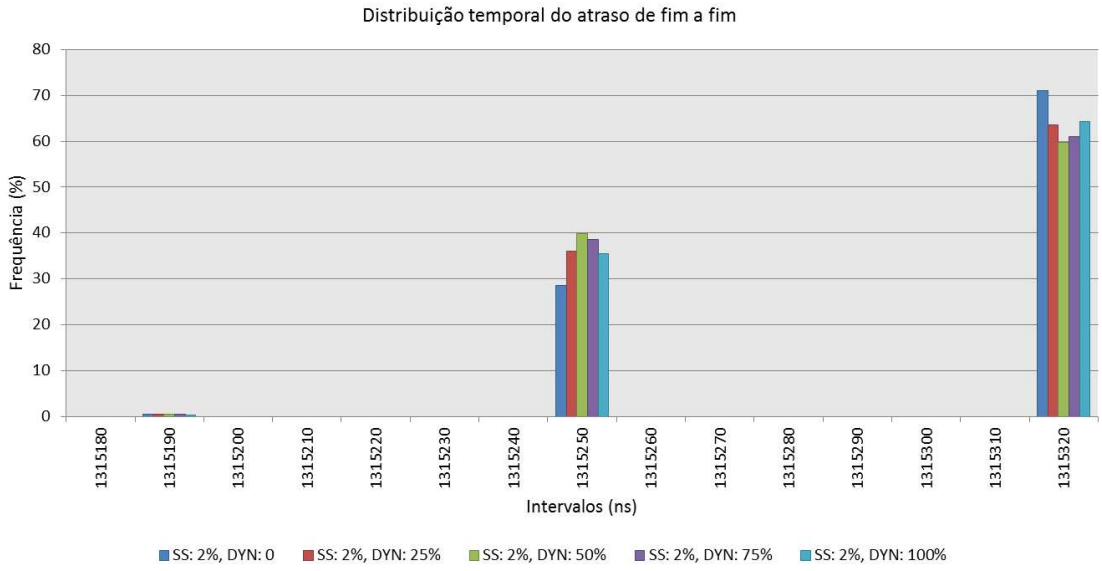

Figura 58: Distribuição temporal do atraso de fim-a-fim.

A Figura 59 exibe os valores máximos, mínimos e médios do atraso de fim-a-fim para as diferentes situações de tráfego no segmento dinâmico.

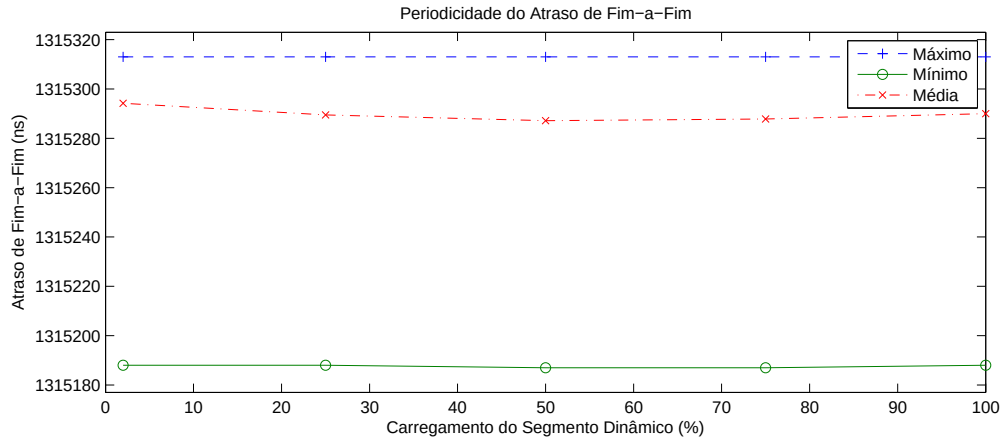

Figura 59: Atraso de fim-a-fim em função de diferentes condições de tráfego no segmento dinâmico.

6.4.2.3 *Avaliação do Desempenho do Sistema de Controle em Função das Condições de Tráfego no Segmento Dinâmico*

A influência do tráfego no segmento dinâmico sobre o desempenho do sistema de controle é ilustrada pela Figura 60, a qual exibe o comportamento do estado x_{def}. Este comportamento é consequência direta dos resultados dos atrasos presentes no laço de controle, evidenciados pelas duas seções anteriores. Entretanto, existe ainda um fator que não foi considerado anteriormente: a perda da *deadline* das tarefas de controle pelas interfaces VN8900. A Figura 61 exibe uma comparação entre os valores atuais do estado x_{def} e aqueles enviados ao controlador através do barramento (demarcado por "Sinal no Barramento" no gráfico) para a situação de maior tráfego no segmento dinâmico. A pequena área demarcada por um círculo e exibida de forma ampliada dentro do quadrado vermelho desta mesma figura, ilustra o efeito da "perda de pacotes" durante a transmissão.

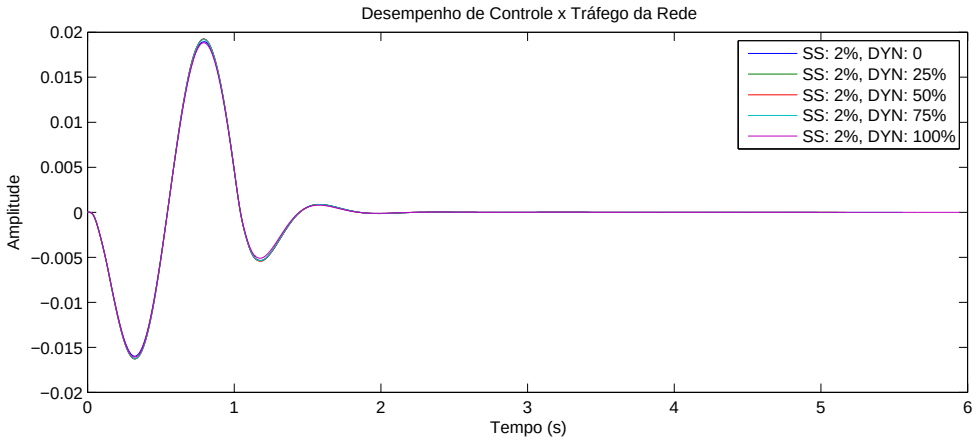

Figura 60: Resposta do estado x_{def} para diferentes condições de tráfego no segmento dinâmico.

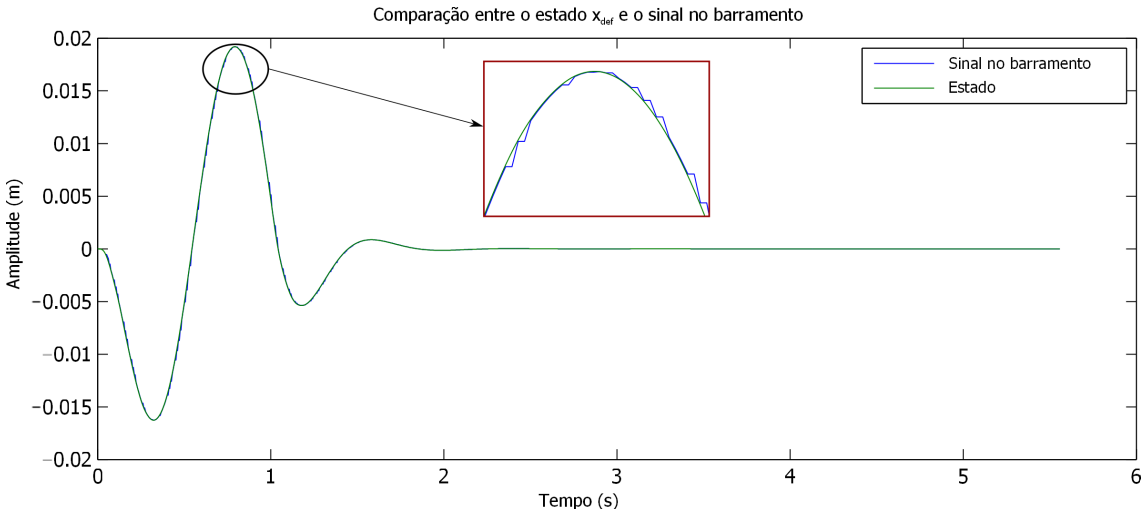

Figura 61: Comparação entre o estado x_{def} com o sinal transmitido através do barramento.

6.4.3 Experimento 3

O experimento 3 segue a mesma linha de avaliações do experimento 2. Entretanto, neste caso, haverá, também, aumento do tráfego de mensagens no segmento estático do protocolo. As mesmas funções e parâmetros do experimento 2 serão utilizados neste cenário.

6.4.3.1 Avaliação da Periodicidade das Mensagens de Controle

Como no experimento anterior, mais uma vez, a medição da periodicidade das mensagens de controle será baseada na leitura do *time-stamp* de cada mensagem, através da função CAPL *messageTimeNS()*.

O histograma apresentado pela Figura 62 exibe a distribuição temporal dos períodos das mensagens de controle para diferentes situações de tráfego no barramento. Desta vez, a condição de tráfego no segmento dinâmico é fixa e mantida em 100% de ocupação.

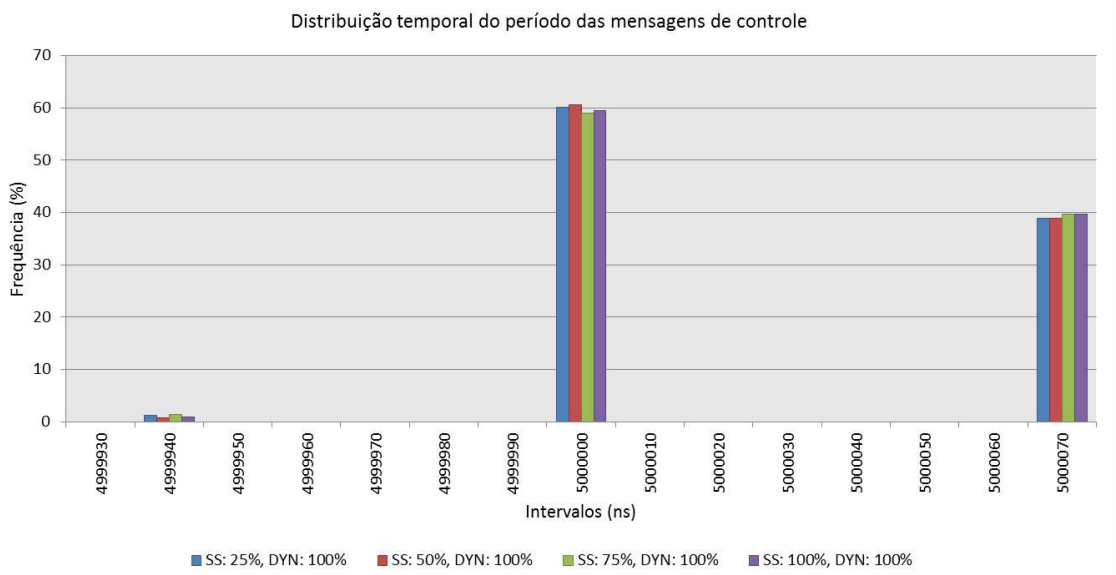

Figura 62: Distribuição temporal dos períodos das mensagens de controle.

A Figura 63 exibe os períodos máximos, mínimos e médios das mensagens de controle para diferentes condições de tráfego no barramento, de forma a melhor ilustrar os limites superior e inferior da periodicidade das mensagens nestas condições. Mais uma vez, independentemente do tráfego de mensagens presentes no barramento, um limite superior foi estabelecido em 5000070 *ns*.

6.4.3.2 Avaliação Temporal do Atraso de Fim-a-Fim

Como a Tabela de escalonamento do sistema não foi alterada do segundo experimento para o terceiro, o atraso de fim-a-fim deve permanecer entre os mesmos limites da análise anterior: $[1,287;\ 1,320]$ *ms*.

A Figura 65 exibe os valores máximos, mínimos e médios do atraso de fim-a-fim para as situações de tráfego no segmento estático.

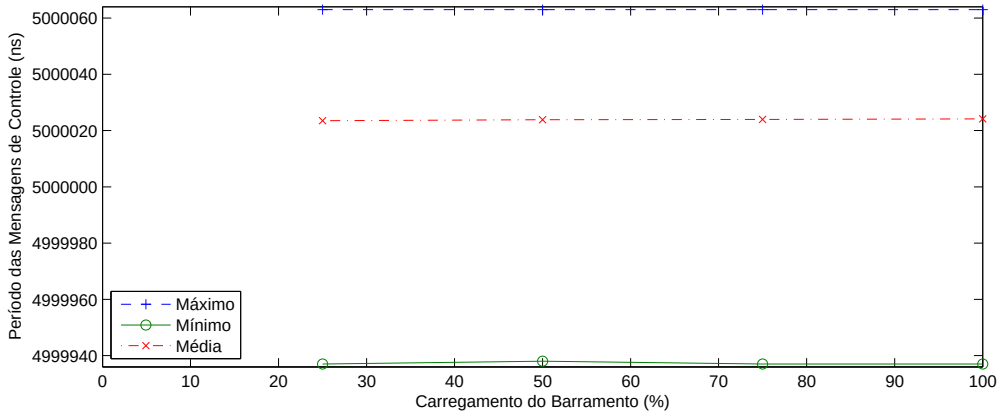

Figura 63: Periodicidade das mensagens de controle.

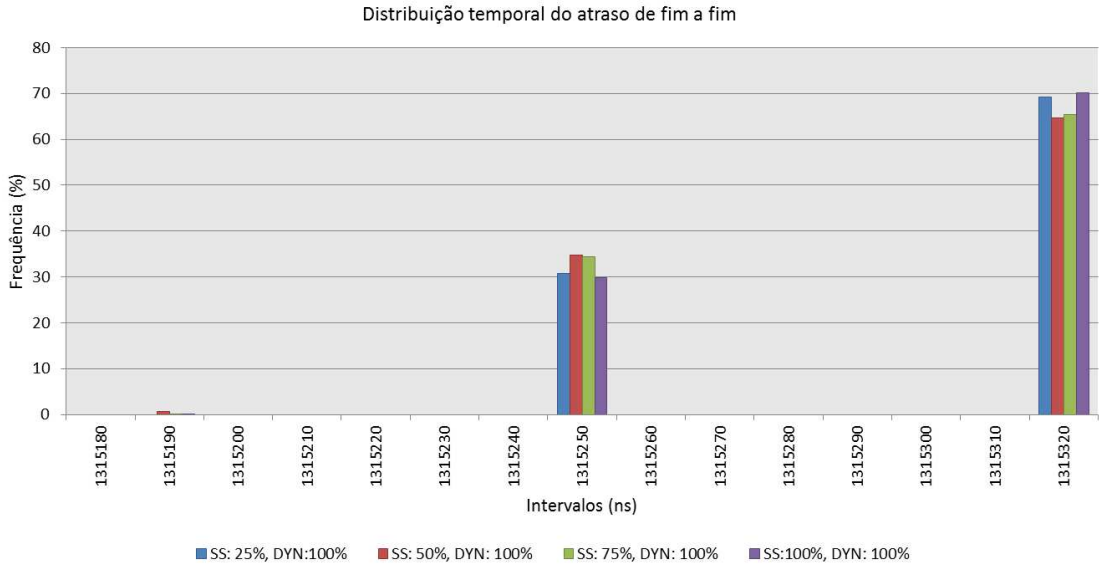

Figura 64: Distribuição temporal do atraso de fim-a-fim.

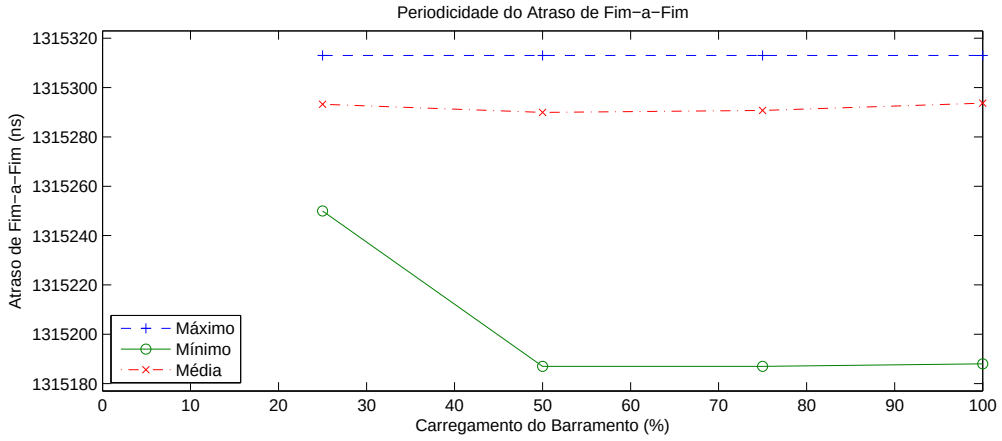

Figura 65: Atraso de fim-a-fim em função de diferentes condições de tráfego no barramento.

6.4.3.3 *Avaliação do Desempenho do Sistema de Controle em Função das Condições de Tráfego no Barramento*

A influência do tráfego no barramento sobre o desempenho do sistema de controle é ilustrado pela Figura 66, a qual ilustra a curva de resposta do estado x_{def}. Como no caso anterior, o comportamento é consequência direta dos resultados apresentados pelas avaliações anteriores. Mais uma vez, houve a ocorrência de poucas perdas de pacotes. É interessante observar que desta vez, a maior degradação de desempenho ocorreu quando o segmento estático apresentava 75% de ocupação, apesar de a degradação não ser significativa.

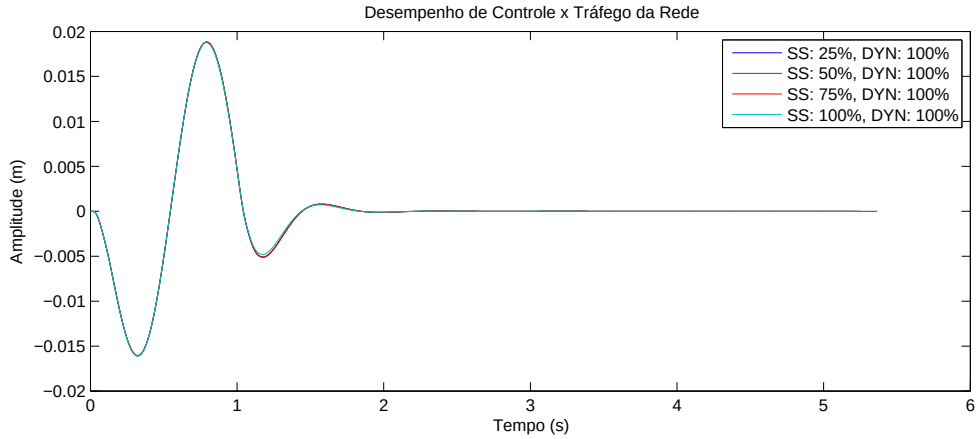

Figura 66: Resposta do estado x_s para diferentes condições de tráfego no barramento.

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, foram estudados os efeitos causados a uma malha de controle quando o laço é fechado através de uma rede de comunicação FlexRay. Como estudo de caso, foi escolhido o sistema de suspensão ativa, através do modelo conhecido por *quarter-car model*, o qual é bastante utilizado na literatura. Com o intuito de avaliar a influência do período de tempo entre o instante de amostragem das saídas da planta até a ação do atuador (denominado durante o texto de τ_{ee}), foram elaborados três controladores baseados em diferentes metodologias de projeto: (i) o primeiro baseado em alocação de pólos através de uma realimentação de estados, (ii) uma realimentação dinâmica de saída baseada no *framework* H_∞ e (iii) uma realimentação de estados baseada na metodologia de dados amostrados. Inicialmente, para a avaliação do comportamento do sistema quando são aplicados os controladores desenvolvidos, foram realizadas simulações que consideram o caso contínuo, ou seja, sem nenhum tipo de atraso no laço de controle.

De forma a avaliar experimentalmente seu comportamento, o sistema foi implementado em uma plataforma distribuída, formada pelas interfaces VN8900 e pelo software CANoe.FlexRay, produzidos pela empresa alemã *Vector Informatik GmbH*. Estas são ferramentas de última geração amplamente utilizadas pela indústria automobilística para desenvolvimento e análise de ECUs e sistemas embarcados. Cada experimento foi elaborado com o objetivo de avaliar qual o desempenho e impacto da comunicação via FlexRay em um sistema de controle que necessita de garantias temporais.

A implementação, no entanto, apresenta vários aspectos que devem ser minuciosamente elaborados, pois existem restrições tanto do sistema de controle quanto da comunicação. Sob a ótica de controle, a principal restrição para implementação está relacionada à taxa de amostragem. A escolha do período de amostragem depende basicamente da dinâmica do sistema a ser controlado.

Do ponto de vista da comunicação, de fundamental importância para o correto funcionamento do sistema, é o adequado escalonamento das mensagens que serão transmitidas por meio do barramento. Esta condição é particularmente importante para as mensagens que carregam as informações do laço de controle (amostragem das saídas da planta e leis de controle, por exemplo). Este foi o principal objetivo do teste de tempo de resposta das interfaces VN8900, pois existe um tempo significativo entre o momento em que a informação é adquirida pelo sistema até o instante em que a mesma está efetivamente disponível para transmissao através do barramento. Este foi um dos pontos críticos do trabalho, pois durante os experimentos, muitas vezes o escalonamento estava correto, mas as informações do laço de controle foram transmitidas com um ciclo de comunicação de atraso, devido à demora da interface em atualizar seu *buffer* de comunicação. Outro ponto importante trata-se do correto ajuste dos parâmetros do protocolo. Como mencionado durante o texto, o protocolo FlexRay possui mais de 70 parâmetros que devem ser ajustados

de acordo com a aplicação que será executada, sendo que muitos desses estão relacionados à sincronização dos elementos do *cluster*. Em sistemas do tipo *time-triggered* (como o segmento estático do protocolo), a sincronização se faz necessária para que todos os componentes do barramento possuam uma única base de tempo global.

Dos resultados obtidos durante o último capítulo, ficam bastante claras as características de garantia temporal oferecidas pelo FlexRay, refletida principalmente através da determinação de limites superiores para a periodicidade das mensagens e, também, pela distribuição temporal das mesmas, independente da condição de tráfego na rede. A partir destes resultados, é possível observar que para o sistema dinâmico utilizado como exemplo, e na ausência de perdas de pacotes, o barramento FlexRay não apresentou influência significativa no comportamento do sistema em malha fechada.

Outro ponto interessante, foi verificar a ocorrência de perda de pacotes durante a execução do experimento (conforme evidenciado pelo experimento 2). Estas perdas de pacotes estão relacionadas ao escalonamento de tarefas pelas interfaces VN8900. Nesta situação, não foi possível que a tarefa concluísse suas funções antes da *deadline* para transmissão (durante o ciclo de comunicação corrente).

Devido a seu caráter multidisciplinar e ao grande interesse recente, ainda existem muitas outras possibilidades para expandir este trabalho. Entre estas, pode-se destacar a utilização de uma metodologia de projeto de controle que considere os efeitos específicos do protocolo, tanto para transmissão de mensagens no segmento estático quanto no segmento dinâmico, de forma a elaborar um projeto integrado entre as partes de comunicação e controle. Conforme mencionado no parágrafo anterior, alguns pacotes foram "perdidos" porque a interface não conseguiu responder a tempo de transmitir no *time-slot* adequado. Uma forma de evitar que isto ocorra, talvez, seja adicionar outros nós físicos ao cluster, de forma a aliviar a carga de processamento imposta às iterfaces e, ainda, elaborar diferentes topologias de rede e analisar qual o impacto destas sobre o desempenho do sistema como um todo. Outra área de estudo interessante, seria procurar otimizar também o escalonamento das mensagens, de forma a conseguir uma melhor utilização da banda disponível. Por fim, como estudo de caso prático, seria interessante desenvolver uma interface FlexRay para o sistema de suspensão ativa do laboratório de controle do DELET. Esta interface seria útil para permitir a validação experimental do sistema, que poderia ser usado em disciplinas de graduação e pós-graduação.

REFERÊNCIAS

ALBERT, A. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. In: EMBEDDED WORLD, 2004, Nürnberg. **Proceedings...** [S.l.: s.n.], 2004. p.235–252.

ANTSAKLIS, P.; BAILLIEUL, J. Special Issue on Technology of Networked Control Systems. **Proceedings of the IEEE**, [S.l.], v.95, n.1, p.5–8, Jan. 2007.

ARNDT, M. **Implementation of a FlexRay Communication Interface for Linux**. 2009. 54p. Trabalho de Conclusão (Graduação) — University of Kaiserslautern, Kaiserslautern, 2009.

ATAIDE, F. H.; PEREIRA, C. E. FTT-CAN - Estudo de caso em aplicação automotiva. **SBA: Controle e Automação Sociedade Brasileira de Automatica**, [S.l.], v.23, n.5, p.621 – 635, Out. 2012.

BAILLIEUL, J.; ANTSAKLIS, P. J. Control and Communication Challenges in Networked Real-Time Systems. **Proceedings of the IEEE**, [S.l.], v.95, n.1, p.9–28, Jan. 2007.

BARTOLS, F. et al. Performance Analysis of Time-Triggered Ether-Networks Using Off-the-Shelf-Components. In: IEEE INTERNATIONAL SYMPOSIUM ON OBJECT/COMPONENT/SERVICE-ORIENTED REAL-TIME DISTRIBUTED COMPUTING WORKSHOP (ISORCW), 14., 2011, Newport Beach. **Proceedings...** [S.l.: s.n.], 2011. p.49–56.

CAN. **Controller Area Network CAN, an In vehicle Serial Communication Protocol**. In: SAE Handbook 1992. [S.l.]: SAE Press, 1992. p.20341–20355.

CAN in Automation (CiA). **Controller Area Network**. Disponível em: <http://www.can-cia.org/index.php?id=can>. Acesso em: 10 dez. 2013.

CHOKSHI, D. B.; BHADURI, P. Performance Analysis of FlexRay-based Systems Using Real-time Calculus, Revisited. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 25., 2010, Sierre. **Proceedings...** New York: ACM, 2010. p.351–356.

CLOOSTERMAN, M. et al. Controller synthesis for networked control systems. **Automatica**, [S.l.], v.46, n.10, p.1584 – 1594, Oct. 2010.

DO, A. et al. Control design for LPV systems with input saturation and state constraints: an application to a semi-active suspension. In: IEEE CONFERENCE ON DECISION

AND CONTROL AND EUROPEAN CONTROL CONFERENCE (CDC-ECC), 50., 2011, Orlando. **Proceedings...** [S.l.: s.n.], 2011. p.3416–3421.

DO, A.-L. **Approche LPV pour la commande robuste de la dynamique des véhicules** : amélioration conjointe du confort et de la sécurité. 2011. 231p. Tese (Doutorado em Engenharia Elétrica) — Institut National Polytechnique de Grenoble, Grenoble, 2011.

DUAN, J.; DENG, H.; YU, Y. Co-simulation of SBW and FlexRay bus based on CANoe_MATLAB. In: IEEE INTERNATIONAL CONFERENCE ON AUTOMATION AND LOGISTICS (ICAL), 2010, Hong Kong e Macau. **Proceedings...** [S.l.: s.n.], 2010. p.202–206.

ELMENREICH, W. Time-Triggered Fieldbus Networks State of the Art and Future Applications. In: IEEE INTERNATIONAL SYMPSIUM ON OBJECT ORIENTED REAL-TIME DISTRIBUTED COMPUTING (ISORC), 11., 2008, Orlando. **Proceedings...** [S.l.: s.n.], 2008. p.436–442.

FIELDBUS FOUNDATION. **Foundation Fieldbus**. Disponível em: <http://www.fieldbus.org>. Acesso em: 12 dez. 2013.

FLEXRAY CONSORTIUM. **FlexRay Electrical Physical Layer Specification - Version 3.0.1**. Disponível em: <http://www.flexray.com>. Acesso em: 28 ago. 2012.

FLEXRAY CONSORTIUM. **FlexRay Communications System - Protocol Specification - Version 3.0.1**. Disponível em: <http://www.flexray.com>. Acesso em: 28 ago. 2012.

FLEXRAY CONSORTIUM. **FlexRay Electrical Physical Layer Application Notes - Version 3.0.1**. Disponível em: <http://www.flexray.com>. Acesso em: 28 ago. 2012.

FRIDMAN, E.; SEURET, A.; RICHARD, J.-P. Robust sampled-data stabilization of linear systems: an input delay approach. **Automatica**, [S.l.], v.40, n.8, p.1441–1446, Aug. 2004.

FÜHRER, T. et al. Time triggered communication on CAN (Time Triggered CAN-TTCAN). In: INTERNATIONAL CAN CONFERENCE, 7., 2000, Amsterdam. **Proceedings...** [S.l.: s.n.], 2000. Não paginado.

GÖHNER, P. **Einführung in das Bussystem FlexRay am Beispiel Steer-by-Wire**. Stuttgart: Universität Stuttgart, 2012. Disponível em: <http://www.ias.uni-stuttgart.de/lehre/praktika/automatisierung/unterlagen/06-Grundlagen_FlexRay-v10_de.pdf>. Acesso em: 20 set. 2013.

GOMES DA SILVA JR., J. M. et al. Stabilisation of neutral systems with saturating control inputs. **International Journal of Systems Science**, [S.l.], v.42, n.7, p.1093–1103, 2011.

GUPTA, R. A.; CHOW, M.-Y. Networked control system: overview and research trends. **IEEE Transactions on Industrial Electronics**, [S.l.], v.57, n.7, p.2527–2535, July 2010.

HAGIESCU, A. et al. Performance analysis of FlexRay-based ECU networks. In: DESIGN AUTOMATION CONFERENCE, 44., 2007, San Diego. **Proceedings...** New York: ACM, 2007. p.284–289.

HANZÁLEK, Z.; BENEŠ, D.; WARAUS, D. Time Constrained FlexRay Static Segment Scheduling. In: INTERNATIONAL WORKSHOP ON REAL-TIME NETWORKS, 10., 2011, Porto. **Proceedings...** [S.l.: s.n.], 2011. p.23–28.

HELLER, C.; REICHEL, R. Enabling FlexRay for avionic data buses. In: IEEE/AIAA DIGITAL AVIONICS SYSTEMS CONFERENCE (DASC), 28., 2009, Orlando. **Proceedings...** [S.l.: s.n.], 2009. p.1.B.1–1 – 1.B.1–14.

HESPANHA, J. P.; NAGHSHTABRIZI, P.; XU, Y. A survey of recent results in networked control systems. **Proceedings of the IEEE**, [S.l.], v.95, n.1, p.138–162, 2007.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **FlexRay Communications System**. Disponível em: <http://www.iso.org/iso/home.htm>. Acesso em: 09 set. 2013.

LI, F.; WANG, L.; LIAO, C. Evaluating the communication impact on quality of service in steer-by-wire systems. In: IEEE VEHICLE POWER AND PROPULSION CONFERENCE (VPPC), 2008, Harbin. **Proceedings...** [S.l.: s.n.], 2008. p.1–4.

LIAN, F.-L.; MOYNE, J. R.; TILBURY, D. M. Performance evaluation of control networks: ethernet, controlnet and devicenet. **IEEE Control Systems**, [S.l.], v.21, n.1, p.66–83, Feb. 2001.

LIAN, F.-L.; MOYNE, J.; TILBURY, D. Network design consideration for distributed control systems. **IEEE Transactions on Control Systems Technology**, [S.l.], v.10, n.2, p.297–307, Mar. 2002.

LIM, H.-T.; WECKEMANN, K.; HERRSCHER, D. Performance study of an in-car switched ethernet network without prioritization. In: INTERNATIONAL WORKSHOP, NETS4CARS/NETS4TRAINS, 3., 2011, Oberpfaffenhofen. **Proceedings...** [S.l.]: Springer Berlin Heidelberg, 2011. p.165–175.

MAKOWITZ, R.; TEMPLE, C. Flexray - A communication network for automotive control systems. In: IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS, 6., 2006, Torino. **Proceedings...** [S.l.: s.n.], 2006. p.207–212.

MOORE, B. On the flexibility offered by state feedback in multivariable systems beyond closed loop eigenvalue assignment. **IEEE Transactions on Automatic Control**, [S.l.], v.21, n.5, p.689–692, Oct. 1976.

MURRAY, R. M. et al. Future directions in control in an information-rich world. **IEEE Control Systems**, [S.l.], v.23, n.2, p.20–33, Apr. 2003.

NAGHSHTABRIZI, P.; HESPANHA, J. Analysis of distributed control systems with shared communication and computation resources. In: AMERICAN CONTROL CONFERENCE 2009 (ACC2009), 2009, St. Louis. **Proceedings...** [S.l.: s.n.], 2009. p.3384–3389.

NETO, A. Z. **Análise do Impacto da Comunicação em Redes Foundation Fieldbus no Desempenho de Sistemas de Controle**. 2007. 93p. Dissertação (Mestrado em Engenharia Elétrica) — Programa de Pós-Graduação em Engenharia Elétrica, Univeridade Federal do Rio Grande do Sul, Porto Alegre, 2007.

NILSSON, J. **Real-time Control Systems with Delays**. 1998. 138p. Tese (Doutorado em Engenharia Elétrica) — Lund Institute of Technology, Lund, 1998.

OGATA, K. **Modern Control Engineering**. 5th. ed. New Jersey: Prentice Hall, 2010.

PARK, I.; SUNWOO, M. FlexRay Network Parameter Optimization Method for Automotive Applications. **IEEE Transactions on Industrial Electronics**, [S.l.], v.58, n.4, p.1449–1459, Apr. 2011.

PEREIRA, C. E.; NEUMANN, P. **Industrial Communication Protocols**. In: NOF, S. Y. (Ed.). Springer Handbook of Automation. [S.l.]: Springer Berlin Heidelberg, 2009. p.981–999.

POUSSOT-VASSAL, C. **Commande Robuste LPV Multivariable de Châssis Automobile**. 2008. 229p. Tese (Doutorado em Engenharia Elétrica) — Grenoble Institut Polytechnique, Grenoble, 2008.

SAE INTERNATIONAL. **AS6802**: time-triggered ethernet. Disponível em: <http://standards.sae.org/as6802/>. Acesso em: 12 dez. 2013.

SAMII, S. et al. Design Optimization and Synthesis of FlexRay Parameters for Embedded Control Applications. In: IEEE INTERNATIONAL SYMPOSIUM ON ELECTRONIC DESIGN, TEST AND APPLICATION (DELTA), 6., 2011, Queenstown. **Proceedings...** [S.l.: s.n.], 2011. p.66–71.

SANDE, T. van der et al. Robust control of an electromagnetic active suspension system: simulations and measurements. **Mechatronics**, [S.l.], v.23, n.2, p.204–212, Mar. 2013.

SCHEDL, A. Goals and Architecture of FlexRay at BMW. In: SLIDES PRESENTED AT THE VECTOR FLEXRAY SYMPOSIUM, 2007, Stuttgart. **Anais...** [S.l.: s.n.], 2007. p.1–21.

SCHERER, C.; GAHINET, P.; CHILALI, M. Multiobjective output-feedback control via LMI optimization. **IEEE Transactions on Automatic Control**, [S.l.], v.42, n.7, p.896–911, July 1997.

SCHMIDT, K.; SCHMIDT, E. Message Scheduling for the FlexRay Protocol: the static segment. **IEEE Transactions on Vehicular Technology**, [S.l.], v.58, n.5, p.2170–2179, June 2009.

SHAW, R.; JACKMAN, B. An introduction to FlexRay as an industrial network. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS (ISIE), 2008, Cambridge. **Proceedings...** [S.l.: s.n.], 2008. p.1849–1854.

SMAR Equipamentos Industriais. **A Foundation Fielbus Technology Overview**. Disponível em: <www.smar.com/PDFs/catalogues/FBTUTCE.pdf>. Acesso em: 12 dez. 2013.

STEINBACH, T.; KORF, F.; SCHMIDT, T. Comparing time-triggered Ethernet with FlexRay: an evaluation of competing approaches to real-time for in-vehicle networks. In: IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS (WFCS), 8., 2010, Nancy. **Proceedings...** [S.l.: s.n.], 2010. p.199–202.

TANENBAUM, A. S.; WETHERALL, D. J. **Computer Networks**. 5th. ed. New Jersey: Prentice Hall, 1999.

TIAN, G.; BAI, P.; CHEN, Q. Response time analysis of FlexRay communication in fuel cell hybrid vehicle. In: IEEE VEHICLE POWER AND PROPULSION CONFERENCE (VPPC), 2008, Harbin. **Proceedings...** [S.l.: s.n.], 2008. p.1–4.

TINDELL, K.; BURNS, A.; WELLINGS, A. J. Calculating controller area network (CAN) message response times. **Control Engineering Practice**, [S.l.], v.3, n.8, p.1163–1169, Aug. 1995.

TIPSUWAN, Y.; CHOW, M.-Y. Control methodologies in networked control systems. **Control Engineering Practice**, [S.l.], v.11, n.10, p.1099 – 1111, Oct. 2003.

TROFINO, A. **Controle Robusto**. 2000. 153 p. Apostila de Controle. Disponível em: <http://www.das.ufsc.br/ trofino/>. Acesso em: 01 jan. 2014.

TTA-GROUP. **TTP Overview**. Disponível em: <http://www.ttagroup.org/ttp/overview.htm>. Acesso em: 12 dez. 2013.

VECTOR INFORMATIK GMBH. **Introduction to FlexRay**. Disponível em: https://vector.com/vl_index_en.html>. Acesso em: 20 dez. 2013.

VECTOR INFORMATIK GMBH. **VN8900 - FlexRay/CAN/LIN/J1708 Network Interface**. Disponível em: <https://vector.com/vi_vn8900_en.html>. Acesso em: 12 dez. 2013.

VECTOR INFORMATIK GMBH. **Vector Knowledge Base**. Disponível em: <http://vector.com/kb/index.php?q=47>. Acesso em: 15 nov. 2013.

VECTOR INFORMATIK GMBH. **Serial Bus Systems In the Motor Vehicle**. Disponível em: <https://vector.com/index.php?seite=vl_sbs_introduction_en&kanal=html>. Acesso em: 08 jan. 2014.

VECTOR INFORMATIK GMBH. **CANoe Option .FlexRay**. Disponível em: <http://vector.com/vi_canoe_flexray_en.html>. Acesso em: 08 jan. 2014.

WALSH, G.; YE, H. Scheduling of networked control systems. **IEEE Control Systems**, [S.l.], v.21, n.1, p.57–65, Feb. 2001.

XIA, F.; SUN, Y. **Control and Scheduling Codesign**. [S.l.]: Zhejiang University Press – Springer, 2008.

ZHANG, L.; GAO, H.; KAYNAK, O. Network-Induced Constraints in Networked Control Systems - A Survey. **IEEE Transactions on Industrial Informatics**, [S.l.], v.9, n.1, p.403–416, Feb. 2013.

ZHANG, W.; BRANICKY, M. S.; PHILLIPS, S. M. Stability of networked control systems. **IEEE Control Systems**, [S.l.], v.21, n.1, p.84–99, Feb. 2001.

APÊNDICE A (SCRIPTS MATLAB)

Neste apêndice, estão relacionados os códigos MATLAB utilizados para síntese dos controladores e simulações para validação do comportamento do sistema em malha fechada.

A.1 Controladores

A.1.1 Alocação de Pólos

```
%% Active Suspension Controller
% Pole Placement with State Feedback
% Quarter-car Model
% Author: Thiago Michelin
clear all;
clc;

%% System Static Parameters
% Sprung mass - Vehicle's Body Mass
ms = 315;
% Unsprung mass - Vehicle's Suspension Set
mus = 37.5;
% Suspension Damping Parameter
bs = 1000;
% Suspension Spring Stiffness
ks = 29500;
% Spring Stiffness from Tyre Model
kt = 210000;

%% State Space Representation
A = [0 1 0 -1; -ks/ms -bs/ms 0 bs/ms; ...
        0 0 0 1; ks/mus bs/mus -kt/mus -bs/mus];
Bu = [0 1/ms 0 -1/mus]';
Bw = [0 0 -1 0]';
B = [Bw Bu];
Cy = [1 0 0 0];
Cz = A(2,:);
C = [Cy; Cz];
Dy = zeros(1,2);
Dz = [Bw(2,1) Bu(2,1)];
D = [Dy; Dz];

%% Open Loop Analysis
qcar = ss(A,B,C,D);
```

```
35 qcarol = ss(A,Bw,C,[0; Bw(2,1)]);
36 disp('Damping Factor:')
37 damp(qcar)
38
39 w = logspace(-1,3,10000);
40
41 %% State-Feedback Development
42
43 % Desired Damping for the faster poles
44 ddamp1 = 0.2;
45 % Desired Damping for the dominant poles
46 ddamp2 = 0.7;
47 % Desired place for the real part of faster poles
48 wn1 = 85;
49 % Desired place for the real part of dominant poles
50 wn2 = 20;
51 % System's eigenvalues
52 P = eig(A);
53 % Desired Poles Matrix
54 Pd = zeros(4,1);
55
56 % Faster Poles Calculation
57 Pd(1) = -ddamp1*wn1 + wn1*sqrt(1-ddamp1^2)*1i;
58 Pd(2) = -ddamp1*wn1 - wn1*sqrt(1-ddamp1^2)*1i;
59
60 % Dominant Poles Work
61 Pd(3) = -ddamp2*wn2 + wn2*sqrt(1-ddamp2^2)*1i;
62 Pd(4) = -ddamp2*wn2 - wn2*sqrt(1-ddamp2^2)*1i;
63
64 % Moore's Method
65 w1 = [1 2]';
66 w2 = [3 4]';
67 V1 = inv([real(Pd(1))*eye(4)-A -imag(Pd(1))*eye(4); ...
68         imag(Pd(1))*eye(4) real(Pd(1))*eye(4)-A])*[Bu zeros(4,1); ...
69         zeros(4,1) Bu]*w1;
70
71 V2 = inv([real(Pd(3))*eye(4)-A -imag(Pd(3))*eye(4); ...
72         imag(Pd(3))*eye(4) real(Pd(3))*eye(4)-A])*[Bu zeros(4,1); ...
73         zeros(4,1) Bu]*w2;
74
75 V = [V1(1:4,:) V1(5:8,:) V2(1:4,:) V2(5:8,:)];
76 W = [1 2 3 4];
77
78 Kpp = W/V;
79
80 %% Closed Loop
81 Acl = A + Bu*Kpp;
82 Bcl = Bw;
83 Ccl = [1 0 0 0; Acl(2,:); Kpp];
84 Dcl = [0; Bw(2,1); 0];
85
86 syscl = ss(Acl,Bcl,Ccl,Dcl);
87
88 disp('Pólos em malha aberta:')
89 P
90 disp('Pólos desejados em malha fechada:')
91 Pd
92 disp('Pólos em malha fechada:')
```

```
eig(Acl)

% Bode Plot for Comparison Purposes
bode_closed_loop1 = figure;
centerfigure(bode_closed_loop1);
bodemag(qcar(1,1),syscl(1,1),w)
legend('Malha Aberta','Malha Fechada','Location','SouthWest')
title('Resposta em Frequência da FT x_{def}/dx_r')
grid on

bode_closed_loop2 = figure;
centerfigure(bode_closed_loop2);
bodemag(qcar(2,1),syscl(2,1),w)
legend('Malha Aberta','Malha Fechada','Location','SouthWest')
title('Resposta em Frequência da FT d^2x_s/dx_r')
grid on

%% Simulation
tf = 5;
tstep = 0.001;
tsim = 0:tstep:tf;
dxr = disturbance(tf, tstep, 'dcucoruto_v2');
[yo,~,xo] = lsim(qcarol,dxr,tsim);
[yc,~,xc] = lsim(syscl,dxr,tsim);

sim1 = figure;
centerfigure(sim1);
plot(tsim,yo(:,1),tsim,yc(:,1))
xlabel('Tempo (s)')
legend('Malha Aberta','Malha Fechada','Location','SouthEast')
title('Deflexão da Suspensão (x_{def})')

sim2 = figure;
centerfigure(sim2);
plot(tsim,yo(:,2),tsim,yc(:,2))
legend('Malha Aberta','Malha Fechada','Location','SouthEast')
xlabel('Tempo (s)')
title('Aceleração Vertical do Corpo do Automóvel (d^2x_s/dt)')
```

A.1.2 Realimentação Dinâmica de Saída

```
%% Active Suspension Robust Controller (Hinf technique via LMIs)
% Suspension Model 1
% Dynamic Output Feedback Controller
% Author: Thiago Michelin
clear all;
clc;

%% System Parameters
% Vehicle's Sprung mass (car body)
ms = 315;
% Vehicle's Unsprung mass (car's suspension: suspension + wheel + tire)
mus = 37.5;
% Suspension Spring Constant
ks = 29500;
% Tyre Spring Constant
kt = 210000;
```

```
% Suspension Damper Parameter
bs = 1000;

%% Plant State Space Representation
A = [0 1 0 -1; -ks/ms -bs/ms 0 bs/ms; ...
        0 0 0 1; ks/mus bs/mus -kt/mus -bs/mus];
[rA,cA] = size(A);

Bu = [0 1/ms 0 -1/mus]';
[rBu,cBu] = size(Bu);

Bw = [0 0 -1 0]';
B = [Bw Bu];
Cy = [1 0 0 0; 0 1 0 -1];
[rCy,cCy] = size(Cy);

Cz = [1 0 0 0; A(2,:)];
Dyw = zeros(2,1);
Dzu = [0; Bu(2,1)];
Dzw = [0; Bw(2,1)];

qcar = ss(A,Bw,Cz,Dzw);

%% Weighting Functions for Performance Conditions
% Weighting Function for disturbance input (xr)
Wroad = 0.157;

wab_g = 10;
num_ab = wab_g*[0.4901 1563 360.9];
den_ab = [1 217.7 788.9];

[a1,b1,c1,d1] = tf2ss(num_ab,den_ab);
[ra1,ca1] = size(a1);
[rb1,cb1] = size(b1);
[rc1,cc1] = size(c1);
[rd1,cd1] = size(d1);

% Weighting Function for output xdef
wsd_g = 0.1;
num_sd = wsd_g;
den_sd = [1/(2*pi*20) 1];

[a4,b4,c4,d4] = tf2ss(num_sd,den_sd);
[ra4,ca4] = size(a4);
[rb4,cb4] = size(b4);
[rc4,cc4] = size(c4);
[rd4,cd4] = size(d4);

pond1 = figure;
centerfigure(pond1)
bode(tf(num_ab,den_ab))
grid on

pond2 = figure;
centerfigure(pond2)
bode(tf(num_sd,den_sd))
grid on

```

```
Ao = [a1 zeros(ra1,ca4); zeros(ra4,ca1) a4];
Bo = [b1 zeros(rb1,cb4); zeros(rb4,cb1) b4];
Co = [c1 zeros(rc1,cc4); zeros(rc4,cc1) c4];
Do = [d1 zeros(rd1,cd4); zeros(rd4,cd1) d4];

[rAo,cAo] = size(Ao);
[rBo,cBo] = size(Bo);
[rCo,cCo] = size(Co);
[rDo,cDo] = size(Do);

%% Augmented System
Ab = [A zeros(rA,cAo); Bo*Cz Ao];
B1b = [Bw*Wroad; Bo*Dzw*Wroad];
B2b = [Bu; Bo*Dzu];
C1b = [Do*Cz Co];
C2b = [Cy zeros(rCy, cAo)];
D11b = Do*Dzw*Wroad;
D12b = Do*Dzu;
D21b = Dyw*Wroad;

%% Controller Synthesis
[rAb,cAb] = size(Ab);
[rB2b,cB2b] = size(B2b);
[rC2b,cC2b] = size(C2b);
[rD12b,cD12b] = size(D12b);

In = eye(rAb);
Inu = eye(cBu);

%% LMIs Constraints
setlmis([]);

% Variables
X = lmivar(1,[rAb 1]);
Y = lmivar(1,[rAb 1]);
At = lmivar(2,[rAb cAb]);
Bt = lmivar(2,[cC2b rC2b]);
Ct = lmivar(2,[cB2b rB2b]);
Dt = lmivar(2,[cD12b rC2b]);
G = lmivar(1,[1 0]);

% LMIs Definitions
LMI_XY1 = newlmi;
lmiterm([-LMI_XY1 1 1 X],1,1);
lmiterm([-LMI_XY1 2 1 0],In);
lmiterm([-LMI_XY1 2 2 Y],1,1);

LMI_Hinf1 = newlmi;
lmiterm([LMI_Hinf1 1 1 X],Ab,1,'s');
lmiterm([LMI_Hinf1 1 1 Ct],B2b,1,'s');
lmiterm([LMI_Hinf1 2 1 At],1,1);
lmiterm([LMI_Hinf1 2 1 0],Ab');
lmiterm([LMI_Hinf1 2 1 -Dt],C2b',B2b');
lmiterm([LMI_Hinf1 2 2 Y],1,Ab,'s');
lmiterm([LMI_Hinf1 2 2 Bt],1,C2b,'s');
lmiterm([LMI_Hinf1 3 1 0],B1b');
lmiterm([LMI_Hinf1 3 1 -Dt],D21b',B2b');
lmiterm([LMI_Hinf1 3 2 Y],B1b',1);
```

```
lmiterm([LMI_Hinf1 3 2 -Bt],D21b',1);
lmiterm([LMI_Hinf1 3 3 G],-Inu,1);
lmiterm([LMI_Hinf1 4 1 X],C1b,1);
lmiterm([LMI_Hinf1 4 1 Ct],D12b,1);
lmiterm([LMI_Hinf1 4 2 0],C1b);
lmiterm([LMI_Hinf1 4 2 Dt],D12b,C2b);
lmiterm([LMI_Hinf1 4 3 0],D11b);
lmiterm([LMI_Hinf1 4 3 Dt],D12b,D21b);
lmiterm([LMI_Hinf1 4 4 G],-1,1);

lmis = getlmis;

n = decnbr(lmis);
c = zeros(n,1);

for j = 1:n
    Gj = defcx(lmis,j,G);
    c(j) = Gj;
end

options = [0,0,0,0,0];
[copt,xopt] = mincx(lmis,c,options);

Gopt = dec2mat(lmis,xopt,G);
Xopt = dec2mat(lmis,xopt,X);
Yopt = dec2mat(lmis,xopt,Y);
Aopt = dec2mat(lmis,xopt,At);
Bopt = dec2mat(lmis,xopt,Bt);
Copt = dec2mat(lmis,xopt,Ct);
Dopt = dec2mat(lmis,xopt,Dt);

%% Calculate M and N
[M,Nt] = qr(eye(rAb) - Xopt*Yopt);
N = Nt';

%% Controller Reconstruction
Dc = Dopt;
Cc = (Copt - Dc*C2b*Xopt)/(M');
Bc = N\(Bopt - Yopt*B2b*Dc);
Ac = (N\(Aopt - Yopt*Ab*Xopt - Yopt*B2b*Dc*C2b*Xopt - ...
    N*Bc*C2b*Xopt - Yopt*B2b*Cc*M'))/(M');

hinfcontrol = ss(Ac,Bc,Cc,Dc);

%% Closed Loop System
[rAc,~] = size(Ac);

Acl = [(A+Bu*Dc*Cy) Bu*Cc;Bc*Cy Ac];
Bcl = [Bw+Bu*Dc*Dyw;Bc*Dyw];
Ccl = [Cz+Dzu*Dc*Cy Dzu*Cc];
Dcl = Dzw+Dzu*Dc*Dyw;

Ccl_m = [1 0 0 0 zeros(1,rAc); 1 0 -1 0 zeros(1,rAc); ...
    Acl(2,:)];
Dcl_m = [0; 0; Bcl(2,:)];

SysCl = ss(Acl,Bcl,Ccl,Dcl);

```

```
SIMK = ss(Acl,Bcl,Ccl_m,Dcl_m);
figure;
pzmap(SysCl)
[Wnh,zh,Ph] = damp(SIMK);

ts = 5;
ti = 0.001;
t = 0:ti:ts;
w = disturbance(ts,ti,'dcucoruto_v2');

[ycl,~,xcl] = lsim(SIMK,w,t);
[yol,~,xol] = lsim(qcar,w,t);

%% Plotting Area
figure;
plot(t,ycl(:,1),t,yol(:,1))
legend('Malha Fechada','Malha Aberta')
title('Deflexao da Suspensao')

figure;
plot(t,ycl(:,3),t,yol(:,2))
legend('Malha Fechada','Malha Aberta')
title('Aceleracao Vertical')
```

A.1.3 Metodologia de Dados Amostrados

```
%% Active Suspension Controller
% State Feedback with DDE (Delayed Differential Equations)
% Continuous Case
% Suspension Model 1
% Author: Thiago Michelin

clear all;
clc;

%% System Parameters

% Vehicle's Body mass
ms = 315;
% Vehicle's Unsprung mass (car's suspension: suspension + wheel + tyre)
mus = 37.5;
% Suspension Spring Constant
ks = 29500;
% Tyre Spring Constant
kt = 210000;
% Suspension Damper Parameter
bs = 1000;

%% State Space Representation
A = [0 1 0 -1; -ks/ms -bs/ms 0 bs/ms; ...
        0 0 0 1; ks/mus bs/mus -kt/mus -bs/mus];
[ra,ca] = size(A);

Bu = [0 1/ms 0 -1/mus]';
[rbu,cbu] = size(Bu);

Bw = [0 0 -1 0]';
```

```
Cy = [1 0 0 0; A(2,:)];
Dy = [0 0; Bw(2,:) Bu(2,:)];

qcar = ss(A,Bw,Cy,Dy(:,1));

%% LMIs Constraints

% Upper delay bound
h = 10.5e-3;
zeta = 0.5;

% Exponential Stability
alpha = 1.7;

setlmis([]);

% Matrix Variables
Q1 = lmivar(1,[ra 1]);
Q2 = lmivar(2,[ra ca]);
R = lmivar(2,[ra ca]);
Y = lmivar(2,[cbu ca]);

% LMI_Q
LMI_Q = newlmi;
lmiterm([-LMI_Q 1 1 Q1],1,1);

% LMI_R
LMI_R = newlmi;
lmiterm([-LMI_R 1 1 R],1,1);

% LMI_1
LMI_1 = newlmi;
lmiterm([LMI_1 1 1 Q1],2*alpha,1);
lmiterm([LMI_1 1 1 Q2],1,A','s');
lmiterm([LMI_1 1 1 R],-2*h*exp(-2*alpha*h),1);
lmiterm([LMI_1 1 2 Q1],1,1);
lmiterm([LMI_1 1 2 -Q2],-1,1);
lmiterm([LMI_1 1 2 Q2],zeta,A');
lmiterm([LMI_1 1 3 Y],Bu,1);
lmiterm([LMI_1 1 3 R],2*h*exp(-2*alpha*h),1);
lmiterm([LMI_1 2 2 R],h^2,1);
lmiterm([LMI_1 2 2 Q2],-zeta,1,'s');
lmiterm([LMI_1 2 3 Y],zeta*Bu,1);
lmiterm([LMI_1 3 3 R],-2*h*exp(-2*alpha*h),1);

lmisys = getlmis();

options = [0 0 0 0 0];
[tmin,xfeas] = feasp(lmisys,options);

if(tmin < 0)
    Q2opt = dec2mat(lmisys,xfeas,Q2);
    Yopt = dec2mat(lmisys,xfeas,Y);

    Kdde = Yopt/(Q2opt');
else
    disp('LMI Constraints not feasible')
end
```

```
90
91  %% Closed Loop
92  Acl = A + Bu*Kdde;
93  Bcl = Bw;
94  Ccl = [1 0 0 0; Acl(2,:)];
95  Dcl = [zeros(1,1); Bcl(2,1)];
96
97  disp('Autovalores em Malha Fechada:')
98  [wn,z,p] = damp(ss(Acl,Bcl,Ccl,Dcl))
99
100 syscl = ss(Acl,Bcl,Ccl,Dcl);
101
102 %% Simulation
103 t = 0:0.001:5;
104 u = disturbance(5,0.001,'dcucoruto_v2');
105
106 [yo,~,xo] = lsim(qcar,u,t);
107 [yc,~,xc] = lsim(syscl,u,t);
108
109 figdde = figure;
110 centerfigure(figdde)
111 plot(t,xo(:,1),t,xc(:,1))
112 legend('Open Loop','Closed Loop')
113 xlabel('Tempo (s)')
114 ylabel('Amplitude')
115 title('Suspension Deflection')
116
117 figdde2 = figure;
118 centerfigure(figdde2)
119 plot(t,yc(:,2),t,yo(:,2))
120 title('Body Acceleration')
121 legend('Closed Loop','Open Loop')
```

APÊNDICE B (SCRIPTS CAPL)

Neste apêndice, encontram-se os scripts utilizados pelo software *CANoe* para modelar o comportamento dos dispositivos VN8910.

B.1 Planta

B.1.1 Experimento 1

```
includes
{
}

variables
{
  // Future State x(k+1)
  float fst0 = 0;
  float fst1 = 0;
  float fst2 = 0;
  float fst3 = 0;

  // Plant States x(k)
  float st0 = 0;
  float st1 = 0;
  float st2 = 0;
  float st3 = 0;

  // Plant Inputs
  float u0 = 0;   // Disturbance
  float u1 = 0;   // Actuator Force

  frPDU FrPDUState1 pdustate1;
  frPDU FrPDUState2 pdustate2;
  frPDU FrPDUState3 pdustate3;
  frPDU FrPDUState4 pdustate4;

  // Timer for simulation
  timer t_plant;
}

on start
{
  setTimer(t_plant,0,100000);
}
```

```
36
37 on preStart
38 {
39   frSetSendPDU(pdustate1);
40   frSetSendPDU(pdustate2);
41   frSetSendPDU(pdustate3);
42   frSetSendPDU(pdustate4);
43 }
44
45 on timer t_plant
46 {
47   u0 = getValue(disturbance);
48
49   st0 = fst0;
50   st1 = fst1;
51   st2 = fst2;
52   st3 = fst3;
53
54   fst0 = 9.9999953e-01*st0+9.9984127e-05*st1+4.6311103e-07*st2
55             +1.5872489e-08*st3+4.4442968e-09*u0+1.5849333e-11*u1;
56   fst1 = -9.3511065e-03*st0+9.9968255e-01*st1+9.2622206e-03*st2
57             +3.1744977e-04*st3+8.8885936e-05*u0+3.1698666e-07*u1;
58   fst2 = 3.9274098e-06*st0+1.3332890e-07*st1+9.9996811e-01*st2
59             +9.9865273e-05*st3+2.7962276e-05*u0-1.3313253e-10*u1;
60   fst3 = 7.8548195e-02*st0+2.6665781e-03*st1-6.3779372e-01*st2
61             +9.9730546e-01*st3+5.5924553e-01*u0-2.6626507e-06*u1;
62
63   setTimer(t_plant,0,100000);
64
65   putValue(state1,st0);
66   putValue(state2,st1);
67   putValue(state3,st2);
68   putValue(state4,st3);
69 }
70
71 on frStartCycle *
72 {
73   pdustate1.xb = st0;
74   frUpdatePDU(pdustate1,1,-1);
75
76   pdustate2.dxb = st1;
77   frUpdatePDU(pdustate2,1,-1);
78
79   pdustate3.xw = st2;
80   frUpdatePDU(pdustate3,1,-1);
81
82   pdustate4.dxw = st3;
83   frUpdatePDU(pdustate4,1,-1);
84 }
85
86 on frPDU FrPDUControlLaw
87 {
88   u1 = this.cl;
89 }
```

B.1.2 Experimentos 2 e 3

```
includes
{
}

variables
{
  // Future State x(k+1)
  float fst0 = 0;
  float fst1 = 0;
  float fst2 = 0;
  float fst3 = 0;

  // Plant States x(k)
  float st0 = 0;
  float st1 = 0;
  float st2 = 0;
  float st3 = 0;

  // Plant Inputs
  float u0 = 0;   // Disturbance
  float u1 = 0;   // Actuator Force

  frPDU FrPDUState1 pdustate1;
  frPDU FrPDUState2 pdustate2;
  frPDU FrPDUState3 pdustate3;
  frPDU FrPDUState4 pdustate4;

  // Timer for simulation
  timer t_plant;
}

on start
{
  setTimer(t_plant,0,100000);
}

on preStart
{
  frSetSendPDU(pdustate1);
  frSetSendPDU(pdustate2);
  frSetSendPDU(pdustate3);
  frSetSendPDU(pdustate4);
}

on timer t_plant
{
  u0 = getValue(disturbance);

  st0 = fst0;
  st1 = fst1;
  st2 = fst2;
  st3 = fst3;

  fst0 = 9.9999953e-01*st0+9.9984127e-05*st1+4.6311103e-07*st2
            +1.5872489e-08*st3+4.4442968e-09*u0+1.5849333e-11*u1;
  fst1 = -9.3511065e-03*st0+9.9968255e-01*st1+9.2622206e-03*st2
            +3.1744977e-04*st3+8.8885936e-05*u0+3.1698666e-07*u1;
  fst2 = 3.9274098e-06*st0+1.3332890e-07*st1+9.9996811e-01*st2
```

```
            +9.9865273e-05*st3+2.7962276e-05*u0-1.3313253e-10*u1;
  fst3 = 7.8548195e-02*st0+2.6665781e-03*st1-6.3779372e-01*st2
            +9.9730546e-01*st3+5.5924553e-01*u0-2.6626507e-06*u1;

  setTimer(t_plant,0,100000);

  putValue(state1,st0);
  putValue(state2,st1);
  putValue(state3,st2);
  putValue(state4,st3);
}

on frStartCycle *
{
  pdustate1.xb = st0;
  frUpdatePDU(pdustate1,1,-1);

  pdustate2.dxb = st1;
  frUpdatePDU(pdustate2,1,-1);

  pdustate3.xw = st2;
  frUpdatePDU(pdustate3,1,-1);

  pdustate4.dxw = st3;
  frUpdatePDU(pdustate4,1,-1);

  putValue(tps, timeNowNS());
}

on frFrame FrControlLaw
{
  u1 = this.cl;
  putValue(tua, timeNowNS());
  putValue(ctrl_msg_time, messageTimeNS(this));
}
```

B.2 Controladores

B.2.1 Controlador 1

```
  includes
{
}

variables
{
  float a11 = 1;
  float a12 = 0;
  float a21 = 0;
  float a22 = 1;

  float b11 = -0.01;
  float b12 = 0;
  float b13 = 0.01;
  float b14 = 0;
  float b21 = 0;
  float b22 = -0.01;
```

```
  float b23 = 0;
  float b24 = 0.01;

  float c11 = 5.681e4;
  float c12 = 3.248e4;

  float d11 = -284;
  float d12 = -4651;
  float d13 = -41.77;
  float d14 = 4142;

  float u1 = 0;
  float u2 = 0;
  float u3 = 0;
  float u4 = 0;

  float x1 = 0;
  float x2 = 0;

  float fx1 = 0;
  float fx2 = 0;

  frPDU FrPDUControlLaw resp;
}

on preStart
{
  frSetSendPDU(resp);
}

on frPDU FrPDUState1
{
  u1 = this.xb;
}

on frPDU FrPDUState2
{
  u2 = this.dxb;
}

on frPDU FrPDUState3
{
  u3 = this.xw;
}

on frPDU FrPDUState4
{
  u4 = this.dxw;
  fx1 = a11*x1 + a12*x2 + b11*u1 + b12*u2 + b13*u3 + b14*u4;
  fx2 = a21*x1 + a22*x2 + b21*u1 + b22*u2 + b23*u3 + b24*u4;
  resp.c1 = c11*x1 + c12*x2 + d11*u1 + d12*u2 + d13*u3 + d14*u4;
  frUpdatePDU(resp,1,-1);

  x1 = fx1;
  x2 = fx2;
}
```

B.2.2 Controlador 2

```
includes
{
}

variables
{
  // State x(k+1)
  float fst0 = 0;
  float fst1 = 0;
  float fst2 = 0;
  float fst3 = 0;
  float fst4 = 0;
  float fst5 = 0;
  float fst6 = 0;
  float fst7 = 0;
  float fst8 = 0;

  // Control States
  float st0 = 0;
  float st1 = 0;
  float st2 = 0;
  float st3 = 0;
  float st4 = 0;
  float st5 = 0;
  float st6 = 0;
  float st7 = 0;
  float st8 = 0;

  // Inputs
  float u0 = 0;
  float u1 = 0;
  float u2 = 0;
  float u3 = 0;

  // Control Inputs
  float ci1 = 0;
  float ci2 = 0;

  frPDU FrPDUControlLaw resp;
}

on preStart
{
  frSetSendPDU(resp);
}

on frPDU FrPDUState1
{
  u0 = this.xb;
}

on frPDU FrPDUState2
{
  u1 = this.dxb;
}

on frPDU FrPDUState3
{
```

```
  u2 = this.xw;
}

on frPDU FrPDUState4
{
  u3 = this.dxw;

  ci1 = u0 - u2;
  ci2 = u1 - u3;

  fst0 = -0.0118*st0 + 7.96e-6*st1 - 1.95e-5*st2 + 0.000277*st3
           - 0.0001089*st4 - 1.4e-5*st5 + 9.28e-8*st6 - 3.502e-8*st7
           - 4.166e-7*st8 + 1,596e-5*ci1 + 7,384e-6*ci2;
  fst1 = -83,29st0 + 0.985*st1 - 0.00167*st2 + 0.023*st3
           - 0.0091*st4 - 0.0012*st5 + 7.434e-6*st6
           - 2.846e-6*st7 - 3.49e-5*st8 + 0.00663*ci1
           + 0.00096*ci2;
  fst2 = 1835.322*st0 - 4.087*st1 + 0.339*st2 - 0.84*st3
           - 0.175*st4 - 0.0227*st5 + 0,00015*st6
           - 5.745e-5*st7 - 0.000667*st8 + 0.0267*ci1
           - 0.014*ci2;
  fst3 = 2541.94*st0 - 5.92*st1 - 0.71*st2 + 0.19*st3 - 0.22*st4
           - 0.0284*st5 + 0.000189*st6 - 7.055e-5*st7
           - 0.000844*st8 - 0.047*ci1 - 0.02*ci2;
  fst4 = 6172.497*st0 - 14.824*st1 - 1.768*st2 - 1.804*st3
           + 0.378*st4 - 0.082*st5 + 0,000597*st6 - 0.00023*st7
           - 0.0022*st8 -0.111*ci1 - 0.0479*ci2;
  fst5 = 98,46*st0 - 0.276*st1 - 0.0465*st2 - 0.223*st3
           + 0.037*st4 + 0.974*st5 - 0.00071*st6 - 0.000163*st7
           - 3.766e-5*st8 + 0.0142*ci1 - 0.00057*ci2;
  fst6 = -158.614*st0 + 0.711*st1 + 0.047*st2 + 0.3397*st3
           - 0.091*st4 - 0.0409*st5 + 0.9996*st6 - 0.00055*st7
           + 6.896e-5*st8 + 0.00223*ci1 + 0.0015*ci2;
  fst7 = -569.16*st0 - 10.4*st1 + 1.052*st2 - 11.977*st3
           + 4.675*st4 - 0.263*st5 + 0,01*st6 + 0.994*st7
           + 0.000336*st8 + 0.0161*ci1 + 0.0051*ci2;
  fst8 = -192778.203*st0 + 768.527*st1 + 13.468*st2 + 77.472*st3
           - 16.567*st4 - 1.1395*st5 + 0.044*st6 + 0.000722*st7
           + 0.8094*st8 + 16.71*ci1 + 1.65*ci2;

  resp.cl = 3860150619.67214*st0 - 125356.060206106*st1
           + 74273.8355102560*st2 - 1055929.89173690*st3
           + 415039.441694827*st4 + 53548.0298178308*st5
           - 355.506448523836*st6 + 133.870258332483*st7
           + 1586,09221239423*st8 - 57759.3585251614*ci1
           - 26054.4935193367*ci2;

  frUpdatePDU(resp,1,-1);

  st0 = fst0;
  st1 = fst1;
  st2 = fst2;
  st3 = fst3;
  st4 = fst4;
  st5 = fst5;
  st6 = fst6;
  st7 = fst7;
  st8 = fst8;
```

```
}
```

B.2.3 Controlador 3

```
includes
{
}

variables
{
  float k0 = 2624.69879380247;
  float k1 = 92.4171122172592;
  float k2 = -20974.5141287378;
  float k3 = -114.120983875527;

  float st0 = 0;
  float st1 = 0;
  float st2 = 0;
  float st3 = 0;

  frPDU FrPDUControlLaw resp;
}

on preStart
{
  frSetSendPDU(resp);
}

on frPDU FrPDUState1
{
  st0 = this.xb;
}

on frPDU FrPDUState2
{
  st1 = this.dxb;
}

on frPDU FrPDUState3
{
  st2 = this.xw;
}

on frPDU FrPDUState4
{
  st3 = this.dxw;
  resp.cl = k0*st0 + k1*st1 + k2*st2 + k3*st3;
  frUpdatePDU(resp,1,-1);
}
```