RICARDO DE ANDRADE

**Sistemas de comunicação CAN FD: modelamento por software e
análise temporal**

São Paulo
2014

RICARDO DE ANDRADE

**Sistemas de comunicação CAN FD: modelamento por software e
análise temporal**

Dissertação apresentada ao programa de
pós-graduação em Engenharia Elétrica da
Escola Politécnica da Universidade de São
Paulo como parte dos requisitos para a
obtenção do Título de Mestre em Ciências.

Orientador:

Prof. Dr. João Francisco Justo Filho

São Paulo

2014

RICARDO DE ANDRADE

**Sistemas de comunicação CAN FD: modelamento por software e
análise temporal**

Dissertação apresentada ao programa de
pós-graduação em Engenharia Elétrica da
Escola Politécnica da Universidade de São
Paulo como parte dos requisitos para a
obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia Elétrica - Microeletrônica

Orientador:
Prof. Dr. João Francisco Justo Filho

São Paulo
2014

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 29 de outubro de 2014.

Assinatura do autor _____

Assinatura do orientador _____

Catálogo-na-publicação

Andrade, Ricardo de

**Sistemas de comunicação CAN FD: modelamento por software e análise temporal / R. de Andrade. -- versão corr. – São Paulo, 2014.
102 p.**

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Redes de comunicação (Modelo; Desenvolvimento) 2.Sistemas de comunicação I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

DEDICATÓRIA

Dedico este trabalho aos meus pais, a minha irmã, a minha família, a minha noiva, aos meus amigos e colegas, e aos meus professores que estiveram junto comigo em toda esta jornada.

AGRADECIMENTOS

Agradeço aos meus pais, irmã e família primeiramente. Agradeço aos Profs. Drs. João Francisco Justo Filho e Armando Antônio Maria Laganá pela orientação e apoio durante todas as etapas deste mestrado.

Agradeço à USP e ao LSI por disponibilizarem verba para aquisição de ferramenta para este trabalho e por incentivar à pesquisa, e ao laboratório da FATEC que sempre esteve aberto para a realização de meus experimentos.

Agradeço ao Prof. Me. Kleber Nogueira Hodel, amigo e professor da FATEC Santo André, pelo total apoio, conselhos, ajuda e compreensão.

Agradeço aos amigos Bruno Casagrande, Marcos Carvalho Guedes, ao pessoal que trabalhou comigo na Continental Automotive Systems, aos colegas da Zael Eletroeletrônica Ltda, aos amigos e colegas da DSW Automotive, aos colegas de trabalho da UFABC pelo apoio e conselhos em especial ao Jarriv Alcantara de Oliveira Reginaldo pela ajuda em alguns pontos deste trabalho e aos meus amigos que considero muito.

Agradeço a toda equipe da empresa Vector Automotive da Alemanha, especialmente ao Sr. Guido Grothues, pelo suporte técnico e ao Sr. Jutta Schmidt, pelo suporte comercial da ferramenta.

Por fim, mas com muito amor, agradeço a minha noiva Juliana Shizue Fukuzaki que me ajudou na finalização deste trabalho e me proporcionou um aprendizado bem grande e intenso em muitos setores do conhecimento humano. Agradeço pela paciência e carinho que você teve comigo durante a finalização desta dissertação e em todos os outros momentos.

“É melhor tentar e falhar, que preocupar-se e ver a vida passar, é melhor tentar, ainda que em vão, que sentar-se fazendo nada até o final.

Eu prefiro na chuva caminhar que em dias tristes em casa me esconder.

Prefiro ser feliz, embora louco, que em conformidade viver...”

Martin Luther King

RESUMO

O CAN (*Controller Area Network*) é um padrão no barramento de comunicação, amplamente difundido em aplicações industriais, particularmente em sistemas automotivos. Atualmente, um dos principais problemas no ramo automotivo é que esse barramento está com muitas mensagens no barramento, resultado da incorporação incremental de sistemas eletrônicos em automóveis, visto que há uma exigência maior de conectividade devido às exigências da sociedade e mercado. Como alternativa, vem sendo desenvolvida uma nova rede de comunicação, conhecida como *CAN with Flexible Data-Rate* (CAN-FD), que é um barramento com velocidade de transmissão de informação mais alta e maior capacidade de transporte de dados. Este projeto tem por objetivo principal explorar as funcionalidades da rede CAN-FD, através de simulações do trânsito de mensagens numa rede CAN-FD usando os dados de uma rede real CAN, e verificando a previsibilidade de ambas no âmbito de um protocolo que possa atender à demanda de sistemas complexos. A comparação é executada a partir de um conjunto de mensagens adicionadas na rede, para verificar os limites de transmissão de cada uma das redes, e os respectivos tempos de atraso das mensagens. Como um segundo estudo de caso, uma rede de controle em malha fechada foi desenvolvida, conectada a um barramento CAN e um barramento CAN-FD. Essa técnica de controle permitiu eliminar os ruídos que interferem no controle, e checar o limite em que o protocolo de comunicação consegue manter em uma malha de controle funcionando. Os resultados mostraram que é possível transmitir uma imensa quantidade de dados com o menor uso do *busload* (quantidade de mensagens transmitidas) no veículo através do uso do barramento CAN-FD, porém ainda não foi lançado no mercado um controlador do CAN-FD para realizar essa tarefa. Por outro lado, os dois protocolos, CAN-FD e CAN, tem suas previsibilidades comprometidas pois não conseguem enviar a mensagem quando o barramento está superior a 98,86% de carga.

Palavras-chave: Eletrônica Automotiva. Comunicação Serial de Dados. CAN. CAN-FD. Redes Automotivas. Eletrônica Embarcada. Redes de Comunicação CAN-FD. CAN Bus. Controller Area Network.

ABSTRACT

The CAN (Controller Area Network) is a standard in the communication bus, widespread in industrial applications, particularly in automotive systems. Currently, one of the main problems in the automotive industry is that this bus is with many messages on the bus, the result of incremental incorporation of electronic systems in automobiles, since there is a greater demand for connectivity due to the demands of society and the market. Alternatively, it has been developed a new communications network, known as CAN with Flexible Data-Rate (CAN-FD), which is a bus with transmission speeds higher and higher capacity data transport information. This project's main objective is to explore the features of the network CAN-FD, through simulations of the traffic of messages on a CAN network FD using data from a real CAN network, and verifying the predictability both in the context of a protocol that can meet the demand complex systems. The comparison is performed from a set of messages added to the network to verify the boundaries of each of the transmission networks and the respective delay times of the messages. As a second case study, a network of closed-loop control was developed, connected to a CAN bus and CAN bus FD. This control technique has eliminated the noises that interfere with the control and check the extent that the communication protocol can keep a control loop running. The results showed that it is possible to transmit a huge amount of data with the lowest usage busload (amount of transmitted messages) to the vehicle through the use of CAN bus FD, but not yet released to market a CAN controller FD to accomplish this task . Moreover, both protocols, CAN-FD and CAN has its predictability compromised because they are unable to send the message when the bus is more than 98.86% load.

Keywords: Automotive Electronics. Serial Communication of Data. CAN. CAN-FD. Automotive networks. Embedded Electronics. Communication Networks CAN-FD. CAN Bus. Controller Area Network.

LISTA DE ILUSTRAÇÕES

Figura 1 - Impacto ao usuário com as novas funções sob o custo total do veículo. .	12
Figura 2 - Diagrama de arquitetura <i>stand-alone</i> . (Hodel, 2008).	18
Figura 3 - Diagrama de uma arquitetura centralizada.	19
Figura 4 - Diagrama de arquitetura distribuída.	20
Figura 5 - <i>Frames</i> do CAN-FD normal e CAN-FD estendido.....	26
Figura 6 - Início do quadro (<i>Start of Frame</i>).	26
Figura 7 - Campo de <i>arbitration</i>	27
Figura 8 - Campo de controle.	28
Figura 9 - Dados até 64 <i>Bytes</i> no CAN-FD.....	29
Figura 10 - Campo do CRC do CAN-FD.	30
Figura 11 - Campo do ACK do CAN-FD.....	30
Figura 12 - Segmento do tempo de <i>bit</i> no CAN e CAN-FD.	32
Figura 13 - Curva de deadline crítico.....	36
Figura 14 - Curva de deadline suave.....	36
Figura 15 - Taxa de transmissão dos dados no CAN-FD.	43
Figura 16 - <i>Busload</i> em % (eixo y) versus tempo em segundos (eixo x) comparando CAN e CAN-FD.	52
Figura 17 - Painel de configuração da simulação do CANoe da Vector	53
Figura 18 - Rede CAN com <i>busload</i> a 30%.....	55
Figura 19 – Dados e <i>busload</i> a 30% (eixo x) versus tempo em segundos (eixo y). .	56
Figura 20 - Mensagens adicionais para <i>busload</i> a 50%.	56
Figura 21 - <i>Busload</i> a 50% (eixo x) versus tempo em segundos (eixo y).	57
Figura 22 - Mensagens adicionais para obter um <i>busload</i> de 70%.	58
Figura 23 - <i>Busload</i> da rede CAN para 70% (eixo x) versus tempo em segundos (eixo y).	58
Figura 24 - Conversão de mensagens em linguagem de programação CAPL.....	63
Figura 25 - Gráfico em % do <i>busload</i> (eixo y) e tempo em segundos (eixo x) da rede CAN (linha roxa) e CAN-FD (linha dourada) de acordo com a primeira estratégia. .	64
Figura 26 - Gráfico em % do <i>busload</i> (eixo y) e tempo em segundos (eixo x) da rede CAN (linha roxa) e CAN-FD (linha dourada) de acordo com a segunda estratégia. .	65
Figura 27 - Mensagens atrasadas em milissegundos versus <i>bytes</i> versus <i>busload</i> do CAN de acordo com <i>busload</i>	66
Figura 28 - Gráfico do <i>busload</i> na rede CAN-FD.....	67
Figura 29 - Comparação da rede real CAN com a rede simulada CAN-FD.....	67
Figura 30 - Malha de controle fechada de um controlador PID.	70
Figura 31 – Modelo de motor CC.	71
Figura 32 - Controle de malha fechada na rede CAN e CAN-FD com controlador PID.	74
Figura 33 - Diagrama de blocos do sistema de controle CAN/CAN-FD.	75
Figura 34 - Diagrama de tempo do sistema de controle na rede CAN e CAN-FD....	76
Figura 35 - Comparação de 4 tipos de configurações de controle em uma rede CAN.	81
Figura 36 - Comparação de uma malha fechada com variação do <i>busload</i> na rede CAN-FD.....	84
Figura 37 - <i>Busload</i> limiar de uma rede CAN-FD	85
Figura 38 - Gráfico para orientação ao projetista	86

LISTA DE TABELAS

Tabela 1 - Protocolo ISO e sua descrição.	21
Tabela 2 - Comparação de características do CAN e CAN-FD.	23
Tabela 3 - Prioridade na transmissão de mensagem no barramento CAN-FD.	25
Tabela 4 - Cálculo teórico de uma rede CAN a 30% de <i>busload</i> utilizando a equação de Tindell.	46
Tabela 5 - Cálculo de uma rede com busload a 50%	47
Tabela 6 - Cálculo de uma rede CAN com <i>busload</i> de 70%.	48
Tabela 7 - Tabela de dados da rede CAN.	50
Tabela 8 - Dados da rede CAN-FD.	51
Tabela 9 - Mensagens convertidas de CAN para CAN-FD	54
Tabela 10 - Número de <i>bytes</i> em CAN e CAN-FD de acordo com o DLC.	54
Tabela 11 - Tempo das mensagens com CAN a 30% de <i>busload</i>	60
Tabela 12 - Tempo das mensagens com CAN a 50% de <i>busload</i>	61
Tabela 13 - Tempo das mensagens com CAN a 70% de busload.	62
Tabela 14 - Tarefas de controle no CAN e CAN-FD	77
Tabela 15 - Tempo de fila das mensagens na malha fechada na rede CAN.	79
Tabela 16 – Tempo de fila das mensagens na malha fechada na rede CAN-FD.	82

LISTA DE ABREVIATURAS E SIGLAS

ABS	Anti locking-blocking System
ACK	Acknowledge
BBW	Brake-By-Wire
BRS	Bit Rate Switch
BRP	Baud Rate Prescaler
CAN	Controller Area Network
CAN-FD	Controller Area Network with Data Flexible
CDMA/CD+AMP	Carrier Sense Multiple Access/Collision Detection + Arbitration on Message Priority
Cluster	Painel de Instrumentos
CRC	Checksum redundancy cyclic
DLC	Data Length Code
EBS	Electronic Brake System
EDL	Extended Data Length
EOF	End of Frame
ESI	Error State Indicator
ESC	Electronic Stability Control
FRAME	Quadro
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
MAC	Media Access Control
ms	Milisegundos
OSI	Open System Interconnection
Rx	Recepção
SOF	Start of Frame
SDS	Smart Distributed System
Tx	Transmissão
USP	Universidade de São Paulo

SUMÁRIO

1	INTRODUÇÃO	12
1.1	MOTIVAÇÃO	12
1.2	OBJETIVOS	15
1.3	CONTEÚDO E ORGANIZAÇÃO	16
2	ARQUITETURAS ELETRO/ELETRÔNICAS E REDE CAN.....	17
2.1	ARQUITETURAS ELETRO/ELETRÔNICAS.....	17
2.1.2	ARQUITETURA <i>STAND-ALONE</i>	17
2.1.3	ARQUITETURA CENTRALIZADA	18
2.1.4	ARQUITETURA DISTRIBUÍDA	19
2.1.5	NORMALIZAÇÃO E HISTÓRICO DO CAN	20
2.2	APLICAÇÕES INICIAIS	22
2.3	VANTAGENS GERAIS DO CAN.....	23
2.4	HISTÓRICO DO CAN-FD	23
2.4.1	CARACTERÍSTICAS GERAIS DO CAN-FD.....	24
2.4.2	FUNCIONAMENTO DO CAN-FD	25
2.4.3	FRAME CAN-FD.....	25
2.4.4	CAMPO DO <i>START OF FRAME</i>	26
2.4.5	CAMPO DE ARBITRAÇÃO	27
2.4.6	CAMPO DE CONTROLE	28
2.4.7	CAMPO DE DADOS.....	29
2.4.8	CAMPO DO CRC E <i>BIT STUFFING</i>	29
2.4.9	CAMPO DO ACK	30
2.5	FILTRAGEM E VALIDAÇÃO DE MENSAGENS.....	31
2.5.1	CONFIGURAÇÃO DE SINCRONIZAÇÃO E VELOCIDADE.....	31
2.6	SISTEMAS DE TEMPO REAL	34
2.6.1	CONCEITOS FUNDAMENTAIS DE SISTEMAS DE TEMPO REAL.....	35
2.6.2	TAREFAS DE TEMPO REAL E <i>DEADLINE</i> CRÍTICO E SUAVE.....	35
2.6.3	TIPOS DE TAREFA.....	37
2.7	CAMADAS ISO/OSI.....	37
2.7.1	CAMADA FÍSICA	37
2.7.2	CAMADA DE ENLACE	37
2.7.3	CAMADA DE REDE.....	38
2.7.4	CAMADA DE TRANSPORTE	38
2.7.5	CAMADA DE SESSÃO	38
2.7.6	CAMADA DE APRESENTAÇÃO	39
2.7.7	CAMADA DE APLICAÇÃO.....	39
3	METODOLOGIA: CÁLCULO DE TEMPOS E RESPOSTAS DO FRAME CAN E CAN-FD.....	40
3.1	CONCEITOS INTRODUTÓRIOS DA METODOLOGIA	40
3.2	DESENVOLVIMENTO DO MODELO DE CÁLCULO DO <i>BUSLOAD</i>	40
3.3	ESTUDO DE CASO 1.....	44
4	ESTUDO DE CASO 1 E RESULTADOS.....	49
4.1	MENSAGENS IDENTIFICADAS NO BARRAMENTO CAN	49

4.2	ANÁLISE TEMPORAL DE UMA REDE CAN A 30%, 50% E 70% DE <i>BUSLOAD</i>	55
4.3	CONVERSAO DE MENSAGENS CAN PARA CAN-FD EM CAPL.....	62
4.4	ANÁLISE GRÁFICA TEMPORAL DE MENSAGENS NA REDE.....	65
5	ESTUDO DE CASO 2	69
5.1	DEFINIÇÕES DO EXPERIMENTO.....	70
5.2	ANÁLISE TEMPORAL E CONTROLE DA MALHA FECHADA DA REDE CAN.....	79
5.2.1	RESULTADOS DA ANÁLISE TEMPORAL DE CONTROLE DA MALHA FECHADA DA REDE CAN	80
5.3	ANÁLISE TEMPORAL E CONTROLE DA MALHA FECHADA DA REDE CAN-FD	82
5.4	VALIDAÇÃO DO MÉTODO DEFINIDO PARA CÁLCULO DO <i>BUSLOAD</i>	85
6	CONCLUSÃO.....	87
	TRABALHOS DECORRENTES	89
	REFERÊNCIAS.....	90
	APÊNDICE A.....	95
	APÊNDICE B.....	97

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

No início da década de 70, a crise mundial do petróleo marcou a história da indústria automotiva, fomentando a busca por uma maior eficiência nos automóveis. Mais recentemente, questões ambientais apresentam novos desafios para a redução de poluentes. Esses dois fatores, eficiência e emissão de poluentes, são fundamentais no desenvolvimento de motores automotivos a combustão, para atender às demandas da legislação nacional. Uma trajetória para se construir automóveis com essas características é incorporar sistemas eletrônicos, desde redes de sensores e atuadores até sistemas de controle, o que se denomina eletrônica embarcada automotiva. A incorporação dessa eletrônica automotiva teve grande impacto nas últimas duas décadas devido, principalmente, à disseminação dos microcontroladores (NEVERS, 1995).

Desde 1990, o setor de eletrônica embarcada, especialmente aquele associado a *software*, vem aumentando a uma taxa de aproximadamente 10% ao ano (NEVERS, 1995). Em 2006, o custo médio de um sistema eletrônico embarcado de um automóvel era de aproximadamente 25% do seu preço final, chegando a 35% em veículos de alta tecnologia. A Figura 1 - Impacto ao usuário com as novas funções sob o custo total do veículo, mostra a curva de evolução do custo da eletrônica embarcada em um veículo de acordo com a tecnologia utilizada em cada momento específico.

Figura 1 - Impacto ao usuário com as novas funções sob o custo total do veículo.

Fonte: Santos(2000).

Os avanços da indústria automobilística seguem diversas vertentes, buscando melhorias em sua manufatura em termos de desempenho, eficiência, segurança, conforto e conveniência. Essas melhorias geralmente requerem o desenvolvimento de novas tecnologias, que envolvem diversos sistemas de *hardware* e *software*. Devido a esses fatos, a eletrônica vem sendo incorporada nos automóveis continuamente. Nos veículos são inseridos módulos eletrônicos que são chamados de ECU (*Electronic Control Unit*) as quais existem em carros com até 70 desses módulos diferentes e 500 MB de memória total, sendo que todos esses dispositivos estão conectados em redes de comunicação automotivas, tal como o *Controller Area Network* (CAN) (BOSCH, 2005).

As ECU's são responsáveis por ler as informações originadas em sensores colocados em pontos estratégicos do veículo, processá-las e utilizá-las como base para controlar diversos atuadores. Esses módulos funcionam como um computador, possuindo uma placa de circuito impresso com um microprocessador ou

microcontrolador e um *software* gravado em seus respectivos bancos de memória (SCHMIDT, 2007). Os microprocessadores são circuitos integrados que possuem maior capacidade de operação lógica e espaço para armazenamento de memória RAM. (SCHMIDT, 2007). Já os microcontroladores são mais limitados, com menor espaço de endereçamento da memória e dedicados a sistemas embarcados.

Com legislações ambientais cada vez mais rígidas, foram desenvolvidos complexos sistemas eletrônicos para o controle de emissões e redução no consumo de combustível. Adicionalmente, a redução dos custos dos circuitos integrados e dos microprocessadores possibilitou a sua aplicação em outras funções dos veículos e, conseqüentemente, houve um grande desenvolvimento na área da eletrônica veicular embarcada. (PLOSS; MUELLER; LETEINTURIER, 2008).

O desempenho e a confiabilidade oferecidos pelos componentes eletrônicos, aliados às possibilidades apresentadas por técnicas de *software*, permitiram a implantação de novas funções (NAVET e SIMONOT-LION, 2009). Exemplos clássicos são os sistemas ABS (*Anti Blocking System*) e ESC (*Electronic Stability Control*), que atuam respectivamente como controle anti-travamento das rodas durante uma frenagem brusca do veículo e mantém a estabilidade no caso de uma situação de possível capotagem. Nos Estados Unidos, por exemplo, um estudo realizado pela Administração Nacional de Segurança no Tráfego em Rodovias (NHTSA) revelou que o sistema ABS reduziu a taxa de colisões em 6% nos veículos de passeio (KAHANE; DANG, 2009), enquanto que outro estudo realizado pela mesma agência mostrou uma redução de 34% graças ao sistema ESC (DANG, 2007).

Quando os módulos de gerenciamento eletrônicos (ECU's) começaram a ser desenvolvidos nos automóveis, elas não trocavam informações entre si, funcionando de forma independente (NOLTE; HANSSON; BELLO, 2005). Neste tipo de arquitetura, apesar da vantagem de se ter um *hardware* simples, a falta de integração entre os sistemas leva, dentre outras desvantagens, à necessidade de utilização de vários equipamentos de diagnóstico e, em alguns casos à duplicidade de sensores que capturam a mesma informação, mas enviam para sistemas diferentes (VAZ, 2011).

Portanto, houve a necessidade de pesquisas serem realizadas por Universidades e indústrias. Devido a esse fato, a microeletrônica vem evoluindo e, com isso, houve uma diminuição do número de cabos no veículo. Desta forma, foram criados os sistemas distribuídos, que consistem em ECU's específicas para o controle

de diferentes partes do veículo, interligadas por uma rede veicular de dados. Semelhante a uma rede de computadores, a troca de informações em uma rede veicular do tipo distribuída garante que os sistemas eletrônicos trabalhem de forma unificada, compartilhando as informações provenientes dos sensores e, portanto, reduzindo a quantidade de cabos e dispositivos no veículo e, conseqüentemente, o seu peso e consumo de combustível (MURPHY, 2009).

Os sistemas distribuídos estão presentes em painéis de instrumento, limpadores, luzes, portas, janelas, bancos, retrovisores, e controle de climatização. Eles vem sendo cada vez mais controlados por *softwares* baseados em rede de comunicação e microcontroladores que utilizam fios ao invés de peças mecânicas (NAVET; SONG; SIMONOT-LION; WILWERT, 2005)

Os sistemas, que antes eram controlados mecanicamente, vem sendo gradativamente substituídos por sistemas puramente eletrônicos, como, por exemplo, o acelerador eletrônico (*throttle-by-wire*), que substituiu o sistema com cabo. Provavelmente, nos próximos anos, haverá uma substituição dos sistemas mecânicos de direção e freios atuais por sistemas puramente eletroeletrônicos. O sistema “*brake-by-wire*” vai substituir os circuitos hidráulicos ou pneumáticos do freio por cabos elétricos, enquanto que o sistema “*steer-by-wire*” vai implantar um comando eletrônico na coluna de direção (HODEL, 2010). Os veículos de luxo atuais já possuem sistemas inteligentes que integram diversos itens de segurança para atuarem em conjunto, a fim de minimizar os efeitos provenientes de falhas mecânicas ou dos motoristas. Essa nova geração de sistemas ativos de segurança é capaz de identificar situações de risco, através da fusão dos sinais de sensores em diversas partes do veículo, que reconhecem o ambiente que está ao redor do mesmo, sendo capazes de emitir alertas sonoros, visuais e, em um futuro próximo, até mesmo intervir no veículo para evitar ou minimizar os efeitos dos acidentes (SCHULZE *et al.*, 2008).

Desta forma, a fim de implementar um sistema capaz de ter todas essas funções, o CAN torna-se crucial para permitir um funcionamento harmônico de todos os dispositivos automotivos (PINTO; RUFINO; ALMEIDA, 2011).

1.2 OBJETIVOS

O principal objetivo deste trabalho é explorar os sistemas de comunicação de dados e suas limitações, desenvolvendo ferramentas para executar a investigação e demonstrar se existem vantagens do protocolo CAN-FD sobre o CAN convencional. Além disso, foi desenvolvido um controle de malha fechada, a fim de identificar os limites desses protocolos quando inseridos em um ambiente com perturbações de sinais e ruídos. A investigação explorou, ainda, se o sistema tem sua confiabilidade temporal adequada ao sistema automotivo.

1.3 CONTEÚDO E ORGANIZAÇÃO

Este trabalho está organizado em cinco capítulos. O capítulo 1 apresenta uma introdução e motivação do estudo, o capítulo 2 discute as arquiteturas eletro/eletrônicas, incluindo um histórico do sistema e uma revisão bibliográfica. O capítulo 3 apresenta os elementos de metodologia e modelagem que serão usados no trabalho, enquanto que o capítulo 4 apresenta o sistema de estudo e resultados de simulações. O capítulo 5 apresenta o estudo de caso 2 e seus resultados. Finalmente, o capítulo 6 mostra a conclusão deste trabalho.

2 ARQUITETURAS ELETRO/ELETRÔNICAS E REDE CAN

2.1 ARQUITETURAS ELETRO/ELETRÔNICAS

As arquiteturas eletrônicas são importantes na estrutura da rede de comunicação. Visto isso, é importante estudá-las para entender como o CAN e o CAN-FD funcionam. O conjunto de vários módulos eletrônicos conectados em uma rede embarcada é denominado de arquitetura elétrica. Os módulos tem como função controlar todos os dispositivos eletrônicos do veículo que são ligados através de uma arquitetura. Diversos modelos de arquitetura, encontrados com maior frequência em veículos, serão discutidos neste capítulo, como arquitetura individual, arquitetura centralizada e arquitetura distribuída (MIESTERFIELD, 1999).

2.1.2 Arquitetura *stand-alone*

A arquitetura individual, que foi a primeira a ser utilizada em aplicações industriais, especialmente no setor automotivo, é conhecida como analógica e não possui uma interação entre os módulos do sistema mas somente interação com seus subsistemas. Existem vantagens e desvantagens ao se utilizar o sistema *stand-alone*. Como vantagens, pode-se mencionar a independência para o desenvolvimento de sistemas e a simplicidade do *hardware* associado. Por outro lado, há inúmeras desvantagens, cada sistema tem seu próprio diagnóstico com variadas ferramentas para aquisição de dados, além da dificuldade de manutenção, deixando o projeto com custo proibitivo. O grande número de cabos, necessários no sistema para conexão de sensores e atuadores, também dificulta a sua manutenção. A Figura 2 ilustra o conceito desta arquitetura.

Figura 2 - Diagrama de arquitetura *stand-alone*. (Hodel, 2008).

Fonte: Guimarães (2007).

2.1.3 Arquitetura centralizada

Este tipo de arquitetura pode receber todos os sinais dos módulos integrados, chamados como ECU do sistema, conforme apresentado na Figura 3. Essa foi uma evolução da arquitetura analógica pois todas entradas e saídas se encontram em apenas um módulo. As vantagens dessa arquitetura são diversas: permite um diagnóstico único para o sistema, apresenta um *hardware* simples, além do custo reduzido para executar o diagnóstico. Por outro lado, como desvantagem, o sistema requer uma grande quantidade de cabos para conectar os módulos, dificultando a manutenção da rede.

Figura 3 - Diagrama de uma arquitetura centralizada.

Fonte: Guimarães (2007).

2.1.4 Arquitetura distribuída

A arquitetura distribuída é usada atualmente pelas principais empresas automotivas, principalmente devido às possibilidades de se adicionar módulos do tipo *plug & play*, que sempre está pronto para funcionar. A principal desvantagem dessa arquitetura é a necessidade de um *software* de controle da rede, uma vez que precisa de um sistema robusto para gerenciar todo o barramento. A Figura 4 apresenta um diagrama dessa arquitetura, que é a utilizada pela rede CAN, e que será explicada no próximo tópico. A representação abaixo possui “n” módulos (ECU’s) que estão todos conectados em um único barramento. Esses nós possuem entradas e saídas de sinais que estão ligadas aos seus respectivos sensores e atuadores. É necessário apenas um *software* de diagnóstico para fazer análise das informações.

Figura 4 - Diagrama de arquitetura distribuída.

Fonte: Guimarães (2007).

2.1.5 Normalização e histórico do CAN

Em 1983, os pesquisadores Wolfhard Lawren, da Universidade de Ciências Aplicadas Braunschweig-Wolfenbüttel, e Horst Wettstein, da Universidade de Karlsruhe aplicaram o conceito das arquiteturas eletroeletrônicas, especificamente a de rede distribuída na rede CAN (*Controller Area Network*) em veículos automotores. Esses pesquisadores foram contratados pela empresa alemã Mercedes-Benz, uma das líderes mundiais no setor automotivo, para atuarem juntamente com engenheiros da empresa Bosh, procurando desenvolver um novo protocolo serial, uma vez que naquela época nenhum protocolo existente atendia aos requisitos exigidos. Desta forma, a Mercedes-Benz e a Bosh uniram-se para implementar esse novo sistema (WEI LUN *et al.*, 2010). Em fevereiro de 1986, no Congresso da SAE (*Society of Automotive Engineering*), em Detroit nos Estados Unidos da América, foi apresentado o estudo de uma rede *multi-master* serial síncrona, ou seja, uma rede com vários módulos que transmitem *bits* sequencialmente. Em 1989, um grupo de usuários de CAN, associado ao ramo de indústrias têxteis, desenvolveu alguns um protocolo de alto nível que ajudaram a formar o CAN Kingdom (CIA, 2012). A Philips também se

juntou ao grupo de usuários do CAN Industrial, empregando essa rede em aparelhos de raios-X (KIENCKE, 1986).

A rede de comunicação CAN surgiu primeiramente para veículos de passeio em 1992, através da montadora Mercedes-Benz, sendo posteriormente utilizada em diversos outros segmentos. O sistema CAN passou a ser aplicado com bastante frequência em toda a Europa, sendo que em alguns países, como a Finlândia, a rede foi utilizada para comunicação na indústria de elevadores, e na Suécia na indústria de máquinas têxteis.

Esse barramento foi apresentado como uma rede baseada em mensagens com alta prioridade na transmissão. No ano de 1992 foi fundado um grupo de usuários de CAN, chamado de *CAN in Automation* (CiA). Esse grupo era formado primordialmente por engenheiros do mundo todo, que vem debatendo, até os dias de hoje, qualquer mudança técnica necessária para a rede. Naquela época, ficou estabelecido que a principal funcionalidade era a redução de fios no sistema, uma vez que o sistema utiliza apenas dois fios trançados para transmitir os dados (CAN High e CAN Low) (CIA, 2012). Nesse início de pesquisas, os engenheiros da Mercedes-Benz estiveram bastante envolvidos, a ponto de serem considerados os pioneiros na implementação automotiva. Naquela época, foi escolhida a empresa Intel como a principal fornecedora de semicondutores, sendo a primeira a fornecer o controlador 82526, seguida pela Philips, com o controlador 82C200, que possuía até 16 MHz de frequência e suportava temperaturas entre -40°C e 85°C, o que já representava algo bastante satisfatório até mesmo considerando aplicações atuais (CHEN; TIAN, 2009).

Em 1993, foi implementado o padrão com a ISO 11898-2 (alta velocidade), ISO 11898-3 (tolerante a falhas/baixa velocidade), ISO 11992-1 e SAE J2411 (meio de transmissão que utiliza apenas uma linha para transmissão e recepção de mensagens (CIA, 2001). Pode ser visualizado na Tabela 1 os padrões ISO.

Tabela 1 - Protocolo ISO e sua descrição.

Padrão ISO	Descrição
11898-1	Especifica camada de dados
11898-2	Especifica alta velocidade taxa até 1Mbps
11898-3	Especifica baixa velocidade taxa de 40Kbps até 125Kbps
11898-4	Especifica <i>time-triggered</i>
11898-5	Especifica camada física para taxa até 1Mbps e sistemas que requerem baixo consumo de energia
11898-6	Especifica camada física para taxa até 1Mbps e sistemas que requerem um mecanismo de <i>Wake-up</i> configurável no <i>frame</i> do CAN

Fonte: Elaborado pelo autor.

2.2 APLICAÇÕES INICIAIS

Dois sistemas de baixo nível surgiram na década de 90 para o domínio de aplicação em automação industrial, criados pelos engenheiros Allen-Bradley e Honeywell Microswitch. Foram denominados de *DeviceNet* e *Smart Distributed System* (SDS). O SDS é um sistema de comunicação de controlador lógico baseado no sistema CAN. É capaz de fazer leitura de dados e tomar decisões na rede. Esse protocolo possibilita ao *framework* comunicação do tipo *peer-to-peer*, ou seja, comunicação de ponto a ponto com todos os nós na rede. (RAMKUMAR *et al.*, 1998). Entretanto, somente o *DeviceNet* se popularizou e chegou a concorrer diretamente com outros protocolos de comunicação da época que utilizavam as redes Profibus-DP (BAO, 2009) e Interbus, que possui uma arquitetura do tipo anel. Esse protocolo possui cabos de transmissão e recepção e um sistema que está ligado em série a todos os nós na rede de forma contínua, ou seja, um nó recebe e envia a informação para o outro, sendo todos dependentes do nó anterior. A desvantagem é essa dependência do nó anterior, pois se um nó tiver falha, o próximo não receberá os dados (LI HUI *et al.*, 2009). O *DeviceNet* também é baseado no CAN (SCHIFFER *et*

al., 2000). Devido a sua conectividade *plug-and-play*, ele se tornou o sistema de comunicação de dados na liderança para o mercado específico de aplicações nos Estados Unidos (CIA, 2001). Todos esses controladores foram construídos baseados na arquitetura *multi-master*, com tamanho de dados com até 8 *Bytes* (LUO *et. al.*, 2008).

2.3 VANTAGENS GERAIS DO CAN

Algumas características do protocolo CAN foram determinantes para o sistema ter se disseminado nos sistemas de comunicação de dados em aplicações industriais (NEVERS, 1995). As principais características são a redução de volume de cabos e a facilidade de diagnóstico.

2.4 HISTÓRICO DO CAN-FD

O CAN-FD (*Controller Area Network with Flexible Data-Rate*) é uma evolução do CAN e foi apresentada na 13ª Conferência Internacional do CAN, realizada em Nuernberg-Alemanha em 2012, com as empresas Bosch e Vector. Nessa conferência, foi executada uma simulação em bancada da transmissão de dados de um veículo, onde se demonstrou que a taxa de transmissão podia chegar até 15 Mbps, utilizando a *interface* do CANoe que é uma ferramenta de diagnóstico desenvolvida pela Vector (CIA, 2012). O CAN-FD tem toda sua estrutura baseada no CAN, a diferença principal é a quantidade de dados (até 64 *Bytes*) que o CAN-FD suporta, enquanto o CAN suporta apenas 8 *Bytes* no seu *Frame*, como mostrado na Tabela 2. Além disso, o *hardware* muda a parte do controlador. O CAN-FD tem como principal diferença, em relação ao CAN, o suporte de dois *bits* que são responsáveis por alterar a velocidade de transmissão da mensagem (VECTOR, 2013).

Tabela 2 - Comparação de características do CAN e CAN-FD.

Características	CAN	CAN-FD
Tamanho de dados máximo	8 Bytes	64 Bytes
Máxima taxa de transmissão de dados	1Mb/s	10Mb/s
<i>Bits</i> para CRC	15	17 ou 21

Fonte: Elaborado pelo autor.

2.4.1 Características gerais do CAN-FD

O CAN-FD tem flexibilidade na configuração da rede, sendo possível acrescentar vários nós na rede, sem causar problemas técnicos no sistema. A rede trabalha no conceito multi-mestre, eliminando a necessidade de um módulo principal responsável pelo gerenciamento da rede. Adicionalmente, em seu *frame*, ele tem um campo de detecção de erros, retransmissão de mensagens corrompidas, como no barramento CAN. Uma característica relevante é a compatibilidade com o CAN, ou seja, todo nó do CAN-FD está apto a receber e transmitir mensagens de acordo com a norma ISO 11898-1 (CIA, 2012).

2.4.2 Funcionamento do CAN-FD

O CAN-FD, conforme discutido anteriormente, é um sistema *multi-master*, ou seja, pode ter mais de um nó controlador, o que facilita a criação de um sistema redundante. Ele usa um protocolo de acesso conhecido como CDMA/CD+AMP (*Carrier Sense Multiple Access/Collision Detection + Arbitration on Message Priority*). Por esse motivo, o CAN-FD trabalha de forma semelhante à ethernet comum, mas ao invés de corrigir as colisões de transmissão, faz com que um dos dois nós em conflito pare de transmitir. A rede do CAN utiliza o arbítrio de comparação binária, que define qual será a prioridade de envio da mensagem de acordo com a lógica dos ID's (*identifiers*), onde ID é o endereço de cada módulo eletrônico.

Quanto menor for o valor do ID, maior será sua prioridade no barramento. O bit 0 recebe a denominação de bit dominante e o bit 1 recebe a denominação de bit recessivo. A Tabela 3 ilustra como funciona a prioridade no barramento do CAN. É possível inferir que ao analisar a rede, o nó A (ID 205) é o ganhador da prioridade ao transmitir uma mensagem, pois tem o menor ID do barramento. O segundo com maior prioridade seria o nó B (ID 250) e por último o nó C (ID 400).

Tabela 3 - Prioridade na transmissão de mensagem no barramento CAN-FD.

Nós na Rede	ID														
A	205	0	0	1	0	0	0	0	0	0	0	1	0	1	
B	250	0	0	1	0	0	1	0	1	0	0	0	0	0	
C	400	0	1	0	0	0	0	0	0	0	0	0	0	0	
Ganha Barramento	205	0	0	1	0	0	0	0	0	0	0	1	0	1	

Fonte : Elaborado pelo autor

2.4.3 Frame CAN-FD

A Figura 5 apresenta o *Frame* do CAN-FD normal e do CAN-FD estendido. A diferença entre eles é apenas a quantidade de *bits* do ID. São 11 *bits* para o CAN padrão e 29 *bits* para o CAN-FD estendido (VECTOR, 2013). Nessa figura, cada número representa a quantidade de *bit* que deve ser inserido ali. A diferença entre as

regiões laranja e azul é que a parte laranja é onde o valor do *clock* é fixa igual no CAN e a parte azul é onde o *clock* pode ser alterado. Neste trabalho, ele foi configurado em 4MHz, que será discutido mais detalhadamente nas próximas seções.

Figura 5 - Frames do CAN-FD normal e CAN-FD estendido.

Fonte: Vector (2013).

2.4.4 Campo do *start of frame*

A Figura 6 apresenta o *Frame* constituído de duas fases, *Arbitration Phase* e *Data Phase*, além de possuir 7 (sete) diferentes campos para os *bits* (VECTOR, 2013).

- *Start of Frame*: é o início da transmissão de dados, por *default* o *bit* deve ser sempre dominante (*bit* 0), como ilustrado na Figura 6.

Figura 6 - Início do quadro (*Start of Frame*).

Fonte: Vector (2013).

2.4.5 Campo de arbitração

O campo *Arbitration* está localizado entre o *Start of Frame* (SOF) e o r1. O *Identifier* é o endereço da mensagem (destinatário e remetente), e o r1, que deve ser sempre um *bit* recessivo, como mostrado na Figura 7 (VECTOR, 2013). Esse tipo de padrão na comunicação vem sendo usado mesmo antes de 1992, através de outras redes de comunicação, como Profibus e DeviceNet. Antes de enviar o dado da mensagem, é necessário saber o destinatário, representado pelo ID, senão a comunicação não pode ser estabelecida. Neste estudo, verificou-se, através de simulações, que não há vulnerabilidade dessa forma.

Figura 7 - Campo de *arbitration*.

Fonte: Vector (2013).

2.4.6 Campo de controle

O campo de Controle (*Control*) é composto dos seguintes *bits*: IDE (*Identifier extension bit*) ou *bit* de extensão do identificador, EDL (*Extended Data Length*) ou tamanho de dado estendido, r0 (*Reserved Bit*) ou *bit* de reserva, BRS (*Bit Rate Switch*) ou chaveamento da taxa de *bit*, ESI (*Error State Indicator*) ou indicador de estado de erro e DLC (*Data Length Code*). O IDE é aquele que mostra se o frame é 11 *bits* ou 29 *bits* no ID, o qual depende da aplicação. Se um sistema com muitas mensagens precisa ser desenvolvido com maior robustez e maior quantidade de nós no barramento, então seria necessário utilizar 29 *bits* no ID. Caso contrário, em sistemas de menor porte, menor quantidade de nós interligados, deve-se utilizar um ID de 11 *bits*. Para selecionar a rede CAN é necessário colocar 0 no *bit* EDL (*Extended Data Length*) e 1 para selecionar a rede CAN-FD, padrão adotado pelas empresas Vector e Bosh. R0 é *bit* reserva, sempre recessivo (1) definido na construção da arquitetura do *software*. O BRS (*Bit Rate Switch*) é o responsável pela taxa de transmissão de velocidade dos dados. E quando recessivo, a taxa de *clock* é chaveada para um valor que o desenvolvedor planejar alterar. O ESI (*Error State Indicator*) é o *bit* indicador de erro passivo e ativo. A Figura 8 mostra o Campo de Controle, onde começa a transmissão no SOF, o endereço (ID), o tamanho de nós na rede (IDE), o tipo de comunicação (CAN ou CAN-FD), (*Error State Indicator*), a taxa de transmissão dos dados (*Bit Rate Switch*), o tipo de erro (ESI) e a quantidade de dados que será transmitido (*Data Length Code*). Essa ordem não poderia ser diferente, pois até enviar os dados, o controlador precisa identificar o tipo de dado que ele transmitirá (VECTOR, 2013).

Figura 8 - Campo de controle.

Fonte: Vector (2013).

2.4.7 Campo de dados

O *bit* DLC (*Data Length Control*) mostra a quantidade de *bytes* que pode ser inserido no campo de dados. Ele é composto de 4 *bits* que determinam a quantidade de dados que serão enviados. Para enviar 8 *bytes* de dados, deve-se introduzir 1000 no frame, ou para enviar 64 *bytes*, configura-se para 1111. Desta forma, o CAN-FD pode enviar 8, 12, 16, 20, 24, 32, 48 e 64 *bytes*, permitindo grande flexibilidade e robustez. O máximo que o *hardware* suporta atualmente é essa configuração de 4 *bits*. Se fosse inserido mais *bits* no DLC, o *hardware* não suportaria devido a sua máxima taxa de transmissão de dados. De acordo com a Figura 9, o CAN-FD consegue receber até 64 *bytes* no seu *frame*. Os dados são transferidos de acordo com os *bits* mais significativos (VECTOR, 2013).

Figura 9 - Dados até 64 Bytes no CAN-FD.

Fonte: Vector (2013).

2.4.8 Campo do CRC e *bit stuffing*

O tamanho do CRC (*Cyclic Redundancy Checksum*), que é o campo que faz o cálculo de erro na transmissão de dados, é diferente daquele encontrado no CAN. São 15 *bits* para o CAN, 17 *bits* para CAN-FD, onde *Data Field* ≤ 16 Bytes e 21 Bytes para o CAN-FD quando o *Data Field* > 16 Bytes porque quanto maior a quantidade de dados a ser transmitidos, maior deve ser a capacidade do CRC de executar os cálculos, padrão o qual foi definido pela Vector. O cálculo de *bit stuffing* é executado por padrão a cada 5 *bits* consecutivos. Então, após o quinto *bit*, é introduzido um *bit* invertido, a fim de evitar que estejam sendo enviados dados errados ao sistema. Esse mecanismo permite sincronizar todas as mensagens no barramento.

No exemplo abaixo, verifica-se que a cada sequência de 5 (cinco) zeros, deve-se inserir um *bit* recessivo (1).

000001000001000001...

O CRC, conforme a Figura 10, é calculado depois do *bit stuffing* (STEVENSON *et al.*, 2012).

Figura 10 - Campo do CRC do CAN-FD.

Fonte: Vector (2013).

2.4.9 Campo do ACK

O ACK ou *Acknowledge* é o campo que envia um *bit* para confirmação de recebimento de cada mensagem. Ele envia esse *bit* no final do campo CRC. Houve uma pequena mudança de comportamento em relação ao CAN, pois no CAN-FD, o módulo receptor reconhece até 2 (dois) tempo de *bits* como ACK. O *End of Frame* é o campo depois do ACK que possui 7 *bits* recessivos e termina a transmissão de uma mensagem, como mostrado na Figura 11 (VECTOR, 2013).

Figura 11 - Campo do ACK do CAN-FD.

Fonte: Vector (2013).

2.5 FILTRAGEM E VALIDAÇÃO DE MENSAGENS

A fim de receber uma mensagem transmitida no barramento, o nó necessitará que o seu filtro esteja preparado para recebê-la. São as máscaras do próprio ID que executam esse filtro. Ela compara os bits do identificador esperado e do identificador recebido. Quando uma mensagem não conter erro, o nó guardará os dados e sinalizará para que a mensagem seja recebida com sucesso, caso haja erro na transmissão, os dados não serão recebidos.

No transmissor, se uma mensagem é corrompida, a retransmissão será feita automaticamente de acordo com a priorização. O número de tentativas de retransmissão pode ser limitado por configuração para um específico valor sem limite de número (VECTOR, 2013). Essas configurações são feitas no controlador do CAN.

2.5.1 Configuração de sincronização e velocidade

Esta é uma parte crítica da rede, na qual uma configuração errada poderá comprometer todo o desempenho da rede ou até mesmo uma falha geral. A fim de atingir a velocidade de transmissão requerida, há uma configuração nos registradores do controlador CAN que devem ter seus valores respeitados e alterados de acordo com o meio de transmissão. Tanto o CAN como o CAN-FD possuem uma unidade de tempo chamada “*time quantum*” ou *tq*. Essa unidade é definida pelo selecionador como a taxa de transmissão do subsistema CAN, ou seja, é o tempo do *clock* do sistema CAN, que é derivado de um *clock* de microcontrolador.

Para calcular o *tq*, são necessários alguns passos, como mostrados nas equações 1 e 2.

$$\text{Tempo de 1 bit} = \frac{1}{\text{Taxa de transmissão}} \quad (1)$$

A equação 2 mostra como é o cálculo do tq . O BRP (*baud rate prescaler*) é uma incógnita que representa o valor interno da pré-escala do oscilador de acordo com a equação de (WILFRIED, 2005).

$$tq = \frac{[2(BRP+1)]}{F_{osc}} \quad (2)$$

na qual: F_{osc} é a frequência do oscilador.

Essa equação mostra que é necessário multiplicar o *prescaler* por 2, pois é devido à construção da arquitetura do controlador CAN.

Posteriormente, é possível saber o número de tq 's necessários no sistema CAN através da equação 2. Ao dividir o tempo de 1 *bit* pelo tempo de 1 tq , o resultado será o número total de tq o qual deve ser distribuído de acordo com as características dos segmentos do tempo de *bit* que serão ilustrados na Figura 12.

$$\text{Número de } tq = \frac{\text{tempo de 1 bit}}{\text{tempo de 1 } tq} \quad (3)$$

Sabe-se que, no CAN-FD, são definidos dois *bits* de taxa de transmissão, o primeiro para a fase de arbitração (*Arbitration-Phase*) com tempo de *bit* maior e o segundo para a fase de dados (*Data Phase*). No CAN-FD há a definição de dois valores de *prescaler*, um chamado de Tempo de *Bit* Nominal e outro Tempo de *Bit* de Dados. O Tempo de *Bit* de Dados pode ser dividido de acordo com a Figura 12.

Figura 12 - Segmento do tempo de *bit* no CAN e CAN-FD.

Fonte: Elaborado pelo autor.

De acordo com a Figura 12 há algumas divisões no *bit*. A divisão do Segmento de Sincronismo possui no máximo 1 *tq*. Essa parte do segmento é responsável por sincronizar vários nós no barramento, sincronizando a entrada na rede com o relógio no sistema. O segundo segmento é o de Propagação, que possui até 8 *tq*'s. Esse segmento é responsável por compensar todos os atrasos que o meio físico pode causar, como tempo de transmissão, propagação do sinal no barramento e tempo para recepção do dado. O terceiro segmento é da Fase 1, que mostrará onde será feita a aquisição do valor do sinal que está sendo lido no momento. Essa fase tem um campo de *tq* que varia entre 1 a 8. O último segmento é a Fase 2, que tem a mesma configuração da Fase 1, tem seu tempo de 1 a 8 *tq*'s.

O *Sample Point* é o ponto de amostra (em tempo) no qual o nível do barramento é lido e interpretado como o valor do respectivo *bit*. Está localizado no final do segmento da Fase 1. No sistema do CAN-FD, o Segmento de Sincronismo tem apenas 1 *tq*, enquanto o de Propagação possui um *range* de 1 a 32 *tq*'s. Os segmentos das Fases 1 e 2 também seguem a mesma configuração da Propagação, indo até 32 *tq*'s. A primeira parte do Frame do CAN-FD, até o *bit* BRS, é transmitida pelo Tempo de *Bit* Nominal. O *bit* é chaveado se o BRS for recessivo até que o delimitador do CRC seja alcançado ou o controlador do CAN-FD identifique alguma condição de erro em que haja a necessidade de iniciar o *Error Frame*. O *Error Frame*, assim como o ACK, *End of Frame*, *Overload Frame* e todos os *Frames* no CAN são transmitidos pelo Tempo de *Bit* Nominal (BOSH, 2012).

Esse protocolo também pode ser aplicado a qualquer área que necessite de comunicação instantânea por fios, como nas áreas de aviação, marítima, automação e robótica.

2.6 SISTEMAS DE TEMPO REAL

O sistema de tempo real tem sua interação com o ambiente, não só através da *interface* homem-máquina, mas também através de dispositivos que captam informações e atuam sobre elas, através de sensores e atuadores. Eles estão presentes desde sistemas mais simples, como o controle de rotação de um motor, até sistemas mais complexos, como o controle automático da rede eletrônica de uma aeronave (KOPETZ, 1997). Nesse tipo de sistema, o dado necessita ser enviado e recebido com o menor tempo sem atraso possível, sendo que qualquer atraso na ordem de milissegundos pode causar graves acidentes. O tempo real é exigido quando há a necessidade de segurança e preservação da vida em qualquer ambiente tecnológico.

Dentro do sistema de tempo real, existe um termo chamado de previsibilidade, que significa a capacidade de um sistema em executar ações desejadas em intervalos de tempo pré-estabelecidos, geralmente programados no próprio sistema computacional que está inserido (BURNS e WELLINGS, 2001). O gerenciamento eletrônico desse tipo de situação deve prever inúmeras situações para evitar atrasos imprevistos. Por exemplo, caso haja muitas mensagens, o sistema deve determinar aquela que tem maior prioridade de execução. (BURNS, 2002).

É importante ressaltar que, no gerenciamento de informações, deve-se inserir uma heurística que tome ações de acordo com a prioridade da informação. Por exemplo, uma mensagem com atraso num sistema de usina nuclear não é tolerável, a previsibilidade nesse tipo de situação deve ser muito próxima de 100%. Por outro lado, num sistema de conforto e conveniência automotiva, essa condição pode ser amenizada, sendo toleráveis tempos de atraso um pouco maiores, sem que se comprometa a satisfação do produto com o consumidor.

Para garantir a previsibilidade do sistema de tempo real, existem métodos que podem contribuir para tal tarefa, como utilizar linguagens de programação robusta com poucos atrasos nas funções que precisem de *delay* em seu comando, assim como prover redundância dos componentes do sistema, para aumentar a confiança no seu funcionamento.

2.6.1 Conceitos fundamentais de sistemas de tempo real

Aplicações que exijam tempo real, como um acionamento de freio de um veículo ou um botão que desliga instantaneamente uma máquina industrial de corte, precisam expressar seus requisitos em função do tempo. Isto significa que é imprescindível a existência de uma referência de tempo comum para todos os elementos envolvidos na execução das tarefas e, para isso, os relógios do sistema devem estar sincronizados entre si (sincronização interna) e deve haver uma referência externa (sincronização externa). Com relação à sincronização interna, ela é suficiente caso seja utilizada apenas para medição de duração das operações. Por outro lado, se for necessária uma medição de horas do dia, aí é importante que a sincronização externa esteja ativada (ZUBERI e SHIN, 1996).

2.6.2 Tarefas de tempo real e *deadline* crítico e suave

Para os sistemas que possuem muitas tarefas, é possível descrevê-las como:

- *Deadline* crítico - é a especificação temporal de uma tarefa que tem um intervalo de tempo que ocorre quase instantaneamente para ser executado. Nesse sistema, entram dispositivos que são cruciais para a vida, como gerenciamento de motores, gerenciamento e controle eletrônico do freio, da direção, dentre outros como pode ser visualizado na Figura 13 (HODEL, 2008).

Figura 13 - Curva de deadline crítico.

Fonte: Hodel (2008).

- *Deadline* Suave - é a especificação temporal de uma tarefa que tem um intervalo de tempo não instantâneo para ser executado mostrado na Figura 14 como, por exemplo, sistemas de multimídia, GPS, travas elétricas (HODEL, 2008).

Figura 14 - Curva de deadline suave.

Fonte: Hodel (2008).

2.6.3 Tipos de tarefa

Há duas abordagens conhecidas sobre como as tarefas são ativadas no sistema. A primeira é orientada a eventos ou *event-triggered*, que é quando a tarefa é executada e lida instantaneamente pelos módulos que estão na rede. A outra é a tarefa de tempo pré-determinado que acontece com um tempo programado no sistema.

2.7 CAMADAS ISO/OSI

O CAN e o CAN-FD baseiam-se nas camadas de rede ISO/OSI, que possui até 7 camadas diferentes, porém ambos usam apenas duas delas, camadas física e de enlace. Esse conceito de camadas foi criado pela ISO (*International Organization for Standardization*), sendo que inicialmente foi criado para desenvolver arquiteturas de *Framework*, ou seja ele foi criado para organizar uma estrutura para conteúdo e processo a fim de garantir consistência e completude de um pensamento. A ideia de camadas surgiu pelo trabalho de Charles Bachman, pesquisador da ISO.

As camadas existentes são camada física, de enlace, de rede, de transporte, de sessão, de apresentação e de aplicação (HODEL, 2008) que serão apresentadas a seguir.

2.7.1 Camada física

A camada física é responsável pela parte dos dispositivos com *interface* entre *hardware* e *software*. Ela possui características do material tais como comprimento do cabo, tensão de operação do dispositivo eletrônico, normas de temperatura, dentre outros (RUFINO, 1997).

2.7.2 Camada de enlace

Essa camada é a sequência da anterior. Ela trata as topologias de rede, interfaces, dentre outros, e é responsável por todo processo de chaveamento. Após receber os *bits*, ela os converte de maneira que o sistema entenda, transforma-os em

unidade de dado, subtrai o endereço físico e encaminha para a camada de rede que continua com o processo. Essa camada já pode entender o endereço físico (MAC Address – *Media Access Control* ou Controle de Acesso à Mídia). O MAC é um endereço único para cada dispositivo e é composto por 48 *bits*, divididos em duas etapas, conforme exemplo abaixo (WOZNIAK, 2012).

2.7.3 Camada de rede

Essa camada é responsável pelo roteamento dos dados na rede. Ela determina qual é o melhor caminho para os dados, fazendo as melhores rotas possíveis (ZIMMERMANN, 1980).

2.7.4 Camada de transporte

A camada de transporte é responsável por receber os dados enviados pela camada de sessão e segmentá-los para que sejam enviados à camada de rede, que por sua vez, transforma esses segmentos em pacotes. No receptor, a camada de transporte realiza o processo inverso, ou seja, recebe os pacotes da camada de rede e junta os segmentos para enviar à camada de sessão.

Isso inclui controle de fluxo, ordenação dos pacotes e a correção de erros, tipicamente enviando para o transmissor uma informação de recebimento, garantindo que as mensagens sejam entregues sem erros na sequência, sem perdas e duplicações.

2.7.5 Camada de sessão

Após a camada de transporte ter recebido os bits, obtido endereço, definido um caminho para o transporte dos dados, inicia-se a sessão responsável pelo processo da troca de dados/comunicação. Essa camada inicia, gerencia e termina a comunicação entre os hosts, que são os servidores que armazenam os dados. É importante destacar que essa camada deve sincronizar os dados entre os hosts para manter tudo funcionando adequadamente.

2.7.6 Camada de apresentação

A camada de apresentação converte o formato do dado recebido pela camada de aplicação em um formato comum a ser usado na transmissão desse dado, ou seja, um formato entendido pelo protocolo usado.

2.7.7 Camada de aplicação

É a camada que mais sofre interação, pois é nela em que os programas de mensagens instantâneas e *softwares* gerenciadores de *e-mails* são utilizados. É nela que basicamente ocorre a *interface* de recepção e transmissão de dados.

3 METODOLOGIA: CÁLCULO DE TEMPOS E RESPOSTAS DO FRAME CAN E CAN-FD

3.1 CONCEITOS INTRODUTÓRIOS DA METODOLOGIA

Este capítulo mostra os métodos que foram utilizados para o desenvolvimento de simulações de uma rede CAN e CAN-FD inseridos em sistemas de controle em malha fechada, com o objetivo de identificar os limites de previsibilidade dos dois protocolos. Este projeto compara o comportamento do *busload* (a quantidade de *bytes* enviados por segundo) nas redes CAN e CAN-FD. Foram desenvolvidos cálculos teóricos e testes a fim de determinar a porcentagem de dados que serão transmitidos no barramento e o tempo de envio dessas mensagens. Para isso, o estudo de caso usou a equação de Tindell (TINDELL;BURN; WELLINGS, 1995) que é uma equação criada em 1995 por esse pesquisador e vem sendo utilizada pelo meio acadêmico e industrial. Essa equação mostra o tamanho da mensagem em tempo na rede (C_m), que é calculado em segundos. Partindo dessa equação do sistema CAN e de testes simulando a rede CAN-FD, foi desenvolvida uma nova equação, que determina o *busload* no CAN-FD. A partir dessa equação, são levantados os dados do tamanho da mensagem em relação ao tempo de seu envio ao barramento da rede. Esses dados são inseridos em uma tabela, que compara com os respectivos dados da rede CAN.

Os dados de identificação dos módulos, quantidade de *bytes*, sinais, tempo de envio e mensagens foram adquiridos em um veículo Polo Volkswagen 2004, que serve como base de estudos para esta pesquisa. Os dados da rede CAN do automóvel foram comparados com os respectivos dados simulados em uma rede CAN-FD.

3.2 DESENVOLVIMENTO DO MODELO DE CÁLCULO DO *BUSLOAD*

Para simular dados na rede CAN, é essencial um estudo teórico para calcular o tamanho da mensagem no barramento. Para calcular esse tamanho da mensagem C_m do CAN, foi utilizada a equação de Tindell (TINDELL;BURN; WELLINGS, 1995),

que permite calcular o *busload* no barramento. O tamanho da mensagem em tempo na rede (C_m) é calculado em segundos.

$$C_m = \left[\frac{B_s + 8 * D_f}{5} + B_s + B_e + 8 * D_f \right] * \frac{1}{\tau_{\text{bit}}} \quad (4)$$

onde:

C_m é o tamanho da mensagem em tempo na rede (em segundos);

B_s é o número total de *bits* até o último *bit* de CRC, o qual contabiliza os números de *bit stuffing*;

B_e é *bits* a partir do CRC;

D_f é a quantidade de *bytes* no campo de dados;

τ_{bit} é a taxa de transmissão.

Esse cálculo precisa da quantidade de *bits* que estão sujeitos ao *bit stuffing* na rede (B_s). A divisão deve ser feita por 5, que é o número máximo de *bits* repetidos e sequenciais que o sistema permite. Desta forma, na pior hipótese, a cada 5 *bits* consecutivos e repetidos, o sistema ganhará um *bit* de ordem lógica invertida à sequência. Isso dará o pior tempo de *busload* da equação. Calculada a primeira parte da equação, é necessário introduzir os *bits* restantes após o campo CRC e somar com a quantidade de dados, que é ($8 * D_f$) e multiplica-se pela taxa de transmissão no barramento.

Simplificando-se a equação, o que se tem é a soma de todos os *bits* do *frame* do CAN somados com os possíveis *bits stuffing* do sistema.

No caso do CAN-FD, foi necessário desenvolver uma nova equação para calcular o *busload* na rede. A equação se baseou na equação original Tindell e nos testes que foram simulados e realizados em laboratório com o auxílio da ferramenta de diagnóstico CANoe (TINDELL; BURN; WELLINGS, 1995), que checa quanto tempo o barramento está ocupado pelo *frame* de transmissão por segundo. A equação 5 desenvolvida será descrita a seguir.

$$\beta = \frac{\tau}{\omega} \quad (5)$$

$$\tau = T_s + T_f \quad (6)$$

onde:

β : é o *Busload*

τ : tempo de ocupação da carga no barramento

ω : tempo de medição em segundos

T_s : Tempo dos *bits* mais lentos no *frame* CAN.

T_f : Tempo mais rápido dos *bits* no *frame* CAN FD.

$$T_s = \frac{(\text{SOF}+\text{ID}+\text{r1}+\text{IDE}+\text{EDL}+\text{r0}+\text{BRS}/2+\text{CRCdel}/2 * 1,2)+\text{ACK}+\text{DEL}+\text{EOF}+\text{IFS}}{t_x}, \quad (7)$$

na qual $\text{SOF}+\text{ID}+\text{r1}+\text{IDE}+\text{EDL}+\text{r0}+\text{BRS}/2+\text{CRCdel}/2 = 17 \text{ bits}$, 1,2 é o fator de pior caso de *bit stuffing*, na qual é igual a dividir por 5 e somar ao seu número de origem. O $\text{ACK}+\text{DEL}+\text{EOF}+\text{ITM} = 12 \text{ bits}$ sem *bit stuffing*. Toda essa equação deve ser dividida pela taxa de transmissão naquele *frame* t_x que, no caso deste projeto, é 500 Kbps.

Para a transmissão de *bits* mais rápidos e com capacidade de dados menores que 16 *bytes*, a seguinte equação se aplica:

$$T_f = \frac{[(D_f+\text{BRS}/2+\text{ESI}+\text{DLC}+\text{CRCdel}/2)*1,2]+\text{CRC}_{17}+5}{t_y} \quad (8)$$

onde $\text{BRS}/2 + \text{ESI} + \text{DLC} + \text{CRCdel}/2$ é 6 *bits*, $\text{CRC}_{17} + 5$ é CRC *bits* +*bit stuffing* até 16 *bytes* de dados. D_f é a quantidade de *bits* no campo de dados; D_f é a quantidade de *bits* no campo de dados;

Para a transmissão de *bits* mais rápidos, com capacidade de dados maiores que 16 *bytes*, a seguinte equação se aplica:

$$T_f = \frac{[(D_f + BRS/2 + ESI + DLC + CRC_{del}/2) * 1,2] + CRC_{21} + 6}{t_y}, \quad (9)$$

onde $CRC_{21} + 6$ é o CRC *bits* + o *bit stuffing* quando o campo de dados é maior que 16 *bytes*. Ele deve ser 21 devido ao fato do campo possuir um espaço maior de *bytes*, assim surge a necessidade de aumentar o campo que faz o cálculo de erro na transmissão dos dados. A taxa de transmissão dos dados na configuração rápida desta pesquisa t_y é de 4000000 *bits* por segundo.

Desta forma, a equação para dados menores que 16 *bytes* é:

$$\beta = \frac{(SOF + ID + r1 + IDE + EDL + r0 + BRS/2 + CRC_{del}/2 * 1,2) + ACK + DEL + EOF + IFS}{t_x} + \frac{[(D_f + BRS/2 + ESI + DLC + CRC_{del}/2) * 1,2] + CRC_{17} + 5}{t_y} \quad \omega$$

(10)

E para maior que 16 *bytes* :

$$\beta = \frac{(SOF + ID + r1 + IDE + EDL + r0 + BRS/2 + CRC_{del}/2 * 1,2) + ACK + DEL + EOF + IFS}{t_x} + \frac{[(D_f + BRS/2 + ESI + DLC + CRC_{del}/2) * 1,2] + CRC_{21} + 6}{t_y} \quad \omega$$

(11)

A Figura 15 mostra onde são enviados os *bits* com maior e menor velocidades. No caso da região em laranja, os *bits* são com velocidades menores de 500Kbps e na região em azul dentro da marcação vermelha são os *bits* com velocidade maior, no caso de 4Mbps. O máximo que pode ser configurado no CAN-FD é uma taxa de transmissão de aproximadamente 10Mbps, que neste estudo será limitado a 4Mbps para uma visualização e análise mais clara na obtenção dos resultados.

Figura 15 - Taxa de transmissão dos dados no CAN-FD.

Fonte: (VECTOR, 2012).

3.3 ESTUDO DE CASO 1

Este estudo explora o uso da rede CAN-FD em um ambiente automotivo, a fim de descobrir se essa rede conseguirá atender a todas as mensagens que exigem respostas muito rápidas de comunicação, sem atrasos na rede. Para testar a rede CAN, o estudo utilizou um veículo da marca Volkswagen, modelo Polo, ano 2004 2.0. Neste automóvel, foram feitas análises da rede CAN e a partir de 5 módulos do veículo e 19 *frames*, sendo cada módulo com a sua mensagem, dados como ID's, tempos de transmissão da mensagem, conteúdo da mensagem, dentre outros. Através desses parâmetros, foi desenvolvida uma simulação da rede CAN-FD. Essa simulação foi executada neste veículo, porém pode usar o conceito da simulação da rede CAN-FD em qualquer outra aplicação de transmissão de dados seja ela automotiva, aeronáutica, naval, médica, dentre outras.

O primeiro passo foi mapear o sistema automotivo de estudo com o *software* CANoe, que é uma ferramenta de diagnóstico da empresa Vector que lê todos os dados de qualquer dispositivo que possua esse protocolo de comunicação, com o objetivo de identificar as centrais eletrônicas de comunicação CAN. Para analisar as mensagens entre os módulos, foi feita uma engenharia reversa, desconectando os módulos de ECU, conforto, *Cluster* (Painel de Instrumentos), ar-condicionado (*climatronic*) e analisando *byte a byte* todo o barramento CAN.

A Tabela 4 foi construída com 5 mensagens adicionais identificadas como ID 610, 620, 630, 640 e 650, segundo a equação 4, para chegar em um *busload* de 30% pois é a média da taxa de *busload* utilizada pelos automóveis na atualidade, então foram estabelecidos alguns valores de *busload* a 30%, 50% e 70%. Na primeira coluna da tabela, estão localizados os 19 identificadores dos módulos (ID's), na segunda está a classe (ECU) que cada ID está alocado. Em seguida, DLC que mostra o tamanho da mensagem que será enviado no barramento; na quarta coluna há o tempo medido pela ferramenta CANoe e na quinta o tempo aproximado da transmissão da mensagem. Esse tempo foi arredondado para facilitar os cálculos do *busload*. A sexta coluna apresenta o tamanho da mensagem em tempo na rede (C_m), dado em segundos. A sétima coluna mostra quantas vezes a mensagem ocupa dentro de um tempo de 1000 ms e a última coluna mostra os valores de *busload* de cada mensagem. Para facilitar a leitura, foram definidas cores para cada conjunto de módulos de ECU.

Tabela 4 - Cálculo teórico de uma rede CAN a 30% de *busload* utilizando a equação de Tindell.

ID/ Frame	ECU's	DLC	Time Trigger (ms)	Período teórico das mensagens (ms)	Tamanho da mensagem em tempo na rede (Cm) em s	Quantas vezes cada mensagem entra dentro do espaço de 1000ms, que é o tempo maior de transmissão (pode ser calculado como mmc).	Busload
280 Motor		8	9,9993	10	0,0002532	100	0,0253
288 Motor		8	9,9993	10	0,0002532	100	0,0253
380 Motor		8	9,9993	10	0,0002532	100	0,0253
480 Motor		8	19,972	20	0,0002532	50	0,0127
488 Motor		8	9,9995	10	0,0002532	100	0,0253
588 Motor		8	19,975	20	0,0002532	50	0,0127
580 Motor		8	1009,6	1000	0,0002532	1	0,0003
320 Cluster		8	25,068	25	0,0002532	40	0,0101
420 Cluster		8	199,93	200	0,0002532	5	0,0013
520 Cluster		8	199,89	200	0,0002532	5	0,0013
5E0 AC		8	19,776	20	0,0002532	50	0,0127
388 Conforto		3	19,965	20	0,0001572	50	0,0079
38A Conforto		4	19,969	20	0,0001764	50	0,0088
470 Conforto		5	51,725	50	0,0001956	20	0,0039
5D0 Conforto		6	99,732	100	0,0002148	10	0,0021
5D8 Conforto		8	100,07	100	0,0002532	10	0,0025
570 Conforto		4	99,9	100	0,0001764	10	0,0018
50 N/A		2	20,9	20	0,000138	50	0,0069
3D0 N/A		4	20,7	20	0,0001764	50	0,0088
610 Módulo_teste1		8	9,9993	10	0,0002532	100	0,0253
620 Módulo_teste2		8	9,9993	10	0,0002532	100	0,0253
630 Módulo_teste3		8	9,9993	10	0,0002532	100	0,0253
640 Módulo_teste4		8	9,9993	10	0,0002532	100	0,0253
650 Módulo_teste5		8	9,9993	10	0,0002532	100	0,0253
Busload=							29,62%

Fonte: Elaborado pelo autor.

A Tabela 5 segue a mesma ideia da Tabela 4, porém com 12 mensagens adicionais com o propósito em ter-se um *busload* de 50%. A tabela 5 necessitou 20 mensagens adicionais para alcançar a carga de 70%. Em todas as tabelas, foram inseridas mensagens adicionais com o ID de teste e o menor tempo possível em todas as mensagens de 9,9993 ms, que foi uma condição de sistemas prioritários (aqueles que exigem tempo menor por serem mais importantes na execução de tarefas).

Tabela 5 - Cálculo de uma rede com busload a 50%

ID/ Frame	ECU's	DLC	Time Trigger (ms)	Período teórico das mensagen s	Tamanho da mensagem em tempo na rede (Cm) em s	Quantas vezes cada mensagem entra dentro do espaço de 1000ms, que é o tempo maior de transmissão (pode ser calculado como mmc).	Busload
280 Motor		8	9,9993	10	0,0002532	100	0,02532
288 Motor		8	9,9993	10	0,0002532	100	0,02532
380 Motor		8	9,9993	10	0,0002532	100	0,02532
480 Motor		8	19,972	20	0,0002532	50	0,01266
488 Motor		8	9,9995	10	0,0002532	100	0,02532
588 Motor		8	19,975	20	0,0002532	50	0,01266
580 Motor		8	1009,6	1000	0,0002532	1	0,00025
320 Cluster		8	25,068	25	0,0002532	40	0,01013
420 Cluster		8	199,93	200	0,0002532	5	0,00127
520 Cluster		8	199,89	200	0,0002532	5	0,00127
5E0 AC		8	19,776	20	0,0002532	50	0,01266
388 Conforto		3	19,965	20	0,0001572	50	0,00786
38A Conforto		4	19,969	20	0,0001764	50	0,00882
470 Conforto		5	51,725	50	0,0001956	20	0,00391
5D0 Conforto		6	99,732	100	0,0002148	10	0,00215
5D8 Conforto		8	100,07	100	0,0002532	10	0,00253
570 Conforto		4	99,9	100	0,0001764	10	0,00176
50 N/A		2	20,9	20	0,000138	50	0,0069
3D0 N/A		4	20,7	20	0,0001764	50	0,00882
610 Módulo_Testes1		8	9,9993	10	0,0002532	100	0,02532
620 Módulo_Testes2		8	9,9993	10	0,0002532	100	0,02532
630 Módulo_Testes3		8	9,9993	10	0,0002532	100	0,02532
640 Módulo_Testes4		8	9,9993	10	0,0002532	100	0,02532
650 Módulo_Testes5		8	9,9993	10	0,0002532	100	0,02532
660 Módulo_Testes6		8	9,9993	10	0,0002532	100	0,02532
670 Módulo_Testes7		8	9,9993	10	0,0002532	100	0,02532
680 Módulo_Testes8		8	9,9993	10	0,0002532	100	0,02532
690 Módulo_Testes9		8	9,9993	10	0,0002532	100	0,02532
700 Módulo_Testes10		8	9,9993	10	0,0002532	100	0,02532
710 Módulo_Testes11		8	9,9993	10	0,0002532	100	0,02532
720 Módulo_Testes12		8	9,9993	10	0,0002532	100	0,02532
Busload=							49,88%

Fonte: Elaborado pelo autor.

4 ESTUDO DE CASO 1 E RESULTADOS

4.1 MENSAGENS IDENTIFICADAS NO BARRAMENTO CAN

Este capítulo apresenta o procedimento para obter os resultados da pesquisa, o desenvolvimento do estudo de caso do CAN-FD e o modelamento dos dados na rede. Aqui serão discutidos os dados de todas as mensagens identificadas no veículo, o que elas representam, a finalidade e como serão utilizadas na simulação do CAN-FD.

A Tabela 7 mostra que há 7 colunas de dados obtidos como ID (identificador do *frame*), ECU (qual o módulo que está enviando a mensagem), DLC (tamanho do dado no *frame*), *time trigger* (intervalo de transmissão entre *frames* em ms), período teórico da mensagem, tamanho da mensagem em segundos e quantas vezes a mensagem entra no espaço de 1000 ms. Na quinta coluna, há o período teórico das mensagens em milissegundos. Esse período teórico, para esse caso específico, é um valor único, pois cada montadora possui um valor pré-determinado. Através de experimentos anteriores (HODEL, 2008), é recomendável aproximar os valores do *time trigger* para números inteiros, o que facilita consideravelmente os cálculos. A segunda coluna da tabela mostra o ID de cada um dos 19 módulos do veículo. O tamanho da mensagem foi calculado seguindo a equação 4 de Tindell, explicada anteriormente. A sexta coluna mostra os valores do tamanho da mensagem de cada módulo. A sétima coluna mostra os valores que foram obtidos através do MMC (mínimo múltiplo comum) da quinta coluna, porque é o menor período teórico em comum das mensagens. No cálculo do MMC, o resultado foi 1000. A próxima coluna multiplica o valor de cada mensagem pelo tamanho que cada uma coube em 1000 ms. A última coluna é apenas o valor de cada mensagem em ms. É possível observar que o cálculo do *busload* levou a um valor de 19,49%. Esse valor mostra que o resultado está muito próximo do resultado da equação de Tindell, que foi calculado em 18,50%. Essa diferença de aproximadamente 1% é devido a aproximações que foram feitas nos tempos de transmissões das mensagens nas tabelas acima, a fim de facilitar o cálculo do *busload*.

Tabela 7- Tabela de dados da rede CAN.

ID	DLC	Time Trigger (ms)	Período teórico das mensagens (ms)	Tamanho da mensagem em tempo na rede (Cm) em s	Quantas vezes cada mensagem entra dentro do espaço de 1000ms, que o tempo maior de transmissão (pode ser calculado como mmc).	
280 Motor	8	9,9993	10	0,0002532	100	0,02532
288 Motor	8	9,9993	10	0,0002532	100	0,02532
380 Motor	8	9,9993	10	0,0002532	100	0,02532
480 Motor	8	19,972	20	0,0002532	50	0,01266
488 Motor	8	9,9995	10	0,0002532	100	0,02532
588 Motor	8	19,975	20	0,0002532	50	0,01266
580 Motor	8	1009,645	1000	0,0002532	1	0,00025
320 Cluster	8	25,068	25	0,0002532	40	0,01013
420 Cluster	8	199,93	200	0,0002532	5	0,00127
520 Cluster	8	199,891	200	0,0002532	5	0,00127
5E0 AC	8	19,776	20	0,0002532	50	0,01266
388 Conforto	3	19,965	20	0,0001572	50	0,00786
38A Conforto	4	19,969	20	0,0001764	50	0,00882
470 Conforto	5	51,725	50	0,0001956	20	0,00391
5D0 Conforto	6	99,732	100	0,0002148	10	0,00215
5D8 Conforto	8	100,07	100	0,0002532	10	0,00253
570 Conforto	4	99,9	100	0,0001764	10	0,00176
50 N/A	2	20,9	20	0,000138	50	0,0069
3D0 N/A	4	20,7	20	0,0001764	50	0,00882
					Busload	19,49%

Fonte: Elaborado pelo autor.

Conforme pode ser visto nos resultados anteriores, existem 5 módulos CAN no veículo de ano 2004 com 19 *frames* e a tendência é sempre aumentar o número desses dispositivos nos automóveis. Consequentemente, para a rede de comunicação conseguir suportar tantas mensagens com pequenos atrasos é necessário uma rede mais robusta com maior capacidade de transmissão de dados e pouco atraso. Deste modo, a Tabela 8 foi desenvolvida através da equação 11, construída aqui para calcular o *busload* da rede CAN-FD. Os cálculos foram executados de acordo com as configurações do CAN-FD, utilizando os mesmos parâmetros de comparação do CAN da Tabela 6 (mesmos módulos, mesmos dados, mesmo tempo, mesma taxa de transmissão) a fim de mostrar a diminuição do *busload*. A tabela mostra que há 5 ID's diferentes do CAN-FD. O primeiro ID do motor totaliza 64 *Bytes*, transmitidos em 10 ms de período teórico. Há outras 3 mensagens do *cluster* (Painel de Instrumentos) com 24 *Bytes* transmitindo a 25 ms. A tabela possui também uma mensagem de ar-condicionado de 8 *Bytes* transmitida em 20 ms e, para finalizar, mais 6 módulos de conforto no tamanho de 30 *Bytes* transmitidos em 20 ms. Há ainda dois módulos não identificados (N/A) 50 e 3D0 sendo enviados pelo ID 50, ambos com transmissão a 20 ms. O *busload* calculado nos mesmos parâmetros da rede CAN foi de 4,43%, é quase 5 vezes menor que o antecessor, que foi de 19,49%. Isso mostra a capacidade do

desempenho do CAN-FD, uma vez que ele conseguiu enviar os dados em menor tempo e menor quantidade de mensagens. A partir desse resultado, a pesquisa buscou o limite do CAN-FD para identificar se há problemas de atrasos das mensagens em condições severas, como o acréscimo de mais motores no barramento.

Tabela 8 - Dados da rede CAN-FD.

ID	Módulos	Tempo de bit mais lento	Tempo de bit mais rápido	Tempo de carga do barramento	Tempo da mensagem (s)	Busload	ATRASOS/ Pior caso (ms)	Quantidade de Bytes/ segundo
280	Motor	0,0000648	0,00016215	0,00022695	0,01	0,022695	0,082	6400
320	Cluster	0,0000648	0,00006615	0,00013095	0,025	0,005238	0,000	960
5E0	A/C	0,0000648	0,0000285	0,0000913	0,02	0,004585	0,035	400
388	Conforto	0,0000648	0,00008535	0,00015015	0,02	0,0075075	0	1600
50	N/A	0,0000648	0,0000217	0,0000865	0,02	0,004325	0	300
BUSLOAD=						4,43%	TOTAL DE BYTES=	9660

Fonte: Elaborado pelo autor.

A Figura 16 apresenta um esquema da simulação do *busload* da rede CAN em comparação com a rede CAN-FD. A linha roxa do gráfico mostra a quantidade de mensagens em % no barramento CAN, enquanto a linha dourada mostra a porcentagem de mensagens transmitidas no barramento CAN-FD. Ao analisar os gráficos, é possível observar que o CAN está em torno dos 18,5% da rede e o CAN-FD está em torno dos 4,1%, valores muito próximos dos cálculos teóricos apresentados no capítulo anterior. Os valores não são exatos, uma vez que os cálculos teóricos utilizam a pior hipótese de existir *bit stuffing*, escalonamentos e arredondamentos. Os resultados mostram que o CAN-FD está com o *busload* quase 5 vezes menor que o do CAN. Esse é um resultado bastante satisfatório, e será testado posteriormente para checar até aonde o CAN-FD consegue ter um bom desempenho quando comparado com o CAN.

Figura 16 - *Busload* em % (eixo y) versus tempo em segundos (eixo x) comparando CAN e CAN-FD.

Fonte: Elaborado pelo autor.

A Figura 17 apresenta o painel de simulação da empresa Vector. É possível ver os gráficos de uma rede CAN identificando todos os *Bytes* sendo transmitidos em tempo real, além de ser possível desenvolver via programação CAPL (um tipo de programação similar ao C). A aplicação deve ser programada em CAPL no próprio editor de código do CANoe, que possui compilador interno para tais tarefas. Por essa *interface*, é possível ver os gráficos, os dados, a estatística e as mensagens sendo simuladas.

Figura 17 - Painel de configuração da simulação do CANoe da Vector

Fonte: Elaborado pelo autor.

A lógica do desenvolvimento é que todos os dados, após serem recebidos pelo automóvel Polo, precisam ser reaproveitados para a simulação, então um dos blocos possui a função de *replay block*, que é exatamente utilizar um *loop* para repetição de dados laboratoriais.

Neste desenvolvimento, foi identificada uma dificuldade que é a quantidade de *bytes* a ser enviada. É impossível enviar mensagens de 56 *bytes* de forma única, uma vez que há um limite de *bytes*, de acordo com a tabela 8. Nesse caso, por exemplo, poderia ser enviada em duas etapas, dividindo a mensagem em várias sequências. O ideal é dividir em menor quantidade possível, pois assim podem ser transmitidos mais dados em uma única mensagem. Para a transformação, foi utilizado o canal CAN1 para o CAN e o canal CAN2 para o CAN-FD. O bloco de repetição faz um *loop* das mensagens originais do CAN1. O programa CAPL lê todas as mensagens de CAN1, muda o formato para CAN-FD, enviando-os para o canal CAN2. A taxa de transmissão é 500 Kbps, enquanto que para o CAN-FD durante a transmissão de dados é 4 Mbps. O apêndice A apresenta o código fonte do *software* desenvolvido.

A Tabela 9 apresenta o campo das mensagens que foram convertidas da rede CAN para a rede CAN-FD. É possível observar que para todos os ID's CAN do

gerenciamento de motor (*ECU*) foram divididos em 5 mensagens do CAN-FD. Foram organizados os dados de mensagens CAN através da aquisição dos ID's que encontra-se no campo de arbitração da seguinte maneira: 280, 288, 380, 480, 488, 588 e 580 no *ID* 280 do CAN-FD, que são mensagens do motor do veículo. Nesta pesquisa foi definido a escolha do *ID* mais prioritário em todas as mensagens pois é o que tem o menor tempo na transmissão das mensagens. As mensagens do Cluster no CAN 320, 420 e 520 foram enviadas pelo ID mais prioritário, ID 320 do CAN-FD, a mensagem do ar-condicionado 5E0 foi enviada individualmente no barramento. Já as mensagens de conforto 388, 38A, 470, 5D0, 5D8 e 570 são representadas através do ID 388 no barramento do CAN-FD. Finalmente, as mensagens de módulos não identificados 50 e 3D0 foram representadas pelo ID 50.

Tabela 9 - Mensagens convertidas de CAN para CAN-FD

ID do Formato CAN	ID do Formato CAN FD
280,288,380,480,488,580,588	280
320,420,520	320
5E0	5E0
388,38A,470,5D0,5D8,570	388
50,3D0	50

Fonte: Elaborado pelo autor.

Para combinar todos os dados das 7 mensagens do motor para uma só no CAN-FD, o critério da Tabela 10 deve ser respeitado. Essa tabela mostra o máximo DLC possível associado com o máximo número de *bytes* tanto para o CAN quanto para o CAN-FD. O *software* é mostrado no Apêndice B.

Tabela 10 - Número de *bytes* em CAN e CAN-FD de acordo com o DLC.

DLC	CAN BYTES	CAN-FD BYTES
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	8	12
10	8	16
11	8	20
12	8	24
13	8	32
14	8	48
15	8	64

Fonte: Elaborado pelo autor.

4.2 ANÁLISE TEMPORAL DE UMA REDE CAN A 30%, 50% E 70% DE *BUSLOAD*

A Figura 18 mostra as mensagens adicionais que foram desenvolvidas com os *ID's* 610, 620, 630 e 640 à rede CAN, com o objetivo de chegar a uma carga de 30% no veículo e, posteriormente, comparar com outras cargas, a 50% e 70% e identificar possíveis problemas na transmissão de dados.

Figura 18 - Rede CAN com *busload* a 30%.

	Message Parameters				Triggering			Data Field							
	Identifier	Channel	DLC	Send		Cycle Time [ms]		0	1	2	3	4	5	6	7
	610	CAN 1	8	now	☒	10		0	20	0	0	0	0	0	0
	620	CAN 1	8	now	☒	10		0	0	30	0	0	0	0	0
	630	CAN 1	8	now	☒	10		0	0	0	40	0	0	0	0
>	640	CAN 1	8	now	☒	10		0	0	0	0	50	0	0	0

Fonte: Elaborado pelo autor.

A Figura 19 mostra o *busload* a 31,08%. Esses resultados mostram como os dados dos cálculos teóricos, que foram obtidos a partir da equação de Tindell, estão muito próximos com os valores obtidos em laboratório.

Figura 19 – Dados e *busload* a 30% (eixo x) versus tempo em segundos (eixo y).

Fonte: Elaborado pelo autor.

A Figura 20 mostra 12 mensagens adicionais para alcançar o *busload* de 50%.

Figura 20 - Mensagens adicionais para *busload* a 50%.

Message Name	Msg Params		Triggering		Data Field							
	Channel	DLC	Send	Cycle Time [ms]	0	1	2	3	4	5	6	7
610	CAN 1	8	now	10	0	20	0	0	0	0	0	0
620	CAN 1	8	now	10	0	0	30	0	0	0	0	0
630	CAN 1	8	now	10	0	0	0	40	0	0	0	0
640	CAN 1	8	now	10	0	0	0	0	50	0	0	0
650	CAN 1	8	now	10	0	0	0	0	0	60	0	0
660	CAN 1	8	now	10	0	0	0	0	0	0	70	0
670	CAN 1	8	now	10	0	0	0	0	0	0	0	80
680	CAN 1	8	now	10	10	0	0	0	0	0	0	0
690	CAN 1	8	now	10	0	20	0	0	0	0	0	0
700	CAN 1	8	now	10	0	0	30	0	0	0	0	0
710	CAN 1	8	now	10	0	0	0	40	0	0	0	0
720	CAN 1	8	now	10	0	0	0	0	50	0	0	0

Fonte: Elaborado pelo autor.

A Figura 21 mostra o *busload* de 48,70%, valor próximo do cálculo teórico apresentado no capítulo de Metodologia que foi apresentado com um valor de 49,88% de *busload*.

Figura 21 - *Busload* a 50% (eixo x) versus tempo em segundos (eixo y).

Fonte: Elaborado pelo autor.

A Figura 22 mostra parte das 20 mensagens adicionais para se obter um *busload* de 70%

Figura 22 - Mensagens adicionais para obter um *busload* de 70%.

	Message Parameters				Triggering			Data Field								
	Identifier	Channel	DLC	Send		Cycle Time [ms]		0	1	2	3	4	5	6	7	
	690	CAN 1 ▾	8 ▾	now	☒	10		0	20	0	0	0	0	0	0	0
	700	CAN 1 ▾	8 ▾	now	☒	10		0	0	30	0	0	0	0	0	0
	710	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	40	0	0	0	0	0
	720	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	50	0	0	0	0
	730	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
	740	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
	750	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
>	760	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
	770	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
	780	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
	790	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0
	795	CAN 1 ▾	8 ▾	now	☒	10		0	0	0	0	0	0	0	0	0

Fonte: Elaborado pelo autor.

A Figura 23 mostra que o *busload* deu 68,62% quando foram inseridas 20 mensagens adicionais, prevendo um tráfego de mensagens em torno de 70%, que está muito próximo do resultado obtido teoricamente.

Figura 23 - *Busload* da rede CAN para 70% (eixo x) versus tempo em segundos (eixo y).

Fonte: Elaborado pelo autor.

Todos os resultados acima foram valores próximos à teoria. E para complementar o estudo foi desenvolvido um estudo de caso 2 na qual foi inserida uma

rede de CAN-FD numa malha de controle fechada. O objetivo é de verificar a capacidade de envio de dados na rede analisando se há atraso de transmissão das mensagens. Serão testadas 4 situações de *busload* diferentes, em que será verificado o limite da rede em enviar dados sem comprometer o tempo de transmissão. Foram criadas 3 tabelas com os tempos de mensagens de transmissão e atrasos de todos os ID's seguindo as condições de configuração do busload (30%, 50% e 70%). As tabelas 11,12 e 13 mostram os tempos que as mensagens levam para serem transmitidas do emissor ao receptor. O tempo de mensagem é considerado de acordo com o período teórico, o valor real e a diferença entre eles. As tabelas mostram esses tempos de acordo com o identificador do módulo, o *ID*.

Tabela 11 - Tempo das mensagens com CAN a 30% de *busload*.

ID's	Nº Bytes de dados	Período teórico das mensagens (ms)	Período real das mensagens (ms)	Tempo Teórico - tempo real
280	8	10	9,9993	0,0007
288	8	10	9,9993	0,0007
380	8	10	9,9993	0,0007
480	8	20	19,972	0,028
488	8	10	9,9995	0,0005
588	8	20	19,98	0,02
580	8	1000	1009,771	-9,771
320	8	25	25,292	-0,292
420	8	200	200,31	-0,31
520	8	200	200,31	-0,31
5E0	8	20	20,445	-0,445
388	3	20	19,965	0,035
38A	4	20	19,969	0,031
470	5	50	51,849	-1,849
500	6	100	100,062	-0,062
508	8	100	100,274	-0,274
570	4	100	99,9	0,1
50	2	20	20,9	-0,9
300	4	20	20,7	-0,7
610	8	10	10,223	-0,223
620	8	10	9,9993	0,0007
630	8	10	9,9993	0,0007
640	8	10	9,9993	0,0007
650	8	10	9,9993	0,0007

Fonte: Elaborado pelo autor

Tabela 12 - Tempo das mensagens com CAN a 50% de *busload*.

ID's	Nº Bytes de dados	Período teórico das mensagens (ms)	Período real das mensagens (ms)	Tempo Teórico - tempo real
280	8	10	10,044	-0,044
288	8	10	10,046	-0,046
380	8	10	10,045	-0,045
480	8	20	20	0
488	8	10	9,9995	0,0005
588	8	20	20,037	-0,037
580	8	1000	1009,645	-9,645
320	8	25	25,068	-0,068
420	8	200	199,93	0,07
520	8	200	199,891	0,109
5E0	8	20	19,776	0,224
388	3	20	19,965	0,035
38A	4	20	19,969	0,031
470	5	50	52,203	-2,203
5D0	6	100	101,09	-1,09
5D8	8	100	101,343	-1,343
570	4	100	99,9	0,1
50	2	20	20,9	-0,9
3D0	4	20	20,7	-0,7
610	8	10	9,9993	0,0007
620	8	10	10,788	-0,788
630	8	10	10,722	-0,722
640	8	10	10,788	-0,788
650	8	10	10,722	-0,722
660	8	10	11,222	-1,222
670	8	10	9,9993	0,0007
680	8	10	9,9993	0,0007
690	8	10	9,9993	0,0007
700	8	10	10,788	-0,788
710	8	10	10,788	-0,788
720	8	10	11,222	-1,222

Fonte: Elaborado pelo autor.

Tabela 13 - Tempo das mensagens com CAN a 70% de busload.

ID's	Nº Bytes de dados	Período teórico das mensagens (ms)	Período real das mensagens	Tempo Teórico - tempo real
280	8	10	10,101	-0,101
288	8	10	10,099	-0,099
380	8	10	10,099	-0,099
480	8	20	20,069	-0,069
488	8	10	10,103	-0,103
588	8	20	19,975	0,025
580	8	1000	1009,645	-9,645
320	8	25	25,77	-0,77
420	8	200	200,032	-0,032
520	8	200	199,891	0,109
5E0	8	20	22,975	-2,975
388	3	20	19,965	0,035
38A	4	20	19,969	0,031
470	5	50	52,633	-2,633
5D0	6	100	99,967	0,033
5D8	8	100	100,07	-0,07
570	4	100	100,026	-0,026
50	2	20	20,9	-0,9
3D0	4	20	20,7	-0,7
610	8	10	9,9993	0,0007
620	8	10	12,538	-2,538
630	8	10	9,9993	0,0007
640	8	10	9,9993	0,0007
650	8	10	12,542	-2,542
660	8	10	10,603	-0,603
670	8	10	11,102	-1,102
680	8	10	12,04	-2,04
690	8	10	10,355	-0,355
700	8	10	10,852	-0,852
710	8	10	9,9993	0,0007
720	8	10	10,853	-0,853
730	8	10	11,105	-1,105
740	8	10	10,35	-0,35
750	8	10	10,852	-0,852
760	8	10	9,9993	0,0007
770	8	10	9,9993	0,0007
780	8	10	11,102	-1,102
790	8	10	10,987	-0,987
795	8	10	11,105	-1,105

Fonte: Elaborado pelo autor.

4.3 CONVERSAO DE MENSAGENS CAN PARA CAN-FD EM CAPL

Esta seção aborda como foi desenvolvida a conversão dos dados da rede CAN em CAN-FD, de acordo com algumas estratégias. A primeira estratégia seria desenvolver um *software* que separa as mensagens em três partes de acordo com seus tamanhos, conforme mostrado na Figura 24. Durante o estudo, foram executados diversos testes em bancada, referentes a essas estratégias, a fim de identificar se haveria variação nos atrasos das mensagens de acordo com cada metodologia adotada. Nesta figura, por exemplo, para escrever uma mensagem de 56 *bytes* foram usados três tipos de mensagens, uma com 32 *bytes*, uma com 16 *bytes* e outra com 8 *bytes*. As 19 mensagens da rede CAN no “Trace 1” foram otimizadas em 10 mensagens no formato do CAN-FD, que podem ser visualizadas no “Trace 2”. O Apêndice B mostra o *software* de envio dos dados.

Figura 24 - Conversão de mensagens em linguagem de programação CAPL.

Fonte: Elaborado pelo autor.

A Figura 25 mostra o gráfico da primeira estratégia citada acima, onde é possível observar que o valor do *busload* no barramento do CAN-FD (em linhas douradas) ficou na média de 3,8%. Esse valor representa um número abaixo da estratégia adotada, porém não é satisfatório para esta pesquisa pois, apesar de ter um *busload* baixo, há mais mensagens sendo transmitidas no barramento e o desejado neste estudo é o menor número de mensagens com a maior quantidade de dados. O mesmo gráfico mostra, em linhas roxas, o *busload* da rede CAN em torno de 18,5%. Portanto, essa figura mostra que o CAN-FD pode reduzir até 5 vezes do valor original das mensagens em rede CAN.

Figura 25 - Gráfico em % do busload (eixo y) e tempo em segundos (eixo x) da rede CAN (linha roxa) e CAN-FD (linha dourada) de acordo com a primeira estratégia.

Fonte: Elaborado pelo autor.

A segunda estratégia seria escrever mensagens inteiras sem dividi-las em mais partes, o que pode ser verificado na Figura 26. A média do *busload* ficou em torno de 4,1%, valor 0,3% maior que a primeira estratégia. Então, é possível concluir que, independentemente da estratégia adotada, o *busload* é praticamente o mesmo, porém a melhor estratégia é aquela na qual o barramento receba o menor número possível de mensagens com uma capacidade maior de dados. A mínima variação só ocorre por conta de serem adicionados alguns *bytes* “00” na estratégia e a condição de pior caso para o *bit stuffing*. Por outro lado, esses *bytes*, que podem ser chamados de *bytes* extras, não mudam o *busload*.

Como conclusão, este projeto adotou a 2ª estratégia de não quebrar as mensagens, pois assim a simulação conta com o menor número possível de mensagens sendo transmitidas. E quanto menor o número de mensagens melhor o sistema de comunicação, pois haverá menos atraso no envio dos dados como poderá ser visto nas simulações realizadas na próxima seção.

Figura 26 - Gráfico em % do busload (eixo y) e tempo em segundos (eixo x) da rede CAN (linha roxa) e CAN-FD (linha dourada) de acordo com a segunda estratégia.

Fonte: Elaborado pelo autor.

4.4 ANÁLISE GRÁFICA TEMPORAL DE MENSAGENS NA REDE

A Figura 27 mostra a análise da quantidade de *bytes* transmitidos e o tempo de atraso dos dados (em ms) na rede CAN em modo normal de operação e quando a mesma sofreu um acréscimo no *busload* com 3 cargas distintas. Em modo normal, a primeira mensagem da rede, com 30%, teve em média 5 ms de atraso, a segunda mensagem, com 50%, teve aproximadamente 13 ms de atraso nas mensagens. A mais afetada foi a de 70% do *busload*, que teve mais de 20 ms de atraso.

Figura 27 - Mensagens atrasadas em milissegundos versus *bytes* versus *busload* do CAN de acordo com *busload*.

Fonte: Elaborado pelo autor.

A Figura 28 mostra a análise da quantidade de *bytes* transmitidos e o tempo de atraso dos dados (em ms) na rede CAN-FD em modo normal de operação e quando a mesma sofreu um acréscimo no *busload* com 3 cargas distintas. É possível identificar que, numa condição normal, uma rede CAN que antes possui 19,49% de *busload*, convertendo-a para CAN-FD obteve uma rede de 4,43% de *busload*, além de atraso menor que 0,2 ms com um envio de dados de 9660 *bytes*. Enquanto na rede CAN foram inseridas mensagens a fim de alcançar um *busload* de 29,62%, na rede do CAN-FD o *busload* equivaleria a 6,11%, um valor quase 5 vezes menor e um tempo de atraso muito pequeno, algo em torno de 0,3 ms. Para a rede CAN com 49,88% de *busload*, obteve-se um CAN-FD com *busload* de 8,18% com atraso de 0,5 ms e finalmente para CAN com 70,13% de *busload*, que foi o pior caso devido ao grande atraso de até 25 ms, obteve-se um *busload* no CAN-FD de 10,44% e um tempo de atraso quase nulo de 0,778 ms. Isso mostra que o CAN tem muito problema de atraso quando existe uma taxa alta de tráfego de dados e com a tecnologia atual automotiva,

na qual os veículos já são desenvolvidos com muitos módulos de segurança, conforto e conveniência, torna-se crucial um sistema de comunicação que tenha atrasos insignificantes como é o caso da rede de comunicação CAN-FD.

Figura 28 - Gráfico do *busload* na rede CAN-FD.

Fonte: Elaborado pelo autor.

Através dos dois gráficos anteriores, foi desenvolvido um gráfico comparativo dos dados reais da rede CAN com os dados simulados da rede CAN-FD conforme pode ser visualizado na Figura 29.

Figura 29 - Comparação da rede real CAN com a rede simulada CAN-FD.

Fonte: Elaborado pelo autor.

5 ESTUDO DE CASO 2

O segundo estudo de caso se refere a um sistema de malha fechada. O objetivo de inserir uma malha fechada é remover as incertezas que um sinal pode gerar no sistema, fazendo com que esse sistema se torne mais robusto e imune a perturbações e ruídos que possam aparecer na rede.

É esperado que ocorram problemas nos sinais das mensagens na rede CAN, pois quanto maior o *busload* maior a probabilidade de atrasos, conforme diversos experimentos (HODEL, 2008). Por outro lado, o objetivo desse estudo de caso é verificar o comportamento da rede CAN-FD, a fim de observar se existirão atrasos na rede quando for inserido motor de corrente contínua e controlador PID na rede. A comparação entre a malha fechada do CAN e do CAN-FD é feita através de medições nos sinais de saída dos sensores acoplados à malha de controle. Para desenvolver essa malha, será necessário um sensor, um atuador e um motor, todos conectados ao controlador através da rede. O barramento CAN é o mesmo utilizado no primeiro estudo de caso, considerando a mesma configuração de cada um dos testes. A fim de entender o objetivo da malha de controle fechado ser inserida no barramento, a Figura 30 ilustra o esquema de ligação dessa malha. Nela observa-se, por exemplo, que o controlador envia as informações de acionamento na rede CAN e na rede CAN-FD, que reenvia os dados para um atuador. Quando o atuador é acionado, ele transmite movimento ao motor de corrente contínua. O sensor recebe as informações do motor e as envia na rede novamente para o controlador fazer os ajustes necessários no sinal ficando em um ciclo de malha fechada. Esse controle foi necessário, pois um sistema de malha fechada elimina as perturbações que possam surgir na rede. Portanto, considerando os atrasos das mensagens, de acordo com cada configuração do *busload*, será possível identificar se a rede de comunicação conseguirá enviar todas as mensagens sem atrasos críticos.

Figura 30 - Malha de controle fechada de um controlador PID.

Fonte: Elaborado pelo autor.

5.1 DEFINIÇÕES DO EXPERIMENTO

Com o objetivo de testar o limite da rede CAN-FD, foi considerado um motor CC (corrente contínua) no barramento de comunicação. Isto foi motivado pelo fato de todos os dispositivos automotivos possuírem motores desse tipo (motores de tensão 12 V e 24 V para veículos pesados), sensor, atuador e controlador. Posteriormente, serão definidas as equações que descrevem o comportamento dinâmico do motor, de acordo com o modelo apresentado na Figura 31.

Figura 31 – Modelo de motor CC.

Fonte: (TIPSUWAN, CHOW, VANIJJIRATTKHN, 2003).

No controle em malha fechada a saída é utilizada em um ciclo realimentado, no qual tem a participação do controlador que determina a correção do sinal. Isto é feito a partir de uma realimentação da saída para a entrada, formando um ciclo, conforme a figura 31. Esse experimento é rico em informações, pois, caso a rede sofra atrasos devido ao controle, a rede CAN-FD poderia não ser a melhor escolha para mensagens que exijam tempos instantâneos.

As equações que descrevem o modelo dinâmico do Motor CC são:

$$e_a = e_b + L_a \frac{di_a}{dt} + R_a i_a \quad (12)$$

$$e_b = k_1 \frac{d\theta}{dt} \quad (13)$$

onde E_a , R_a , L_a e i_a são respectivamente a tensão de entrada, a resistência, a indutância e a corrente da armadura do motor. K_1 é a constante de força contra eletromotriz e θ é a posição angular do motor

$$T = J \frac{d^2\theta}{dt^2} + F \frac{d\theta}{dt} \quad (14)$$

$$T = k_2 i_a \quad (15)$$

onde T é o torque do motor, J é o momento de inércia, F é constante de atrito viscoso e k_2 é a constante de torque do motor.

A fim de facilitar os cálculos matemáticos das equações e para poder montar o modelo matemático em simuladores, foi utilizado o Matlab/Simulink. Para isto, é necessário usar as equações (12) e (15) no domínio de *Laplace*, que levam a:

$$\frac{\theta(s)}{E_a(s)} = k \frac{k}{s[L_a J s^2 + (L_a F + R_a J)s + R_a F + k_2 k_1]} \quad (16)$$

Desprezando a indutância de armadura L_a , a função de transferência simplificada, para a posição do eixo como saída, resulta:

$$\frac{\theta(s)}{E_a(s)} = \frac{k}{s(\tau s + 1)} = Gp(s) \quad (17)$$

onde :

$$K = \frac{k_2}{(R_a F + k_2 k_1)} \quad (18)$$

$$\tau = \frac{R_a J}{(R_a F + k_2 k_1)} \quad (19)$$

Foi desenvolvida uma simulação no Matlab com *interface* do Simulink, a fim de criar a malha fechada do motor em uma rede CAN-FD e desenvolver uma simulação para checar a limitação dessa mesma rede e compará-la com uma rede CAN.

Para o estudo do sistema de controle na rede CAN, foi utilizado o mesmo modelo de motor de (TIPSUWAN; CHOW; VANIJJIRATTKHN, 2003), assim como os respectivos valores dos parâmetros. Neste estudo, foram aproveitadas as equações do motor e do controlador, que foram modificadas de acordo com o tempo de envio de cada mensagem do barramento.

$$G_p(s) = \frac{2029,826}{(s+26,29)(s+2,296)} \quad (20)$$

Para o controle do motor, utilizar-se-á um controlador PID, definido pela seguinte equação:

$$G_c(s) = \frac{\beta K_p [s + (\frac{K_i}{K_p})]}{s} \quad (21)$$

A equação 20 descreve o motor e a equação 21 descreve o controlador conforme a Figura 32. Utilizando as redes CAN e CAN-FD, é possível realizar o controle de um motor (módulo do atuador) e receber o seu posicionamento (módulo de respostas dos sensores) para processamento de seu controle.

É enviado um sinal de degrau para o sistema, que então passa pelo controlador e efetua o controle sobre o motor (atuador). O controlador através de sensores transmite para o sistema de gerenciamento a sua posição atual (ou velocidade), para que possa realimentar o controlador. Neste sistema ocorre atraso entre os envios da informação de posição atual do motor, e o atraso do sistema de controle para gerenciamento do atuador.

Este modelo, desenvolvido em MATLAB/Simulink®, representa uma simulação de controle de um motor utilizando ou uma rede CAN ou CAN-FD. O objetivo é explorar os limites de transmissão de dados, os atrasos na rede para envio de sinal

de controle e recebimento do sinal atual de posição do motor para realimentação do controlador, sem que prejudique o funcionamento adequado deste motor.

Figura 32 - Controle de malha fechada na rede CAN e CAN-FD com controlador PID.

Fonte: Elaborado pelo autor.

$$K_p = 0,1701 \text{ e } K_1 = 0,378$$

K_p é o ganho proporcional

K_1 é o ganho integral

$G_p(s)$ é a planta do motor DC

b é o parâmetro de ajuste do K_p e K_1 , neste caso será utilizado $b = 1$.

Para verificar o comportamento do sistema de controle sob a rede CAN, foram simuladas as funções de transferência do controlador e motor, representadas na Figura 33. O controlador transmite um sinal para um conversor A/D que necessita transformar o sinal analógico em digital que, por sua vez, envia para o barramento do CAN-FD. O sinal é convertido novamente de digital para analógico e enviado ao motor. Como é uma malha fechada, esse sinal é convertido para digital novamente, enviado ao barramento de comunicação, convertido em analógico, voltando para o controlador ajustando o sinal da melhor forma possível (CHOW; TIPSUWAN, 2003).

Figura 33 - Diagrama de blocos do sistema de controle CAN/CAN-FD.

Fonte: Elaborado pelo autor

Conforme descrito anteriormente, existem duas formas de transmitir as mensagens, periodicamente ou por evento. Este exemplo utiliza mensagens periódicas. O comando de acionamento do motor está conectado ao controlador e é transmitido periodicamente através da Ms1, e o atuador possui um filtro para receber essa mensagem e acionar o motor, como mostrado na Figura 34. O controlador do motor terá a mensagem Ms2 do sensor que será transmitida periodicamente para a rede.

Com a introdução do sistema de controle nas redes CAN e CAN-FD, algumas variáveis temporais são adicionadas. Na mesma figura são apresentadas variáveis, como tarefas a serem executadas na rede de controle de malha fechada.

Figura 34 - Diagrama de tempo do sistema de controle na rede CAN e CAN-FD.

Fonte: Elaborado pelo autor.

Essa malha fechada das redes CAN e CAN-FD possui algumas tarefas. Essas tarefas são as mensagens transmitidas entre o barramento de comunicação e o controle do motor. Para isso, há uma ordem de execução das tarefas descritas na Tabela 14.

Tabela 14 - Tarefas de controle no CAN e CAN-FD

Tarefa 1a (T1a) - Medir a saída do motor DC e gerar a mensagem CAN com a informação.
Tarefa 1b (T1b) – Transmitir a mensagem do sensor (Tempo na fila para transmissão).
Tarefa 1c (T1c) – Transmitir da mensagem fisicamente no barramento.
Tarefa 2a (T2a) – Receber a mensagem do sensor, processá-la e gerar uma mensagem de correção para o atuador.
Tarefa 2b (T2b) – Transmitir da mensagem do atuador (Tempo na fila para transmissão).
Tarefa 2c (T2c) – Transmitir a mensagem fisicamente no barramento.
Tarefa 3 (T3) - Receber a mensagem do controlador e atuar.

Fonte: Elaborado pelo autor.

Para cada uma das tarefas, é considerado um atraso no *loop* de controle. Para esse estudo de caso, serão considerados os atrasos das T1a, T2a e T3 como zero, pois o tempo gasto para essas tarefas não estão ligados ao desempenho da rede CAN.

O tempo total do *loop* de controle é descrito pela equação:

$$\sum \tau_{total} = \tau_{T1a} + \tau_{T1a} + \tau_{T1b} + \tau_{T1c} + \tau_{T2a} + \tau_{T2b} + \tau_{T2c} + \tau_3 \quad (22)$$

Como:

$$\tau_{T1a} = 0, \tau_{T2a} = 0 \text{ e } \tau_{T3} = 0$$

$$\sum \tau_{total} = \tau_{T1b} + \tau_{T1c} + \tau_{T2b} + \tau_{T2c} \quad (23)$$

Na qual :

τ_{T1a} = tempo da mensagem T1a

τ_{T1b} = tempo da mensagem T1b

τ_{T1c} = tempo da mensagem T1c

τ_{T2a} = tempo da mensagem T2a

τ_{T2b} = tempo da mensagem T2b

τ_{T2c} = tempo da mensagem T2c

τ_{T3} = tempo da mensagem T3

Individualizando os atrasos para cada uma das mensagens, é possível estabelecer que:

$$\text{Atraso Ms1} = \tau_{T1b} + \tau_{T1c}$$

$$\text{Atraso Ms2} = \tau_{T2b} + \tau_{T2c}$$

Os tempos físicos de transmissão das mensagens no barramento τ_{T1c} e τ_{T2c} são calculados de acordo com as equações apresentadas no subcapítulo 3.1 da Metodologia. Para este estudo de caso, foi utilizado o CAN 2A de 500 Kbps e o tamanho das mensagens de 8 bytes com $C_m = 0,25$ ms. Como as duas mensagens ms1 e ms2 utilizam o mesmo protocolo e possuem o mesmo tamanho de dados, é possível estabelecer que:

$$\tau_{T1b} = \tau_{T2b} = 0,5 \text{ ms}$$

5.2 ANÁLISE TEMPORAL E CONTROLE DA MALHA FECHADA DA REDE CAN

Os tempos de atraso τ_{T1c} e τ_{T2c} são gerados pelo atraso na fila, que podem ser obtidos nas Tabelas 11, 12 e 13 do primeiro estudo de caso, que é a diferença entre o valor teórico de transmissão da mensagem e o valor real. Foram estudados dois casos de mensagens retirados da rede CAN do veículo Polo. No segundo estudo de caso, o objetivo é identificar o limiar dos tempos na rede CAN, ou seja, o limite na qual o sistema começa a ficar totalmente instável e com altíssimos atrasos. O *busload* limiar foi acima de 98,86% onde a transmissão dos dados para, conforme a Tabela 15.

Tabela 15 - Tempo de fila das mensagens na malha fechada na rede CAN.

	1ªConfig.	2ªConfig.	3ªConfig.	4ªConfig.
Busload (%)	29,62	49,88	70,13	98,86
Tempo de Fila da Ms1 (ms)	0,445	-0,224	2,975	1000+
Tempo de Fila da Ms2 (ms)	-0,0007	0,778	2,538	1000+

Fonte: Elaborado pelo autor.

Na tabela 12, quando há um valor negativo significa que a mensagem foi transmitida antecipadamente, enquanto que um valor positivo significa que chegou

atrasada. As ms1 e ms2 serão transmitidas a cada 20ms (*time triggered*) para Ms1 e a 10 ms para Ms2. Por isso, além do tempo gasto na fila para transmissão, os valores de 20 ms e 10 ms respectivamente serão considerados um atraso na malha de controle.

Para a primeira configuração:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 20\text{ms} \rightarrow 0,25\text{ms} + 0,445\text{ms} + 20\text{ms} = 20,695\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 10\text{ms} \rightarrow 0,25\text{ms} - 0,0007\text{ms} + 10\text{ms} = 10,2493\text{ms}$.

Para a segunda configuração:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 20\text{ms} \rightarrow 0,25\text{ms} - 0,224 + 20\text{ms} = 20,026\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 10\text{ms} \rightarrow 0,25\text{ms} + 0,778\text{ms} + 10\text{ms} = 11,028\text{ms}$.

Para a terceira configuração:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 20\text{ms} \rightarrow 0,25\text{ms} + 2,975 + 20\text{ms} = 23,225\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 10\text{ms} \rightarrow 0,25\text{ms} + 2,538\text{ms} + 10\text{ms} = 12,788\text{ms}$.

Para a quarta configuração:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 20\text{ms} \rightarrow 0,25\text{ms} + 1000\text{ms} + 20\text{ms} = 10020,25\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 10\text{ms} \rightarrow 0,25\text{ms} + 1000\text{ms} + 10\text{ms} = 10010,25\text{ms}$.

5.2.1 Resultados da análise temporal de controle da malha fechada da rede CAN

Foram simuladas quatro configurações na rede CAN, que foram submetidas a uma carga de mensagens até chegar ao limite de 98,86%. A primeira com *busload* de 29,62%, a segunda com 49,88%, a terceira com 70,13% e a quarta com 98,86%,

conforme a Figura 35. Existem 4 gráficos para cada configuração, referentes aos sinais de entrada, do controlador, do sensor e do sensor com atraso na rede. Os sinais são apresentados de acordo com o tempo em execução da simulação(eixo x).

Figura 35 - Comparação de 4 tipos de configurações de controle em uma rede CAN.

Fonte: Elaborado pelo autor.

De acordo com os resultados, a estabilidade do sistema de controle está diretamente ligada ao mpo de atraso das mensagens. Quando as mensagens estão sem atrasos, o sistema de controle funciona perfeitamente, porém no último teste (4ª configuração), no qual o sistema de controle funcionou em uma rede com uma taxa de utilização muito alta (*busload* de 98,86%), obteve-se um atraso maior nas mensagens de menor prioridade e como o sistema utilizou essas mensagens, o sistema controlado ficou totalmente instável.

Então a partir desses resultados, conclui-se que a rede CAN não deve ser utilizada atualmente devido a sua limitação de transmissão dos dados que são

enviados com atrasos significantes por causa da grande quantidade de dados na rede (*busload* alto).

5.3 ANÁLISE TEMPORAL E CONTROLE DA MALHA FECHADA DA REDE CAN-FD

Para a malha fechada na rede CAN-FD, o cálculo segue o mesmo princípio que o CAN, porém com tempos diferentes, conforme mostrado na Tabela 16.

Tabela 16 – Tempo de fila das mensagens na malha fechada na rede CAN-FD.

	1ªConfig.	2ªConfig.	3ªConfig.	4ªConfig.
Busload (%)	6,11	8,18	10,44	13,46
Tempo de Fila da Ms1 (ms)	-0,082	-0,082	-0,082	-0,082
Tempo de Fila da Ms2 (ms)	-0,031	- 0,031	-0,043	-0,043

Fonte: Elaborado pelo autor.

Essa tabela é referente ao tempo de envio das mensagens na rede CAN-FD, de acordo com o *busload* de cada configuração. Para essa configuração, será considerada uma taxa de transmissão a 4 MHz e até 64 bytes de dados. Desta forma, τ_{T1b} e τ_{T2b} será de 0,22 ms.

Para a primeira configuração do CAN-FD:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 10\text{ms} \rightarrow 0,22\text{ms} - 0,082\text{ms} + 10\text{ms} = 10,138\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 20\text{ms} \rightarrow 0,22\text{ms} - 0,031\text{ms} + 20\text{ms} = 20,189\text{ms}$.

Para a segunda configuração do CAN-FD:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 20\text{ms} \rightarrow 0,22\text{ms} - 0,082 + 10\text{ms} = 10,138\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 10\text{ms} \rightarrow 0,22\text{ms} - 0,031\text{ms} + 20\text{ms} = 20,189\text{ms}$.

Para a terceira configuração do CAN-FD:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 10\text{ms} \rightarrow 0,22\text{ms} - 0,082 + 10\text{ms} = 10,138\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 20\text{ms} \rightarrow 0,22\text{ms} - 0,043\text{ms} + 20\text{ms} = 20,177\text{ms}$.

Para a quarta configuração do CAN-FD:

Atraso total de Ms1 = $\tau_{T1b} + \tau_{T1c} + 10\text{ms} \rightarrow 0,22\text{ms} - 0,082\text{ms} + 10\text{ms} = 10,138\text{ms}$;

Atraso total de Ms2 = $\tau_{T2b} + \tau_{T2c} + 20\text{ms} \rightarrow 0,22\text{ms} - 0,043\text{ms} + 20\text{ms} = 20,177\text{ms}$.

De acordo com a Figura 36, as quatro configurações de *busload* do CAN-FD tiveram resultados muito melhores, quando comparadas com as respectivas do CAN. Enquanto que o *busload* no CAN foi de 98,86% ele foi de 13,46% no CAN-FD. Através desses resultados, é possível inferir que essa nova rede de comunicação não causará problemas de atraso das mensagens aos consumidores por permitir uma previsibilidade e segurança altas.

Figura 36 - Comparação de uma malha fechada com variação do *busload* na rede CAN-FD.

Fonte: Elaborado pelo autor.

Além dessas medições, o projeto testou o limite do CAN-FD na malha fechada. Primeiramente, o projeto testaria quantos motores a malha consegue suportar, porém foi identificado que a malha pode suportar quantos motores houverem desde que a rede CAN-FD consiga enviar as informações sem atrasos, então não tem como introduzir motores em série na rede e dizer quantos podem funcionar. O que foi testado neste estudo é que a rede CAN-FD tem um limite de transmissão de dados em 98,86% do seu *busload*. A Figura 37 mostra o atraso das mensagens em “Bus

Busy”, o tempo de atraso é superior a 10 segundos e isso compromete todo o sistema de controle independente do número de motores na rede.

Time	Diff Time	Chn	ID	Event Type	Dir	DLC	Data length	Data	Bus Busy	Bus Idle
0.931960				Main Queue Load => Normal						
0.019945	0.019945	CAN 2	500	CAN FD Frame	Tx	8	8	20 99 1D 00 00 00 80 00	10523.650 ms	0.000 ms
0.009982	0.009982	CAN 2	280	CAN FD Frame	Tx	15	64	21 A5 00 00 28 00 25 28 0C 61 12 00 00 60 67 9C 00 65 00 00 80 00 0E 56 24 00 00 00 00 41 33 ...	10530.299 ms	0.000 ms
0.000888	0.000888	CAN 2	600	CAN FD Frame	Tx	15	64	01 00 00 00 00 00 01 00 ...	10532.298 ms	0.000 ms
0.000888	0.000888	CAN 2	601	CAN FD Frame	Tx	15	64	00 01 00 00 00 00 01 00 ...	10532.520 ms	0.000 ms
0.000888	0.000888	CAN 2	602	CAN FD Frame	Tx	15	64	00 00 01 00 00 00 01 00 ...	10532.741 ms	0.000 ms
0.006020	0.006020	CAN 2	636	CAN Frame	Tx	15	8	00 00 00 00 00 00 00 00	4.233 ms	0.000 ms
0.019945	0.019945	CAN 2	50	CAN FD Frame	Tx	6	6	00 00 18 18 00 00	10520.104 ms	0.000 ms
0.019945	0.019945	CAN 2	388	CAN FD Frame	Tx	13	32	83 02 81 32 02 30 00 80 11 00 FF 0C 80 02 41 A0 17 41 00 51 0C 00 3F 10 00 00 00 07 A0 84 00 00 ...	10520.456 ms	0.000 ms
0.025263	0.025263	CAN 2	320	CAN FD Frame	Tx	12	24	01 06 28 00 00 00 00 00 82 98 98 00 61 00 FF 00 EF 86 01 40 79 F3 F2 00	10510.476 ms	0.000 ms
0.000888	0.000888	CAN 2	603	CAN FD Frame	Tx	15	64	00 ...	10532.963 ms	0.000 ms
0.001998	0.001998	CAN 2	604	CAN FD Frame	Tx	15	64	00 ...	10529.198 ms	0.000 ms

Figura 37 - *Busload* limiar de uma rede CAN-FD

Fonte: Elaborado pelo autor.

5.4 VALIDAÇÃO DO MÉTODO DEFINIDO PARA CÁLCULO DO BUSLOAD

Nesta pesquisa foram executados dois estudos de caso sobre a rede CAN e CAN-FD. Este projeto executou a validação do método das mensagens definido pelo cálculo do *busload*. Além das simulações e desenvolvimentos realizados, houve a necessidade de criar um método que orienta os projetistas da arquitetura de rede CAN-FD nos veículos. Foram divididos em 3 seções para o projetista. A primeira é chamada de “Verde”, na qual o *busload* vai até 30%. Isso mostra que o desenvolvimento de novos módulos pode prosseguir sem problemas de afetar barramento. Entre 30% e 70% do *busload* é a seção “Amarela” na qual há necessidade de observar os atrasos das mensagens no barramento. A partir dos 70% de *busload* é determinada a zona “vermelha” a qual deve ser analisada e identificada a necessidade de substituição do CAN para o CAN-FD. Neste caso, o projetista sabe que a zona amarela pode receber até 20 mensagens adicionais com alta prioridade e tempo de 10 ms cada sem se preocupar com a zona vermelha. Sabendo disso, o desenvolvedor pode projetar a troca da rede CAN para o CAN-FD tendo um prazo adequado para isso. Para haver o “*overhead*” da rede, são necessárias 32 mensagens

adicionais o que pode levar cerca de 8 anos dependendo do projeto, da quantidade de módulos a serem adicionados e de outros fatores do fabricante.

A Figura 38 mostra a curva da rede CAN de um veículo polo com simulação até 70% do *busload* na qual pode ser visualizado a estratégia de orientação ao projetista citada.

Figura 38 - Gráfico para orientação ao projetista

Fonte: Elaborado pelo autor.

6 CONCLUSÃO

O principal objetivo deste trabalho foi de explorar os sistemas e suas limitações, desenvolvendo métodos e ferramentas que possam executar essa investigação e demonstrar se houveram vantagens do protocolo CAN-FD sobre o CAN convencional. O protocolo CAN-FD foi concebido para aperfeiçoar as aplicações de sistemas eletrônicos dentro de um ambiente com inúmeros módulos interligados garantindo um alto grau de previsibilidade.

Ao analisar o comportamento temporal do CAN, verificou-se um problema, pois quando um sistema é adicionado ou modificado dentro da rede, toda a rede pode ser afetada além de haver falta de previsibilidade. Por outro lado, no caso do CAN-FD podem ser adicionados muitos módulos com muitas mensagens e um *busload* alto. Entretanto, há falta de previsibilidade quando inseridas muitas mensagens no barramento, pois de acordo com os testes realizados neste projeto, as mensagens param de transmitir com *busload* acima de 98%. O sistema de tempo fica em um estado de atraso infinito. Nesta pesquisa, a simulação verificou um atraso superior a 400 segundos. Isso mostra que com esse *busload* alto, as mensagens não serão enviadas.

A análise temporal em uma rede de controle em malha fechada mostrou que o CAN teve seu comportamento prejudicado, enquanto o CAN-FD obteve um desempenho muito melhor, porém também limitado à quantidade de dados no barramento.

Este estudo explorou as funcionalidades do CAN-FD e suas limitações, o que poderá permitir o desenvolvimento de sistemas utilizando microcontroladores que suportam essa tecnologia de comunicação.

Além disso, existe a possibilidade para que novas pesquisas sejam desenvolvidas baseadas no entendimento do protocolo do CAN-FD, demonstrados aqui, e também que seja explorada mais detalhadamente essa rede em uma malha fechada de controle. Como desenvolvimento adicional, foi criado um método representado através de um gráfico que pode orientar os projetistas de uma rede CAN na hora do desenvolvimento de um veículo em bancada. Esse tipo de orientação possibilita ter um controle de qual rede usar ao longo dos anos tendo em vista a capacidade de expansão dos módulos e o fator custo/benefício do projeto.

Esta pesquisa da rede CAN-FD também contribuiu com o meio acadêmico, visto que essa rede surgiu do setor industrial o qual detém muitas informações em segredo e este trabalho desenvolveu método de validação do protocolo através de equações que foram extraídas das simulações de dados na rede entre outros desenvolvimentos. Essa validação comprovou os resultados simulados com os cálculos teóricos.

TRABALHOS DECORRENTES

Publicação no IEEE-Engineering Education Conference (EDUCON-2013) com apresentação do trabalho: “A Didactic Platform to study of CAN-FD Bus”, publicado no IEEE - Berlin– Alemanha, Março/2013 página 318-323, ISSN 2165-9559.

Publicação no International Conference on Engineering Education (ICEE 2012) com apresentação do trabalho: “THE STUDY OF CAN BUS BY A DIDACTIC KIT”, publicado no do International Conference on Engineering Education, página 975/981, ICEE 2012, Turku – Finlândia, Julho/2012.

REFERÊNCIAS

- BAO, W. et al. Analysis and research on the real-time performance of Profibus Fieldbus. In: WRI WORLD CONGRESS ON SOFTWARE ENGINEERING, Xiamen, 2009. **WCSE '09 Proceedings**. Washington: IEEE Computer Society, 2009. p. 136-140. DOI: 10.1109/WCSE.2009.181
- BOSCH. **CAN with Flexible Data-Rate**. Gerlingen: Robert Bosch GmbH, 2011. Disponível em: <http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can_fd.pdf> Acesso em: 17 abril 2012.
- BOSCH. **Automotive Handbook**. 6th ed., Robert Bosh GmbH, 2005.
- BURNS, A. **Real-Time systems**. Academic Press, 2002.
- BURNS, A. & WELLINGS, A. **Real-Time systems and programming languages**. 3th ed. MA- USA: Addison Wesley, 2001. 738p.
- CHEN, H.; TIAN, J. Research on the Controller Area Network. In: IEEE - International Conference on Networking and Digital Society, 9., Guiyang, Guizhou, 2009. **ICNDS'09 Proceedings**. 2009. IEEE, p. 251-254.
- CHOW. M. & TIPSUWAN, Y. Gain adaptation of networked DC motor controllers based on QoS variations. **IEEE Transactions on Industrial Electronics**, v. 50, n. 5, p. 936-943. Oct. 2003.
- CIA (CAN In Automation). Disponível em: <<http://www.can-cia.org/index.php?id=522>>. Acesso em 23 de março de 2013.
- DANG, J.N. **Statistical analysis of the effectiveness of electronic stability. 2007**. (DOT HS 810 794 July 2007 - NHTSA Technical Report) Disponível em: <<http://www-nrd.nhtsa.dot.gov/Pubs/810794.pdf>>. Acesso em: 23 de março de 2013.
- GUIMARÃES, A. A. **Eletrônica embarcada automotiva**. São Paulo: Érica. 2007.
- HODEL, K. N. **Limites do protocolo CAN (Controller Area Network) para aplicações que exigem alto grau de confiabilidade temporal**. 2008. 77p. Dissertação (Mestrado). Escola Politécnica, Universidade de São Paulo, São Paulo, 2008.
- HODEL, K. N. **Limits of the CAN protocol (Controller Area Network) for applications that require high reliability of time**. SAE Brasil 2010. Disponível em: <http://www.saebrasil.org.br/livraria/vitrine.asp?codigo_produto=2444>. Acesso em: 21 de outubro de 2014.
- INTERNATIONAL TELECOMMUNICATION UNION. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. **ITU-T Recommendation X.224**. 1995. Disponível em: <<http://www.itu.int/rec/T-REC-X.224-199511-I/en/>>. Acesso em: 11 jun. 2013.

KAHANE, C. J., & DANG, J. N.. **The long-term effect of ABS in passenger.** **Washington DC, Columbia.** Washington, DC: NCSA/NHTSA, 2009. 89p. Disponível em: <<http://www-nrd.nhtsa.dot.gov/Pubs/811182.PDF>>. Acesso em: 20 de junho de 2013.

KIENCKE, U., DAIS, S., LITACHEL, M. **Automotive serial Controller Area Network.** SAE International, 1986.

KOPETZ, H. **Real-Time systems: Design principles for distributed embedded applications.** Kluwer Academic Publishers, 1997.

LI HUI, Z. H. (2009). Research and application on INTERBUS operator terminal. In: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND INFORMATION TECHNOLOGY, 2., 2009, Beijing. **ICCSIT 2009 Proceedings.** IEEE, 2009. p. 309-312. DOI:10.1109/ICCSIT.2009.5234561

LUO, F. et al. CAN transceivers conformance test system development. In: VEHICLE POWER AND PROPULSION CONFERENCE, 2008, Harbin. **VPPC'08 Proceedings.** IEEE, 2008. DOI: 10.1109/VPPC.2008.4677522

MICROCHIP TECHNOLOGY INC. (2005). **Stand-Alone CAN controller with SPI interface – MCP 2515.** Disponível em: <<http://ww1.microchip.com/downloads/en/DeviceDoc/21801d.pdf>>. Acesso em: 10 de agosto de 2013.

MIESTERFELD, F. The next generation vehicle architecture. In: MURGEN, R. K; **multiplexing and networking: automotive electronics series.** 1999. p. 645-652.

MURPHY, R. **A CAN to FlexRay migration framework.** 2009, 325p. MSc. Thesis . Waterford Institute of Technology Awards Council, Ireland, 2009. Disponível em: <http://repository.wit.ie/1398/1/A_CAN_to_FlexRay_Migration_Framework.pdf>. Acesso em: 18 de Agosto de 2013.

NAVET, N. & SIMONOT-LION, F. eds. **Automotive embedded systems handbook.** Florida: CRC Press/Taylor and Francis. 2008.

NAVET, N. et al. Trends in automotive communication systems. **Proceedings of the IEEE** v. 96, n.6 p. 1204-1223. June 2005.

NEVERS, N. D. **Air pollution control engineering.** Long Grove, IL: McGraw-Hill. 1995.

NOLTE, T., HANSSON, H., & BELLO, L. Automotive Communications - Past, Current and Future. In: CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 10., 2005. **ETFA 2005 Proceedings.** 2005, p. 984-992.

PINTO, R., RUFINO, J., & ALMEIDA, C. High availability in controller area networks. In: INTERNATIONAL CONFERENCE ON COMPUTER AS A TOOL. 2011, Lisboa,

EUROCON Proceedings. IEEE, 2011. p. 1-4. DOI: 10.1109/EUROCON.2011.5929399

PLOSS R.; MUELLER,A.; LETEINTURIER, P. Solving automotive challenges with electronics. In: VLSI TECHNOLOGY, SYSTEMS AND APPLICATIONS, 2008, Hsinchu. **VLSI-TSA 2008 Proceedings.** IEEE, 2008. p. 1-2. DOI:10.1109/VTSA.2008.4530772

RAMKUMAR, S. M. (1998). A capstone experience-development of a flexible manufacturing system (FMS) using smart distributed sensing system. In: FRONTIER IN EDUCATION CONFERENCE, 28.1998, Tempe. **FIE'98 Proceedings.** IEEE, 1998. p. 399-404. DOI: 10.1109/FIE.1998.736883

RUFINO, J. An overview of the Controller Area Network. In: CiA Forum CAN for Newcomers. **Proceedings.**, Braga, Portugal, CiA. Jan. 1997.

SANTOS, G., GALVÃO, B., ONUSIC, H., & SARTORI, C. (2000). Comercial vehicles - EMC evaluations in South America. In: EUROPEAN SYMPOSIUM ON ELECTROMAGNETIC COMPATIBILITY, 4., 2000, Brugue. **EMC Europe 2000 Proceedings.** 2000, p. 61-65.

SCHIFFER, V.; VANDESTEEG, K.W.; VASKO, D. A. Introduction to DeviceNet safety. In: INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS. 2000, Porto. **WFCS2000 Proceedings.** 2000. P.293-300. DOI:10.1109/WFCS.2000.882561

SCHMIDT, C. Automotive electronics - enabling the future of individual mobility. In: INTERNATIONAL ELECTRON DEVICES MEETING. 2007, Washington, DC. **IEDM 2007 Proceedings.** IEEE International, 2007. p. 3-8. DOI: 10.1109/IEDM.2007.4418850

Schulze, M., Mäkinen, T., Irion, J., Flament, M. & Kessel, T. (2008). *IP PreVent Final Report*. Retrieved 25 september 2008, European Commission, Brussels. Disponível em < http://www.transport-research.info/Upload/Documents/201012/20101208_184029_81524_PReVENT_Final_Report.pdf.> Acesso em: 14 de outubro de 2014.

STEVENS, K. R. (**TCP/IP illustrated the protocols.** 2nd Ed., Addison-Wesley, 2012. v.1. Kindle Edition loc 3505.

THE CAN-FD Protocol overcomes CAN limits.**CNL Online**, March 2012. Disponível em: <http://can-newsletter.org/catalogue/do/printArticle/id/nr_stand_can-fd_2012-03-08/nr_stand_can-fd_2012-03-08.pdf>. Acesso em: 12 de setembro de 2013.

TINDELL, K.; BURNS, A. **Guaranteed message latencies for distributed safety critical hard real-time networks.** England: University of York, 1994. Technical Report YCS 299.

TINDELL, K.; BURNS, A.; WELLINGS, A. Calculating Controller Area Network message response times. **Control Engineering Practice**. v.3, n. 8 p.1163-1169, 1995.

TIPSUWAN, Y., CHOW, M., VANIJJIRATTKHN, R. An implementation of a networked PI controller over IP network. In: CONFERENCE ON INDUSTRIAL ELECTRONICS SOCIETY, 29., 2003. **IECON'03 Proceedings**. IEEE, 2003 v.3, p.2805-2810. DOI: 10.1109/IECON.2003.1280692

VAZ, R. M. **Automatizando o agendamento de mensagens no protocolo Flexray**. Monografia. FATEC Santo André, 2011. 100p.

VECTOR. **CAN-FD Flexible Data Rate**. Disponível em: < <http://vector.com>>. Acesso em: 22 jan. 2013.

WEI LUN, N.; CHEE KYUN, N.; NOODIN, N. K. N.; ROKHANI, F. Z.; ALI, B.M. Home appliances controller using wireless controller area network (WCAN) system. In: INTERNATIONAL CONFERENCE ON COMPUTER AND COMMUNICATION ENGINEERING , 2010, Kuala Lumpur. **ICCCE 2010 Proceedings**. IEEE, 2010,p. 1-6. DOI: 10.1109/ICCCE.2010.5556759.

WILFRIED, V. **A comprehensible guide to Controller Area Network**. Massachusetts: Copperhill Media Corporation, 2005.

WOZNIAK, J. Mobility management solutions for IP networks. In: INTERNATIONAL TELECOMMUNICATIONS NETWORK STRATEGY AND PLANNING SYMPOSIUM, 15., 2012, Roma. **Proceedings**. IEEE, 2012. P.1-11. DOI:10.1109/NETWKS.2012.6381665

ZIMMERMANN, H. OSI reference model — The ISO model of architecture for open systems interconnection. **IEEE Transactions on Communications**, v. 28, n.4, p. 425–432. April 1980.

ZUBERI, K.M. & SHIN, K.G.. Real-time decentralized control with CAN. In: CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION. 1996. **EFTA'96 Proceedings**. IEEE, 1996. v.1, p.93-99. DOI:10.1109/ETFA.1996.573261

APÊNDICE

APÊNDICE A

Programa para transformar mensagens dos módulos CAN em CAN-FD.

```
#-----
# Projeto desenvolvido por Ricardo de Andrade#
#-----
variables
{
    message * mymsg;
}
on message CAN1.*
{
    mymsg.ID = this.ID;      // copia o ID original para o CAN-FD mymsg.DLC =
this.DLC; // copia o DLC original para o CAN-FD mymsg.CAN = 2;      //
para ter a saída da mensagem no CAN mymsg.EDL = 1;      // para mudra
para CAN-FD no CAN2
    mymsg.BRS = 1;          // para chavear para bit de maior taxa de
transmissão.
    switch (this.DLC)
    {
        case 1:
            mymsg.byte(0) = this.byte(0);    // para copiar o primeiro byte
original no campo de dados no campo de dados do CAN-FD
            break;

        case 2:
            mymsg.word(0) = this.word(0);     // 2 bytes
            break;

        case 3:
            mymsg.word(0) = this.word(0);     // 3 bytes
            mymsg.byte(2) = this.byte(2);
            break;

        case 4:
            mymsg.dword(0) = this.dword(0);   // 4 bytes
            break;

        case 5:
            mymsg.dword(0) = this.dword(0);   // 5 bytes
            mymsg.byte(4) = this.byte(4);
            break;

        case 6:
            mymsg.dword(0) = this.dword(0);   // 6 bytes
            mymsg.word(4) = this.word(4);
            break;

        case 7:
            mymsg.dword(0) = this.dword(0);   // 7 bytes
            mymsg.word(4) = this.word(4);
            mymsg.byte(6) = this.byte(6);
            break;

        case 8:
            // 8 bytes
```

```
        mymsg.qword(0) = this.qword(0);  
        break;  
    }  
    output (mymsg);  
}
```

APÊNDICE B

Programa para converter mensagens dos módulos CAN em CAN-FD conforme estratégia número 1.

```
#-----
# Projeto desenvolvido por Ricardo de Andrade#
#-----

variables
{
    /*****
    Motor mensagens
    *****/
    message 0x280 my1msg; // bloco de mensagens de 10 ms
    qword a1;
    qword b1;
    qword c1;
    qword d1;
    msTimer my1tim;

    message 0x480 my2msg; // bloco de mensagens de 20 ms
    qword a2;
    qword b2;
    msTimer my2tim;

    message 0x580 my3msg; // variável do ID 580 maior que 1 segundo.

    /*****
    Cluster msgs
    *****/
    message 0x420 msg2can420;
    qword msg1can420;
    qword msg1can520;
    msTimer timer420;

    message 0x320 msg2can320;

    /*****
    Conforto mensagens
    *****/

    message 0x388 msg2can388;
    qword msg1can388;
    qword msg1can38A;
    msTimer timer388;

    message 0x470 msg2can470;

    message 0x5d0 msg2can5d0;
    dword msg1can5d0dword;
    word msg1can5d0word;
```

```

    qword msg1can5d8;
    dword msg1can570;
    msTimer timer5d0;

    /*****
    Ar-Condicionado mensagens
    *****/
    message 0x5e0 msg2can5e0;

    /*****
    Módulo não Identificado mensagens
    *****/
    message 0x50 msg2can50;
    qword msg1can50;
    qword msg1can3d0;
    msTimer timer50;
}

on start
{
    setTimerCyclic (myltim, 10);    // tempo para todas as 4 mensagens de 10
ms de tempo.
    setTimerCyclic (my2tim, 20);    // tempo para duas mensagens de 20 ms de
tempo.
    setTimerCyclic(timer420,200);    // tempo para o ID 420 e 520 de 200 ms.
    setTimerCyclic(timer388,20);    // tempo para o ID 388 e 38 A de 20 ms.
    setTimerCyclic(timer5d0,100);    // tempo para ID 5D0,5d8 e 570 de 100 ms.
    setTimerCyclic(timer50,20);    // tempo do ID 50 e d30 de 20 ms.
}

/* ***** Motor msgs ***** */

on message can1.0x280
{ a1 = this.qword(0); } // pega o valor dos oito bytes de dados da
mensagem 280.
on message can1.0x288
{ b1 = this.qword(0); } // pega o valor dos oito bytes de dados da
mensagem 288.
on message can1.0x380
{ c1 = this.qword(0); }
on message can1.0x488
{ d1 = this.qword(0); }

on message can1.0x480
{ a2 = this.qword(0); } // pega o valor dos oito bytes de dados da
mensagem 480.
on message can1.0x588
{ b2 = this.qword(0); } // pega o valor dos oito bytes de dados da
mensagem 588.

on message can1.0x580
{
    my3msg = this;
    my3msg.can=2;
    my3msg.brs=1;
    my3msg.edl=1;

```

```

    output (my3msg);          // envia a msg FD 580 no CAN 2 bus,original de 8
    bytes de dados.
}

on timer my1tim
{
    my1msg.can = 2;
    my1msg.dlc = 13;          // comprimento de dado 4x8 bytes, 32 bytes
    my1msg.brs = 1;
    my1msg.edl = 1;
    my1msg.qword(0) = a1; // escreve os primeiros 8 bytes na mensagem FD.
    my1msg.qword(8) = b1; // escreve os segundos 8 bytes na mensagem do
    FD.
    my1msg.qword(16) = c1;
    my1msg.qword(24) = d1;
    output (my1msg);          // envia a mensagem FD 280 no CAN 2 bus (
    somente simulado).
}

on timer my2tim
{
    my2msg.can = 2;
    my2msg.dlc = 10;          // tamanho de 2x8 bytes,16 bytes.
    my2msg.brs = 1;
    my2msg.edl = 1;
    my2msg.qword(0) = a2; //escreve os primeiros 8 bytes na mensagem FD.
    my2msg.qword(8) = b2; //escreve os segundos 8 bytes na mensagem do FD.
    output (my2msg);          //envia a mensagem FD 480 no CAN 2 bus ( somente
    simulado).
}

/*****
Cluster msgs
*****/
on timer timer420
{
    // monta cabecalho
    msg2can420.can = 2;
    msg2can420.dlc = 10;
    msg2can420.brs = 1;
    msg2can420.edl = 1;

    // monta dados
    msg2can420.qword(0) = msg1can420;
    msg2can420.qword(8) = msg1can520;

    // exhibe as msgs
    output (msg2can420);
}

on message can1.0x420
{
    msg1can420 = this.qword(0);
}

on message can1.0x520
{
    msg1can520 = this.qword(0);
}

```

```

on message can1.0x320
{
    // copia tudo
    msg2can320 = this;

    // monta cabecalho
    msg2can320.can=2;
    msg2can320.brs=1;
    msg2can320.edl=1;

    //envia para o canal 2
    output(msg2can320);
}
/*****
MENSAGEM DO MÓDULO DE CONFORTO
*****/
//PREPARAÇÃO DE TIMER.

on timer timer388
{
    // monta cabecalho
    msg2can388.can = 2;
    msg2can388.dlc = 7;
    msg2can388.brs = 1;
    msg2can388.edl = 1;

    // monta dados
    msg2can388.dword(0) = msg1can388;
    msg2can388.dword(3) = msg1can38A;

    // exibe as msgs
    output(msg2can388);
}
on timer timer5d0
{
    // monta cabecalho
    msg2can5d0.can = 2;
    msg2can5d0.dlc = 11;
    msg2can5d0.brs = 1;
    msg2can5d0.edl = 1;

    // monta dados
    // msg2can5d0.qword(0) = msg1can5d0;
    // msg2can5d0.word(5) = msg1can5d0;
    //
    // msg2can5d0.qword(8) = msg1can5d8;
    // msg2can5d0.qword(16) = msg1can570;
    msg2can5d0.dword(0) = msg1can5d0dword ;
    msg2can5d0.word(4) = msg1can5d0word;
    msg2can5d0.qword(6) = msg1can5d8;
    msg2can5d0.dword(14) = msg1can570;

    // exibe as msgs
    output(msg2can5d0);
}

on message can1.0x388
{

```

```

    msg1can388 = this.qword(0);
}
on message can1.0x38A
{
    msg1can38A = this.qword(0);
}

on message can1.0x470
{
    // copia tudo
    msg2can470 = this;

    // monta cabecalho
    msg2can470.can=2;
    msg2can470.brs=1;
    msg2can470.edl=1;

    //envia para o canal 2
    output(msg2can470);
}

on message can1.0x5d0
{
    msg1can5d0dword = this.dword(0);
    msg1can5d0word = this.word(4);
}
on message can1.0x5d8
{
    msg1can5d8 = this.qword(0);
}

on message can1.0x570
{
    msg1can570 = this.qword(0);
}

/*****
AR - CONDICIONADO
*****/
on message can1.0x5e0
{
    // copia tudo
    msg2can5e0 = this;

    // monta cabecalho
    msg2can5e0.can=2;
    msg2can5e0.brs=1;
    msg2can5e0.edl=1;
    msg2can5e0.dlc =8;

    //envia para o canal 2
    output(msg2can5e0);
}

/*****/

```



```

MÓDULO NÃO IDENTIFICADO
*****/

on timer timer50
{
    // monta cabeçalho
    msg2can50.can = 2;
    msg2can50.dlc = 6;
    msg2can50.brs = 1;
    msg2can50.edl = 1;

    // monta dados
    msg2can50.dword(0) = msg1can50;
    msg2can50.word(4) = msg1can3d0;
    // exibe as msgs
    output(msg2can50);
}

on message can1.0x50
{
    msg1can50 = this.qword(0);
}
on message can1.0x3d0
{
    msg1can3d0 = this.qword(0);
}%

```