EDUARDO SILVEIRA OLIVEIRA

INFLUÊNCIA DO AUMENTO DA UTILIZAÇÃO DA LARGURA DA BANDA PASSANTE DA REDE DE COMUNICAÇÃO DE UM VEÍCULO AUTOMOTIVO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Automotiva.

São Paulo
2006

EDUARDO SILVEIRA OLIVEIRA

INFLUÊNCIA DO AUMENTO DA UTILIZAÇÃO DA LARGURA DA BANDA PASSANTE DA REDE DE COMUNICAÇÃO DE UM VEÍCULO AUTOMOTIVO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Automotiva.

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Lucas Antônio Moscato

São Paulo
2006

FICHA CATALOGRÁFICA

Oliveira, Eduardo Silveira
Influência do aumento da utilização da largura da banda passante da rede de comunicação de um veículo automotivo / E.S. Oliveira. -- São Paulo, 2006.
p.

Trabalho de curso (Mestrado Profissionalizante em Engenharia Automotiva) - Escola Politécnica da Universidade de São Paulo.

1.Sistema de comunicação veicular 2.Protocolo de comunicação serial digital I.Universidade de São Paulo. Escola Politécnica II.t.

Dedico este trabalho à minha esposa Fabiana, pela paciência, compreensão e incentivo, à minha filha Laura, que nasceu durante o curso, aos meus pais Antônio e Delúdia e às minhas irmãs Gisele e Rejane, pelo apoio irrestrito. Dedico também a Deus, que sempre está presente em minha vida, direcionando-me para o melhor caminho.

AGRADECIMENTOS

Ao meu orientador, o Prof. Dr. Lucas Antônio Moscato, pela orientação e diretrizes precisas, pela paciência e pela colaboração.

Ao Prof. Dr. Fábio Takase, pela colaboração e sugestões muito importantes para o sucesso do trabalho.

Aos meus amigos de curso Rogério Gimenez, Jairo Lima de Souza, Flávio Liviero e José Maria Muniz, pelo apoio, pelas caronas e pelas horas dedicadas de estudo e trabalho.

Aos amigos da Scania da Suécia, Daniel Thuresson, Niklas Bruce e Jonny Johansson pelo interesse do meu desenvolvimento e suporte ao meu trabalho.

Ao amigo Marcelo Saravalle, pela ajuda nas medições no caminhão e pelo conhecimento em CAN.

Ao Prof. Dr. Roger Johansson, da Universidade Chalmers, em Gotemburgo, na Suécia, pelo profundo conhecimento em sistemas X-By-Wire.

Aos meus cunhados Dorival Spolon e Fábio Medeiros, aos meus sogros, o Sr. Gilberto Nascimento e Dna. Lúcia Medeiros, à minha co-cunhada, Carla Nascimento e ao filho do meu cunhado Dorival, Leandro Spolon, pelo suporte e pela torcida pelo sucesso.

Ao Prof. Dr. Max Mauro, da Universidade Leste de Minas Gerais, pelo conhecimento em CAN.

Aos demais colegas da Scania, que me ajudaram no desenvolvimento deste trabalho.

RESUMO

A dependência do uso de computadores em veículos automotivos tem crescido muito nos últimos 20 anos, desde o desenvolvimento do primeiro sistema eletrônico de segurança ativa, o ABS. No entanto, novos sistemas e novas funções são adicionadas a cada ano e sistemas de aeronaves, tais como Fly-By-Wire, Brake-By-Wire, Steer-By-Wire, caracterizados por serem sistemas críticos de segurança (safety-critical), nos quais os sistemas puramente mecânicos dão lugar a sistemas mecatrônicos, começam a ser utilizados em veículos terrestres, que demandam alta taxa de transmissão de mensagens, determinismos, redundância e mecanismos de detecção de falhas. Um caminhão Scania contém atualmente mais de 20 computadores embarcados ou ECUs (Electronic Control Unit ≡ Unidade Eletrônica de Controle), que controlam as diversas funções utilizadas para sua aplicação. Estima-se que, em dez anos, esse número venha a crescer ainda mais e que, no futuro, sistemas X-By-Wire também serão contemplados em veículos pesados, como os caminhões. Sendo assim, faz-se necessária uma avaliação dos protocolos atualmente utilizados pela Scania para saber se as novas demandas serão atendidas. Este trabalho tem como objetivo estudar a influência do aumento da demanda de mensagens na largura da banda passante do sistema de comunicação de um veículo pesado e propor alternativas para a solução do problema.

ABSTRACT

The dependence of the use of electronics in automotive vehicles has increased a lot in the last twenty years, since the development of the first active safety system called ABS. However, new systems and new functions are developed each year and systems from airplanes, such as Fly-By-Wire, Brake-By-Wire, Steer-By-Wire, characterized as safety-critical, in which systems purely mechanical are replaced by mecatronic ones, start to be implemented in ground vehicles as well, raising demands for fast communication, determinism, redundancy and fault tolerances mechanisms. A Scania truck has more than twenty built-in computers or ECUs (Electronic Control Units), which control all functions used for a truck application. The forecast for the next ten years is that the number of ECUs will increase even more and, X-By-Wire systems will also be implemented in trucks. Therefore, it is necessary to evaluate today's communication protocols used by Scania to know whether the new demands will be fulfilled. This thesis has the objective to check the influence of the messages increase in the bandwidth of a heavy truck communication system and propose alternatives to solve the problem.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

ABS	- Antilock Brake System
ACK	- Acknowledge
APS	- Air Processing System
CAN	- Controller Area Network
CAN_H	- CAN High
CAN_L	- CAN Low
COO	- Coordinator
CPU	- Central Processing Unit
CRC	- Cyclic Redundancy Check
CSMA	- Carrier Sense Multiple Access
CSMA/CA	- CSMA with Collision Avoidance
CSMA/CD	- CSMA with Collision Detection
DA	- Destination Address
DLC	- Data Logic Control
DP	- Data Page
DS	- Directory Service
ECU	- Electronic Control Unit
EDF	- Earliest Deadline First
EMS	- Engine Management System

EOF - End of Frame

E/S - Dispositivos físicos para entrada e saída

ET-CAN - Event-Triggered CAN

FCS - Frame Check Sequence

FDM - Frequency Division Multiplexing

FMS - Fleet Management System

FSE - Format Synchronization Entity

FTAM - File Transfer, Access and Management

GE - Group Extension

ICL - Instrument Cluster

IDE - Identifier Extension

ISO - International Organization for Standardization

LLC - Logical Link Control

MAC - Médium Access Control

MDI - Medium Dependent Interface

MHS - Message Handling System

NRZ - Non Return to Zero

NTU - Network Time Unit

OSI - Open System Interconnection

PCI - Protocol Control Information

PDU - Protocol Data Unit

PF - PDU Format

PGN - Parameter Group Number

PLS - Physical Signaling

PMA - Physical Medium Attachment

RAM - Random Access Memory

RM-OSI - Reference Model OSI

RMS - Rate Monotonic Scheduling

RTR - Remote Transmission Request

RX - Recepção

SA - Source Address

SAE - Society of Automotive Engineers

SDU - Service Data Unit

SOF - Start of Frame

TCO - Tacograph

TDM - Time Division Multiplexing

TT-CAN - Time-Triggered CAN

TTP - Time Triggered Protocol

TUR - Time Unit Ratio

TX - Transmissão

Vdiff - Tensão Diferencial

1. INTRODUÇÃO

1.1. Motivação

Tem-se visto que, com o passar dos anos, tanto em carros de passeio quanto em veículos comerciais, a quantidade e a sofisticação de sistemas eletrônicos têm crescido em ritmo exponencial. Dentre os sistemas mais conhecidos estão o sistema anti-travamento de freio (ABS), o Airbag e o sistema de injeção eletrônica de motores. Hoje, a quantidade de fios necessários para interligar esses sistemas também tem crescido de forma considerável, chegando, em alguns casos, a atingir 4 km de cabos, se comparado aos 45 metros em veículos fabricados em 1955 (MURPHY, 2004).

Aumentar a quantidade de cabos representa aumento de peso, custo e diminuição da confiabilidade e desempenho dos sistemas eletrônicos, assim como representa a limitação na expansão de funções nos veículos. Do ponto de vista do Pós-Vendas, pode-se destacar a dificuldade de diagnose de falhas, que além de causar irritação, devido ao tempo de parada do veículo, é muito custosa para o cliente.

A partir das dificuldades acima descritas, fabricantes de sistemas eletrônicos começaram a pensar em uma solução que, embora diminuísse a quantidade de cabos, deveria permitir a ampliação de sistemas sem diminuição da confiabilidade, trabalho em condições severas, diagnóstico fácil das falhas e redução do tempo e custo de reparo, além da disponibilidade das informações de sensores e atuadores para todos os módulos eletrônicos do veículo.

À solução encontrada deu-se o nome de CAN (Controller Area Network), que é um canal de comunicação serial multiplexado no qual informações entre módulos eletrônicos distribuídos são compartilhadas.

Uma grande empresa de veículos pesados, a Scania, possui atualmente mais de 20 ECUs em seu sistema de comunicação. Muitas outras ECUs serão montadas em um caminhão Scania. Sistemas inteligentes que “liberem” o motorista de tarefas rotineiras, suportando-o em situações críticas, chamados de sistemas X-By-Wire (Steer-By-Wire, Brake-By-Wire, Drive-by-wire), nos quais sistemas puramente mecânicos dão lugar a sistemas mecatrônicos, demonstram ser uma tendência futura. A implementação desses sistemas exige fortes demandas de comunicação, mais rápida, eficiente e previsível.

1.2 Descrição do problema

A Scania utiliza como protocolo de comunicação o ET-CAN com o protocolo de nível alto a SAE J1939, que atualmente trabalham bem. Porém, o ET-CAN e a J1939 possuem limitações, tais como ECUs assíncronas, baixa largura de banda e mecanismos de tolerância de falhas.

1.3 Objetivos

Os objetivos deste trabalho são estudar a influência do aumento da utilização da largura da banda passante da rede de comunicação de um caminhão Scania e propor soluções para resolver esse problema.

1.4 Método

Para atender aos objetivos do trabalho, uma pesquisa literária foi realizada com as normas que estabelecem as diretrizes para implementação dos protocolos de comunicação, que são a SAE J1939, nos módulos 11 e 21 e a ISO 11898, nos módulos 1, 2 e 4.

Entrevistas com especialistas em sistemas de comunicação embarcada da Scania em sua matriz na Suécia, da Universidade Chalmers, em Gotemburgo, também na Suécia e da

Universidade Leste de Minas Gerais, foram usadas para adquirir conhecimento em comunicação veicular.

Foi feita uma análise de escalonamento na arquitetura dos veículos Scania para verificar como a largura de banda está sendo utilizada à taxa de transmissão (baud-rate) de 250 Kbits por segundo.

Por fim, um experimento em um caminhão Scania foi realizado na área de Protótipo na fábrica em sua planta localizada em São Bernardo do Campo para examinar o protocolo ET-CAN, permitir que se faça uma validação prática e propor soluções para o problema.

1.5 Organização da Dissertação

Esta dissertação procura analisar como o atraso de mensagens de alta prioridade é influenciado pelo aumento da carga da banda passante em um veículo automotivo.

No capítulo 2 é feito um breve histórico sobre a evolução dos sistemas de computação e das arquiteturas de computadores. São apresentadas as topologias mais comuns utilizadas nas redes de computadores. São discutidos conceitos de banda passante, modulação, multiplexação na freqüência e no tempo, codificação de sinais digitais, assim como técnicas de transmissão e detecção de erros. Neste capítulo também é apresentado o modelo OSI da ISO (RM-OSI) e é descrita cada camada do modelo. E por fim, uma introdução é feita para o protocolo CAN (Controller Area Network).

No capítulo 3 é detalhado o estudo sobre escalonamento de processos de sistemas operacionais de computadores. Tópicos como regiões críticas e condições de disputa e impedimento são descritos. Algoritmos de escalonamento para sistemas de tempo real são estudados e comparados.

No capítulo 4 o protocolos de comunicação serial digital, por evento (CAN), é descrito, através das normas ISO 11898, em seus módulos 1, 2 e 4, e SAE J1939, em seus módulos 11 e 21.

No capítulo 5 são realizados os estudos de escalonamento para a arquitetura utilizada atualmente e executados os testes para verificar se o objetivo deste trabalho foi alcançado e os resultados são analisados.

Por fim, no capítulo 6, são apresentadas conclusões, propostas de solução do problema e sugestões para pesquisas futuras.

2. SISTEMAS EMBARCADOS DE COMPUTADORES

2.1 Introdução

A comunicação é uma necessidade da sociedade humana que se iniciou desde os primórdios de sua existência. Várias formas de comunicação foram criadas, tais como sinais de fumaça e pombos-correio, permitindo a aproximação entre povos distantes (SOARES, L. F.G.; LEMOS, G.; COLCHER, S., 1995).

A partir de 1938, com a invenção do telégrafo por Samuel F. B. Morse, iniciou-se uma nova época de comunicação. Com o telégrafo, mensagens eram codificadas em cadeias de símbolos binários e transmitidas manualmente através de pulsos elétricos.

A tecnologia de computadores, iniciada na década de 50, que permitiu um avanço muito grande no tratamento de informações, associada à tecnologia da comunicação de dados, determinou uma revolução nas formas de comunicação que são utilizadas atualmente.

Redes de computadores são hoje uma realidade e suas utilizações são estendidas às áreas industrial, veicular e de aviação, entre outras.

2.2 Sistemas de Computação

A partir dos anos 60, avanços nos sistemas de computadores possibilitaram que seus usuários tivessem acesso remoto ao computador central, dando início aos sistemas chamados de Teleprocessamento (EMBRATEL, 1994), ou seja, permitia a comunicação entre o elemento central, que alojava todas as aplicações, e os terminais remotos.

Mudanças na caracterização dos sistemas de comunicação ocorreram em meados dos anos 70, quando começaram a sair de um modo centralizado para um processo de distribuição do poder computacional. Seguindo essa tendência, a necessidade de interconexão entre vários sistemas para o uso compartilhado de periféricos tornou-se extremamente importante. Pesquisas de novas arquiteturas foram impulsionadas para encontrar soluções que permitissem que os sistemas se tornassem mais rápidos, mais confiáveis e modulares.

Das várias propostas de arquiteturas apresentadas, a partir dos anos 80, duas são consideradas principais: Sistemas de Multiprocessadores Fortemente Acoplados e Sistemas de Processamento Distribuído (Fracamente Acoplados).

Os Sistemas de Multiprocessadores Fortemente Acoplados são caracterizados por processar instruções de seqüências múltiplas e independentes, sendo compostos por elementos de processamento que compartilham um espaço de memória comum, controlados por um único sistema operacional.

Os Sistemas de Processamento Distribuído, ou Fracamente Acoplados, possuem vários elementos de processamento interconectados, física e logicamente, para executar programas de aplicação de maneira cooperativa e descentralizada.

Assim, várias são as razões para a utilização desses dois sistemas, como seguem:

- Custo/Desempenho: O custo dos microprocessadores tem se tornado cada vez mais baixo com o avanço da tecnologia e sua produção em escala, portanto, os sistemas que utilizam microprocessadores se tornam alto potenciais na relação entre o custo e o desempenho;

- Responsividade: Como um sistema de múltiplos processadores pode ser moldado à aplicação, ele pode apresentar um grande potencial de responsividade e processamento;
- Modularidade: Vários componentes básicos podem ser utilizados para se compor um sistema de computação. Um sistema modular permite sua expansão com a simples inclusão de processadores. Ainda, pode-se adequar o sistema à carga de utilização, ou seja, para um sistema com pouca carga, utilizam-se poucos processadores, e para um sistema com grande volume de carga, utilizam-se mais processadores;
- Confiabilidade: como a redundância é uma necessidade em sistemas confiáveis, uma arquitetura que tenha o máximo de componentes idênticos constitui uma ótima estrutura, sem que ele seja duplicado como um todo;
- Concorrência: para aplicações que exigem alto desempenho de processamento, é necessária a utilização de elementos de processamento concorrentes.

Como principais desvantagens, podemos citar:

- O desenvolvimento de softwares para sistemas de múltiplos processadores pode ser mais complexo e, portanto, mais caro;
- A decomposição do sistema é mais complexa, independente de quem estiver processando, seja pelo software ou pelo programador;
- Um sistema distribuído depende muito do sistema de comunicação;
- A taxa de transmissão deve ser tal que não ultrapasse os limites de tolerância entre os processadores;
- Pode haver reflexão de falha de um processador nos demais que estão acoplados na estrutura de comunicação;
- Em sistemas distribuídos, ocorre uma certa perda de controle, como a dificuldade de gerenciar os recursos, forçar padronizações para o software e gerenciar informações disponíveis.

De maneira resumida, a interconexão de sistemas com poder computacional (sistemas distribuídos) veio a atender duas principais necessidades distintas:

- A construção de sistemas mais confiáveis e com maior desempenho;
- O compartilhamento de recursos.

Alguns autores classificam os sistemas distribuídos de Redes de Computadores para atender somente a primeira necessidade, enquanto outros classificam todos os sistemas como distribuídos e os subdividem entre Máquinas de Arquitetura Distribuída e Redes de Computadores.

Máquinas de Arquitetura Distribuída são aquelas compostas por um número ilimitado, porém finito de processadores autônomos, interconectados para formar um único sistema, no qual o processamento global é realizado pela cooperação de elementos descentralizados.

Uma Rede de Computadores também é formada por um número ilimitado de módulos autônomos de processamento (ECUs), porém finito, interconectados, mas que possuem independência na tarefa de compartilhar informações e recursos.

2.3 Redes de Computadores

Uma Rede de Computadores é formada por um conjunto finito de ECUs que compartilham informações e recursos, interligadas através de um sistema de comunicação, como mostra a figura a seguir:

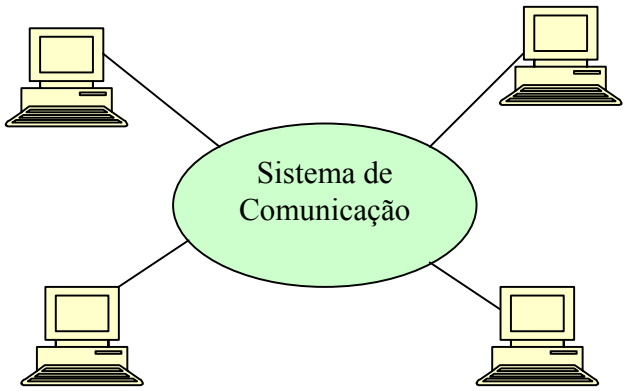


Figura 1 – Rede de Computadores

O sistema de comunicação é um arranjo de topologia que interliga as várias ECUs através de um meio de transmissão (fios, fibra óptica, etc) e de um Protocolo de Comunicação.

Um Protocolo de Comunicação é um conjunto de regras que as ECUs que transmitem e recebem devem seguir para que a correta comunicação se realize. Essas regras são geralmente colocadas em um documento chamado de Protocolo. O protocolo é na verdade um "contrato" entre as ECUs que especifica como a comunicação deve ser feita.

2.4 Estrutura física – Topologias

A topologia de rede descreve a estrutura da conexão física entre as ECUs da rede. A estrutura aplicada será determinante para os custos de implementação, limites de aplicação e parâmetros físicos da rede (ETSCHBERGER, 2001).

2.4.1 Linhas de Comunicação

Há dois tipos de ligações físicas em sistemas de comunicação: ponto a ponto ou multiponto. As ligações do tipo ponto a ponto são caracterizadas pela conexão nas duas

extremidades das ECUs e as ligações do tipo multiponto são caracterizadas pela utilização da mesma conexão física (enlace).

Figura 2 - Tipos de ligação física

A comunicação no enlace possui a seguinte classificação:

- Simplex: o enlace é utilizado somente em um dos sentidos de transmissão;

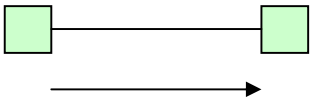


Figura 3 - Comunicação Simplex

- Half-Duplex: o enlace é utilizado nos dois sentidos de transmissão, porém apenas um por vez;

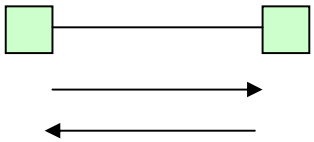


Figura 4 - Comunicação Half-Duplex

- Full-Duplex: o enlace é utilizado nos dois sentidos ao mesmo tempo.

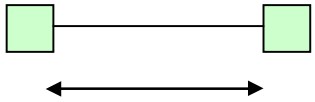


Figura 5 - Comunicação Full-Duplex

As topologias mais importantes são explanadas a seguir (LUPINI, 2004):

Topologia em Estrela - Essa topologia é considerada a mais simples, na qual todos os módulos são conectados no modo ponto-a-ponto.

Como vantagens, temos:

- Cada ECU tem sua própria conexão ao módulo central;
- Integração simples para futuras ECUs;
- Fácil implementação com meio óptico de transmissão (caso seja utilizado).

Como desvantagens, temos:

- O módulo central necessita de tantas interfaces quantas forem as ECUs conectadas a ele;
- A comunicação entre ECUs só é possível via o módulo central e, caso haja problemas nele, não haverá comunicação.

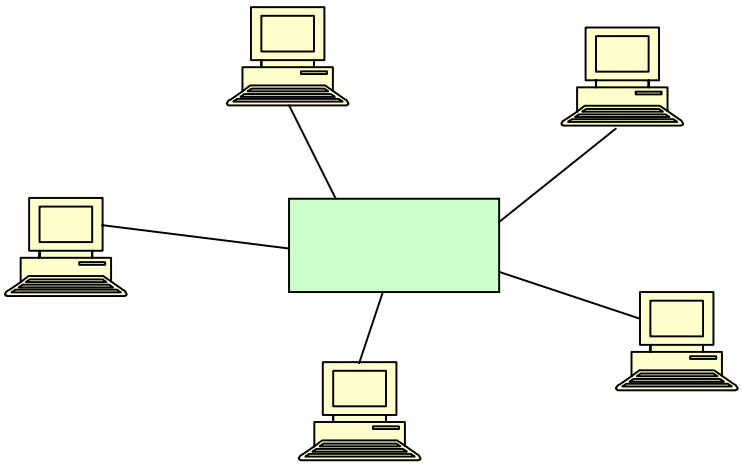


Figura 6 - Topologia em Estrela

Topologia em Barra - Nessa topologia os dados transmitidos por uma ECU estão disponíveis em todas as outras ECUs e possui configuração multiponto.

Existem vários mecanismos para o controle de acesso à barra, que pode ser centralizado ou descentralizado. No caso de controle centralizado, o direito ao acesso é determinado por uma entidade especial da rede. No caso de controle descentralizado, a responsabilidade de acesso é distribuída entre todas as ECUs.

As principais vantagens dessa topologia são:

- Baixo custo de chicotes em aplicações com ECUs geograficamente ordenada em linha;
- Conexão simples de uma ECU;
- Facilidade de aumento de ECUs sem interrupção de operação;
- Em caso de problemas em uma ECU, a comunicação entre as outras ECUs não será afetada.

Seguem as principais desvantagens dessa topologia:

- Comprimento do barramento e número de ECUs limitados em caso da não aplicação de repetidores;
- Limitação do comprimento do tronco (stub) da linha para conexão das ECUs.

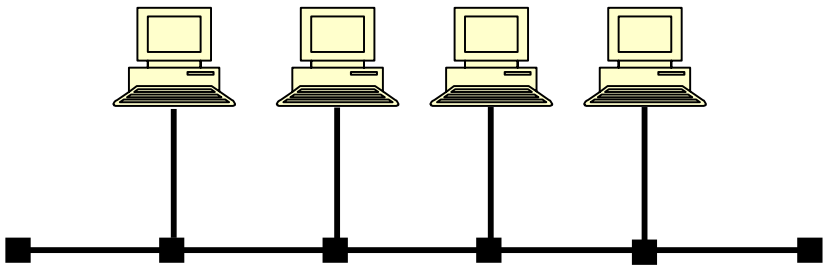


Figura 7 - Topologia em Barra.

Topologia em Árvore - Essa topologia é usada quando ramos arbitrários são possíveis via elementos passivos ou ativos. A grande vantagem dessa topologia comparada à topologia em Barra é a excelente adaptação das ECUs às exigências geográficas no veículo e, conseqüentemente, comprimento pequeno de chicotes e baixo custo.

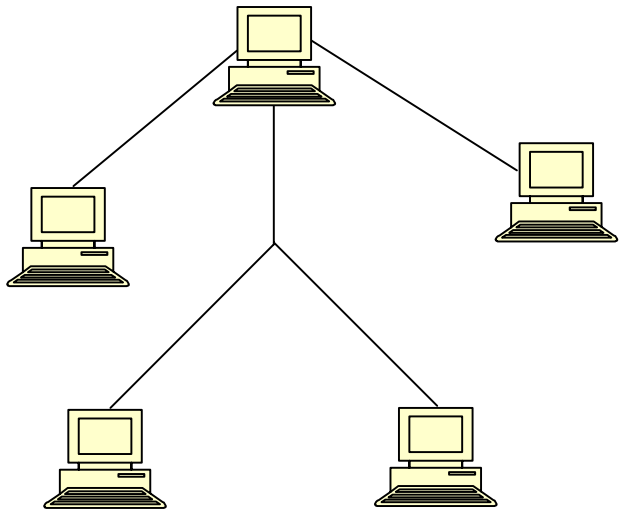


Figura 8 - Topologia em Árvore.

Topologia em Anel - Essa topologia é caracterizada por uma corrente fechada de conexões direcionadas ponto-a-ponto.

Tem como principais vantagens a implementação de ECUs adicionais, já que cada ECU provê regeneração de sinal e excelente uso de meio de transmissão óptico devido à conexão ponto-a-ponto.

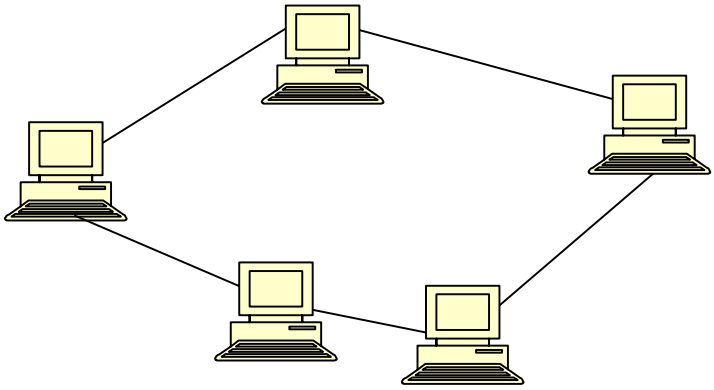


Figura 9 - Topologia em Anel.

2.5 Transmissão de informação

2.5.1 Informação e Sinal

Os processos envolvidos em transmissão de informação de um ponto ao outro são os seguintes:

- A geração da informação a ser transmitida. Pode ser um conjunto de dados armazenados no computador;
- A descrição dessa informação, através de símbolos;
- A codificação desses símbolos para que sejam adequados ao meio físico utilizado;
- A transmissão propriamente dita;
- A decodificação e reprodução desses símbolos;
- A recriação dessa informação pelo receptor – como uma possível degradação de qualidade.

Os sistemas de comunicação utilizam sinais ou ondas eletromagnéticas que trafegam através de meios físicos de transmissão. Sinais são ondas que se propagam através de algum meio físico, seja ele ar ou par de fios. Eles podem possuir amplitude que varia ao longo do tempo correspondendo à codificação da informação transmitida (função do tempo). Informação, no entanto, está associada aos dados que são transmitidos.

2.5.2 Banda Passante

No século XIX, o francês Jean Fourier provou que qualquer sinal periódico, expresso como uma função do tempo *g(t),* com período T_0, pode ser considerado como uma soma (possivelmente infinita) de senos e cossenos de diversas freqüências. A essa soma deu-se o nome de Série de Fourier, representada como:

$$g(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n sen(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \quad \textbf{(1)}$$

Os componentes são chamados harmônicos do sinal com as respectivas amplitudes a_n e b_n e freqüências *nf*.

A representação de um sinal periódico pela Série de Fourier é o equivalente à apresentação dos seus vários harmônicos, ou seja, um sinal periódico pode sempre ser descrito de duas formas equivalentes: através de uma representação no domínio do tempo, onde *g(t)* deve ser escrito em função do tempo, e uma representação no domínio da freqüência, onde o sinal é definido em termos de suas componentes.

Na prática, os sinais encontrados nas transmissões raramente são periódicos. Porém, se considerarmos que os dados a serem transmitidos têm duração limitada e que se repetem de tempos em tempos, cuja representação no tempo durante um período é igual ao sinal original, podemos imaginar que se está analisando um sinal periódico.

Caso essas repetições se tornem afastadas por um período infinito, chega-se às fórmulas que representam a Transformada de Fourier *G(f),* análoga à Série de Fourier para sinais não periódicos, que representa a energia do sinal em cada um de seus componentes:

$$G(f) = \int_{-\infty}^{\infty} g(t).e^{-j2\pi ft} dt \quad \textbf{(2)}$$

A transformada inversa da função G(f) é a própria função g(t), que pode ser obtida da primeira através da seguinte formula:

$$g(t) = \int_{-\infty}^{\infty} G(f).e^{j2\pi ft} df \quad \textbf{(3)}$$

Banda Passante de um sinal é o intervalo de freqüências que compõem esse sinal. A Largura de Banda é o tamanho de sua banda passante, ou seja, a diferença entre a maior e a menor freqüência que compõem o sinal.

Para a transmissão digital, quanto mais harmônicos o sinal transmitido tiver, mais o sinal recebido se aproxima do sinal original (onde todos os harmônicos estão presentes). Da mesma maneira, à medida que a largura de banda do meio se torna mais estreita, a representação do sinal original se torna impossível (com sinal de poucos harmônicos).

Deve-se definir a banda passante, para a transmissão de sinais digitais, como a largura de banda mínima capaz de garantir que a ECU receptora consiga recuperar a informação digital original transmitida.

2.5.3 Multiplexação e Modulação

Geralmente a banda passante utilizada para a transmissão de sinais digitais é menor que a banda passante necessária para o sinal, conforme figura a seguir:

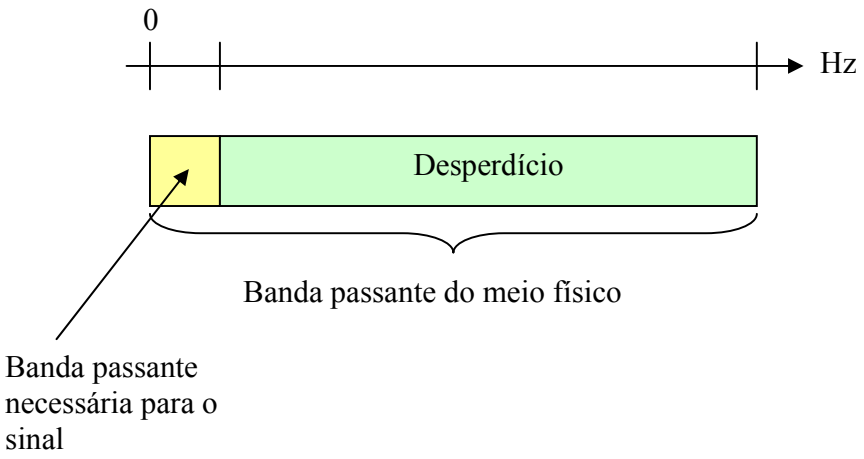


Figura 10 - Meio físico com banda passante maior que a necessária para o sinal.

Como a figura 10 mostra, o sinal está utilizando somente uma pequena parte da banda passante do meio físico, e o restante da banda está inutilizado, representado pelo desperdício.

Para que se possa aproveitar adequadamente a banda passante disponível, com a transmissão de mais de um sinal simultaneamente, uma técnica utilizada é a Multiplexação. Há duas formas básicas de multiplexação: a multiplexação na freqüência (*Frequency Division Multiplexing – FDM*) e a multiplexação no tempo (*Time Division Multiplexing – TDM*).

2.5.3.1 Multiplexação na Freqüência

A técnica de Multiplexação na freqüência consiste em deslocar os sinais que estão ocupando uma mesma banda (técnicas de modulação). Assim, os três sinais podem ser transmitidos no meio físico, cada um deles ocupando uma banda ou canal distinto com tamanho necessário para a sua transmissão.

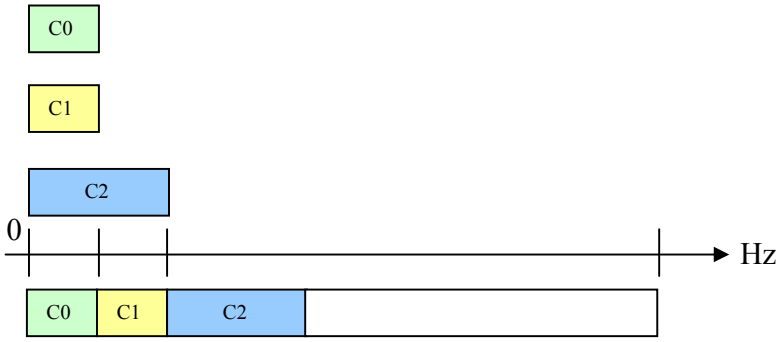


Figura 11 - Multiplexação na freqüência (FDM).

2.5.3.2 Multiplexação no Tempo

Há também a possibilidade de compartilhar um meio físico por várias ECUs pela multiplexação no tempo. A multiplexação por divisão do tempo (*Time Division Multiplexing* – TDM) tem a vantagem de a capacidade (em quantidade de bits por segundo) do meio de transmissão exceder a taxa média de geração de bits das ECUs conectadas à rede. Assim, vários sinais podem ser transmitidos por um único caminho físico, intercalando-se porções de cada sinal no tempo. A multiplexação no tempo pode ser classificada como síncrona e assíncrona.

2.5.3.2.1 TDM Síncrono

No TDM síncrono, o domínio do tempo é dividido em intervalos de tamanho fixo *T* chamados *frames*; cada *frame* é subdividido em *N* subintervalos $\{t_1, ..., t_n\}$ denominados *slots* ou segmentos que formam uma partição dos *frames* que, por sua vez, formam uma partição do tempo infinito.

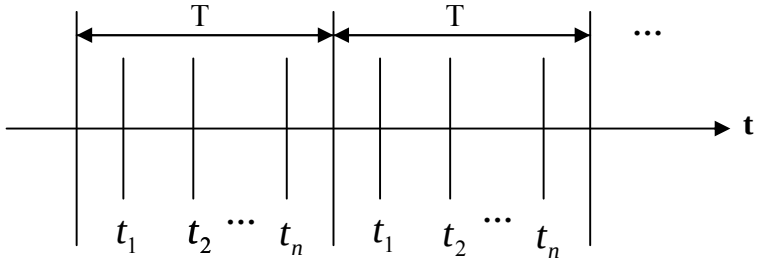


Figura 12 - TDM síncrono.

Assim como canais de freqüência em rede são alocados através do FDM, os canais de tempo são alocados às diferentes ECUs, em redes que utilizam TDM.

Os segmentos de tempo dentro de um *frame* não precisam ter o mesmo tamanho. O segmento no frame determinará a taxa de transmissão máxima efetiva no canal correspondente.

No TDM síncrono com canais chaveados, quando um canal é alocado, a conexão permanece à ECU até que a mesma resolva desfazê-la. Quando uma ECU que alocou um canal não estiver transmitindo, há um desperdício de capacidade do meio físico, já que esse canal não poderá ser utilizado por outra ECU até sua desconexão.

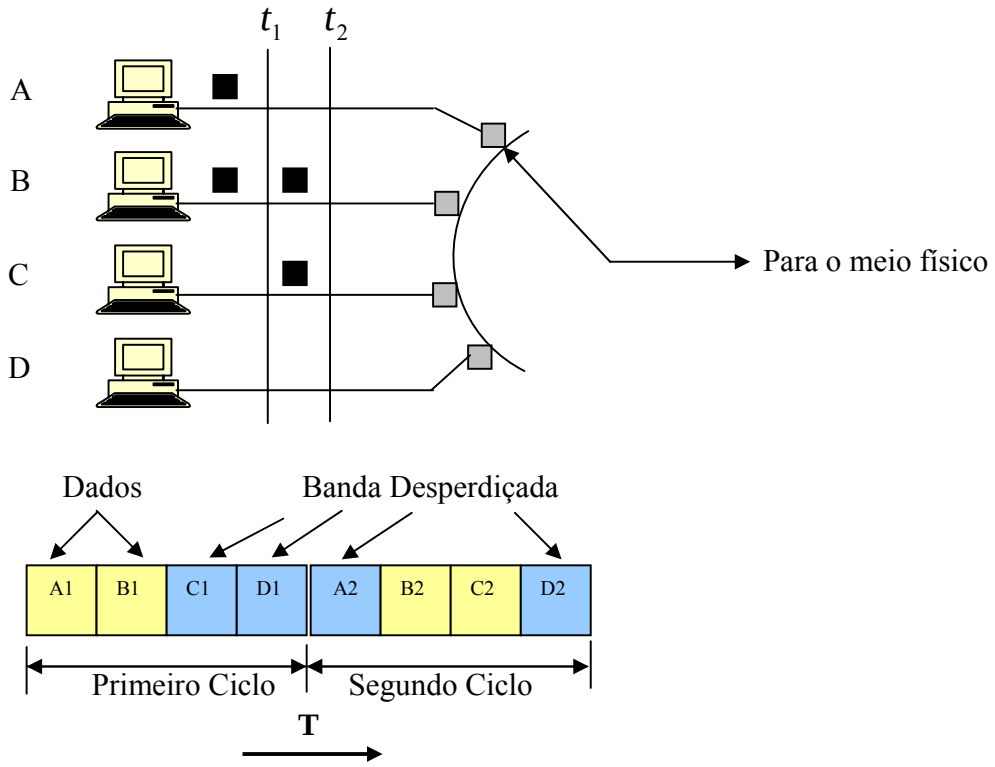


Figura 13 - Desperdício de capacidade em sistemas com TDM síncrono.

2.5.3.2.2 TDM Assíncrono

O TDM assíncrono é uma alternativa ao TDM síncrono para evitar o desperdício de capacidade. Nesse esquema não há alocação de canal nem estabelecimento de conexão. Nenhuma capacidade é desperdiçada, pois o tempo não utilizado está sempre disponível caso alguma ECU queira transmitir. Adesvantagem desta alternativa é que cada unidade de informação transmitida deve conter um cabeçalho com os endereços de origem e destino.

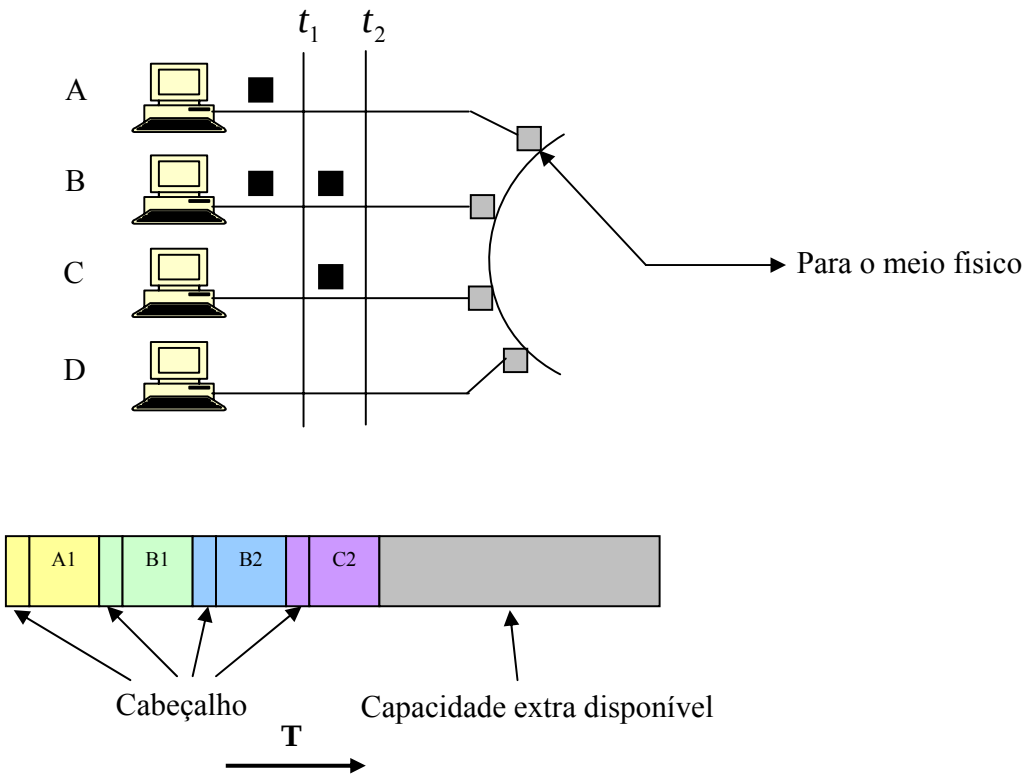


Figura 14 - TDM assíncrono.

2.5.3.3 Codificação e Transmissão de Sinais Digitais

O mais simples e mais empregado sistema de codificação de sinais digitais é conhecido como NRZ (*Non Return to Zero*), no qual dois níveis de tensão ou corrente estão presentes, o zero (0) e o um (1). Nesse esquema, o nível do sinal é mantido constante durante o período de um bit, ou um segmento de tempo, de forma a caracterizar o bit transmitido.

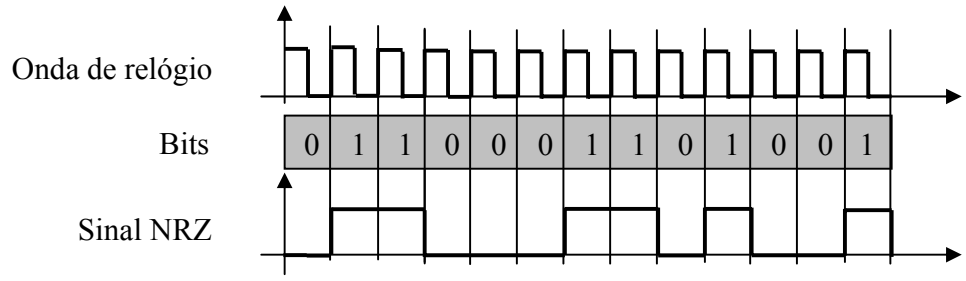


Figura 15 - Codificação NRZ.

2.5.3.3.1 Transmissão Assíncrona

Na transmissão assíncrona, admite-se que a referência de tempo de transmissor e receptor não é única, mas próxima, e tenta-se administrar essa diferença. NRZ é utilizada e a técnica de transmissão é a seguinte:

A freqüência de relógio do receptor é múltipla da freqüência de relógio do transmissor, ou seja, $f_r = nf_t$. Como conseqüência, $T_r = T_t / n$. No inicio da recepção, a amostragem do sinal será feita após $n/2$ pulsos de relógio do receptor. Como o relógio pode apresentar erro de precisão, ocorrerá uma defasagem que não ultrapassará T_t / n. No entanto, essa defasagem será maior à medida que o tempo de transmissão aumenta,

mudando o instante de amostragem do centro do intervalo de sinalização. Assim, a transmissão assíncrona é orientada à transmissão de *caracteres* (pequenas unidades de dados que variam, em geral, entre 5 e 8 bits) de forma a não permitir longas seqüências de bits.

Um mecanismo de detecção de início da recepção é necessário para que não haja erro de fase, que possa ter vindo de caractere anterior. Sendo assim, a transmissão assíncrona é caracterizada pela transmissão de caracteres delimitados por bits, chamados de bit de start e bit de stop, conforme mostra a figura seguinte:

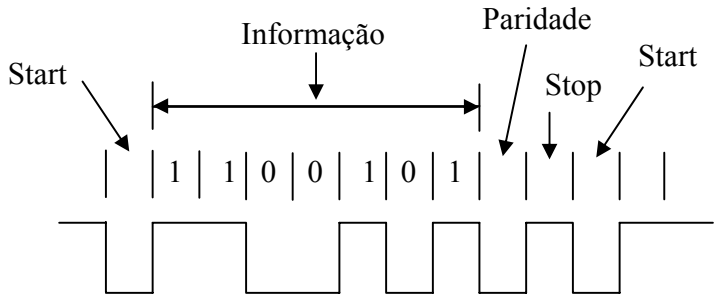


Figura 16 - Caractere na transmissão assíncrona.

O bit de *start* deve sempre ter a transição de 1 para 0 para marcar sua presença e permitir o disparo da contagem no oscilador de recepção (que deve contar $n/2$ pulsos para chegar ao instante de amostragem). No final, um bit de stop é colocado para marcar o fim de um caractere, para permitir que a ECU receptora possa ter acesso ao seu registro de recepção e para garantir a transição para o próximo caractere.

2.5.3.3.2 Transmissão Síncrona

A transmissão síncrona é caracterizada pela existência de uma referência de tempo para as ECUs transmissora e receptora durante uma transmissão. Há duas maneiras de se garantir essa referência:

A primeira consiste em enviar o relógio da ECU transmissora por um canal separado para ser utilizado pela ECU receptora como base para a amostragem correta dos dados transmitidos. Porém, essa técnica incorre na necessidade de dois canais de transmissão, um para dados e outro para o relógio, assim como todo o circuito em duplicidade, aumentando o custo da transmissão. Além disso, as diferenças que os dois relógios possam apresentar determinam a impossibilidade prática do sistema, devido aos retardos de propagação.

A segunda maneira é enviar dados e informação de sincronismo pelo mesmo canal, de maneira que a ECU receptora possa recuperar o sinal enviado. A ECU receptora tem então que separar os dados e a informação de sincronismo e, a partir do relógio recuperado, realizar a amostragem dos dados.

2.5.3.4 Técnicas de Detecção de Erros

Como se torna quase impossível a eliminação total de fenômenos que possam gerar erros, os sistemas de comunicação devem ser contemplados com técnicas que permitam a detecção de possíveis erros para que se possa recuperar o sinal ou informação perdida.

A detecção de erros é imprescindível nos sistemas de comunicação para que, ao perceber um erro, as devidas providências sejam tomadas.

Todos os métodos de detecção de erros são baseados na inserção de bits extras na informação transmitida, que são redundantes. Quando se transmite uma informação, um algoritmo calcula os bits extras e os insere no quadro de mensagem. Ao receber esse quadro, a ECU receptora, reconhecendo o algoritmo de transmissão, recalcula os bits de redundância e os compara com os bits recebidos. Caso sejam diferentes, um erro é detectado.

2.5.3.4.1 Paridade

Essa técnica de detecção de erros consiste na inserção de um bit de paridade no final de cada caractere de um quadro da mensagem. O valor desse bit é determinado de forma que todos os caracteres de um quadro tenham um número par ou um número ímpar de bits. Como exemplo, se uma ECU deseja transmitir o caractere "1110001", com paridade ímpar, ela deverá acrescentar um bit "1" ao caractere, de forma que o resultado final deverá ter um número ímpar de bits "1" (nesse caso, 5 bits "1"). Na recepção, a ECU deverá encontrar um número ímpar de bits "1". Caso não seja encontrado, um erro de Paridade ocorrerá.

2.5.3.4.2 CRC

Nesse mecanismo de detecção de erro, um quadro de mensagem de *k* bits é representado por um polinômio em X, de ordem *k -1,* sendo que o coeficiente do termo X^i é dado pelo (i + 1)-ésimo bit da seqüência de *k* bits. Por exemplo, se um quadro de mensagem for 10110001, ele será representado pelo polinômio $X^7 + X^5 + X^4 + 1$.

Ao transmitir, a ECU divide o polinômio em aritmética módulo 2, por um polinômio gerador de ordem *n*, tendo como resultado um quociente e um resto de ordem *n – 1.* A ECU gera então os *k* bits originais, seguidos dos bits correspondentes ao polinômio obtido como resto da divisão (chamado de *Frame Check Sequence* – FCS).

Quando a ECU receptora recebe um quadro de mensagem, um processo semelhante é executado. Com os *k* primeiros bits recebidos, a ECU receptora efetua a divisão do polinômio correspondente, de ordem *k -1*, pelo mesmo polinômio gerador usado pela ECU transmissora. O resto da divisão é comparado com os *n* últimos bits recebidos do quadro. Se os bits forem iguais, a ECU receptora assume que não houve erros no quadro. Caso algum bit seja contrário, um erro de CRC será gerado.

Seguem alguns polinômios geradores mais comumente utilizados:

$$\boldsymbol{CRC-12} = X^{12} + X^{11} + X^{3} + X^{2} + X + 1 \quad \textbf{(4)}$$

$$\boldsymbol{CRC-16} = X^{16} + X^{15} + X^{2} + 1 \quad \textbf{(5)}$$

2.6 Arquiteturas de Redes de Computadores

2.6.1 Introdução

Dos princípios utilizados em projetos de redes, o que mais de destaca é a estrutura de níveis hierárquicos, ou camadas, cada um utilizando os serviços e funções dos níveis inferiores.

Cada camada (ou nível) deve ser implementada por hardware ou software, que se comunica com o processo correspondente na outra máquina. Os dados transferidos de uma máquina para outra "descem", até o nível físico, para então "subirem" ao mesmo nível (destino). A arquitetura de rede é formada por níveis, protocolos e interfaces (limites entre níveis). Os níveis fornecem aos níveis superiores seus serviços, utilizando suas funções e serviços disponíveis nos níveis inferiores.

Essa é a maneira mais eficiente de se estruturar uma rede. Porém, como cada fabricante de computador criava sua própria estrutura de rede, houve a necessidade de se criar um padrão mundial para definir uma arquitetura única e que fosse aberta e pública. Dessa maneira, a International Organization for Standadization (ISO), definiu o modelo denominado *Reference Model for Open Systems Interconnection* (OSI), que propõe uma estrutura de rede com sete níveis como referência para a arquitetura dos protocolos de redes de computadores.

2.6.2 O Modelo OSI da ISO

O objetivo do padrão internacional 7498, denominado *Open Systems Interconnection Reference Model* (RM-OSI), é fornecer uma base comum para o desenvolvimento coordenado de padrões para a interconexão de sistemas. É também a finalidade do RM-OSI permitir a manutenção da consistência entre os padrões de comunicação de computadores. Como o RM-OSI não especifica os serviços e protocolos de cada camada, mas somente o que cada uma deve fazer, ele não define a arquitetura de uma rede. Pode ocorrer que dois sistemas que seguem o RM-OSI não consigam se comunicar, porque as opções adotadas são incompatíveis. Sendo assim, eles devem escolher opções compatíveis de serviços/protocolos para todas as camadas. A partir dessa necessidade, a ISO elaborou o conceito Perfis Funcionais.

O modelo OSI possui sete camadas de protocolos, que são descritos a seguir (ISO, 2003) :

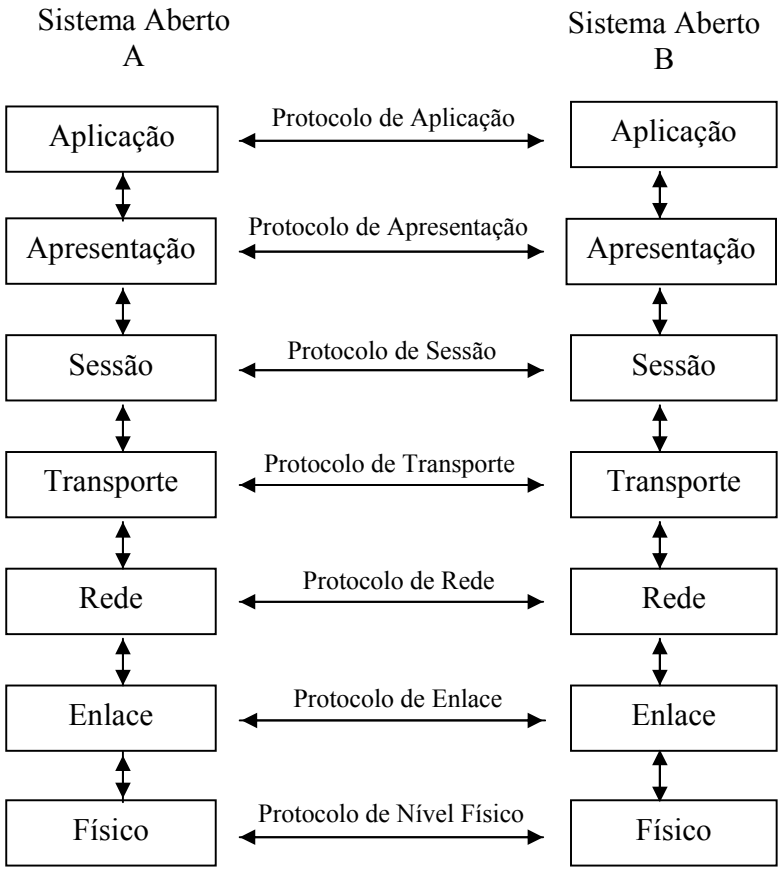


Figura 17 - Níveis do Modelo OSI.

2.6.2.1 Nível Físico

O nível físico especifica as características elétricas e mecânicas de um sistema de comunicação, tais como, tipos de cabos e conectores, além das propriedades elétricas dos sinais, como codificação dos bits, sincronização e sinais de controle. O protocolo de nível físico dedica-se à transmissão de uma cadeia de bits.

2.6.2.2 Nível de Enlace de Dados

Sua principal função é a construção do formato da mensagem. É a camada que controla e protege a comunicação no nível de quadro de mensagem. O formato de mensagem contém informações adicionais que servem para realizar checagem de erros e confirmação da mensagem pela ECU receptora, entre outras.

Outra função principal é o controle de acesso ao meio (barramento). Quando mais de uma ECU tem a intenção de transmitir uma mensagem ao mesmo tempo, a camada de enlace reconhece o caso e resolve o conflito de maneira que uma mensagem não seja perdida.

2.6.2.3 Nível de Rede

Esse nível controla a comunicação lógica através de um sistema de comunicação físico, na qual conecta duas aplicações de comunicação. Ela é responsável pelo transporte apropriado de cada mensagem, sendo que a camada de transporte (próxima camada) será a responsável por ordenar as mensagens recebidas.

Existem duas filosofias quanto ao serviço oferecido pelo nível de rede: datagrama (serviço não-orientado à conexão) e circuito virtual (circuito orientado à conexão).

No datagrama, um pacote não possui nenhuma relação de passado nem de futuro com qualquer outro pacote, devendo carregar seu endereço de destino. No circuito virtual, o transmissor deve enviar um pacote para estabelecer conexão.

2.6.2.4 Nível de Transporte

É a camada responsável pelo controle do fluxo de mensagens empacotadas. Por exemplo, na troca de informação que contém dados de 200 bytes, considerando que um

formato de mensagem do CAN é capaz de transportar somente 8 bytes, a camada de transporte tem a função de “quebrar” tal mensagem antes de enviá-la.

O nível de transporte garante uma comunicação fim-a-fim, ou seja, ele vai isolar dos níveis superiores a parte da transmissão da rede e vai permitir a comunicação entre os níveis de transporte das maquinas de origem e destino.

2.6.2.5 Nível de Sessão

Esse nível fornece a estrutura de controle de comunicação entre aplicações. Ela estabelece, gerencia e termina conexões (sessões) entre aplicações.

Em algumas aplicações em que a troca de informação é half-duplex, circuito que permite a transmissão nos dois sentidos, o nível de sessão utiliza o conceito de *token*, no qual somente o proprietário do *token* pode transmitir seus dados.

2.6.2.6 Nível de Apresentação

É o nível responsável por realizar transformações adequadas nos dados, antes de enviá-los ao nível de sessão.

O nível de apresentação deve conhecer tanto a sintaxe de seu sistema local como a do sistema de transferência. Os serviços oferecidos por esse nível são: transformação e formatação de dados, seleção de sintaxes e estabelecimento e manutenção de conexões de apresentação.

2.6.2.7 Nível de Aplicação

O nível de aplicação oferece meios de utilização dos processos de aplicação para utilização do ambiente de comunicação OSI. Aqui são definidas funções de gerenciamento e mecanismos genéricos para elaboração de aplicações distribuídas.

Além dos elementos genéricos, existem os elementos de serviços específicos de cada protocolo de aplicação. Como exemplo, temos o FTAM (*File Transfer, Access and Management*), o DS (*Directory Service*) e o MHS (*Message Handling System*).

2.6.2.8 Transmissão de dados no Modelo OSI

Quando um usuário no sistema A envia uma mensagem para o usuário no sistema B, no ambiente OSI, o seguinte processo tem início:

Os dados são primeiramente entregues para uma entidade do nível de aplicação, denominada SDU (*Service Data Unit* ≡ Unidade de Dados do Serviço), tornando-se a SDU do nível de aplicação. A SDU reúne aos dados recebidos um cabeçalho chamado de PCI (*Protocol Control Information* ≡ Informação de Controle do Protocolo). O resultado dessa união é chamado de PDU (*Protocol Data Unit* ≡ Unidade de Dados do Protocolo), que é a unidade de informação trocada pelas unidades pares. A PDU é, em seguida, transferida à camada de apresentação.

Recebida a PDU da camada de apresentação, a camada de sessão trata essa unidade da mesma forma, ou seja, adiciona seu cabeçalho, compondo sua PDU. Assim, da mesma maneira, a unidade gerada na camada de apresentação é tratada pelas camadas inferiores até chegar ao nível de enlace, que acrescenta um cabeçalho e um fecho. A PDU desse nível é chamada de quadro, que é transmitido pelo meio físico, através do meio de transmissão.

Quando o quadro chega à outra máquina, o processo é inverso, ou seja, cada camada que recebe o quadro retira o cabeçalho e o fecho que foi acrescentado por sua

entidade par na origem até a camada de aplicação, que recebe os dados enviados pelo sistema A.

2.7 Protocolos de Acesso ao Meio

Os protocolos dos níveis inferiores especificam um conjunto de regras para acesso ao meio físico, que é uma das funções do nível de ligação do modelo OSI.

Os métodos de acesso podem ser divididos em dois grandes grupos: os métodos baseados em contenção e os de acesso ordenado sem contenção.

2.7.1 Acesso Baseado em Contenção

Em uma rede baseada em contenção, não existe ordem de acesso e mais de uma ECU pode transmitir ao mesmo tempo, provocando colisão e, possivelmente, a perda da mensagem. A capacidade de detecção e retransmissão da mensagem vai depender do mecanismo adotado pela ECU.

2.7.1.1 CSMA/CD

No método chamado de CSMA (*Carrier Sense Multiple Access*), quando uma ECU deseja transmitir uma informação, ela "ouve" o meio antes para saber se existe alguma transmissão a caminho. Caso na escuta não haja nenhuma ECU controlando o meio, a transmissão pode ser iniciada. Caso contrário, ela espera por um tempo e tenta transmitir novamente. A ocorrência de colisão acontece caso duas ECUs tentem enviar mensagens no mesmo instante.

O mecanismo CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) realiza a detecção durante a transmissão. Ao tentar transmitir, uma ECU escuta o meio e, ao detectar uma colisão, aborta a transmissão.

Duas técnicas de retransmissão são utilizadas: espera exponencial truncada (*truncated exponential back off*) e retransmissão ordenada (*orderly back off*).

Na primeira técnica, a ECU, ao detectar uma colisão, espera por um tempo aleatório, que vai de zero a um limite superior, de forma a minimizar a probabilidade de colisões repetidas. Tal limite é dobrado a cada colisão sucessiva. Porém, há um limite máximo para um número de retransmissões. Caso o intervalo de retransmissões se torne muito grande e, após algumas retransmissões as colisões continuem, a transmissão é abortada.

Na técnica de retransmissão ordenada, após uma colisão, as ECUs só podem voltar a transmitir após um intervalo de tempo pré-alocado a cada ECU.

No mecanismo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), a rede entra em um modo em que as ECUs só podem iniciar uma transmissão em intervalos de tempos pré-determinados, mesmo tendo ocorrido uma colisão.

2.7.2 Acesso Ordenado sem Contenção

Vários protocolos são baseados em acesso ordenado sem contenção para evitar o problema de colisão. Um dos mais importantes é o acesso ordenado por *Slot*.

2.7.2.1 Acesso por *Slot*

Inicialmente desenvolvido para ser utilizado em topologia tipo anel, esse método consiste em dividir o espaço de comunicação em um número inteiro de pequenos segmentos (*slots*), dentro dos quais as mensagens podem ser armazenadas. Cada segmento contém um bit que indica se está vazio ou preenchido. Quando uma ECU quer transmitir, ela deve aguardar um *slot* vazio e preenchê-lo com uma mensagem.

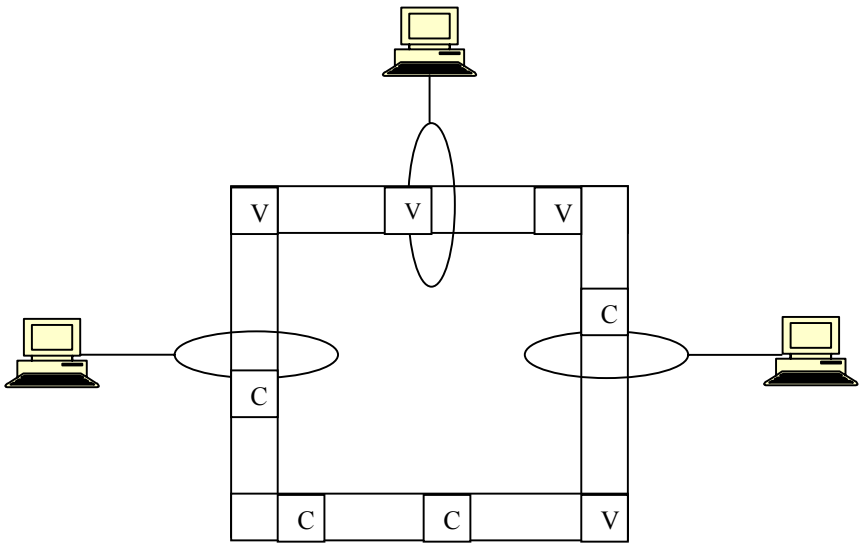


Figura 18 - Anel de *Slots* (ou segmentado).

Como toda ECU sabe quantos *slots* a rede possui, uma ECU detecta o *slot* que transmitiu e retorna-o ao estado vazio.

2.8 O CAN – Controller Area Network

O CAN é um canal de comunicação multiplexado no qual dados são transferidos ao longo de ECUs em sistemas distribuídos (BOSCH, 2005).

O *Controller Area Network* foi oficialmente apresentado pela Bosch em 1986 em Detroit em uma conferência da SAE, e foi originalmente criado para aplicações automotivas (ETSCHBERGER, 2001), sendo montado pela primeira vez em um veículo Mercedes-Benz Classe S (LEEN, G., HEFFERNAN, D., 2001). Depois, várias áreas industriais começaram a aplicar o CAN por causa da sua alta confiabilidade e baixo custo de implementação e, também, devido à disponibilidade de uma infinidade de chips que estão no mercado.

A tecnologia moderna de automação é caracterizada por uma crescente descentralização de processamento de dados. De um lado, o uso de barramentos, ao invés de uma instalação convencional, garante maior flexibilidade em relação a modificações e atualizações. Do outro lado, oferece potencial considerável de redução de custos de instalação.

A conexão entre ECUs via barramento tem prevalecido nas áreas de sistemas eletrônicos internos de veículos. O *Controller Area Network* (CAN) tem assumido um papel extremamente importante nessa referência.

A figura 19 ilustra um exemplo de arquitetura de rede de comunicação baseada no CAN:

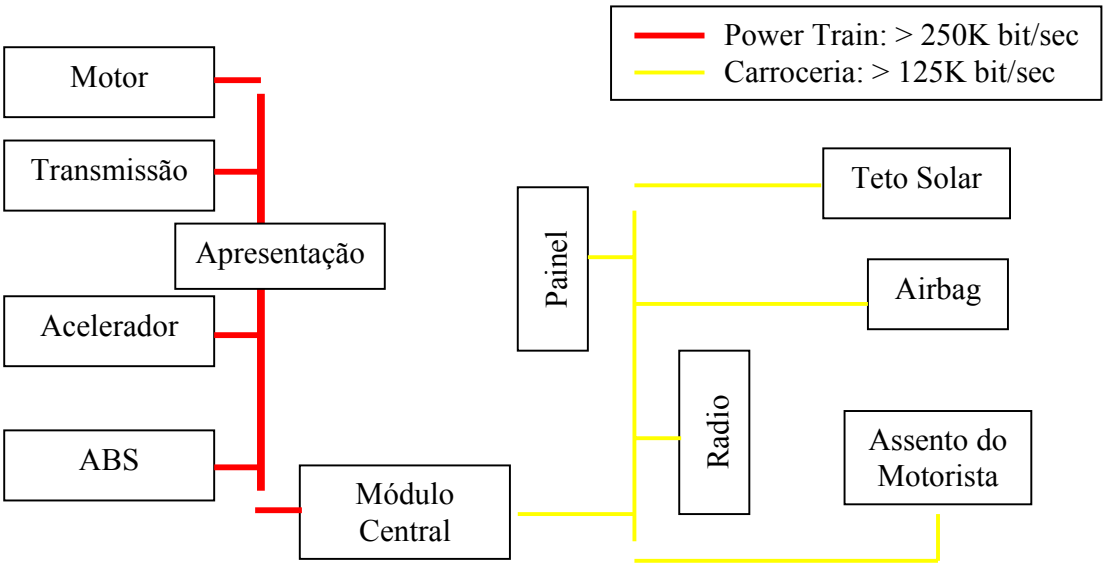


Figura 19 - Exemplo de um sistema multiplexado com diferentes velocidades de transmissão.

Note que existe a possibilidade de interligação de sistemas que exigem diferentes velocidades de transmissão.

2.8.1 Principais Características do CAN

As principais características do CAN são (KOPETZ, 1997); (LAURENZ, 1997):

- Acesso à rede baseado em conceito Multi-mestre - todos os módulos podem transmitir uma mensagem assim que o barramento estiver livre e vários módulos podem solicitar à rede simultaneamente. No momento da transmissão simultânea de vários módulos, o que tiver a mais alta prioridade momentânea recebe o direito de acesso à rede;
- Transmissão do tipo Multicast, ou transmissão para todos os módulos ao mesmo tempo. Um filtro é aplicado para selecionar as informações importantes para cada módulo;
- Arbitragem do barramento sem perda - Filosofia de acesso ao meio CSMA/CD (Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration) na qual é feita uma análise da prioridade de transmissão. Aquele que tiver prioridade maior continuará enviando a sua mensagem sem destruição.
- Protocolo de mensagem orientada – O protocolo CAN é baseado na identificação de uma mensagem através de identificadores de mensagem, e não no endereçamento da mensagem. Baseadas nos identificadores, as ECUs verificam se a mensagem é relevante ou não, se interessa ou não, através de filtros de aceitação integrados nos controladores.
- Solicitação remota de mensagem – Uma ECU pode solicitar uma mensagem a uma outra ECU através do envio de mensagem específica (sem dados) chamada de Remote Request Frame.
- Confirmação de recebimento de uma mensagem – O CAN fornece, no formato da mensagem, um campo de confirmação que assegura que uma mensagem foi recebida sem erros.
- Codificação do bit – O CAN utiliza a codificação do bit chamada de NRZ, na qual cada bit é transmitido por um valor fixo de tensão.

- Sistemas flexíveis – ECUs podem ser adicionadas à rede de comunicação sem a necessidade de mudança em software e hardware, desde que não sejam transmissoras ou necessitem de mensagens adicionais transmitidas.
- Taxa de transmissão programável entre 5Kbps a 1Mbps.

3. ESCALONAMENTO

Quando um computador é multiprogramado, vários processos podem competir pela CPU ao mesmo tempo, quando tais processos estão prontos. A parte do sistema operacional que faz a escolha é chamada de escalonador e utiliza um algoritmo chamado de escalonamento (TANENBAUM, 2005).

3.1 Processos

Atualmente todos os computadores conseguem executar várias tarefas ao mesmo tempo como, por exemplo, ler um disco e mostrar um texto na tela. Em um sistema multiprogramado, a CPU salta de programa em programa, executando cada um por alguns milissegundos.

3.1.1 Modelo de Processos

Nesse modelo, todos os programas que são executados em um computador são organizados em vários processos seqüenciais, ou somente em processos. Um processo é apenas um programa em execução acompanhado dos valores atuais do contador de programa, dos registradores e das variáveis. Na prática, a CPU troca de um processo para o outro, num mecanismo chamado de multiprogramação. A idéia principal de um processo é constituir uma atividade. Ele possui programa, entrada, saída e um estado. Um único processador pode ser compartilhado entre vários processos, com algum algoritmo de escalonamento utilizado para determinar quando parar o trabalho de um processo e iniciar outro.

3.1.2 Hierarquia de Processos

Quando existem processos criados a partir de outros processos, o processo pai e o processo filho continuam, de uma certa maneira, associados. Há ainda processos filhos que criam outros processos, formando uma hierarquia de processos.

Como exemplo de hierarquia de processos, pode-se citar o sistema operacional Unix, no qual um processo, seus filhos e descendentes formam um grupo de processos. Ao ligar o computador, um processo chamado *init* está presente na imagem de carga do sistema. Quando começa a executar, ele lê um arquivo que diz quantos terminais estão presentes. Então, ele entra no processo de cada terminal, que espera por alguma conexão do usuário. Em caso de conexão, um interpretador de comandos atua para aceitar comandos de usuários. Por sua vez, outros processos podem ser criados a partir desses comandos.

3.1.3 Estados de Processos

Mesmo que cada processo seja independente, há, muitas vezes, a necessidade de que os processos se comuniquem e interajam com outros. Um processo pode gerar uma saída para um outro processo, que o utiliza como entrada.

No entanto, deve-se levar em conta as velocidades de cada processo para que não haja atraso ou adiantamento nos processos, o que pode acarretar bloqueio do processo. Isso ocorre porque ele não pode prosseguir, muitas vezes, porque ele está esperando por uma entrada que ainda não está disponível. Pode ocorrer, também, que um processo seja bloqueado, pois, mesmo estando pronto e disponível, o sistema operacional decidiu alocar a CPU para outro processo por algum tempo. Na figura a seguir há um diagrama de estados mostrando os três estados de um processo:

Em execução (utilização da CPU);
Pronto (executável);
Bloqueado (não executável).

1. O processo bloqueia aguardando uma entrada
2. O escalonador seleciona outro processo
3. O escalonador seleciona esse processo
4. A entrada torna-se disponível

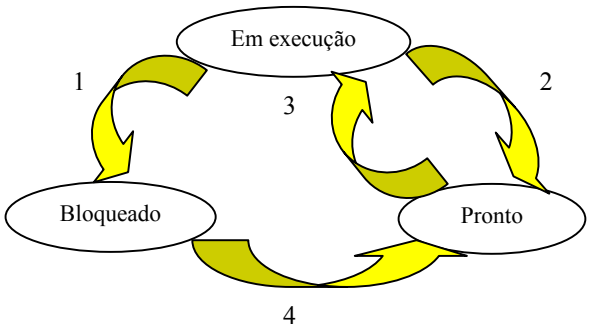


Figura 20 - Estados de um processo.

Conforme mostrado na figura 3.1, há quatro transições possíveis entre três estados de um processo. A transição 1 acontece ao descobrir que o processo não pode prosseguir. As transições 2 e 3 são causadas pelo escalonador de processos, sem que ele saiba disso. A transição 2 ocorre quando o processo que está sendo executado já teve seu tempo suficiente de processamento e que deve passar a executar outro processo. A transição 3 acontece quando todos os processos foram executados e é hora de voltar ao processo 1. Por fim, a transição 4 ocorre quando um evento externo, pelo qual um processo estava aguardando, acontece.

Desta maneira, dá-se o início ao modelo abaixo, onde o escalonador ocupa o nível mais baixo do sistema operacional, com os vários processos acima dele.

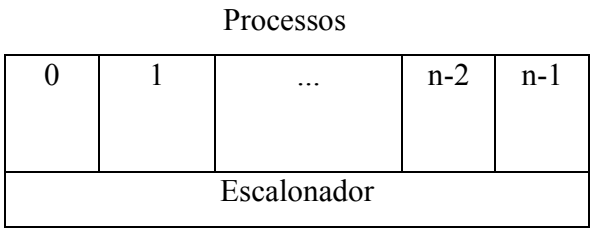


Figura 21 - Modelo de escalonamento dos processos.

3.2 *Threads*

Em sistemas operacionais tradicionais, cada processo tem um espaço de endereçamento e um único fluxo (*thread*) de controle. Na verdade, isso é quase uma definição de processo. No entanto, há situações em que é desejável a utilização de vários *threads* de controle no mesmo espaço de endereçamento, executados quase paralelamente, como se fossem processos separados (exceto para espaços com endereços compartilhados).

3.2.1 Modelo de *Thread*

No modelo de Processo, há dois conceitos separados: agrupamento de recursos e execução. No caso dos *Threads*, esses conceitos são separados. No conceito de processo, pode-se encará-lo como uma forma de agrupar recursos, de modo a facilitar o gerenciamento desses recursos. O *thread* tem um contador de programa que mantém o controle da próxima instrução que ele deve executar. Possui registradores que contêm suas variáveis atuais de trabalho. Possui, também, pilha de histórico de execuções, com estrutura para cada procedimento chamado, mas ainda não retornado. Em suma, o *thread* é a entidade escalonada para a execução sobre a CPU.

Os *threads* acrescentam ao modelo de processo a possibilidade de execução múltipla no mesmo ambiente do processo com independência uma da outra. A execução de múltiplos processos sendo executados em um computador é análoga a ter múltiplos *threads* executados paralelamente, em um processo.

A figura a seguir apresenta três threads, porém de duas maneiras diferentes: com três processos diferentes 3.3 (a) e com um mesmo processo 3.3 (b).

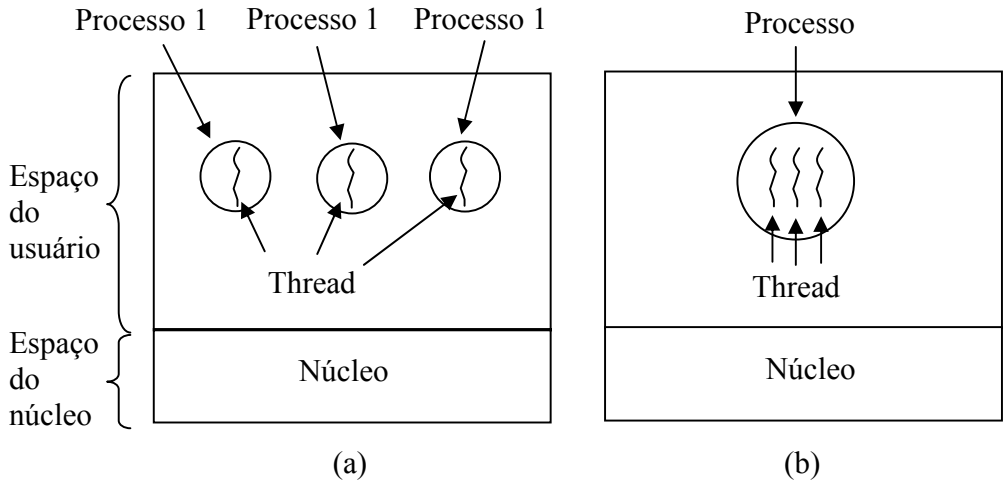


Figura 22 - (a) Três processos, com *thread* individual. (b) Um processo com três *threads*.

3.2.2 O uso de *Threads*

A razão principal da utilização de *threads* é que, em muitas aplicações, podem ocorrer múltiplas atividades ao mesmo tempo, e algumas podem ser bloqueadas de tempos em tempos. Para essas aplicações, os *threads* são essenciais, devido à capacidade das entidades paralelas compartilharem um espaço de endereçamento e todos os seus dados entre elas mesmas.

Outra razão para a utilização é que são mais fáceis de criar e destruir do que os processos, pois não há recursos associados a eles. Além do ganho de desempenho quando o número de entradas e saídas e a quantidade de computação são grandes, eles permitem que essas atividades se sobreponham, aumentando a velocidade da aplicação.

3.3 Comunicação Interprocessos

A necessidade da comunicação entre processos é muito freqüente, porém, deve ocorrer de maneira estruturada e sem interrupções. Há, principalmente, três maneiras de comunicação entre processos:

- Como um processo passa informação para o outro;
- Garantir que um processo não invada o outro em situações críticas;
- Seqüência adequada quando existirem dependências.

3.3.1 Condição de disputa

Em alguns sistemas operacionais, processos que trabalham juntos podem compartilhar algum armazenamento comum, no qual cada processo seja capaz de ler e escrever. Não importa o local da memória compartilhada; pode estar na memória principal ou em um arquivo compartilhado.

Um exemplo comum de como há a comunicação interprocessos é o de um *spool* de impressão, conforme a figura que segue.

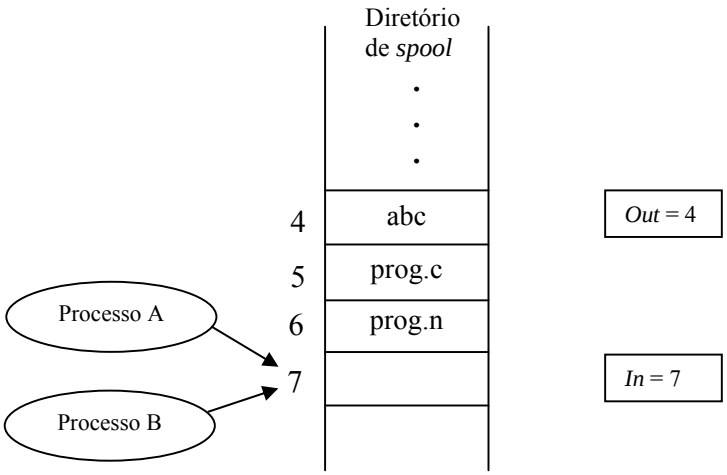


Figura 23 - Dois processos tentando acesso simultaneamente.

Há um número de vagas numeradas para arquivos de impressão a partir de 0 (0, 1, 2, etc) e dois arquivos compartilhados, *out* e *in*, que apontam para o próximo arquivo a ser impresso e para a próxima vaga do diretório, respectivamente. Em um dado momento, as vagas 0 e 6 estão vazias (arquivos já foram impressos) e as vagas de 4 a 6 estão preenchidas (fila de impressão). Quase que simultaneamente, os processos A e B decidem colocar um arquivo na fila de impressão.

O processo A lê *in* e armazena o valor 7. Em seguida, ocorre uma interrupção do relógio, por qualquer motivo, e a CPU decide que o processo A já executou o suficiente e, então, muda para o processo B. Este, da mesma maneira, lê *in* e armazena o valor 7. Ambos agora têm a mesma visão que a próxima vaga é a 7. O processo B executa o mesmo procedimento e atualiza *in* como 8. Ao final, o processo A reinicia de onde parou e armazena o 7, apagando o que o processo B armazenou. Ele calcula a próxima vaga, que é 8, e põe 8 em *in*. Dessa maneira, o processo B nunca receberá nenhuma saída, porque o diretório de impressão não notou nada de errado.

Portanto, são chamadas de condições de disputa, situações como a acima descrita, em que dois processos tentam ler ou escrever algum dado compartilhado e o resultado depende de quem e quando executa precisamente.

3.3.2 Regiões críticas

Para evitar as situações de disputa, é necessário criar mecanismos que impeçam que os processos utilizem arquivos ou variáveis compartilhadas que já estiverem em uso por outro processo.

Durante uma parte do tempo, um processo está ocupado executando outras tarefas, mas, em algum momento, ele necessita acessar uma memória ou arquivo compartilhado. Dessa maneira, a parte do programa em que há acesso à memória compartilhada é chamada de região crítica ou seção crítica.

É necessário satisfazer quatro condições para evitar condições de disputa:

- Nunca dois processos devem estar simultaneamente na região crítica;
- Nada pode se afirmar sobre a velocidade ou sobre o número de CPUs;
- Nenhum processo executando fora da região crítica pode bloquear outros processos;
- Nenhum processo deve esperar eternamente para entrar na região crítica.

A solução ideal seria a descrita na figura a seguir.

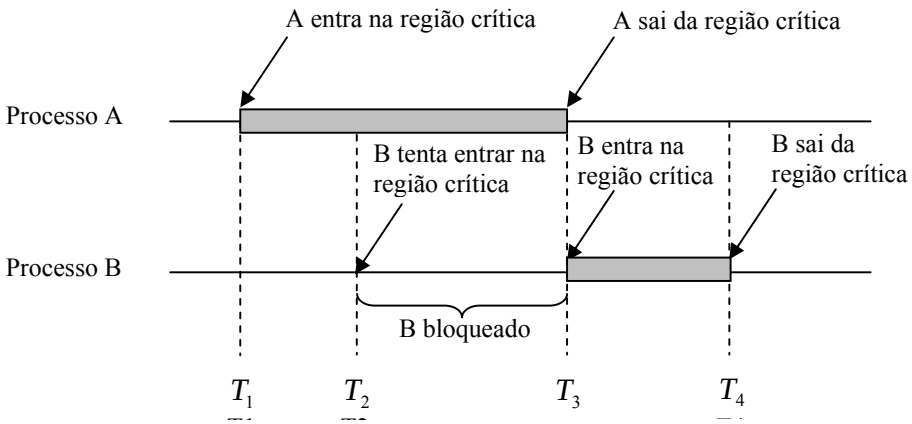


Figura 24 - Exclusão mútua utilizando regiões críticas.

3.3.3 Exclusão mútua com espera ociosa

Segue uma explanação de algumas técnicas de exclusão mútua para evitar que, enquanto um processo estiver ocupando a memória compartilhada em sua região crítica, outro processo venha invadir.

3.3.3.1 Desabilitando interrupções

Essa solução dá aos processos poder de desabilitar todas as interrupções ao entrar na região crítica e desabilitá-las ao sair dela. O grande problema é que podem ocorrer situações em que um processo desabilite as interrupções e não as habilite mais, podendo ocorrer o fim do sistema.

3.3.3.2 Variáveis de impedimento

É uma solução de software, na qual uma variável única compartilhada (*lock*) permite ou não a entrada na região crítica. Se ela estiver com valor 0 inicialmente, o processo altera para valor 1 e entra na região crítica. Se ela estiver com valor 1, o processo aguarda até que ela se torne 1.

Porém, o mesmo problema pode acontecer para o diretório de *spool* de impressão, ou seja, pode acontecer que um processo altere o valor da variável *lock* antes de outro processo e que o primeiro processo nunca consiga entrar na região crítica.

3.3.3.3 Alternância obrigatória

A terceira técnica para tratar o problema de exclusão mútua é exemplificada pelo fragmento de programa, mostrado abaixo:

```
while (TRUE) {                          while (TRUE) {
    while (turn !=0)      /*laço */;         while (turn !=1)      /*laço */;
    critical_region( );                     critical_region( );
    turn = 1;                               turn = 0;
    noncritical_region( );                  noncritical_region( );
          (a)                                     (b)
```

Figura 25 - Exemplo de alternância obrigatória de dois processos.

Nesse exemplo, a variável vez (*turn*), inicialmente com valor 0, serve para controlar a vez de quem entra na região crítica e verifica ou atualiza a memória compartilhada. O processo 0 verifica a variável *turn,* encontra o valor 0 e entra na região crítica. O processo 1 faz a mesma coisa, porém entra em laço (testa continuamente) para esperar que a variável *turn* se torne 1. Ao terminar, o processo 0 põe a variável em 1, permitindo que o processo 1 entre na região crítica.

Caso o processo 1 termine rapidamente sua execução, ele coloca a variável *turn* em 0, deixando os dois processos fora da região crítica. Agora, o processo 0 executa todo seu laço rapidamente , saindo de sua região crítica e pondo a variável *turn* em 1. Assim, a variável *turn* é 1 e os dois processos estão executando em suas regiões críticas. De repente, o processo 0 termina sua região não crítica e volta ao início de seu laço. Então, a ele não será permitido entrar em sua região crítica agora, pois a variável *turn* está em 1 e o processo 1 está ocupando a região não crítica. Ele fica suspenso em seu laço *while* até que o processo 1 coloque a variável *turn* em 0. Assim, alternar a vez não é tão interessante para processos mais rápidos, ou quando um deles for mais lento que o outro.

3.3.3.4 Solução de Peterson

Essa solução, desenvolvida por G.L. Peterson em 1981, consiste em um algoritmo no qual cada processo chama *enter_region* com seu próprio número de processo, 0 ou 1, como parâmetro, antes de entrar em sua região crítica. Dessa maneira, essa chamada fará com que ele fique esperando, se for necessário, até que seja seguro entrar. Após o término do uso da região crítica, o algoritmo usa *leave_region* para permitir que outro processo entre, caso desejar.

3.3.4 Dormir e acordar

As técnicas anteriores verificam se um processo pode entrar ou não na região crítica e caso não possa, ficará em laço ocioso esperando até que seja permitida a entrada. Assim, além de gastar processamento de CPU, pode ocorrer um problema denominado Inversão de Prioridade. Um processo de maior prioridade nunca é escalonado quando um processo de menor prioridade não consegue sair de sua região crítica, e se o processo de maior prioridade estiver executando, ele entra em um laço infinito.

Um processo simples de bloqueio, em vez de gastar tempo de CPU, é o par *sleep* e *wakeup*. *Sleep* é uma chamada ao sistema que faz com que quem a chama durma, ou seja, fique suspenso até que outro processo o desperte. A chamada *wakeup* tem um parâmetro, o processo a ser despertado. Por outro lado, tanto *wakeup* quanto *sleep* podem ter outro parâmetro, um endereço de memória usado para equiparar os *wakeups* a seus respectivos *sleeps.*

Problema Produtor-Consumidor (ou *buffer* limitado) – Quando dois processos compartilham um *buffer* comum e de tamanho fixo, um deles (o produtor) põe uma informação no buffer e o outro (consumidor) a retira. O problema se origina quando o produtor quer pôr um novo item no buffer e ele está cheio. A solução é pôr o produtor para dormir e só despertá-lo quando o consumidor retirar um ou mais itens. Esse método também funciona para o consumidor.

Porém, isso acarretará um problema similar ao do diretório de *spool* e, portanto, para resolvê-lo, uma variável deverá ser criada (*count*), um contador de itens do *buffer*. Assim, o consumidor deverá ler este código (que é o número máximo de itens que o buffer pode conter) e se o buffer estiver cheio, ele deverá dormir. Caso contrário, ele acrescentará um item e incrementará a variável *count*.

Mas ainda há uma situação em que, tanto o produtor quanto o consumidor podem dormir eternamente. Quando o buffer está vazio (*count* é igual a 0), o consumidor acabou de ler a variável *count* para verificar se seu valor é 0. Nesse instante, o escalonador decide interromper o consumidor temporariamente e executar o produtor. O produtor insere um item no buffer, incrementa a variável *count* e se torna igual a 1. Então, o produtor chama o *wakeup* para o consumidor. Porém, o consumidor não está totalmente adormecido e o sinal de wakeup é perdido. Na próxima vez em que o consumidor for executar, ele lerá que o valor do *count* anterior é 0 e dormirá. Até que chegará uma hora em que o produtor preencherá totalmente o *buffer* e dormirá, então os dois dormirão para sempre.

Para evitar esse problema, devem-se modificar as regras, adicionando um bit de espera pelo sinal de acordar. Quando um processo ainda acordado receber um bit de sinal de acordar, esse bit é ligado. Depois, quando o processo tentar dormir e o bit do sinal de acordar estiver ligado, ele será desligado, mas o processo se manterá acordado.

3.3.5 Semáforos

Essa solução utiliza uma variável inteira para contar o número de sinais de acordar salvos para uso futuro, chamada de Semáforo. Ele pode conter o valor 0 para indicar que não há sinal de acordar salvo e outro valor qualquer para indicar que há sinais de acordar pendentes.

Foram criadas duas operações, a *up* e a *down.* A operação *down* sobre um semáforo verifica se seu valor é maior do que 0. Se for, o decrescerá de 1 e prosseguirá. Se for 0, o processo irá dormir, sem terminar o *down*, por enquanto. É garantido que, uma vez iniciada uma operação de semáforo, nenhum outro processo deverá ter acesso ao semáforo até que tenha terminado ou sido bloqueado. A operação *up* incrementa o valor de um semáforo.

3.4 Escalonamento

Antigamente, quando os computadores usavam sistemas em lote, a entrada se dava por meio de fita magnética e o algoritmo de escalonamento era simples, ou seja, executava o próximo *job* da fita. Com o advento dos sistemas compartilhados, o algoritmo de escalonamento tornou-se mais complexo. E como o tempo de processamento da CPU é um recurso escasso, um bom escalonador faz grande diferença no desempenho da máquina e na satisfação do usuário.

Para os computadores pessoais o escalonamento não deve ser problema, uma vez que, atualmente, eles são extremamente rápidos e raramente um usuário executará duas atividades ao mesmo tempo. Porém, no caso de servidores e estações de trabalho que exigem grande desempenho de rede, a situação é um pouco diferente. Nesses casos, vários processos podem ser executados e estarem competindo pela mesma CPU e, portanto, um bom algoritmo de escalonamento torna-se novamente importante.

Além de escolher o processo certo para executar, um escalonador também deve ser capaz de utilizar a CPU de maneira eficiente, pois alternar processos é muito caro.

3.4.1 Comportamento do Processo

Podem-se dividir os processos em dois: processos orientados à CPU e processos orientados à E/S (de disco). Os processos orientados à CPU apresentam longos surtos de uso da CPU e esporádicas esperas à E/S, enquanto os orientados à E/S apresentam poucos surtos de uso da CPU e esperas freqüentes à E/S.

À medida que as CPUs se tornam cada vez mais rápidas, os processos tendem a ficar mais voltados à E/S, ou seja, as CPUs são cada vez mais rápidas que os discos. Basicamente, se um processo orientado à E/S quiser executar, deve ser dada à ele a prioridade, pois assim ele executará suas aquisições de disco, mantendo-o ocupado.

3.4.2 Quando escalonar

Há pelo menos quatro situações para um escalonador tomar uma decisão. A primeira é quando se cria um novo processo, no qual o escalonador deve escolher entre executar o processo pai ou o filho, quando os dois estiverem prontos. A segunda é quando termina um processo e o escalonador deve tomar a decisão de iniciar um outro. Caso não haja mais nenhum outro processo pronto, é executado um processo ocioso gerado pelo sistema. A terceira é a dependência entre processos, quando há um bloqueio sobre um semáforo ou por alguma outra razão, havendo a necessidade de executar outro processo, mas o escalonador não possui informação necessária sobre essa dependência. A quarta situação é quando ocorre uma interrupção de E/S. É o escalonador que deve tomar a decisão se executa o processo que acabou de ficar pronto depois da interrupção, se continua executando o processo atual ou se seleciona um terceiro processo para executar.

Quanto à tratativa de interrupções de relógio, há duas categorias de algoritmo de escalonamento:

- Preemptivo;
- Não-preemptivo.

Os algoritmos de escalonamento preemptivos escolhem um processo e o deixam em execução por um tempo máximo fixado. Caso ele ainda esteja sendo executado ao final desse tempo, ele será suspenso e o escalonador selecionará outro processo para executar (se houver outro disponível). Esse algoritmo deve contemplar uma interrupção de relógio ao fim do intervalo de tempo para que o controle sobre a CPU seja devolvido ao escalonador.

Os algoritmos de escalonamento não-preemptivos escolhem um processo para executar e o deixam executar até que finalize e libere a CPU ou que haja um bloqueio (à

espera de E/S ou de um outro processo). Porém, ele não será compulsoriamente suspenso, mesmo com as interrupções de relógio, que esperam até acabar.

3.4.3 Categorias de algoritmos de escalonamento

Para cada tipo de sistema, devem-se aplicar algoritmos de escalonamento diferentes, pois cada um possui objetivos distintos. Três ambientes merecem especial atenção:

- Lote;
- Interativo;
- Tempo real.

Nos sistemas em lote, podem-se utilizar algoritmos não-preemptivos ou preemptivos de longo intervalo de tempo, uma vez que não há usuários em espera para uma resposta rápida. Para sistemas com usuários interativos, é necessária a utilização de algoritmos com preempção para evitar que um processo tome conta da CPU por um longo período de tempo.

Em sistema de tempo real, a preempção é desnecessária, às vezes, pois os processos sabem que não podem executar por longos períodos e em geral fazem seus trabalhos e bloqueiam rapidamente. Isso porque os programas de tempo real visam somente ao progresso da aplicação.

Vamos descrever aqui, devido à aplicação e objetivo deste trabalho, somente o escalonamento de sistemas de tempo real. Porém, vamos incluir o escalonamento por prioridades, que pertence a sistemas de usuários interativos, por ser utilizado no protocolo CAN.

3.4.3.1 Escalonamento por prioridades

No escalonamento por prioridades é atribuída a cada processo uma prioridade e ao processo executável com a mais alta prioridade é permitido executar.

Para evitar que um processo com a mais alta prioridade execute eternamente, o escalonador pode reduzir a prioridade do processo em execução a cada interrupção de relógio. Dessa forma, a prioridade desse processo cairá abaixo da prioridade do próximo processo com prioridade mais alta, gerando uma alternância de processo. É possível ainda atribuir um quantum máximo que ele poderá executar.

As prioridades podem ser atribuídas estatística ou dinamicamente aos processos. Como exemplo de atribuição estática, um centro de computação comercial pode cobrar R$ 100,00 por hora para trabalhos de alta prioridade, R$ 70,00 para trabalhos de média prioridade e R$ 50,00 para trabalhos de baixa prioridade pelo mesmo período.

Dinamicamente, pode-se dar como exemplo processos que são orientados à E/S, em que a espera pela CPU significa ocupar memória desnecessária por muito tempo. Assim, atribui-se $1 / f$ à prioridade, sendo f a fração do último quantum que o processo usou. Se esse processo usou 1ms de seu quantum de 50ms, sua prioridade foi de 50, mas se ele usou todo o quantum, ele teve prioridade 1.

É também muito comum agrupar processos em classes de prioridade e usar o escalonamento por prioridade entre as classes. Segue um exemplo de um sistema de quatro classes.

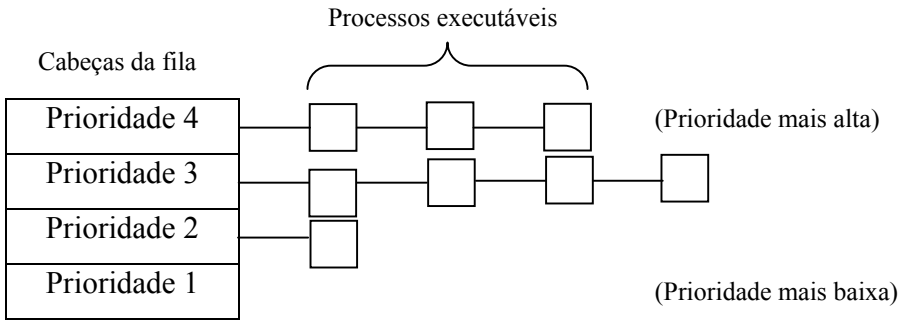


Figura 26 - Exemplo de algoritmo de escalonamento agrupado em classes.

3.4.3.2 Escalonamento em sistemas de tempo real

Num sistema de tempo real, o tempo tem uma função essencial. Nesses sistemas, dispositivos físicos externos geram estímulos ao computador, que deve reagir em determinado intervalo de tempo. Por exemplo, um tocador de CD obtém os bits que chegam do *drive* e precisam convertê-los em música em um intervalo de tempo crítico. Caso contrário, a música soará diferente. Outros exemplos podem ser o piloto automático de aeronaves e os robôs de controle em fábricas automatizadas.

Sistemas de tempo real são geralmente classificados como Sistemas de Tempo Real Críticos, nos quais prazos absolutos devem ser cumpridos, e Sistemas de Tempo Real Não Críticos, quando o descumprimento de um prazo, ocasionalmente, é tolerável.

Os eventos a que um sistema de tempo real deve responder podem ser divididos em Periódicos (acontecem em intervalos regulares) e Aperiódicos (acontecem de modo imprevisível). Um sistema pode ter de responder a múltiplos fluxos de eventos periódicos. Porém, dependendo do tempo necessário para tratar cada evento, talvez não seja possível tratá-los todos. Por exemplo, caso haja *m* eventos periódicos e o evento *i* ocorrer com período *Pi* e requerer *Ci* segundos de CPU para cada evento, então a carga poderá ser tratada somente se:

$$\sum_{i=1}^{m} \frac{Ci}{Pi} \leq 1 \qquad \textbf{(6)}$$

Um sistema de tempo real que satisfaça esse critério é chamado de Escalonável.

Os algoritmos de escalonamento de tempo real podem ser:

- Estáticos;
- Dinâmicos.

Nos sistemas estáticos, as decisões de escalonamento são tomadas antes de o sistema começar a executar, ao passo que nos sistemas dinâmicos, as decisões são tomadas em tempos de execução. Em sistema estático, seu funcionamento só se dá quando há informação perfeita disponível sobre o trabalho a ser feito e os prazos que devem ser cumpridos. Os sistemas dinâmicos não apresentam restrições. Os algoritmos estáticos atribuem antecipadamente uma prioridade fixa a cada processo e então fazem o escalonamento preemptivo priorizado utilizando essas prioridades, ao passo que os algoritmos dinâmicos não apresentam prioridades fixas.

Sistemas multimídias, ou seja, filmes digitais, são um grande exemplo de utilização de sistemas de tempo real, primeiro porque utilizam taxas de dados extremamente altas e segundo porque requerem reprodução em tempo real. A porção de vídeo de um filme digital consiste em alguns quadros por segundo. O sistema NTSC executa 30 quadros por segundo em intervalos de 33,3 ms e o sistema PAL executa 25 quadros por segundo a cada 40 ms. Do contrário, a imagem parecerá fragmentada.

Nos seres humanos, os ouvidos são mais sensíveis que os olhos; assim, uma variação de até mesmo alguns milissegundos na exibição será notada. A variação nas taxas de entrega é chamada de *Jitter* e deve ser estritamente limitada para se obter um bom desempenho. O *jitter* não é o mesmo que atraso. Se a rede atrasar todos os bits por 5 s, de

maneira uniforme, o filme começará mais tarde, mas será visto perfeitamente. Porém, se os quadros forem atrasados em 100 ms, ele se parecerá com um filme de antigamente, podendo-se notar as mudanças de quadros.

3.4.4 Escalonamento geral de tempo real

Vamos considerar três processos *A, B* e *C* competindo pela CPU, cada um com seu trabalho e prazo próprio, conforme a figura 3.8. O processo A executa a cada 30 ms (cada trabalho requer 10 ms de tempo de CPU). Caso não houvesse competição, o processo executaria nos surtos A_1, A_2 , A_3 , etc., cada um iniciando 30 ms depois do anterior. Cada surto da CPU trata um trabalho e tem prazo de terminar antes que o próximo inicie.

O processo *B* executa 25 vezes por segundo e o processo *C* executa 30 vezes por segundo, e o tempo de computação por trabalho é de 15 ms para o *B* e de 5 ms para o *C*.

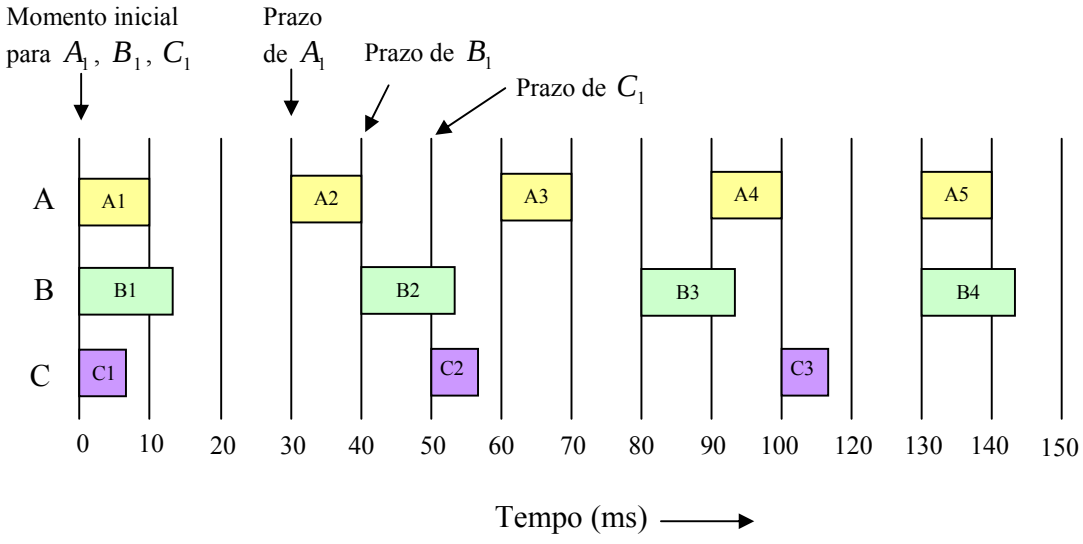


Figura 27 - Três processos periódicos.

Agora, o problema de escalonamento é como fazer que os três processos sejam executados de modo que eles cumpram os prazos determinados. Porém, é necessário saber se os processos são escalonáveis, ou seja, atendem à equação (6). A razão P_i / C_i corresponde a quanto está sendo utilizada a CPU. Nesse exemplo, o processo A está utilizando $10/30$ da CPU, o B está utilizando $15/40$ da CPU e o C está utilizando $5/50$ da CPU. No total, os três processos somam 0,808 de CPU; portanto, o sistema é escalonável.

3.4.4.1 Escalonamento por Taxa Monotônica

O algoritmo clássico de escalonamento estático de tempo real para processos preemptivos e periódicos é o chamado Escalonamento por Taxas Monotônicas (*Rate Monotonic Scheduling – RMS*), definido por Liu e Layland, 1973 (LIU & LAYLAND, 1973 apud TANENBAUM, 2003). As seguintes condições devem ser seguidas pelos processos que utilizam o RMS:

- Cada processo periódico deve terminar dentro de seu período;
- Independência de processos;
- Todo processo utiliza a mesma quantidade de CPU a cada surto;
- Caso o processo seja não periódico, não deve haver prazo estabelecido;
- A preempção ocorre instantaneamente e sem sobrecargas.

O RMS atribui a cada processo uma prioridade igual à freqüência de ocorrência de seu evento de disparo, ou seja, as prioridades são lineares em relação à freqüência. Por exemplo, um processo que dispara a cada 30 ms (ou 33 vezes por segundo), recebe prioridade 33. Um processo que executa a cada 50 ms (ou 20 vezes por segundo), recebe prioridade 20, e assim por diante. Por essa razão, o processo de escalonamento é chamado Monotônico.

3.4.4.2 Escalonamento Prazo Mais Curto Primeiro

O algoritmo de escalonamento chamado de Prazo Mais Curto Primeiro (*Earliest Deadline First – EDF*) é popular para sistemas de tempo real. O EDF é um algoritmo dinâmico em que os processos não necessitam ser periódicos, como o RMS. Outra diferença em relação ao RMS é que ele não exige o mesmo tempo de execução por surto de CPU. Caso necessite de CPU, o processo anuncia sua presença e seu prazo. O escalonador tem uma lista de processos executáveis, ordenados por vencimentos de prazo. O algoritmo executa sempre o primeiro da lista, cujo prazo é o mais curto, ou o mais próximo de vencer.Caso tenha um outro processo que esteja pronto, o sistema verificará se seu prazo vence antes do prazo do processo que está em execução. Em caso positivo, o novo processo faz a preempção do que estiver executando.

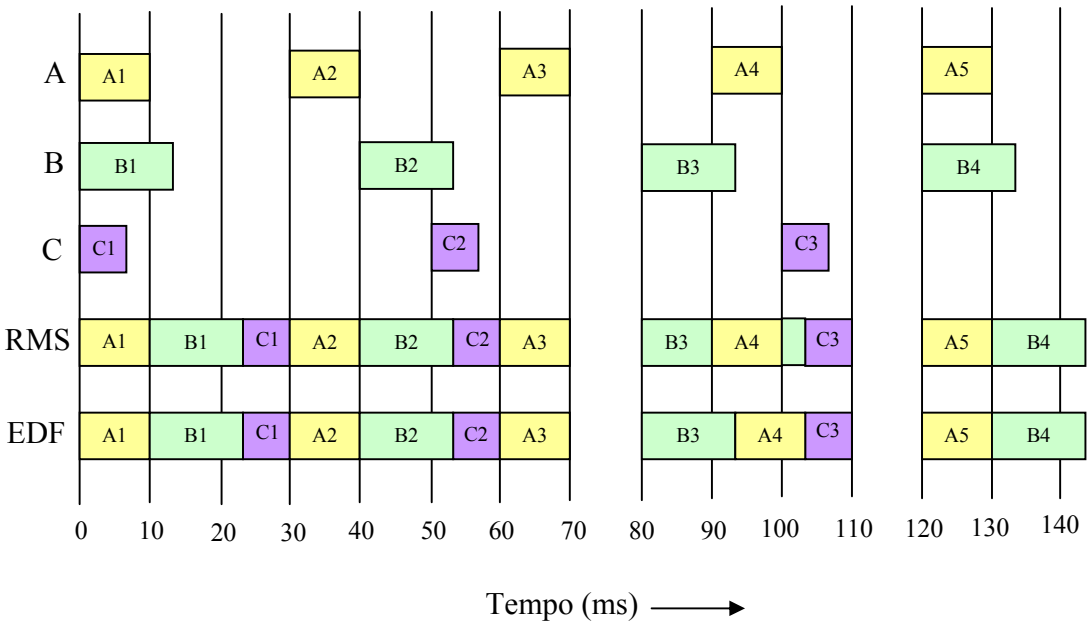


Figura 28 - Comparativo entre os algoritmos RMS e EDF.

Aparentemente, a figura 28 mostra um comportamento parecido entre os dois algoritmos RMS e EDF, mas não é verdade. No caso de aumentarmos a necessidade de CPU do processo *A*, de 10 ms para 15 ms, a situação se complica. Nesse caso, a ocupação da CPU passa de 0,808 para 0,975 (0,500 + 0,375 + 0,100) e, assim, restam apenas 2,5% da CPU. Entretanto, na prática, ainda é possível fazer um escalonamento razoável.

Como as prioridades iniciais dos processos não mudam, para o RMS, os processos *A* e *B* ficam executando e, devido à prioridade baixa de C, C não consegue executar, pois seu prazo vence, determinando assim a falha do algoritmo RMS. No caso do algoritmo EDF, há uma disputa entre *A2* e *C1*. Como o prazo de *C1* vence primeiro em 50 ms e de *A2* vence em 60 ms, *C* é escalonado e, portanto, o algoritmo EDF funciona.

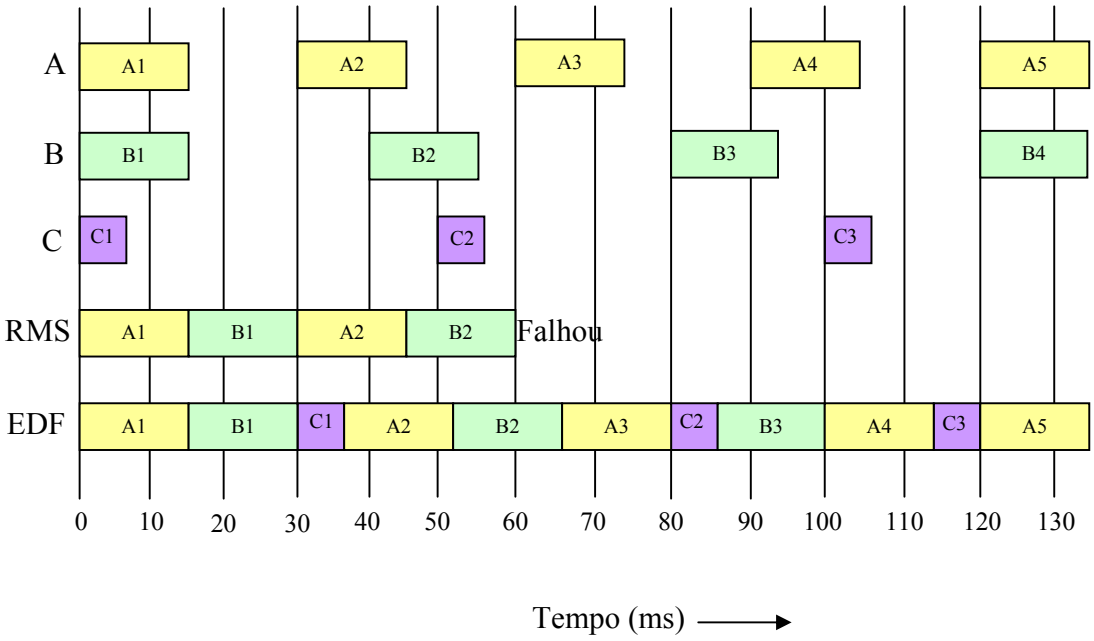


Figura 29 - Exemplo onde RMS falha.

O principal motivo pelo qual o RMS falhou foi porque a utilização de prioridades estáticas só funciona se a ocupação da CPU não for muito grande. Liu e Layland provaram que a utilização máxima da CPU deve ser de 0,780 e o argumento que o exemplo anterior mostrou (utilização de 0,808) foi sorte. No caso do EDF, ele sempre funciona, mesmo com utilização de 100% da CPU.

4. CONTROLLER AREA NETWORK

4.1 Introdução

A rede CAN (*Controller Area Network*) é um sistema de comunicação serial concebida inicialmente para aplicações distribuídas de sistemas automotivos. Sua crescente utilização na indústria automotiva foi motivada pelos benefícios técnicos e econômicos, o que posteriormente tornou-se padronizada pela SAE (*Society of Automotive Engineers*) e pela ISO (*International Organization for Standardization*). Com o sucesso obtido nas aplicações automotivas, a comunidade da indústria de processo e transformação não hexitou em adotá-la também em determinadas aplicações industriais.

As ECUs possuem interfaces de comunicação para troca de mensagens com outras ECUs, interfaces medição e atuação em grandezas físicas como velocidade, temperatura, etc, sendo expressos na forma de sinais digitais inseridas em mensagens trocadas entre nós computacionais sob uma rede de comunicação. As ECUs são sistemas computacionais com tarefas que geram mensagens com propriedades temporais, definidas em função da aplicação.

A Scania Latin América Ltda utiliza a ferramenta computacional Canalyser, do fabricante Vector Informatik, para avaliar e auxiliar projetistas de redes automotivas no desenvolvimento e gerenciamento do desempenho da rede de forma estática, sendo possível desenvolver o projeto e avaliar as características dos nós computacionais, mensagens e sinais digitais sob a rede CAN, avaliando assim o desempenho estático da arquitetura de rede em tempo de projeto.

4.2 Protocolo de Nível Superior

Ao se especificar os Protocolos de Nível Inferior, é útil subdividir as duas camadas de acordo com o Standard LAN ISO 8802-2 e ISO 8802-3 (ANDERSSON et al, 1995), como segue:

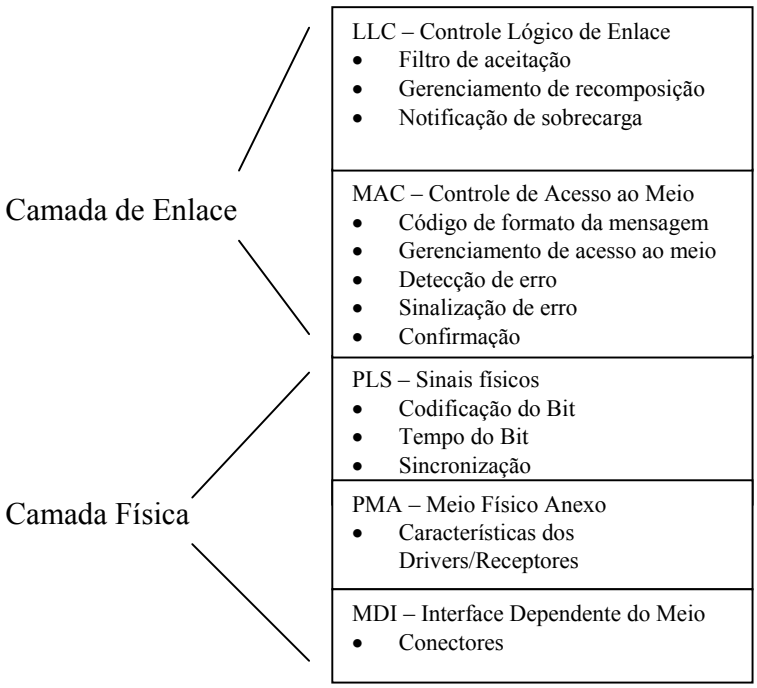


Figura 30 - Subdivisões das Camadas de Enlace e Física conforme LNA ISO 8802-2 e ISO 8802-3.

4.2.1 Meio Físico

O meio de transmissão é o meio em que ECUs são fisicamente conectadas em uma rede e carregam sinais elétricos ou ópticos. Podem ser utilizados vários meios de transmissão, tais como, único fio, dois fios trançados (*unshielded*), dois fios trançados e protegidos (shielded), fibra óptica, entre outros.

4.2.2 Interface Meio-Dependente (MDI)

A sub-camada MDI especifica as propriedades dos conectores elétricos, no caso de uso do meio com fios elétricos. Por causa do conteúdo possível de alta freqüência e baixa tensão do sinal elétrico, os parâmetros como impedância de transmissão e freqüência mínima se tornam importantes.

4.2.3 Meio Físico Anexo (PMA)

A sub-camada PMA especifica as características dos *line-drivers* (dispositivos usados para transferir energia elétrica para um fio ou para um barramento que realiza a parte da função de transmissão do transceptor) e receptores.

4.2.4 Sinais Físicos (PLS)

A sub-camada PLS endereça informações, tais como a representação dos bits, tempo dos bits e sincronização.

4.2.5 Controle de Acesso ao Meio (MAC - Medium Access Control)

MAC ou Controle de Acesso ao Meio é usado para decidir quem ganha acesso à rede para transmitir. Como todas as ECUs podem usar a rede para transmissão, é necessária a utilização de um controle de acesso. Esse mecanismo é situado na sub-camada MAC. Há duas diferentes técnicas de acesso ao meio: CSMA/CD e CSMA/CD-A.

4.2.6 Anexo do Meio Físico de Alta Velocidade (PMA)

As linhas do barramento podem ter dois estados: “recessivo” e “dominante”. O estado “dominante” de uma ECU prevalece sobre o estado “recessivo” de outra. Os dois

estados são definidos por uma diferença de tensão entre os dois fios da rede, chamados de CAN_H e CAN_L, conforme figura a seguir:

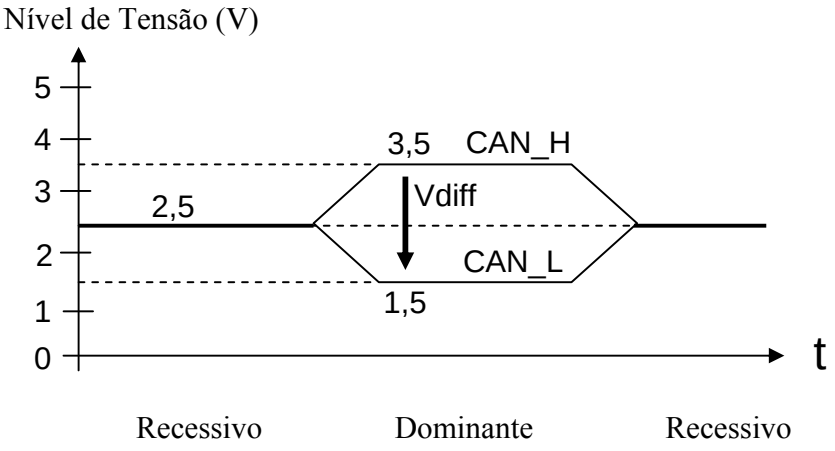


Figura 31 - Valores de tensão para os estados “recessivo” e “dominante”.

Toda ECU deve ser capaz de prover os seguintes valores de tensão de saída diferencial da rede:

- Bit dominante: - 500mV ... +50mV (sem carga)
- Bit recessivo: +1,5 V ... +3,0 V (com 60 Ω de carga)

Uma ECU deve detectar um bit recessivo se a tensão do CAN_H não for maior que a do CAN_L mais 0,5V. Se a tensão do CAN_H for pelo menos 0,9V maior que CAN_L, então um bit dominante deverá ser detectado.

O protocolo CAN especifica um barramento de dois fios terminado em ambos os lados por uma impedância de linha específica, com os seguintes dados:

- - Comprimento máximo da rede a 1Mbits/s 40m
- - Máxima distância da rede a ECU 30cm

- - Impedância característica da linha 120Ω
- - Resistência nominal da linha por m 70mΩ/m
- - Atraso de propagação específica nominal 5ns/m

A topologia da rede deve ser especificada o mais perto possível de uma estrutura de linha única (*Bus Topology*), conforme especificações abaixo:

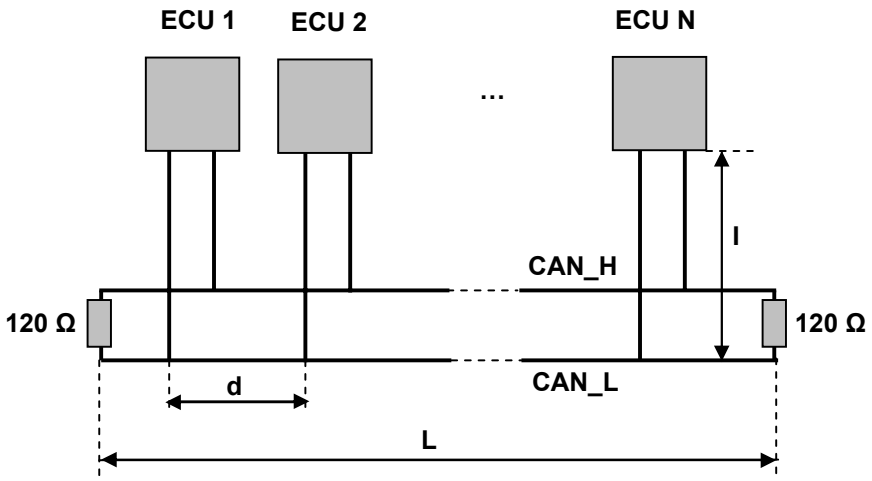


Figura 32 - Modelo de Topologia especificada pela ISO 11898.

4.2.7 Sinais Físicos (PLS)

4.2.7.1 Tempo do Bit (Bit Timing)

O tempo do bit, t_B , é definido como o tempo de duração do bit, conforme figura a seguir:

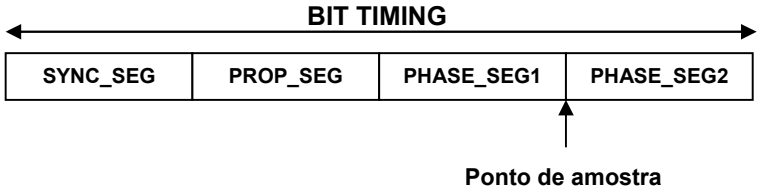


Figura 33 - Subdivisão de um tempo do bit (*Bit Timing*).

Dentro desse formato de tempo do bit, todas as funções de gerenciamento do barramento são executadas, tais como sincronização entre ECUs, compensação de atrasos da rede e posicionamento do ponto de amostra.

Os comprimentos dos diferentes segmentos do bit são definidos como o múltiplo de uma unidade básica de tempo, chamada de *time quantum,* derivado do período do oscilador, programável no circuito de lógica de tempo do bit (250ns with a 16Mhz clock). O tempo do bit para o Protocolo CAN é definido como 4μs, correspondendo a 250Kbits/s.

SYNC_SEG significa segmento de sincronização. É a parte usada para sincronizar várias ECUs no barramento. A duração desse segmento é sempre de um *time quantum.* O segmento de sincronização considera que a borda de descida do sinal (*edge*) é esperada logo após uma unidade de tempo básico (*time quantum*) ter passado e, portanto, ser perdida. A distância entre a borda de descida do sinal, que ocorre fora do segmento de sincronização e o mesmo, é chamada de "erro de fase e" daquele *edge*.

PROP_SEG significa segmento de atraso de sincronização e é a parte que compensa tempos de atraso físico dentro da rede, causado por tempos de atraso na propagação das linhas do barramento e circuitos transceptores.

PHASE_SEG1 e 2 significam segmento de fase de *buffer* e são usados para compensar erros de fase e podem ser aumentados e diminuídos através da re-sincronização.

SAMPLE POINT ou Ponto de Amostra é o ponto do tempo no qual o valor da rede é lido e interpretado como o valor daquele respectivo bit.

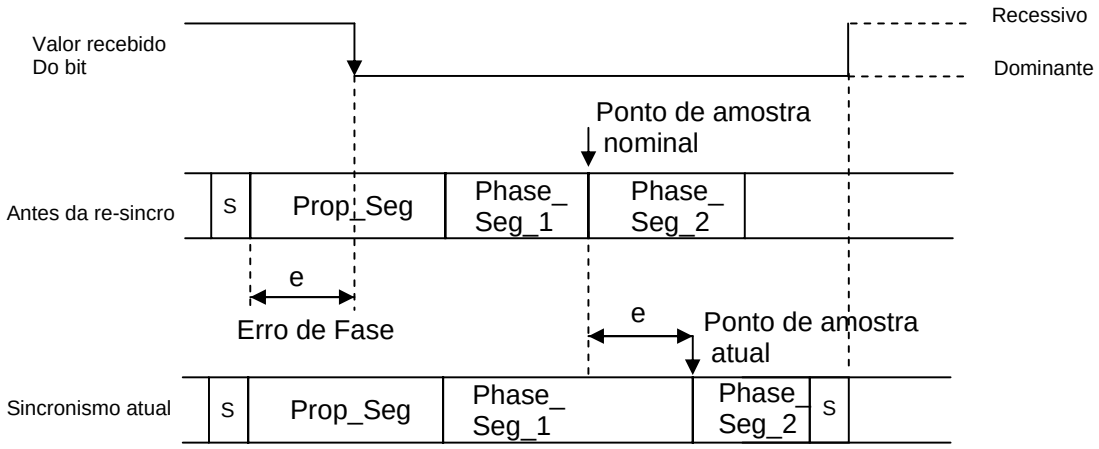
4.2.7.2 Sincronização do bit

Uma sincronização robusta de tempo de bit pode ser realizada quando a rede estiver livre toda vez que o barramento entra em estado “dominante”, ou seja, o tempo do bit é reiniciado com o final do segmento de sincronização, independente do erro de fase.

Todas as outras transições no formato de “recessivo” para “dominante” serão usadas para re-sincronização, o que provoca o aumento ou redução do tamanho do tempo do bit, de maneira que a posição do ponto de amostra é mudada em relação à borda de descida do sinal (*edge*). O número máximo permitido de aumento e redução no comprimento dos segmentos de fase de *buffer* 1 e 2 (*Resyncronization Jump Width*, SJW), por re-sincronização, é limitado e pode ser programado entre 1 e 4 unidades básicas de tempo (*time quantum*).

Apresenta-se, a seguir, o princípio de re-sincronização (ETSCHBERGER, K, 2001):

a. Transmissor mais lento que receptor (descida do sinal está atrasada)



b. Transmissor mais rápido que receptor (descida do sinal está adiantada)

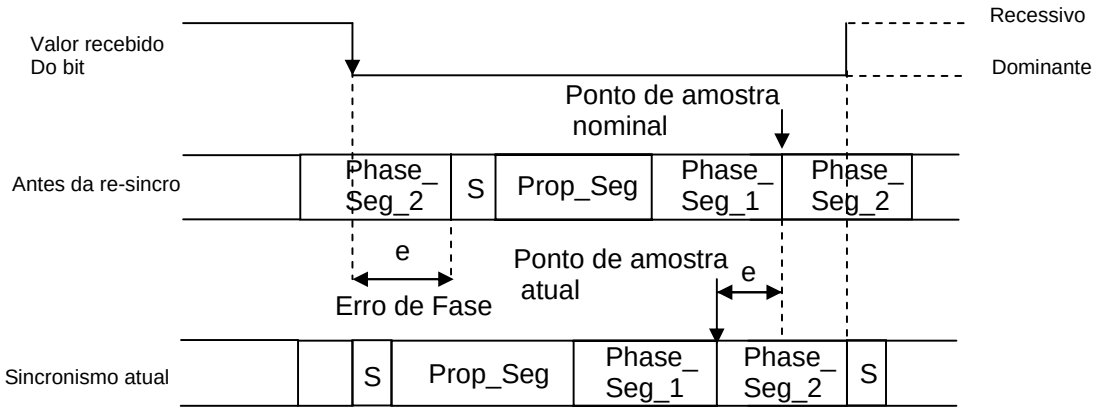


Figura 34 - Princípio da re-sincronização.

4.2.7.3 Codificação do bit (Bit Encoding)

A necessidade da codificação do bit se deve ao fato de que ECUs receptoras devem saber interpretar o sinal enviado para que dados possam ser extraídos. A codificação define as condições (corrente, tensão) que constituem os bits 0 e 1 no barramento. O método utilizado pelo Protocolo CAN para codificação de bits é o chamado NRZ (*Non Return to Zero*), modo Unipolar (LUPINI, 2004).

4.2.8 Controle de Acesso ao Meio (MAC)

Como apresentado anteriormente, a sub-camada MAC, pertencente à Camada de Enlace, é a responsável pelo empacotamento dos dados dentro de Formatos de Mensagem CAN, contendo os bits especificados de diferentes controles. Ela também realiza o gerenciamento de acesso ao meio, tais como serialização, adição *stuff-bits*, detecção erros e reconhecimento de sobrecarga e checagem.

4.2.8.1 Tipos de Formato de Mensagem

O protocolo CAN distingue quatro tipos diferentes de Formato:

- Formato de Dados (Frame Format) – no qual dados são transmitidos de um transmissor para um ou vários receptores através da iniciativa da origem (transmissor);
- Formato Remoto (Remote Frame) – pelo meio do qual ECUs (receptores) podem requisitar a transmissão de um formato de dados de um mesmo identificador por uma origem;
- Formato de Erro (Error Frame) – é usado para sinalizar um erro detectado por uma ECU (transmissor ou receptor) e para destruir o formato;
- Formato de Sobrecarga (Overload Frame) – é usado para prover um atraso extra entre um dado que precede e que sucede, ou entre um formato remoto de solicitação, ou ainda sinalizar uma condição específica de erro.

Os formatos de Dados e Remoto são separados de formatos precedentes por um intervalo de tempo de nível recessivo da rede de pelo menos três bits (Campo *Intermission*). O CAN permite dois tipos diferentes de formatos: o "Formato Base", com 11 bits identificadores, e o "Formato Estendido", com 29 bits identificadores.

Formato de Dados - O Formato de Dados para transmitir mensagens é composto por sete campos principais: *Start-of-Frame (SOF), arbitration, control, data, CRC, acknowledgement* e *End-of-Frame*.

O Formato base começa com o bit inicial chamado de *Start-of-Frame* (SOF), e é representado por um único bit dominante. Uma ECU só pode iniciar o processo de arbítrio quando o barramento estiver livre (*Idle*). Em seguida vem o campo de arbítrio, contendo o identificador de 11 bits e o bit "*Remote Transmission Request*" (RTR), que indica se é um formato de dados ou formato de solicitação (*request*). O formato de solicitação não possui

bytes de dados. O campo de controle (*Control Field*) contém o bit *Identifier Extension* (IDE), que indica o formato base ou estendido. Contém também um bit reservado para uso futuro (r0) e, nos quatro últimos bits, um indicador de bytes de dados no campo de dados (DLC). O campo de dados vai de zero a oito bytes e é seguido pelo campo *Cyclic Redundancy Check* (CRC), que consiste em uma checagem de seqüência de 15 bits e um bit delimitador recessivo para detectar erros de bits. O campo *acknowledgement* (ACK) contém a posição (*slot*) do bit ACK e o delimitador ACK. O *slot* do bit ACK é colocado na rede pelo transmissor como um bit recessivo (nível lógico 1). É sobrescrito como um bit dominante (nível lógico 0) pelos receptores que receberam os dados corretamente. Então a ECU que transmitiu assegura-se de que pelo menos uma outra ECU recebeu corretamente a mensagem. Há uma confirmação (*acknowledged*) pelo receptor se é uma mensagem para o receptor ou não. O campo *End-of-Frame* indica o fim da mensagem. Os bits do campo *Intermission* (Int) indicam o número mínimo de períodos de bits que separam mensagens consecutivas. Se não há mais acesso por qualquer outra ECU, o barramento permanece livre (*idle*).

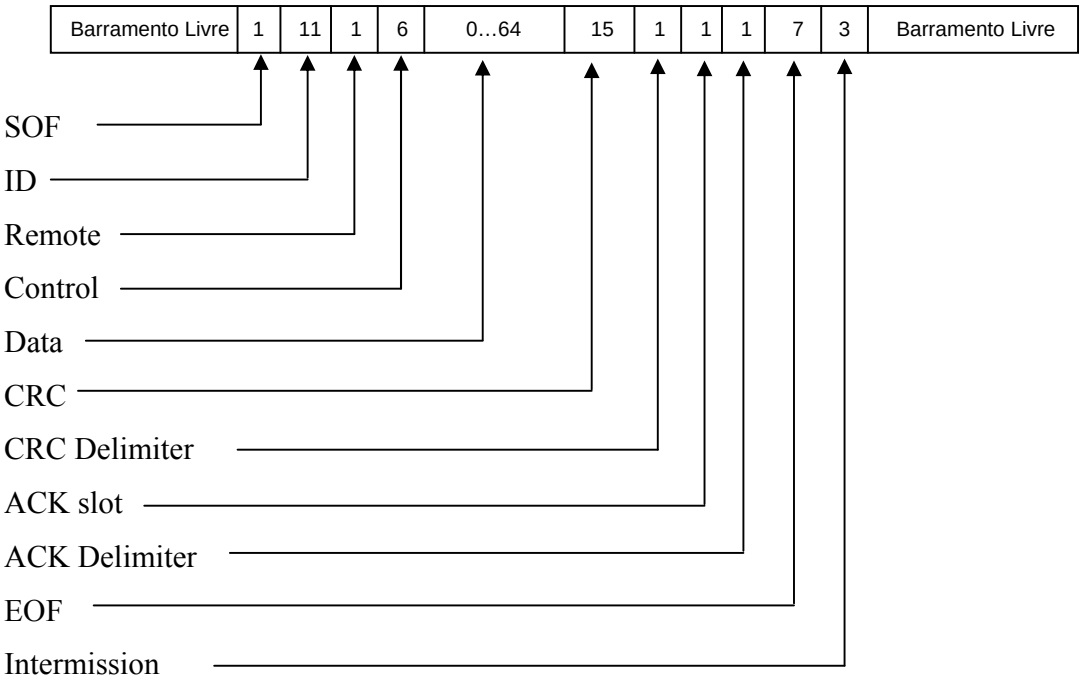


Figura 35 - Formato de Dados Base CAN 2.0A – 11 Bits.

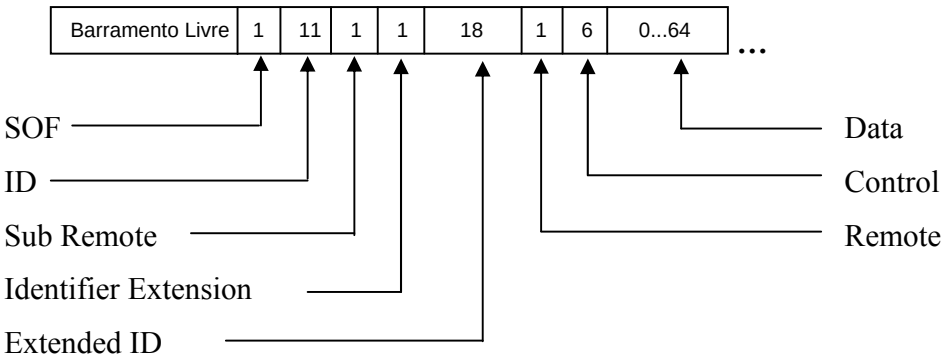


Figura 36 - Formato de Dados Estendido CAN 2.0B – 29 Bits.

Formato Remoto - O formato remoto difere do formato de dados por não conter nenhum campo de dados. Para separar formatos remotos de formatos de dados, o bit RTR localizado no campo Remoto é recessivo para o formato remoto.

O formato remoto é usado por uma ECU para solicitar um formato de dados de uma outra ECU, com o mesmo identificador do formato remoto enviado pela ECU solicitante.

Formato de Erro - A detecção de erro durante a transmissão de dados ou formato remoto é sinalizada por um Formato de Erro, que provoca a retransmissão do formato violando a regra de Bit Adicional (*Bit-Stuffing),* e pode ser detectado por todas as ECUs.

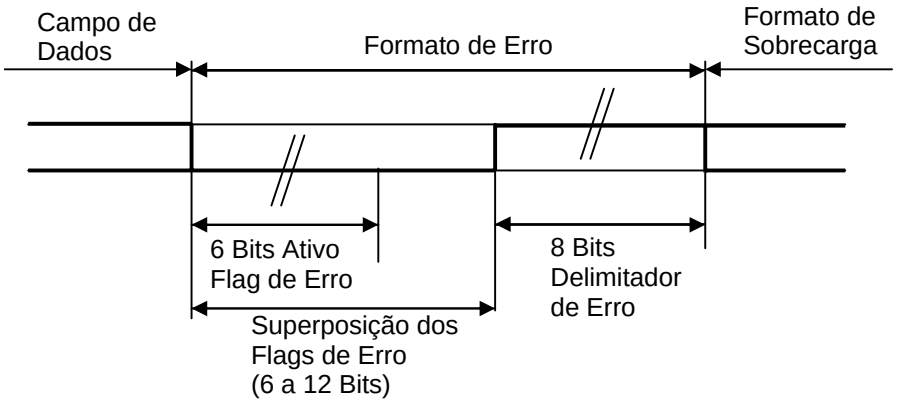


Figura 37 - Formato de erro

O formato de erro consiste em dois campos. O primeiro campo é chamado de Bandeira de Erro (*Error Flag*) e o segundo é chamado de Delimitador de Erro (*Error Delimiter*).

- Bandeira de Erro - Pode ser tanto ativo como passivo, dependendo do estado do Confinamento de Falhas (Fault Confinement). A bandeira de erro ativo consiste em 6 bits dominantes consecutivos e o passivo consiste em 6 bits recessivos.

- Delimitador de Erro - Consiste em 8 bits recessivos. Após a detecção de uma bandeira de erro, cada ECU envia bits recessivos até que elas detectem um bit recessivo no barramento, então elas transmitem mais 7 bits recessivos que criam um delimitador.

Formato de Sobrecarga - O formato de sobrecarga é fornecido para solicitar um atraso de próximas mensagens ou formato remoto pelo receptor de uma ECU ou para sinalizar certas condições de erro relacionadas ao campo *Intermission,* dando tempo para o receptor executar tarefas necessárias. Pode ser iniciado pela sub-camada LLC para indicar uma situação de sobrecarga interna ou pela MAC sobre certas condições de erro.

Consiste em dois campos: bandeira de sobrecarga e delimitador de sobrecarga. A diferença entre os dois é o tempo de transmissão. O formato de sobrecarga começa diretamente após o formato anterior sobrescrevendo o espaço *inter-frame*.

Espaço *Inter-frame* - É o espaço que separa os formatos de dados e remotos de todos os formatos precedentes (dados, erros, sobrecarga, etc). Contém os campos de bits *intermission* e barramento livre (*bus idle*) e de suspender transmissão (*suspend transmission*).

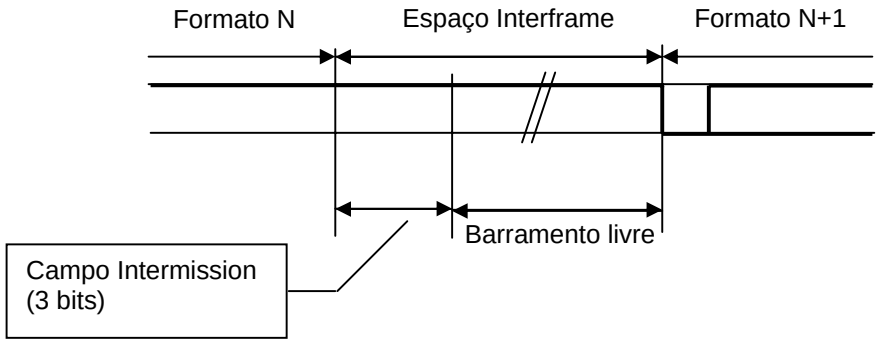


Figura 38 - Espaço Interframe

4.2.8.2 Estrutura da Mensagem – CAN 2.0B 29 Bits (ISO, 2003)

De acordo com a próxima figura, a norma J1939 fornece a definição completa de rede utilizando a estrutura estendida de identificador de 29 bits, informação do comprimento do campo de dados e até 8 bytes de dados.

<table>
<tr><th>28..26</th><th>25</th><th>24</th><th>23...16</th><th>15...8</th><th>7..0</th></tr>
<tr><td rowspan="2">Prioridade</td><td colspan="4">Número do Grupo de Parâmetro</td><td rowspan="2">SA</td></tr>
<tr><td>r</td><td>DP</td><td>PF</td><td>DA/GE</td></tr>
</table>

Onde,

r.....Reservado

DP.....Página de Dados (Data Page)

PF.....Formato PDU

DA.....Endereço de Destino (Destination Address)

GE.....Extensão de Grupo (Group Extension)

SA.....Endereço de Origem (Source Address)

O formato do identificador CAN é dividido nas seguintes partes:

- Prioridade;
- PGN, ou Número de Grupo de Parâmetro, que especifica o tipo de mensagem e os dados transmitidos ou o endereço de destino;
- Endereço de origem da mensagem.

Prioridade - Os três primeiros bits são utilizados para determinar a prioridade da mensagem no processo de arbitragem. O valor 000 é o de maior prioridade, e as mensagens de alta prioridade devem ser usadas para mensagens de controle de alta velocidade, assim como as de baixa prioridade devem ser usadas para mensagens críticas em tempo.

Bit Reservado (r) - Esse bit deve ser definido como 0 e será utilizado para aplicações futuras pela SAE.

Bit de Página de Dados (DP) - É utilizado como um seletor de página. A página 0 indica a mensagem atual que está sendo definida e a página 1 indica o próximo campo PDU (Unidade de Protocolo de Dados).

Formato PDU (PF) - Juntamente com os campos DA/GE forma o PGN (Número do Grupo de Parâmetro), que é usada para definir mensagens, identificar comandos, dados. Como parâmetro, podemos dar o exemplo de rotação do motor (rpm).

Há dois PDUs disponíveis para serviços de comunicação para módulo e mensagem, que são PDU1 e PDU2.

PDU específico (DA/GE) - Esses bits são dependentes do valor do PF. Caso o PF estiver entre 0 e 239 (PDU1), esse PDU específico irá conter o endereço de destino (DA), e caso o PF estiver entre 240 e 255 (PDU2), o PDU específico conterá a Extensão de Grupo (GE). A Extensão de Grupo contém um grande número de valores para identificar mensagens que podem ser transmitidas para todas as ECU's na rede. Esse é o conceito de Multicasting.

Endereço de Origem - Esse campo de 8 bits define o endereço específico de quem está enviando a mensagem. Somente deve haver um endereço de origem e, portanto, os identificadores de dois módulos são sempre diferentes.

4.1.8.3 Métodos de comunicação

Três métodos de comunicação são possíveis dentro da Norma e o uso devido de cada tipo permite o uso efetivo dos PGNs (Números de Grupos de Parâmetros), como segue:

- Destino específico, utilizando o PDU1 (valores de PF de 0 a 239). Ainda inclui o uso do endereço de destino global 255.

- Transmissão, utilizando o PDU2 (valores de PF de 240 a 255).
- Propriedade, usando tanto o PDU1 quanto o PDU2.

Cada um dos métodos tem uma utilização apropriada.

A comunicação de Destino específico é utilizada caso as mensagens devam ser direcionadas para um ou outro destino específico e não para ambos.

A comunicação de Transmissão aplica-se em algumas situações, incluindo mensagens enviadas de uma única ou múltiplas origens para um único destino, e também para mensagens enviadas de uma única ou múltiplas origens para múltiplos destinos.

Finalmente, a comunicação de Propriedade é utilizada em duas situações: uma, na qual comunicações padronizadas não são necessárias, e outra, em que a informação de propriedade de comunicação é importante.

4.2.8.4 Formato de Código (Bit Adicional – *Bit Stuffing*)

No protocolo CAN, os bits de um formato são representados fisicamente de acordo com o código NZR, o que significa que, durante o intervalo de um bit, o bit gerado seja ou dominante ou recessivo. Isso garante eficiência máxima em codificação do bit. No entanto, se houver muitos bits seguidos com o mesmo valor (5 ou mais bits), a sincronização pode ser perdida. Então, usa-se a técnica do *bit stuffing,* onde um bit (*Stuff Bit*) é inserido pelo transmissor após a seqüência de bits de mesmo nível. O bit *stuff,* ou bit adicional, é automaticamente eliminado pelo receptor.

4.2.8.5 Gerenciamento de Acesso ao Meio

Sendo o CAN um sistema multimestre, as ECUs que estão transmitindo são consideradas mestres no sistema. Qualquer ECU está livre para acessar o barramento quando o mesmo estiver com status livre. O barramento é considerado livre se o bit do campo *Intermission* não tiver sido interrompido por um bit dominante.

4.2.8.6 Mecanismos de Detecção de Erro

O protocolo CAN provê cinco mecanismos de detecção de erro:

- Checagem de Bit (Bit Check);
- Checagem de Formato (Frame Check);
- Checagem Cíclica de Redundância (Cyclic Redundancy Check);
- Checagem de Confirmação (Acknowledgement Check);
- Checagem da Regra de Stuff (Stuff Rule Check).

Checagem de Bit (*Bit Check*) - Cada ECU que transmite monitora se o nível do barramento transmitido difere do nível real. Se o valor do bit que foi transmitido for diferente do valor do bit que é monitorado, um erro de bit é detectado.

Checagem de Formato (*Frame Check*) - Checa os campos dos bits em relação ao formato fixo, assim como os seus tamanhos. Os erros detectados são chamados de Erros de Formato.

Checagem Cíclica de Redundância (*Cyclic Redundancy Check*) - A checagem CRC guarda a informação do formato adicionando um bit redundante de checagem ao final da transmissão.

Na ECU receptora esses bits são recalculados e comparados aos bits recebidos. Caso eles não forem os mesmos, ocorreu um erro de CRC.

Checagem de Confirmação (*Acknowledgement Check*) - Uma ECU transmissora espera que na janela ACK ao menos uma ECU receptora reconheça o recebimento do formato transmitido. A falta do reconhecimento gera um erro de confirmação.

Checagem da Regra de *Stuff* (*Stuff Rule Check*) - Se uma ECU receber seis bits consecutivos iguais no formato que deveria ser codificado pelo método do bit adicional, um erro de *stuff* é detectado.

4.2.8.7 Controle Lógico de Enlace – LLC

Essa sub-camada corresponde à parte superior da Camada de Enlace do Modelo OSI. Ela endereça tarefas que são independentes do método de acesso ao meio, como filtro de aceitação, notificação de sobrecarga e gerenciamento de recomposição.

A sub-camada provê dois serviços ao usuário do circuito de interface CAN:

- Transferência de dados não-reconhecidos e;
- Solicitação de dados remotos não-reconhecidos.

A interação entre o usuário e o LLC é realizada usando dois formatos: o Formato de Dados LLC e o Formato Remoto LLC

4.2.8.7.1 Filtragem de Aceitação

Se uma ECU quiser enviar uma mensagem a uma ou mais ECUs, ela passa os dados e seu identificador ao processador CAN. A mensagem é construída dentro de um Formato de Dados pelo processador CAN e é transmitida quando o processador receber a alocação do barramento (arbitragem do barramento). Então, todas as outras ECUs se tornam receptoras do formato de dados e cada uma realiza um teste de aceitação para determinar se o formato é relevante para a ECU. Se for, ela aceita o formato de dados, caso contrário, será ignorado. Esse processo é conhecido como filtragem da mensagem. A filtragem pode ser realizada pela CPU, mas há processadores CAN que podem realizar o processo de filtragem para ajudar a CPU.

4.2.8.7.2 Notificação de Sobrecarga

Caso uma ECU receptora gere atrasos antes de aceitar uma mensagem devido a problemas internos, a sub-camada LLC transmite um formato de sobrecarga para aliviar o receptor.

4.2.8.7.3 Gerenciamento de Recomposição

Caso a ECU que estiver transmitindo uma mensagem perder acesso ao barramento devido ao processo de arbitragem, e não for capaz de transmiti-la novamente, a sub-camada LLC realiza a retransmissão automática até que a mensagem seja transmitida com sucesso, quando uma confirmação de envio é enviada ao usuário.

4.2.8.8 Confinamento de Falha (*Fault Confinement*)

As tarefas mais importantes do sistema Confinamento de Falha são a distinção entre falhas temporárias e permanentes e a desconexão de ECUs com falhas. Isso é resolvido usando contadores de erros, um para erros de transmissão e outro para erros de recepção.

Quando um erro é detectado durante a transmissão ou recepção, o contador correspondente será aumentado com o valor de 1 ou 8, de acordo com um conjunto determinado de regras, dependendo do tipo de erro.

Após uma transmissão ou recepção com sucesso, o contador correspondente será diminuído com o valor 1, exceto na situação em que o contador da recepção tiver o valor de 127, então ele terá o valor estabelecido entre 119 e 127. Isso tem o efeito de que os contadores refletem a freqüência relativa de distúrbios anteriores. Uma falha permanente é notada quando uma média de uma em oito mensagens não for enviada.

Dependendo de valores predeterminados dos contadores, o comportamento das ECUs é modificado. As ECUs, gerenciadas por Confinamento de Falha, podem estar em um dos três estados seguintes:

- Erro ativo é o modo de operação normal de uma ECU. Neste estado a ECU responde com “formatos de erro ativo” nos erros detectados de transmissão ou recepção.
- Erro passivo é o estado adotado quando o valor dos contadores de erros de transmissão ou recepção estiver acima de 127. Nesse estado as ECUs respondem com “formatos de erro passivo”, quando erros de transmissão ou recepção são detectados. Após transmitir a mensagem, a ECU não inicia uma outra transmissão imediatamente.
- Rede indisponível (Bus-off), é o estado assumido caso o valor do contador de erro esteja acima de 255. Nesse estado, à ECU não é permitido ter qualquer influência na rede. Após a reinicialização, ou a recepção do sinal Normal_Mode_Request do maior protocolo da ECU, a ECU terá a permissão de adquirir o status “erro ativo”, após 128 ocorrências de 11 bits recessivos consecutivos.

5. ANÁLISE DA UTILIZAÇÃO DA LARGURA DE BANDA DA ARQUITETURA SCANIA

5.1 Introdução

A rede CAN (*Controller Area Network*) (BOSCH, 1991) é um padrão de fato para as aplicações distribuídas em sistemas automotivos. A arquitetura de rede automotiva dos veículos Scania utiliza a rede CAN como meio de comunicação para suas ECUs. Um mapeamento de mensagens contento sinais de grandezas físicas como velocidade, temperatura, etc, trafegam sob a rede CAN e é definido em função dos requisitos da aplicação.

É apresentada aqui uma análise de escalonabilidade das mensagens baseada no *tempo de resposta no pior caso* (*WCRT*) em condições de operação normal e sob condições de erro e em seguida uma *análise do tempo de resposta com offsets* (*WCRTo*) de mensagens (Szakely, 1993). Uma segunda análise, foi feita a implementação da arquitetura na ferramenta computacional *TrueTime,* sob o MATLAB/Simulink®, com a realização de um estudo do comportamento dinâmico das mensagens através de simulações.

Neste estudo, apresentou-se o comportamento das propriedades temporais das mensagens, que podem ser ajustadas em tempo de projeto e, conseqüentemente, dando ao projetista, condições para análise, projeto e avaliação do desempenho de uma rede de comunicação para aplicação automotiva. Com a ferramenta computacional *TrueTime* é possível realizar-se uma simulação integrada da *rede de comunicação*, do *nó computacional* (na forma de ECU) e da *dinâmica do sistema automotivo*.

Esta análise é estruturada da seguinte forma:

- Primeiramente, são apresentados os requisitos das aplicações de sistemas automotivas para a forma de sistemas distribuídos.

- Depois, é apresentada uma revisão das propriedades da rede CAN com análise do tempo de resposta com e sem offsets de mensagens.
- Em uma terceira etapa, mostra-se uma descrição das principais características da arquitetura de rede automotiva dos veículos Scania.
- Em seguida, apresenta-se a implementação dessa arquitetura com a ferramenta computacional TrueTime (sob o MATLAB/Simulink®), que simula, de forma integrada, um kernel de tempo real, redes de comunicação e dinâmicas de processos físicos.
- Por fim, é feita uma verificação experimental da influência do aumento da utilização da largura da banda passante da rede de comunicação Scania.

5.2 Requisitos das Aplicações Automotivas

A indústria automotiva apresenta a produção e o custo como principais características. Quanto à produção, podemos definir que milhões de carros produzidos por ano com um elevado tempo para o desenvolvimento e produção de um novo modelo. Quanto ao custo, estima-se que, futuramente, a parte eletrônica de um carro irá corresponder entre 30 a 35% do valor total do carro e que a presença de um componente como um micro-controlador poderá resultar em milhões durante um ano.

Por menor que seja o valor de economia para um componente automotivo, deve-se imaginar que, em uma linha de produção automotiva, são fabricados milhões de unidades por ano, e o resultado desta economia terá uma considerável redução dos custos de produção e para usuário final. Deve-se ainda considerar que a redução, relativa à parte mecânica, será motivada pela parte eletrônica, em função da capacidade em agregar uma quantidade maior de funções, que terão um impacto direto no consumidor.

Um carro consiste de um produto que é composto por vários subsistemas complexos. Esses subsistemas podem ser os motores, transmissão, freio, ar-condicionado, etc. Neles, há vários componentes de diferentes tipos como ECUs, sensores, atuadores, etc.

E para que o produto (caminhão) funcione de forma desejável, todos esses componentes devem estar se interagindo, a fim de que o cliente esteja satisfeito com o resultado final. Logo, os subsistemas e a satisfação dos clientes têm que ser diretamente proporcional, no âmbito relacional. Sendo assim, a expectativa do cliente e do fornecedor será atendida.

Atualmente, os sistemas automotivos apresentam uma crescente evolução de inovações tecnológicas motivadas pelo uso de sistemas eletrônicos. Alguns requisitos funcionais devem ser vistos para se ter uma melhor noção a respeito de sistemas automotivos. Esses requisitos indicam as funcionalidades presentes nos automóveis.

Quanto à parte eletrônica automotiva, existe uma distinção nos sistemas automotivos entre *body electronics* e *system electronics*. O *body electronics* diz respeito ao controle e gerenciamento das informações que não estão diretamente relacionadas com o movimento do carro, tais como, vidro elétrico, ar-condicionado, painel, aquecedor, etc. O *system electronics* diz respeito ao controle e gerenciamento das informações que estão diretamente relacionadas com o movimento do carro, tais como, freio ABS, acelerador, embreagem, etc. As unidades com poder de processamento são definidas como ECUs.

O estado atual dos sistemas eletrônicos contidos nos sistemas automotivos consiste em quatro partes principais, que são o *body electronics*, *system electronics*, *redes de comunicação* e *sensores inteligentes*.

O *body electronics* consiste em uma ECU automotiva típica e é composta de um microcontrolador de 8 ou 16 bits, com algumas centenas de bytes de RAM e 16 Kbytes de ROM, alguns processos de E/S para conectar os sensores e para controlar atuadores, e uma simples interface de rede.

O *system electronics* tem como principais funções o controle do motor e o freio ABS (*Antilock Braking System*). Os controladores de motor avançados são baseados em microcomputadores de 16 bits, com cerca de 16 Kbytes de RAM e 256 Kbytes de ROM. O

tamanho da memória está crescendo em cerca de mais de 25% a cada ano para satisfazer os requisitos das aplicações. O controle do motor computadorizado pode melhorar o desempenho do motor, fazendo com que haja uma redução no consumo de combustível e emissão de poluentes.

As *redes de comunicação* servem para interligar as ECUs e ainda não estão totalmente consolidadas, pois existem diversos padrões em função dos requisitos funcionais dos sistemas automotivos. Alguns exemplos dessas redes são: LIN, TTP, CAN, ByteFlight, Flexray, A-BUS, D2B, etc.

Os *sensores inteligentes* são motivados em função do alto volume de fabricação automotiva estão levando as indústrias de sensores a desenvolver novas soluções de baixo custo para muitos problemas de medição sob um carro. A vantagem em utilizar os sensores inteligentes é a habilidade integrada em processamento de grandezas como temperatura, nível de fluído, pressão, etc. e comunicação com outras entidades eletrônicas. Com isto, será permitido algum pré-processamento do sinal medido e simplificar problemas de interface.

Devido à vasta proliferação das novas tecnologias, que foram descritas acima, houve uma grande percepção dos pesquisadores da área computação aplicada em sistemas automotivos. Os requisitos das aplicações automotivas fazem com que seja necessária a utilização de paradigmas de computação tais como tempo real, sistemas distribuídos e tolerância à falhas, entre outros. A utilização de novos paradigmas da computação permitirá grandes benefícios que, somados, reduzirão consideravelmente o custo final de um veículo e irão melhorar o desempenho e, além disso, proporcionarão um melhor desempenho ao término do processo.

Algumas características importantes devem ser descritas para um melhor entendimento sobre as melhorias que os sistemas embarcados automotivos podem trazer para os automóveis. São elas: as funções realizadas por *hardware* em sistemas automotivos

podem ser realizadas pelos *softwares* embarcados, a utilização do paradigma de sistemas de tempo-real contribui para um projeto otimizado de sistemas embarcados, reduzindo assim custos de produção, o uso de redes em veículos automotivos dá uma redução interessante em peso e volume, controle em tempo real distribuído através de ECUs, uso de memória flash – para permitir atualizações dos *softwares* após a venda, o paradigma de sistemas distribuídos tais como, hierárquico, cache de CPU (*Central Processing Unit*), cliente-servidor, conjunto de processadores e orientado ao fluxo de dados.

É importante detalhar o que seriam esses paradigmas de sistemas distribuídos para um melhor entendimento de suas funções. São eles:

- Hierárquico: este se caracteriza por dispor os vários processadores de acordo com uma organização em árvore. É organizado de maneira que a capacidade computacional dos processadores seja maior à medida que esses processadores estejam mais próximos da raiz da árvore. Sendo assim, os processadores mais distantes da raiz tratam de serviços mais específicos e especializados, enquanto que os processadores mais próximos da raiz tratam de serviços mais globais.
- Cachê de CPU: caracteriza-se pelo fornecimento, ao usuário, de dois níveis de capacidade computacional: o primeiro possui menor capacidade e o segundo maior. Isso é conseguido conectando-se o usuário a um computador de menor capacidade, sendo que esse estará conectado a um computador central de grande capacidade. O sistema operacional decidirá em qual computador determinado o serviço será realizado, de acordo com alguns dados, tais como: adequação de cada máquina, custo relativo de cada máquina, taxa de transmissão entre as máquinas e carga de trabalho de cada máquina.
- Cliente-servidor: caracteriza-se pela não existência de um computador central (ao contrário da cache de CPU), e por existir vários computadores clientes e vários computadores servidores conectados através de um subsistema de comunicação. Os computadores clientes comandam a execução da aplicação, abortando quando necessário, os computadores servidores, a fim de que esses

realizem algumas funções específicas. Servidores de arquivos, servidores de banco de dados e servidores de impressão são exemplos típicos de processos que residiriam em processadores servidores.

- Conjunto de processadores: caracteriza-se pela existência de um conjunto de processadores disponíveis a todos os usuários, os quais estão conectados diretamente ao subsistema de comunicação. Esse paradigma é mais otimizado que o "cliente-servidor", pois evita que um processador seja dedicado exclusivamente a um usuário ou a um conjunto de usuários. Dessa forma, é possível uma distribuição balanceada dos serviços requisitados pelos usuários entre os vários elementos processadores, evitando que exista grande diferença de carga entre esses elementos processadores.
- Orientado ao fluxo de dados: caracteriza-se por ser muito mais radical que os outros paradigmas. Nele, um programa é descrito por um grato (instrução do programa), e cada nó do grato é designado a um elemento processador. Os resultados gerados são enviados a outros elementos como mensagens. As instruções são executadas de maneira assíncrona, dependendo apenas da disponibilidade dos dados. Isso significa dizer que a otimização apresenta o desempenho máximo. Esse paradigma objetiva alcançar um paralelismo de granularidade fina, ou seja, os processos que executam em paralelo são aproximadamente de tamanho de uma instrução de uma máquina convencional.

As tecnologias de redes automotivas definida pela Sociedade de Engenharia Automotiva (*Society of Automotive Engineers* – SAE) descrevem uma classificação de redes de comunicação em função dos requerimentos das aplicações automotivas. Esta classificação define três classes de comunicação em aplicações automotivas, que são:

- Classe A: São redes de comunicação com baixa largura de banda utilizada em aplicações não críticas (ou seja, na qual não está presente o risco de algum tipo de perda) no corpo eletrônico do veículo como, por exemplo, controle de

lâmpadas, diagnósticos, etc. Veja o exemplo de uma rede do tipo classe A na figura que segue:

Figura 39 - Uma arquitetura de rede automotiva Classe A

- Classe B: São redes utilizadas para aplicações que são importantes, mas não essenciais, para a operação do automóvel, como display de informação de velocidade e nível de combustível. As Classes A e B são aplicadas no corpo eletrônico de um automóvel. Veja o exemplo de uma rede do tipo classe B na figura abaixo:

Figura 40 - Uma arquitetura de rede automotiva Classe B

- Classe C: São redes utilizadas em aplicações de segurança crítica (ou seja, na qual está presente algum tipo de perda) de tempo real distribuído envolvido no sistema eletrônico de um automóvel, como por exemplo: controle de direção, freios e motor. O volume de dados é alto, exigindo baixa latência e alta taxa de transferência. Veja o exemplo de uma rede do tipo classe C na figura a seguir:

Figura 41 - Uma arquitetura de rede automotiva Classe C

Mais especificamente, o objeto da análise concentra-se numa rede automotiva Classe C da Scania Latin América Ltda.

Como o Capítulo 4 foi dedicado à descrição integral das características de uma rede CAN (*Controller Area Network*), será feita, a seguir, a descrição das mensagens CAN.

5.3 Modelo de Mensagens

Nós assumimos que uma rede CAN possua *r* estações e *n* fluxos de mensagens, sendo cada fluxo definido por:

$$S_m = (C_m, T_m, D_m) \quad \textbf{(7)}$$

Um fluxo de mensagem é uma seqüência temporal de mensagens relativas à, por exemplo, leitura remota de uma variável de processo específico. Um fluxo de mensagem S_m é caracterizado por um único identificador. C_m é a maior tempo de duração da mensagem do fluxo S_m. T_m é a periodicidade de transmissão considerada como o intervalo de tempo mínimo entre duas chegadas consecutivas de requisições para a fila de saída do fluxo S_m. Finalmente, D_m é a meta temporal relativa de uma mensagem (deadline) do fluxo S_m, que consiste no intervalo de tempo máximo admissível entre o instante em que a requisição da mensagem é colocada na fila de saída e o instante quando cada mensagem é completamente transmitida.

5.4 Análise de Escalonabilidade de Mensagens

Liu e Layland (Liu & Layland, 1973) mostraram que o algoritmo RM (*Rate Monotnic*) é um algoritmo ótimo entre todos os algoritmos de prioridade fixa para tarefas independentes e com período igual ao *deadline*, no sentido de que nenhum outro algoritmo de prioridade fixa pode escalonar um conjunto de tarefas que não possa ser escalonado pelo RM.

A mesma condição pode ser verificado se considerarmos um conjunto de mensagens sobre a rede ao invés de um conjunto de tarefas sobre uma CPU, desde que seja considerado um ambiente de escalonamento não-preemptivo (visto que o escalonamento de mensagens é não-preemptivo no nível do formato de mensagem).

Um teste de escalonabilidade foi proposto em Sha & Lechosky (Sha & Lechosky, 1987) e descrito a seguir. Para um conjunto de mensagens não-preemptivas e periódicas, com prioridade decrescente são escalonáveis se para todo $m = 1, \ldots, N$.

$$\frac{C_m}{T_m} + \frac{C_2}{T_2} + \bullet\bullet\bullet + \frac{C_i}{T_i} + \frac{\bar{b}_{l,m}}{T_m} \leq m \cdot \left(2^{\frac{1}{m}} - 1 \right) \qquad \textbf{(8)}$$

onde $b_{l,i}$ é o tempo de bloqueio no pior caso por uma tarefa de prioridade mais baixa ou seja,

$$\bar{b}_{l,m} = \max_{j=i+1,\ldots,N} T_j \qquad \textbf{(9)}$$

5.5 Análise do Tempo de Resposta de Mensagens sem *Offsets*

Em Tindell *et al.* (1995), os autores apresentaram em detalhes as análises do tempo de resposta de redes CAN. Eles assumiram que as mensagens possuíam prioridade fixa para os respectivos fluxos (acesso à rede baseado sobre a prioridade do identificador, assumindo que o cada fluxo de mensagens é atribuído um identificador único na rede) e um modelo de escalonamento não preemptivo (ou seja, as mensagens com menor prioridade sendo transmitidas não serão interrompidas por mensagens de maior prioridade). Considerando tal modelo de escalonamento, eles adaptaram todas as análises de escalonabilidade existentes para o escalonamento de tarefas (Audsley *et al.* (1993)) para o caso de escalonamento de mensagens sobre uma rede CAN.

O tempo de resposta no pior caso de uma mensagem enfileirada, medida desde a chegada de uma requisição da mensagem para a fila de saída até o instante em que a mensagem é completamente transmitida, é dado em **(10):**

$$R_m = J_m + w_m + C_m \qquad \textbf{(10)}$$

O termo C_m representa o tempo gasto para se transmitir uma mensagem *m* fisicamente sobre o barramento. O pacote de uma mensagem CAN básica, contém 47 bits de overhead por mensagem, e um *stuffing* de 5 bits, sendo apenas 34 bits dos 47 bits de overhead que estão sujeitos ao *stuffing*. O termo C_m é apresentado a seguir.

$$C_m = \left(\left\lfloor \frac{34 + 8.s_m}{5} \right\rfloor + 47 + 8.s_m \right) \cdot \tau_{bit} \qquad \textbf{(11)}$$

Em **(11),** temos que o termo s_m *indica* o tamanho limitado da mensagem *m* em bytes, e o termo τ_{bit} é o atraso de propagação, ou seja, o tempo necessário para se transmitir

um bit sobre o barramento (como exemplo, sob um barramento a *1Mbit/s*, tem-se que τ_{bit} = $1\mu s$).

O termo w_m em **(10)** representa o atraso na fila no pior caso (o maior tempo entre a inserção de uma mensagem na fila de prioridades e o início de sua transmissão) e é dado por **(12):**

$$w_m = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m + J_j + \tau_{bit}}{T_j} \right\rceil \cdot C_j \qquad \textbf{(12)}$$

Onde o termo *hp(m)* é o conjunto de mensagens no sistema com prioridade maior que a mensagem *m*, e B_m é o tempo de bloqueio no pior caso da mensagem *m*, e é dado por **(13).**

$$B_m = \max_{\forall k \in lp(m)} (C_k) \qquad \textbf{(13)}$$

De **(7)** tem-se que *lp(m)* é o conjunto de fluxos de mensagens no sistema com prioridade menor que o fluxo de mensagem S_m.

Em **(6),** o termo J_j corresponde ao *Jitter* de transmissão e T_j é o período de ativação do conjunto de mensagens com prioridade maior que a mensagem *m*.

Como o termo w_m em **(6)** aparece em ambos os lados da equação, a solução requer, assim, a utilização de uma relação de recorrência para sua solução, conforme apresentado em **(14).**

$$w_m^{n+1} = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m^n + J_j + \tau_{bit}}{T_j} \right\rceil \cdot C_j \qquad \textbf{(14)}$$

onde o termo *hp(m)* é o conjunto de mensagens com prioridade maior que a mensagem *m*.

Finalmente, o termo J_m em **(10),** representa o *Jitter* da mensagem (tempo médio de espera de uma mensagem na fila de transmissão antes do escalonamento). Em redes CAN é

possível determinar atrasos de transmissão máximos, sob a presença de carga, através de prioridade sob mensagens.

5.6 Arquitetura Scania

O sistema de comunicação de um veículo Scania é dividido em três barramentos, conectados por uma porta (*gateway*) chamada de *Coordinator* (COO), conforme a figura 42. Os barramentos são chamados de Barramento Vermelho (*Red Bus*), Barramento Verde (*Green Bus*) e Barramento Amarelo (*Yellow Bus*). Essa divisão se dá pelo grau crítico dos sistemas a eles conectados. Sendo assim, o Barramento Vermelho é o mais crítico, por conter os sistemas vitais para o veículo, e assim por diante. Dessa maneira, caso alguma ECU do Barramento Vermelho falhe, o veículo pára.

O número de ECUs do veículo pode variar, mas há um número mínimo de ECUs que são necessárias para o funcionamento do veículo, que são:

- APS – Air Processing System (Sistema de Processamento de Ar);
- ICL – Cluster (Instrumento Combinado);
- TCO – Tacógrafo;
- EMS – Engine Management System (Sistema de Gerenciamento do Motor);
- COO – Coordinator (Coordenador).

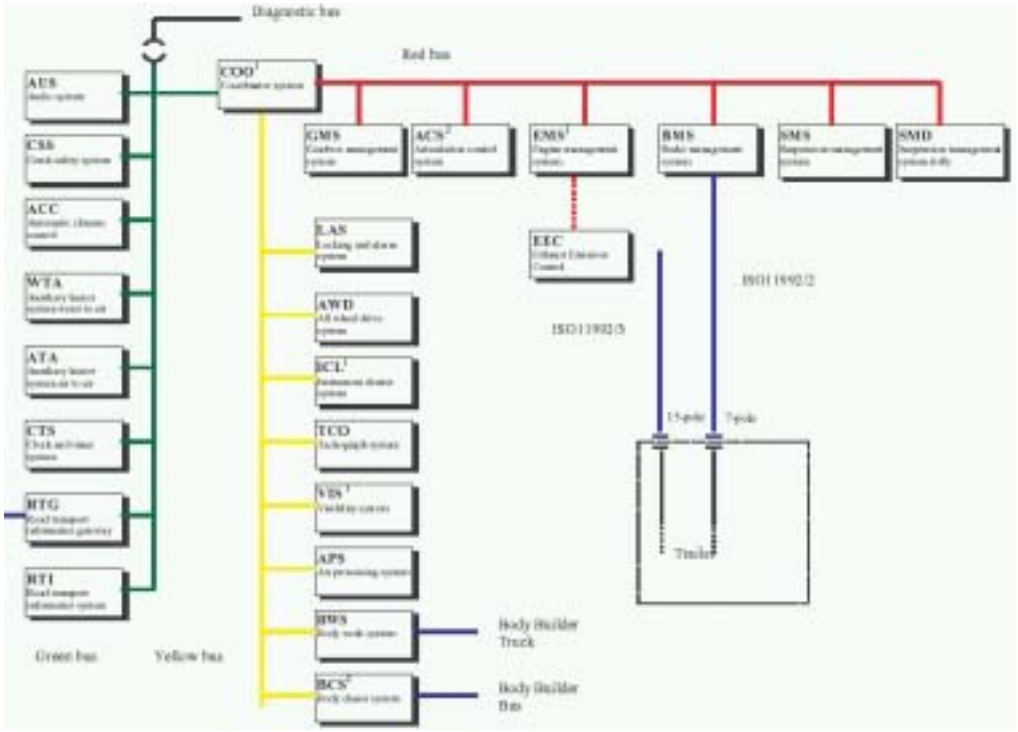


Figura 42 - Arquitetura distribuída de um veículo Scania.

Na arquitetura utilizada atualmente, há uma média de utilização da largura de banda na faixa de 30 a 40%, que corresponde a aproximadamente 10% da largura de banda total (a 250Kbps). Assumindo que, no futuro, o número de ECUs aumente consideravelmente, com o desenvolvimento de sistemas de segurança, tanto ativos quanto passivos, e com a utilização de sistemas *By-Wire,* passaremos a ter um "consumo" de largura de banda em torno de 50 a 80%, o que é considerado excessivo para garantia de que mensagens de prioridade inferior cheguem aos seus destinos.

5.7 Análise do Tempo de Resposta da Arquitetura Scania

A simulação dinâmica da Arquitetura Scania sobre a rede CAN, é realizada com a ferramenta computacional CANalyser (Tindell, 1994) onde as simulações são válidas para controladores Intel 82752.

5.8 Consumo Total de Largura de Banda

A seguir será analisado o teste de escalonabilidade para o conjunto de mensagens apresentado na Tabela 1, para diferentes taxas de transmissão da rede CAN. Será visto também a utilização das mensagens e do barramento com o objetivo de verificar-se a viabilidade de utilização de uma rede CAN como solução para a arquitetura Scania. Será também considerado que o sistema em análise está operando sob uma rede CAN sujeita as certas condições de erro (n_{error}=5, T_{error}=10ms).

ID	Descrição da Menssagem	C_i (byte)	J_i (ms)	T_i (ms)	D_i (ms)	Tipo	Bits/ mensagem	Bits/ segundo
$0C F0 02 03	ETC1	8	1	10	10	P	130	13000
$0C 00 00 0B	TSC1-AE	8	1	10	10	P	130	13000
$18 F0 09 0B	VDC2	8	1	10	10	P	130	13000
$0C F0 02 43	ETC1-CNV	8	1	10	10	P	130	13000
$0C 00 00 43	TSC1-CNVE	8	1	10	10	P	130	13000
$0C 00 00 21	TSC1-LE	8	1	10	10	P	130	13000
$0C 00 00 03	TSC1-TE	8	1	10	10	P	130	13000
Sub-Total								91000
$0C F0 04 00	EEC1	8	1	20	20	P	130	6500
$1C F0 0C 03	ETC8	8	1	20	20	P	130	6500
$0C FE 6C EE	TCO1	8	1	20	20	P	130	6500
$0C FF C8 03	TCTR	8	1	20	20	P	130	6500
Sub-Total								30000
$0C F0 03 00	EEC2	8	1	50	50	P	130	2600
$18 EB FF 00	Engine Configuration	8	1	50	50	P	130	2600
$18 EB FF 10	Retarder Configuration – RD	8	1	50	50	P	130	2600
$18 EB FF 29	Retarder Configuration – REX	8	1	50	50	P	130	2600
$0C 00 10 0B	TSC1-ARD	8	1	50	50	P	130	2600
$0C 00 29 0B	TSC1-AREX	8	1	50	50	P	130	2600
$0C 00 10 27	TSC1 KRD	8	1	50	50	P	130	2600
$0C FF 19 0B	EBC2 Proprietary	8	1	50	50	P	130	2600
$0C 00 29 10	TSC1-RDREX	8	1	50	50	P	130	2600
$0C 00 29 03	TSC1-TREX	8	1	50	50	P	130	2600
Sub-Total								26000
$0C FE 5A 27	ASC1-K	8	1	100	100	P	130	1300
$18 FE 59 2F	ASC3-F	8	1	100	100	P	130	1300
$18 FF FB 21	BCS Message 2	8	1	100	100	P	130	1300
$18 FE F1 00	Cruise Control/Vehicle Speed - E	8	1	100	100	P	130	1300
$18 F00 01 0B	EBC1-A	8	1	100	100	P	130	1300
$18 FD C4 0B	EBC5	8	1	100	100	P	130	1300
$18 FE C4 yy	EBS 22	8	1	100	100	P	130	1300
$18 FE C6 yy	EBS 23	8	1	100	100	P	130	1300
$18 F0 00 10	ERC1-RD	8	1	100	100	P	130	1300
$18 F0 00 29	ERC1-REX	8	1	100	100	P	130	1300
$18 F0 05 03	ETC2	8	1	100	100	P	130	1300
$18 E4 yy 2F	RGE 11	8	1	100	100	P	130	1300

$18 E5 20 yy	RGE 21	8	1	100	100	P	130	1300
$18 FE 4F 0B	VDC1	8	1	100	100	P	130	1300
$18 FE EA 2F	Vehicle Weight	8	1	100	100	P	130	1300
$18 FE F1 27	Cruise Control/Vehicle Speed - K	8	1	100	100	P	130	1300
$18 FE F2 00	Fuel Economy	8	1	100	100	P	130	1300
$0C FF A2 10	Retarder Proprietary 1	8	1	100	100	P	130	1300
Sub-Total								23400
$0C FF B0 27	Coordinator General Information	8	1	200	200	P	130	650
Sub-Total								650
$18 FE DF 00	EEC3	8	1	250	250	P	130	520
$18 FF 90 27	PTO Information - K	8	1	250	250	P	130	520
Sub-Total								1040
$18 F0 06 27	EAC1-K	8	1	500	500	P	130	260
$18 FE EF 00	Engine Fluid Level / Pressure	8	1	500	500	P	130	260
$18 FE F6 00	Inlet/Exhaust Conditions	8	1	500	500	P	130	260
Sub-Total								780
$18 FE F5 27	Ambient Conditions	8	1	1000	1000	P	130	130
$18 EC FF 00	BAM-E	8	1	1000	1000	P	130	130
$18 EC FF 10	BAM-RD	8	1	1000	1000	P	130	130
$18 EC FF 29	BAM-REX	8	1	1000	1000	P	130	130
$18 FF B1 1E	CUV Information	8	1	1000	1000	P	130	130
$18 FE CA 0B	DM1–A	8	1	1000	1000	P	130	130
$18 FE AE 30	Supply Pressure	8	1	1000	1000	P	130	130
$18 FE E6 17	Time/Date	8	1	1000	1000	P	130	130
$18 E0 00 19	CM1-CCE	8	1	1000	1000	P	130	130
$18 E0 00 10	CM1-RDE	8	1	1000	1000	P	130	130
$18 FE FC 27	Dash Display	8	1	1000	1000	P	130	130
$18 FE E5 00	Engine Hours, Revolutions	8	1	1000	1000	P	130	130
$18 FE EE 00	Engine Temperature	8	1	1000	1000	P	130	130
$18 FE E9 00	Fuel Consumption	8	1	1000	1000	P	130	130
$18 FE C1 EE	High Resolution Vehicle Distance	8	1	1000	1000	P	130	130
Sub-Total								1950
$18 FF 60 0B	General Purpose Message 1	8	1	5000	5000	P	130	130
$1C FE AC 0B	Wheel Brake Lining Remaining Inf. EBC4	8	1	5000	5000	P	130	130
Sub-Total								260
$18 DB zz xx	KWP2000 Functional Addressing	8	1	-	-	E	130	130
$18 DA 0B xx	KWP2000 Physical Addressing	8	1	-	-	E	130	130
$18 DA xx 0B	KWP2000 Physical Addressing	8	1	-	-	E	130	130
$0C FF 85 00	DLN3 – E	8	1	-	-	E	130	130
$0C FF 85 27	DLN3 - K	8	1	-	-	E	130	130
Sub-Total								650
Total bits/seg								**175730**

Tabela 1 - Análise do tempo de resposta para mensagens em uma rede CAN – sob operação normal e sob condições de erros

A largura de banda mínima necessária para suportar o conjunto de mensagens apresentado na Tabela 1 é de 175.730 bits/seg. Para um baud-rate de 250Kbps, tem-se uma

utilização da largura de banda da rede 70,3% e, se adotarmos um baud-rate para 500 Kbps, tem-se uma 35,1% de utilização da largura de banda da rede.

É importante verificar que a utilização da rede e análise de escalonabilidade pode ser verificada logo a seguir, de forma previsível e coerente, para ambas as condições de funcionamento da rede CAN.

Para condições normais de funcionamento, os *deadlines* são cumpridos para uma velocidade de transmissão de 250 Kbit/s (para a utilização da rede 70,3% no pior caso).

Para uma rede CAN sujeita a erros de transmissão, a utilização da rede tem uma ligeira subida (devida à retransmissão de mensagens no caso da ocorrência de erros de transmissão), sendo que o conjunto de mensagens agora também não é escalonável para uma velocidade de transmissão de 250 Kbit/s.

5.9 Simulação Dinâmica da Arquitetura Scania

Apresenta-se, nesta seção, as principais características da ferramenta computacional *TrueTime* e em seguida, a implementação da arquitetura Scania, com a realização de uma análise da simulação dinâmica para o conjunto de mensagens.

5.9.1 TrueTime

O TrueTime é uma ferramenta computacional na forma de simulador, baseado em Matlab/Simulink® , desenvolvido inicialmente para aplicações de sistemas de controle em tempo-real, onde o problema de escalonamento de tarefas, com requisitos temporais, e a dinâmica do processo eram necessários realizar-se de forma integrada.

As versões atualizadas do TrueTime disponibilizam mecanismos para a simulação de kernel de tempo real, redes de comunicação e dinâmica de processos físicos, dando condições aos projetista avaliar o desempenho de forma integrada para diferentes

algorítmos de escalonamento em kernel de tempo, diferentes protocolos de redes de comunicação e a implementação de diversos tipos de processos físicos.

Selecionou-se o TrueTime como ferramenta computacional para implementação da arquitetura Scania, em face da facilidade em implementar-se uma arquitetura de rede automotiva.

Como principais características do simulador têm:

- Desenvolvido em C++ MEX (rápido).
- Simulação baseada em eventos (rápido).
- Interrupções externas.
- Bloco de rede (Ethernet, CAN, TDMA, FDMA, Round Robin e rede ethernet comutada).

Possibilidade de escrever tarefas como arquivos **.M** ou funções em C++ e também possível chamar diagrama em blocos no MATLAB/Simulink® dentro de funções de código.

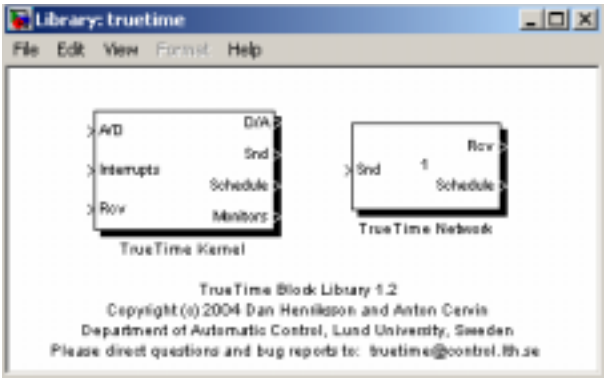


Figura 43 - A biblioteca de blocos do TrueTime.

O bloco *TrueTime Kernel* implementa um kernel de tempo real que pode utilizar de algoritmos de escalonamento de tarefas, tais como executivo cíclico, RM (*Rate*

Monotonic), DM (*Deadline Monotonic*) ou EDF (*Earliest Deadline Monotonic*). Cada bloco pode ter interfaces com conversores A/D (Analógico/Digital) e D/A (Digital/Analógico), tratadores de interrupção, monitores para recursos compartilhados, interfaces para redes de comunicação. Estes blocos podem implementar funções de ECU automotivas.

O bloco *TrueTime Network* implementa um tipo de rede de comunicação que interliga os nós computacionais (Blocos *TrueTime Kernel*) com interfaces de ligação. Podem ser simuladas as seguintes redes de comunicação: Ethernet, CAN, TDMA, FDMA, Round Robin e Switch Ethernet.

5.9.2 Implementação da arquitetura Scania no TrueTime

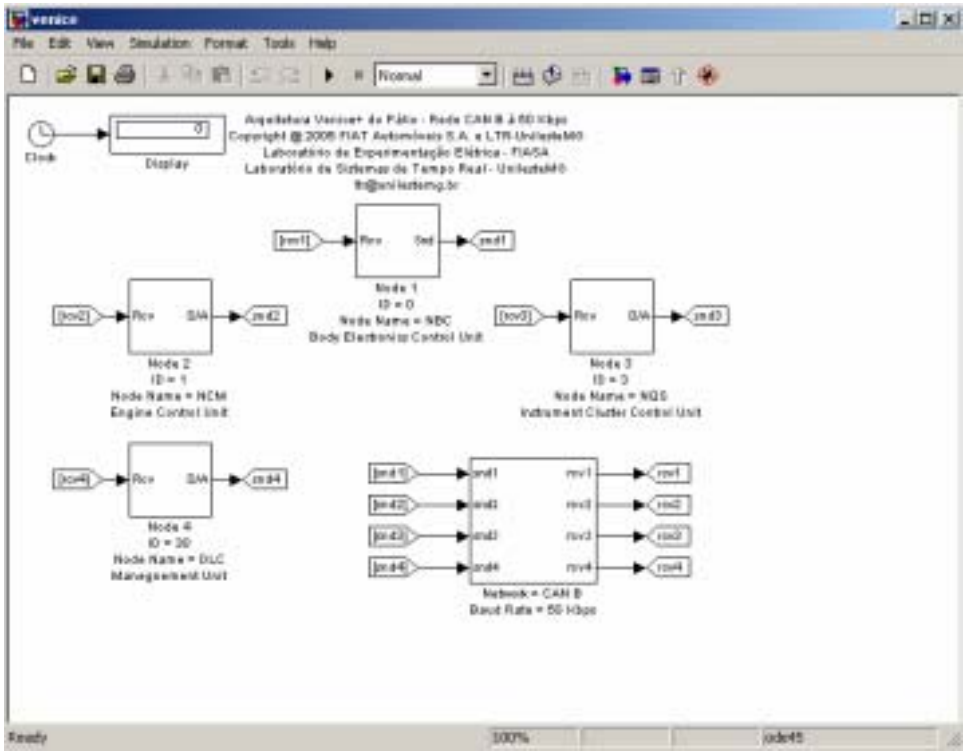


Figura 44 - Implementação da arquitetura Scania no TrueTime.

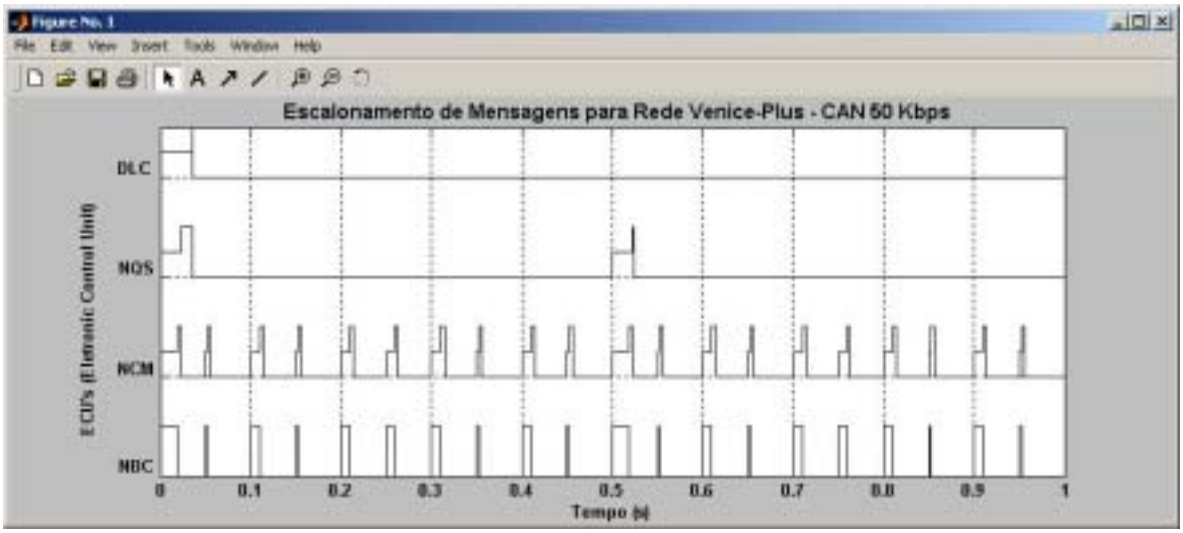


Figura 45 - Escalonamento de mensagens da arquitetura Scania no TrueTime.

5.10 Parte experimental

O objetivo dos testes de verificação é analisar como o atraso de mensagens em um barramento de um veículo Scania se relaciona com o aumento da utilização da largura de banda.

5.10.1 Equipamentos

Para a realização dos testes, foi utilizado um computador portátil (Laptop), que foi conectado à central elétrica do veículo, através de uma interface denominada FMS (*Fleet Management System*), e uma ferramenta de análise denominada de CANalyzer, do fabricante Vector, versão 8.0, de 2005, conforme as figuras 46 e 47.

Figura 39 - Equipamentos utilizados nas medições em um veículo Scania.

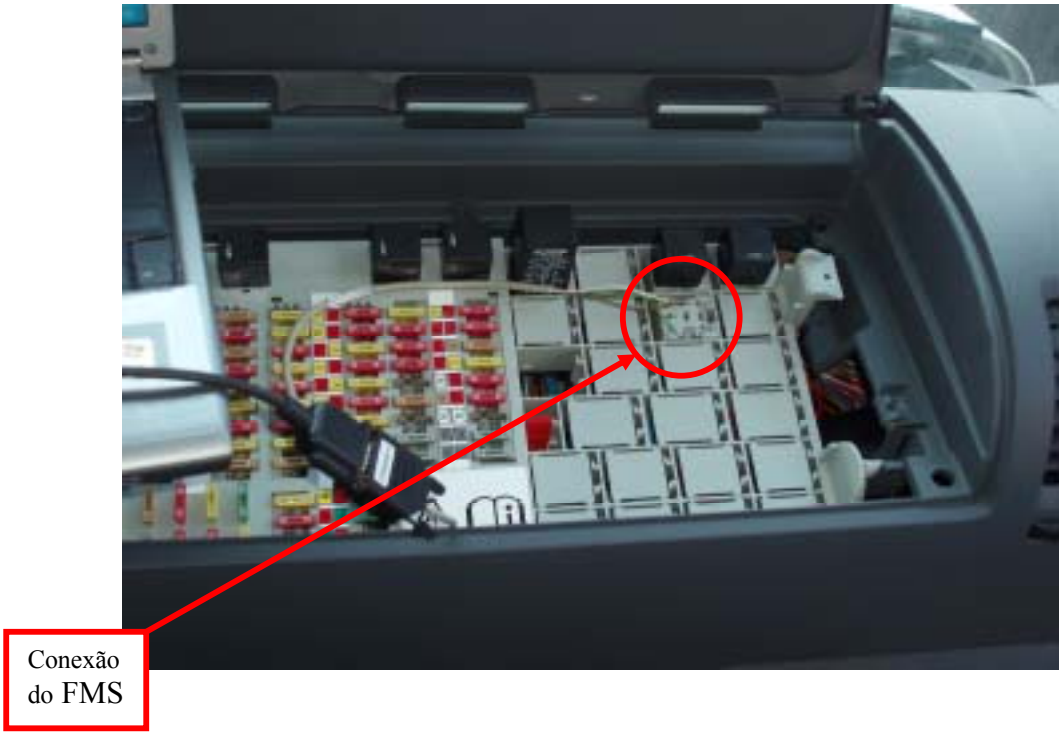


Figura 40 - Conexão da ferramenta de análise (CANalyzer) nos terminais do FMS, para acesso ao barramento verde.

O CANalyzer pode enviar mensagens criadas (*dummy*) e apresentar estatísticas de desvios do período de tempo entre duas mensagens.

5.10.2 Resultados esperados

Espera-se que o atraso em mensagens aconteça à medida que se aumenta a carga do barramento, pois a probabilidade de que elas se bloqueiem é grande. Uma situação de bloqueio ocorre quando uma mensagem é transmitida no barramento e uma outra ECU tenta enviar uma mensagem antes da anterior ser terminada. Mensagens de alta prioridade devem ter atrasos menores que as mensagens de menor prioridade.

A taxa de envio de mensagens pode ser calculada da seguinte forma (WAERN, M., 2003):

$$Taxa\ de\ mensagem = \frac{Taxa\_de\_bit}{Comprimento\_de\_uma\_mensagem} = \frac{250.10^3}{129} = \quad \textbf{(16)}$$

= 1.937 mensagens por segundo ou 0,516 ms por mensagem.

Isso significa que o tempo entre duas transmissões de mensagens com a mais alta prioridade pode variar de duas vezes 0,516 ms mais a variação da ECU que está enviando a mensagem (≈ 1,032 ms).

Mensagens com segunda mais alta prioridade podem ser atrasadas de duas mensagens (2 x 2 x 0,516 ms). Mensagens com prioridade *n* podem ser atrasadas de 2.*n*.0,516 ms.

5.10.3 Coleta de dados

A coleta de dados é feita através do seguinte procedimento:

a. Criar um bloco gerador de mensagens que possui mensagens de acordo com a norma SAE J1939;
b. Ativar o bloco Statistics para cálculo dos desvios-padrão médio, mínimo e máximo;
c. Selecionar as mensagens de interesse e os períodos de envio de cada mensagem;
d. Ligar o veículo e permanecer em marcha lenta;
e. Ativar a janela Bus Statistics para verificar qual é a carga do barramento;
f. Rodar o CANalyzer com as mensagens dummy;
g. Anotar a carga do barramento, em porcentagem;
h. Parar o CANalyzer. Olhar a janela Write. As estatísticas para todas as mensagens estarão informadas no barramento observado. Copiar e colar em uma planilha Excel.
i. Repetir os itens f, g e h para diferentes cargas do barramento.

Será feito o monitoramento de duas mensagens no barramento:

- EEC1 – rotação do motor, com ID 0x1f0f004, de prioridade alta (3), enviado com uma freqüência de 20 ms e;
- Radio Control, que ativa o sinal Mute (mudo) do rádio no painel, com ID 0x1f0ffd0, de prioridade baixa (6), enviado com uma freqüência de 1000 ms.

5.10.4 Resultados e Análise dos Testes

A figura abaixo mostra que os atrasos nas mensagens aumentam à medida que se aumenta a carga do barramento:

- Mensagem EEC1 – Rotação do motor

Busload [%]	81.72
Peakload [%]	82.74
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	1407
Ext. Data [total]	152316
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

```
Start of measurement 02:21:52 pm
CAN 1 Bus with 250000 BPS.
CAN 2 Bus with 250000 BPS.

Statistics report AR0004, 02:21:52 pm
Statistics for transmit spacing of messages in [ms]

                    N     Aver     StdDev   MIN     MAX

f004P 00    RX    7765  20       1.3791   15.79   23.80

End of measurement 02:24:28 pm
```

Figura 41 – Relatório de teste para a mensagem EEC1 – rotação do motor.

Conforme o cálculo teórico feito anteriormente, uma mensagem com prioridade *n* pode ser atrasada de 2.*n*.0,516 ms, podemos calcular que a mensagem EEC1, da rotação do motor com prioridade 3, pode ter atraso máximo de 2.3.0,516 ms, resultando em 3,096 ms. Pelo relatório de medição apresentado acima, podemos verificar que o tempo entre duas mensagens foi de 8,01 ms, ou seja, ela foi bloqueada por duas mensagens de maior prioridade, devendo aguardar um certo tempo para poder ser transmitida.

- Radio Control

Verificou-se que a mensagem começa a atrasar com uma carga de largura de banda em torno de 60%, como mostra a figura abaixo:

Busload [%]	59.40
Peakload [%]	59.68
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	1024
Ext. Data [total]	15262
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

```
Start of measurement 02:18:09 pm
CAN 1 Bus with 250000 BPS.
CAN 2 Bus with 250000 BPS.
-----
Statistics report AR0003, 02:18:09 pm
Statistics for transmit spacing of messages in [ms]

                        N      Aver      StdDev    MIN      MAX

ffd0P 27     RX     62     1000.1    0.90859   997.06   1003.86

End of measurement 02:19:13 pm
```

Figura 42 – Relatório de teste para a mensagem *Radio Control*.

Da mesma maneira, podemos calcular que a mensagem *Radio Control*, de prioridade 6, pode ter atraso máximo de 2.6.0,516 ms, resultando em 6,192 ms. O relatório acima mostra que o atraso atingiu 6,8 ms, o que significa que ela foi bloqueada por uma mensagem de maior prioridade.

Podemos verificar, através do gráfico abaixo, como os atrasos das várias mensagens que trafegam no barramento do veículo, aumentam proporcionalmente com o aumento do número de mensagens.

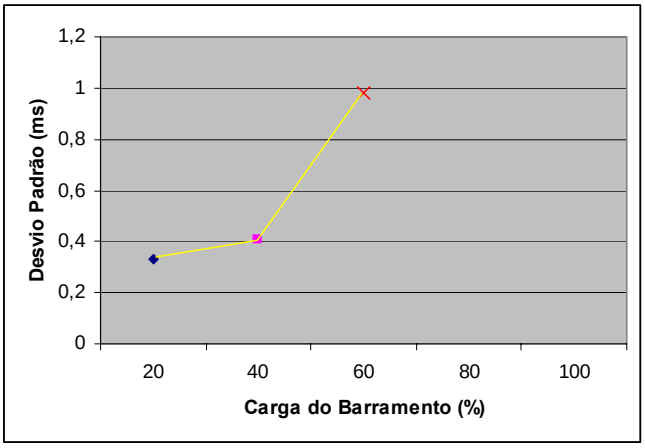


Gráfico 01 – Desvio padrão versus carga do barramento

Os resultados dos testes são dependentes de vários fatores, tais como freqüências de envio de mensagens, número de mensagens enviadas, entre outros. Porém, no veículo, na prática, as situações de envio de mensagens não são “controláveis”, pois variam à medida que os comandos do motorista e as necessidades da utilização do veículo requerem do sistema de comunicação.

O teste prático no veículo demonstra como as mensagens podem se comportar em situações críticas de alta carga do barramento, e que pode ser ainda pior caso as várias ECUs tentem enviar mensagens ao mesmo tempo e em uma mesma freqüência. Assim, à taxa de transmissão especificada atualmente, que é de 250 Kbps, anteriormente analisada por cálculo de escalonamento (verificada como estando em uma posição próxima do limite, para condições de erro), e com o aumento do número de ECUs no sistema de comunicação para atender novas funções da aplicação veicular pesada, é esperado que mensagens de menor prioridade sejam bloqueadas por mensagens de maior prioridade, como aconteceu no experimento, podendo acarretar em problemas para o veículo e seu condutor.

As soluções possíveis para resolver esses problemas serão explanadas a seguir, no capítulo 6.

6. CONCLUSÕES E PROPOSTAS PARA SOLUÇÃO DO PROBLEMA

Como conclusão deste estudo, foi demonstrado que o aumento da carga da largura de banda passante, dos Protocolos CAN e J1939 utilizados atualmente pela Scania, implica no atraso do envio das mensagens de prioridades mais baixas, simulando situações de aumento do número de sistemas e, conseqüentemente, um aumento na quantidade de ECUs no veículo.

Muitos caminhos podem ser tomados para que esse problema seja resolvido, porém indicarei duas alternativas, descritas a seguir:

1) Aumento da taxa de transmissão

Como hoje a taxa de transmissão de um veículo Scania é de 250Kbps, é possível aumentá-la para 500Kbps, fazendo com que a carga da banda passante diminua de 30% para 15%.

Porém, a desvantagem dessa alternativa é que o aumento da taxa de transmissão implica em maior vulnerabilidade à Interferência Eletromagnética (IEM) na área de emissões irradiadas, onde a taxa de transmissão é a mais crítica. Alguns testes realizados em laboratório na Matriz da Scania, na Suécia, têm demonstrado dificuldades em manter o sistema de comunicação suficientemente robusto (JOHANSSON, J., 2006) com a utilização do meio físico adotado, que é o par trançado de fios sem blindagem (*unshielded*).

Uma maneira de tentar atingir a robustez seria a utilização de um par trançado de cabos blindados (*shielded*) de maneira a cancelar a componente elétrica do campo gerado, aliado ao par trançado, que cancela a componente magnética do mesmo campo. A outra maneira é a substituição do meio físico de transmissão, de par trançado para fibra óptica, que não apresenta limitação do aumento da taxa de transmissão.

2) Mudança do Protocolo de Comunicação

A segunda alternativa é a mudança dos protocolos atuais, o CAN clássico e a J1939, caracterizados por serem *Event-Triggered*, como descrito neste trabalho, para um protocolo *Time-Triggered*, que alia a utilização de uma taxa de transmissão de até 10Mbps e o desacoplamento do aumento da taxa de transmissão com o aumento da carga da largura de banda, tais como o Flexray, o TTP e o Time-Triggered CAN. Vários testes comparativos foram feitos na Europa e na Scania também comprovando esta teoria.

Devido à característica de envio de mensagens periódicas em segmentos de tempos (*slots*), mecanismo TDMA, há a garantia de que, mesmo mensagens de baixa prioridade serão transmitidas com atraso mínimo, bastante inferior às mensagens do CAN ativado por tempo (protocolo determinístico).

Outra vantagem está na flexibilidade de enviar, além das mensagens periódicas nas janelas exclusivas, mensagens esporádicas do CAN clássico (PALBUS, 2005).

O lado negativo que considero hoje é o alto custo das ECUs, que fez com que tais protocolos ainda não emplacassem no Mercado Europeu, e a falta de maturidade dos mesmos, mas que apresenta uma forte tendência de se tornarem os protocolos que substituirão o CAN clássico no futuro.

Pesquisas futuras

É possível migrar do sistema CAN para o TT-CAN através da utilização de um *Gateway* desenvolvido até que o protocolo seja introduzido no Mercado (ALBERT, A.; STRASSER, R.; TRÄCHTLER, A., 2005). Fabricantes que utilizam hoje o sistema CAN e sentirem necessidade técnica, através da análise de carga da largura de banda passante e da taxa de transmissão, de mudar para um sistema determinístico, podem aproveitar suas características e utilizar o TT-CAN, ou seja, suporte de programação estática em tempos pré-definidos e gerenciamento de tempo global.

Porém, ao efetuar a migração do CAN para o TT-CAN, deve-se fazer uma análise detalhada da programação das mensagens nas matrizes de tempo (ciclos básicos de transmissão) para cada aplicação, pesada ou leve, para poder extrair o máximo de desempenho com o menor atraso possível. Podemos falar em minimizar o número de ciclos básicos aumentando seu tamanho, ou ainda o contrário, minimizar o tamanho do ciclo básico e aumentar o número de ciclos.

Outra análise deve ser levada em conta: a alocação das mensagens nos sistemas pesados e leves na mesma matriz deve ser observada, pois mensagens regulares CAN (leve) também devem apresentar atraso mínimo. Portanto, o estudo deveria avaliar qual é a penalidade, em porcentagem, se utilizarmos mensagens regulares CAN no TTCAN perdendo largura de banda.

Saindo um pouco dos sistemas CAN e TTCAN e considerando que atualmente os sistemas de transmissão sem fio (*Wireless*) possuem um nível de tecnologia amplamente difundido, uma proposta para pesquisa futura seria o estudo de um sistema de transmissão entre duas ECU's dentro de um veículo através de Wireless. No presente, todos os sistemas, ou protocolos, contemplam como meio físico de transmissão, par de fios

trançados, ou ainda fibra óptica, dependendo da freqüência de transmissão por causa da Interferência Eletromagnética.

Alguns fabricantes de veículos pesados já utilizam a tecnologia *Wireless* para fazer *download* dos programas Flash das ECUs em suas linhas de montagem (*End-of-Line*) e, portanto, seria extremamente interessante entender seu comportamento em ambientes severos enviando e recebendo mensagens através desse sistema.

LISTA DE REFERÊNCIAS

ALBERT, A.; STRASSER, R.; TRÄCHTLER, A. **Migration from CAN to TTCAN for a distributed control system.** In: 9th International CAN Conference - Munich. v05. p.9-16. 2003. Disponível em <http://www.semiconductors.bosch.de/de/20/can/3-literature.asp>. Acesso em 02 nov. 2005.

ANDERSSON, U. et. al. **Multiplexed vehicle electronics tutorial**. High level protocols. Göteborg: Mecel, 1995. p.8-15.

BOSCH. Disponível em < http://www.can.bosch.com >. Acesso em 15 Nov. 2005.

CLAESSON, V. **Efficient and reliable communication in distributed embedded systems**. Thesis for the degree of Doctor of Philosophy – Chalmers University of Technology. Department of Computer Science and Engineering. Göteborg, Sweden, 2002. p.1-74.

EMBRATEL, 1994.

ETSCHBERGER, K. **Controller Area Network: basics, protocols, chips and applications**. Germany: IXXAT Automation GmbH, 2001. p431.

HARTWICH, F. et al. **Timing in the TTCAN network**. Disponível em: <http://www.semiconductors.bosch.de/de/20/can/3-literature.asp>. Acesso em 15 set. 2004.

ISO – INTERNATIONAL STANDARD ORGANIZATION.
Controller Area Network (CAN) for high speed communication – ISO 11898. 2003

____. Road vehicles – Controller Area Network (CAN) – Part 1: Data link layer and physical signaling. **ISO 11898-1**, 2003.

____. Road vehicles – Controller Area Network (CAN) – Part 2: High-speed medium access unit. **ISO 11898-2**, 2003.

____. Road vehicles – Controller Area Network (CAN) – Part 4: Time-triggered communication. **ISO 11898-4**, 2003.

JOHANSSON, J. **CAN/J1939.** [Mensagem pessoal]. Mensagem recebida por <eduardo.oliveira@scania.com> em 2 de jun. 2006.

JOHANSSON, R. **On distributed control-by-wire systems for critical applications.** Thesis for the degree of Doctor of Philosophy – Chalmers University of Technology. Department of Computer Science and Engineering. Göteborg, Sweden, 2005. p.145-165.

KOPETZ, H. **Real-time Systems: Design Principles for Distributed Embedded Applications.** ISBN: 0-7923-9894-7. Boston: Kluwer Academic Publishers, 1997.

LAWRENZ, W. **CAN System Engineering: from theory to practical applications**. New York: Springer, 1997. p.81-102.

LEEN, G.; HEFFERNAN, D.; **TTCAN: a new time-triggered controller area network**. Elsevier, 2002. Disponível em: <http://www.elsevier.com/locate/micpro >. Acesso em 18 set. 2005.

LUPINI, C. A. **Vehicle Multiplex Communication: serial data networking applied to vehicular engineering**. Warrendale: SAE International, 2004. p.13-19.

MURPHY. N. **Embedded Systems Programming: A short trip on the CAN bus**. Disponível em: <http://www.embedded.com/shared/printableArticle.html>. Acesso em 28 jun. 2004.

PALBUS. **Validation of Dependable Bus Systems.** Disponível em: <http://www.sp.se/pne/software&safety/palbus>. Acesso em 15 jul. 2005.

SAE – The Engineering Society For Advancing Mobility Land Sea Air and Space. **Recommended practice for a serial control and communications vehicle network – SAE J1939**. 2003.

____. Recommended practice for a serial control and communications vehicle network – Part 11: Physical Layer, 250K bits/s, Twisted Shielded Pair. **J1939-11**, 1999.

____. Recommended practice for a serial control and communications vehicle network – Part 21: Data Link Layer. **J1939-21**, 2001.

____. Recommended practice for a serial control and communications vehicle network – Part 71: Vehicle Application Layer. **J1939-71**, 2003.

SOARES, L. F.G.; LEMOS, G.; COLCHER, S. **REDES DE COMPUTADORES. Das LANs, MANS e VANs às Redes ATM.** Rio de Janeiro: Campus, 1998. p. 3-183.

TANENBAUM, A. S. **SISTEMAS OPERACIONAIS MODERNOS**. São Paulo: Pearson Prentice Hall, 2003. p. 53-115.

WAERN, M. **REAL-TIME COMMUNICATION. Evaluation of protocols for automotive systems.** Master of Science Thesis, Royal Institute of Technology - KTH, Stockholm, Sweden, 2003. p. 67-92.

APÊNDICE A

DADOS DE MEDIÇÃO

A1: Carga da banda passante a 20%.

Busload [%]	23.66
Peakload [%]	23.79
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	409
Ext. Data [total]	19932
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

Start	of	measurement	01:59:18	pm			
CAN	1	Bus	with	250000	BPS.		
CAN	2	Bus	with	250000	BPS.		

Statistics	report	AR0001,	01:59:18	pm			
Statistics	for	transmit	spacing	of	messages	in	[ms]
			N	Aver	StdDev	MIN	MAX
f003P	0	RX	5613	50.001	18.851	45.93	54.27
f004P	0	RX	14030	20	11.322	16.16	23.62
fe6cP	ee	RX	14029	20.003	18.296	8.68	32.54
ff19P	0b	RX	5612	49.999	30.746	40.26	59.93
ffa2P	10	RX	2806	100	0.44968	98.10	101.03
ffb0P	27	RX	1403	200	0.48298	198.85	201.33
d000P	17	RX	2948	95.187	50.274	78.50	110.73
e000P	19	RX	280	1000.4	0.36329	998.80	1001.74CAN
e000P	19	RX	280	1000.4	0.34249	999.30	1001.29CAN
eb00P	0	RX	1123	249.74	259.76	98.66	700.91
ec00P	0	RX	280	1000	0.26957	999.41	1001.52CAN
f000P	10	RX	2806	100	0.63295	90.50	108.89
f000P	29	RX	2806	100	0.41085	98.63	101.18
f001P	0b	RX	2806	99.999	0.93213	90.45	109.49
feaeP	30	RX	273	1025.4	48.672	1017.38	1033.13CAN
fec1P	ee	RX	281	1000.2	11.849	998.80	1010.32CAN
fecaP	fe	TX	39130	71.709	41.743	0.59	267.57
fecaP	fe	TXRQ	39130	71.709	29.805	0.00	20.00
fecaP	fe	DELAY	38892	11.148	0.33333	0.01	7.43

fee5P	0	RX	280	1000	0.27299	999.36	1001.50CAN
fee6P	17	RX	273	1029.4	169.31	988.92	2000.52CAN
fee9P	0	RX	281	1000	0.27554	999.37	1001.50CAN
feeaP	17	RX	2948	95.191	44.919	79.37	110.11
feeeP	0	RX	280	1000	0.29521	999.37	1001.51CAN
feefP	27	RX	561	500.01	0.41329	498.48	501.52
fef1P	0	RX	2806	100	0.43136	98.07	101.35
fef2P	0	RX	2806	100	0.42423	98.56	101.31
fef5P	19	RX	280	1000.4	0.38904	999.31	1002.08CAN
fef5P	27	RX	281	1000	0.34333	998.95	1001.05CAN
fef6P	0	RX	561	500.01	0.26567	498.44	501.55
fefcP	27	RX	281	1000	0.37783	999.01	1000.96CAN
ff50P	19	RX	280	1000.4	0.46607	998.95	1002.29CAN
ff60P	0	RX	56	5000.1	0.50086	4999.25	5001.32CAN
ff60P	0b	RX	56	4999.9	11.827	4992.11	5000.81CAN
ff60P	10	RX	56	5000.1	0.51089	4999.25	5001.33CAN
ff90P	27	RX	1122	250.01	0.5495	248.50	251.50
ffb1P	1e	RX	281	998.31	36.519	987.02	1002.12CAN
ffd0P	27	RX	281	1000	0.33373	999.02	1001.32CAN
ffd0P	fe	TX	5497	26.912	167.86	2.39	11052.01CAN
ffd0P	fe	TXRQ	5497	26.912	167.78	11.00	11052.69CAN
ffd0P	fe	DELAY	5470	10.268	0.35118	0.19	14.53
ffd1P	2c	RX	389	722.03	383.43	50.86	1006.10CAN
ffd2P	2c	RX	280	1004.4	0.61672	1001.39	1005.99CAN
ffd3P	2c	RX	280	1004.3	0.77423	1000.43	1006.02CAN
ffd5P	17	RX	282	996.96	48.043	987.61	1011.09CAN
e800P	ee	RX	281	1000.5	60.223	978.94	1999.91CAN
feacP	0b	RX	56	4999.9	11.947	4992.09	5001.39CAN
				System	End	of	measurement

A2: Carga da banda passante a 40%.

Busload [%]	41.26
Peakload [%]	47.64
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	712
Ext. Data [total]	116325
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

Bus with 250 000 BP S.
Bus with 250 000 BP S.

stic s report AR0002 , 02:11 :21 pm

stic s for tra nsmit spacing of messa ges in [ms]

			N	Aver	StdDev	MIN	MAX
f003P	0	RX	4969	50.001	13.392	47.86	52.34
f004P	0	RX	12422	20	0.5739	18.24	22.36
fe6cP	ee	RX	12421	20.003	21.066	5.51	32.10
ff19P	0b	RX	4969	49.999	30.745	42.16	57.68
ffa2P	10	RX	2484	100	0.79908	97.63	102.84
ffb0P	27	RX	1242	200	0.4962	198.84	201.36
d000P	17	RX	2610	95.192	51.717	78.03	113.21
e000P	19	RX	248	1000.4	0.39447	999.40	1001.94
e000P	19	RX	248	1000.4	0.34764	999.41	1001.39
eb00P	0	RX	994	249.85	259.84	98.10	701.73
ec00P	0	RX	249	1000	0.32108	999.37	1001.88
f000P	10	RX	2484	100	0.82498	96.92	102.94
f000P	29	RX	2484	100	0.40827	98.62	100.77
f001P	0b	RX	2485	99.998	10.198	90.50	108.95
feaeP	30	RX	243	1025.3	52.588	1015.76	1035.51
fec1P	ee	RX	248	1000.1	14.697	991.87	1009.88
fecaP	fe	TX	87258	28.471	46.733	0.59	264.35
fecaP	fe	TXRQ	87258	28.471	54.399	0.00	20.00
fecaP	fe	DELAY	25041	0.96623	0.55633	0.00	9.74
fee5P	0	RX	249	1000	0.26764	999.40	1001.49
fee6P	17	RX	248	999.98	6.835	986.16	1013.75
fee9P	0	RX	248	1000	0.34474	999.01	1002.04
feeaP	17	RX	2610	95.188	46.003	78.86	110.65
feeeP	0	RX	249	1000	0.28272	999.34	1001.56
feefP	27	RX	497	500.01	0.57864	497.65	502.36
fef1P	0	RX	2485	100	0.39693	98.62	100.78
fef2P	0	RX	2485	100	0.39513	98.75	100.72
fef5P	19	RX	248	1000.4	0.77	993.96	1006.93
fef5P	27	RX	248	1000	0.43784	998.58	1001.42
fef6P	0	RX	497	500.01	0.24761	498.91	501.92
fefcP	27	RX	248	1000	0.35933	998.70	1001.30
ff50P	19	RX	248	1000.4	0.77767	994.41	1005.44
ff60P	0	RX	50	5000.1	0.5516	4999.31	5001.32
ff60P	0b	RX	49	4999.9	0.92928	4994.56	5001.21
ff60P	10	RX	49	5000.1	0.53583	4999.15	5001.75
ff90P	27	RX	994	250.01	0.62228	248.42	251.56
ffb1P	1e	RX	249	998.29	36.876	985.50	1001.25
ffd0P	27	RX	248	1000	0.41516	998.47	1001.47

ffd0P	fe	TX	12422	20	5.219	0.60	270.08
ffd0P	fe	TXRQ	12422	20	0	20.00	20.00
ffd0P	fe	DELAY	12374	34.649	1.628	0.47	13.37
ffd1P	2c	RX	456	544.14	402.34	51.20	1005.09
ffd2P	2c	RX	250	991.71	96.689	52.34	1009.32
ffd3P	2c	RX	247	1004.4	12.931	998.83	1009.78
ffd5P	17	RX	249	997.04	51.638	984.31	1011.85
e800P	ee	RX	249	996.96	50.853	977.95	1013.27
feacP	0b	RX	49	5000	14.969	4993.69	5005.28
	f me	asurement	02:15	:31 pm			

A3: Carga da banda passante a 60%

Busload [%]	59.40
Peakload [%]	59.68
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	1024
Ext. Data [total]	15262
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

	Bus Bus stic stic	with 250 with 250 s report s for tra	000 BP 000 BP AR0003 nsmit	S. S. , 02:18:0 spacing o	9:00 PM f messa	ges in [ms	]	
			N	Aver	StdDev	MIN	MAX	
f003P	0	RX	1250	49.992	78.634	38.54	61.67	CAN 1
f004P	0	RX	3125	19.997	50.314	8.18	31.48	CAN 1
fe6cP	ee	RX	3124	20.004	24.788	8.27	31.52	CAN 1
ff19P	0b	RX	1250	49.999	72.237	38.51	61.68	CAN 1
ffa2P	10	RX	625	100.01	25.732	90.56	108.84	CAN 1
ffb0P	27	RX	313	200.01	0.54961	198.18	201.86	CAN 1
d000P	17	RX	656	95.192	4.918	81.70	102.52	CAN 1
e000P	19	RX	62	1000.4	0.40295	999.41	1001.15	CAN 1
e000P	19	RX	62	1000.4	0.39167	999.47	1001.14	CAN 1
eb00P	0	RX	250	249.36	262.31	90.70	708.71	CAN 1

ec00P	0	RX	63	999.92	16.947	992.81	1008.10	CAN 1
f000P	10	RX	625	100.01	13.977	89.96	109.51	CAN 1
f000P	29	RX	625	99.988	31.344	89.92	109.50	CAN 1
f001P	0b	RX	625	99.985	23.586	88.09	111.62	CAN 1
feaeP	30	RX	61	1025.3	49.956	1017.68	1034.69	CAN 1
fec1P	ee	RX	62	1000.2	1.388	997.62	1007.23	CAN 1
fecaP	fe	TX	37487	16.664	34.113	0.59	265.33	CAN 1
fecaP	fe	TXRQ	37488	16.662	5.527	0.00	20.00	CAN 1
fecaP	fe	DELAY	3098	18.756	0.75362	0.06	16.74	
fee5P	0	RX	63	999.89	10.115	992.76	1001.50	CAN 1
fee6P	17	RX	62	1000.2	96.525	986.00	1013.88	CAN 1
fee9P	0	RX	62	999.88	0.98908	992.78	1001.59	CAN 1
feeaP	17	RX	656	95.194	45.388	78.18	113.40	CAN 1
feeeP	0	RX	63	1000	17.691	993.31	1007.42	CAN 1
feefP	27	RX	125	500	0.53353	498.61	501.19	CAN 1
fef1P	0	RX	625	100	38.324	89.89	110.14	CAN 1
fef2P	0	RX	624	99.989	47.644	89.40	110.06	CAN 1
fef5P	19	RX	62	1000.5	0.98449	997.67	1005.47	CAN 1
fef5P	27	RX	62	1000	0.52807	998.75	1001.31	CAN 1
fef6P	0	RX	125	500	0.28764	499.10	500.93	CAN 1
fefcP	27	RX	62	1000	0.617	998.45	1002.45	CAN 1
ff50P	19	RX	62	1000.5	0.93457	997.88	1005.28	CAN 1
ff60P	0	RX	12	4999.4	23.183	4992.61	5001.33	CAN 1
ff60P	0b	RX	13	4999.1	32.211	4989.13	5001.33	CAN 1
ff60P	10	RX	13	5000.3	0.61098	4999.52	5001.32	CAN 1
ff90P	27	RX	250	250	1.21	246.01	253.94	CAN 1
ffb1P	1e	RX	62	998.23	36.564	986.86	1001.16	CAN 1
ffd0P	27	RX	62	1000.1	0.90859	997.06	1003.86	CAN 1
ffd0P	fe	TX	9370	66.668	72.773	0.60	267.63	CAN 1
ffd0P	fe	TXRQ	9372	66.652	94.281	0.00	20.00	CAN 1
ffd0P	fe	DELAY	3098	3.799	19.135	0.04	17.34	
ffd1P	2c	RX	603	103.65	162.98	48.11	1005.36	CAN 1
ffd2P	2c	RX	62	1004.3	12.617	1001.09	1006.74	CAN 1
ffd3P	2c	RX	62	1004	16.802	999.38	1007.55	CAN 1
ffd5P	17	RX	63	997.14	51.619	985.97	1011.55	CAN 1
e800P	ee	RX	63	997.08	60.011	978.78	1013.54	CAN 1
feacP	0b	RX	13	4999	57.962	4988.52	5011.42	CAN 1
	f me	asurement	02:19	:13 pm				

A4: Carga da banda passante a 80%

Busload [%]	81.72
Peakload [%]	82.74
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	1407
Ext. Data [total]	152316
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

CAN 1	Bus	with 250	000 BP	S.			
CAN 2	Bus	with 250	000 BP	S.			
Statis	tic	s report	AR0004	, 02:21	:52 pm		
Statis	tic	s for tra	nsmit	spacing	of message	s in [ms	]
			N	Aver	StdDev	MIN	MAX
f003P	0	RX	3106	50	14.714	46.19	53.96 CAN
f004P	0	RX	7765	20	13.791	15.79	23.80 CAN
fe6cP	ee	RX	7764	20.003	19.742	5.93	33.60 CAN
ff19P	0b	RX	3106	49.998	36.683	39.80	60.40 CAN
ffa2P	10	RX	1553	100	0.51826	97.51	101.77 CAN
ffb0P	27	RX	777	200	0.53979	198.32	201.84 CAN
d000P	17	RX	1632	95.19	51.714	80.23	112.19 CAN
e000P	19	RX	156	1000.4	0.45287	998.86	1001.36CAN
e000P	19	RX	156	1000.4	0.49558	998.77	1001.85CAN
eb00P	0	RX	621	250.01	260.02	98.27	701.40 CAN
ec00P	0	RX	155	1000	0.32616	999.00	1001.39CAN
f000P	10	RX	1553	100	0.52026	97.84	101.72 CAN
f000P	29	RX	1553	100	0.52145	98.03	101.90 CAN
f001P	0b	RX	1553	99.996	13.393	89.48	111.00 CAN
feaeP	30	RX	152	1025.3	5.196	1016.29	1033.95CAN
fec1P	ee	RX	156	1000.1	13.422	997.05	1013.28CAN
fecaP	fe	TX	137480	11.294	16.855	0.58	263.07 CA
fecaP	fe	TXRQ	137483	11.294	3.059	0.00	20.00 CA
fecaP	fe	DELAY	26560	0.82965	0.65556	0.00	9.82
fee5P	0	RX	155	1000	0.31398	999.41	1001.32CAN
fee6P	17	RX	155	1000	23.946	989.33	1009.97CAN
fee9P	0	RX	155	1000	0.30691	999.39	1001.20CAN
feeaP	17	RX	1632	95.19	43.714	79.75	110.34 CAN
feeeP	0	RX	155	1000	0.41944	998.88	1001.95CAN

feefP	27	RX	311	500.01	0.53528	498.54	501.53 CAN
fef1P	0	RX	1553	100	0.56314	96.36	103.05 CAN
fef2P	0	RX	1552	100	0.43498	98.66	101.20 CAN
fef5P	19	RX	156	1000.4	0.97782	997.04	1005.64CAN
fef5P	27	RX	155	1000	0.50995	998.80	1001.41CAN
fef6P	0	RX	310	500.01	0.31243	498.74	501.44 CAN
fefcP	27	RX	155	1000	0.43164	998.97	1001.08CAN
ff50P	19	RX	156	1000.4	0.93479	997.04	1004.00CAN
ff60P	0	RX	32	5000.1	0.5493	4999.28	5001.32CAN
ff60P	0b	RX	31	4999.8	35.343	4990.80	5008.18CAN
ff60P	10	RX	31	5000.1	0.54648	4999.27	5001.12CAN
ff90P	27	RX	621	250	0.91623	247.72	252.45 CAN
ffb1P	1e	RX	155	998.34	37.042	985.40	1001.32CAN
ffd0P	27	RX	156	1000	0.69735	997.80	1002.33CAN
ffd0P	fe	TX	23291	66.666	64.865	0.60	273.11 CAN
ffd0P	fe	TXRQ	23292	66.661	94.281	0.00	20.00 CAN
ffd0P	fe	DELAY	7725	37.887	20.471	0.19	18.67
ffd1P	2c	RX	1545	100.53	163.22	45.62	1007.27CAN
ffd2P	2c	RX	155	1004.2	17.263	999.52	1011.11CAN
ffd3P	2c	RX	154	1004.1	19.644	999.00	1011.02CAN
ffd5P	17	RX	155	997.03	48.382	985.51	1009.79CAN
e800P	ee	RX	155	997.03	59.478	982.00	1012.58CAN
feacP	0b	RX	31	4999.8	17.415	4991.29	5001.35CAN
End of	me	asurement	02:24	:28 pm			

A5: Carga da banda passante a 100%

Busload [%]	98.00
Peakload [%]	98.11
Std. Data [fr/s]	0
Std. Data [total]	0
Ext. Data [fr/s]	1690
Ext. Data [total]	452756
Std. Remote [fr/s]	0
Std. Remote [total]	0
Ext. Remote [fr/s]	0
Ext. Remote [total]	0
Errorframe [fr/s]	0
Errorframes [total]	0
Chip state	Active

Statis Statis	tics r tics f	eport or tra	AR0006 nsmit	, 02:30:0 spacing o	7:00 PM f message	s in [ms	]
			N	Aver	StdDev	MIN	MAX
f003P	0	RX	5442	50.001	13.247	47.86	52.30
f004P	0	RX	13606	20	0.81081	17.15	23.08
fe6cP	ee	RX	13604	20.003	21.961	7.07	35.01
ff19P	0b	RX	5442	50	2.624	40.30	59.88
ffa2P	10	RX	2721	100	1.044	96.35	104.05
ffb0P	27	RX	1360	200.01	0.60743	198.11	201.88
d000P	17	RX	2858	95.192	50.883	78.06	112.30
e000P	19	RX	273	1000.4	0.52618	998.69	1001.87
e000P	19	RX	273	1000.4	0.48344	998.88	1002.06
eb00P	0	RX	1088	249.59	259.69	98.63	700.86
ec00P	0	RX	272	1000	0.35507	999.42	1001.41
f000P	10	RX	2721	100	11.546	95.22	104.12
f000P	29	RX	2721	100	0.49678	98.64	101.26
f001P	0b	RX	2721	99.998	11.464	89.30	110.50
feaeP	30	RX	266	1025.3	50.134	1014.69	1036.26
fec1P	ee	RX	272	1000.1	15.097	990.87	1010.06
fecaP	fe	TX	369013	0.73742	0.31063	0.58	59.18
fecaP	fe	TXRQ	722154	0.37681	0.98239	0.00	5.00
fecaP	fe	DELAY	176622	0.30363	0.25525	0.00	4.93
fee5P	0	RX	272	1000	0.39351	997.81	1001.40
fee6P	17	RX	272	999.99	59.839	979.14	1020.81
fee9P	0	RX	273	1000	0.43266	997.42	1001.42
feeaP	17	RX	2858	95.19	41.501	80.38	110.48
feeeP	0	RX	272	1000	0.35821	999.42	1001.41
feefP	27	RX	544	500.01	0.53288	498.43	501.44
fef1P	0	RX	2721	100	0.51382	97.69	102.12
fef2P	0	RX	2721	100	0.49234	97.48	102.45
fef5P	19	RX	272	1000.4	10.559	994.68	1007.62
fef5P	27	RX	272	1000	0.63381	997.65	1002.23
fef6P	0	RX	545	500.01	0.39555	498.28	501.82
fefcP	27	RX	272	1000	0.48125	998.60	1001.36
ff50P	19	RX	272	1000.4	12.056	995.85	1008.26
ff60P	0	RX	54	5000.1	0.62601	4997.61	5001.13
ff60P	0b	RX	54	4999.9	11.696	4992.27	5001.16
ff60P	10	RX	54	5000.1	0.62104	4998.88	5002.00
ff90P	27	RX	1089	250.01	0.80408	246.13	253.56
ffb1P	1e	RX	272	998.29	36.229	986.15	1002.51
ffd0P	27	RX	272	1000	0.6427	997.65	1002.40
ffd0P	fe	TX	19451	13.915	112.2	0.60	7905.35
ffd0P	fe	TXRQ	40004	68.022	41.061	0.00	10.00
ffd0P	fe	DELAY	14413	37.869	29.011	0.00	10.00

ffd1P	2c	RX	1104	245.9	337.76	46.15	1008.47
ffd2P	2c	RX	271	1004.3	2.019	996.25	1012.70
ffd3P	2c	RX	271	1004.3	20.222	996.29	1013.14
ffd5P	17	RX	273	996.98	49.656	979.15	1011.30
e800P	ee	RX	273	996.97	54.034	978.38	1010.97
feacP	0b	RX	54	4999.9	19.101	4992.12	5007.07
End of	measu	rement	02:34	:40 pm			