# FTT-CAN - ESTUDO DE CASO EM APLICAÇÃO AUTOMOTIVA

**Fernando Henrique Ataide**[*]
fernando.ataide@fiat.com.br

**Carlos Eduardo Pereira**[†]
cpereira@ece.ufrgs.br

[*]Fiat Automóveis S.A.
Betim - MG

[†]Escola de Engenharia – UFRGS
Porto Alegre – RS – Brazil

## ABSTRACT

**A new approach to Improve the response time of the FFT-CAN protocol**
This paper proposes some enhancements to the FTT-CAN protocol in order to make it more suitable for real-time distributed embedded applications, such as those that occur in the automotive sector. The paper describes the proposed approach, its implementation and results obtained in experimental validation via a case study in the automotive area. The results obtained indicated the effectiveness of the proposed approach in enhancing the real-time behavior of the FTT-CAN protocol, while still preserving its flexibility aspects.

**KEYWORDS**: FTT-CAN, embedded systems, industrial communication protocols

## RESUMO

Este artigo apresenta uma proposta de modificação do protocolo FTT-CAN para melhorar seu desempenho para aplicações automotivas. O artigo descreve a ideia da proposta, sua implementação e validação experimental. Os resultados obtidos indicam que a modificação proposta realmente melhora o desempenho do protocolo FTT-CAN permitindo uma melhor resposta temporal, sem perder a flexibilidade permitida pelo protocolo.

Artigo submetido em 15/03/2011 (Id.: 1302)
Revisado em 07/05/2011, 26/10/2011, 11/01/2012
Aceito sob recomendação do Editor Associado Prof. Luis Antonio Aguirre

**PALAVRAS-CHAVE**: FTT-CAN, RTAI, $\mu$Clinux, sistemas embarcados, sistemas de tempo real.

## 1 INTRODUÇÃO

Sistemas distribuídos de tempo real (SDTR) estão se tornando largamente empregados em diversas áreas de aplicação, tal como controles de processos industriais, automação industrial e residencial e em sistemas de eletrônica embarcada de veículos automotores. Estas aplicações possuem restrições de previsibilidade nas dimensões de tempo e de valor, mesmo em situações de presença de falhas, visto que tais aplicações podem estar intrinsecamente ligadas a integridades de pessoas ou custos de infra-estrutura. A indústria automobilística, por exemplo, possui um grande interesse em produzir veículos populares com controle eletrônico de direção, onde dispositivos mecânicos e hidráulicos serão substituídos por atuadores e sensores elétricos. Este interesse tem foco na diminuição de peso e de consumo de combustível.

Os principais requisitos envolvidos em SDTR para este tipo de aplicação são: comportamento determinístico das execuções de tarefas e das transmissões de mensagens, baixa variabilidade temporal - *jitter* (mesmo em situações de carga de processamento/transmissão), suporte a técnicas de tolerância a falhas e também flexibilidade quanto a futuras adaptações instigadas por novos requisitos funcionais após a fase de projeto. Considerando que um SDTR é formado por um conjunto de unidades de processamento/controle eletrônico distribuídos espacialmente, estas devem ter um protocolo de

comunicação provendo serviços confiáveis e seguros para a camada de software superior, a aplicação propriamente dita.

Neste contexto, na área automotiva, alguns protocolos comerciais sustentam o cumprimento destes requisitos. O primeiro é o protocolo TTP/C (TTA-GROUP, 2003) e sua *Time-Triggered Architecture* (TTA) - um resultado de muitos anos do trabalho iniciado em 1979 na Universidade de Tecnologia de Viena com o projeto MARS. Outro é o protocolo de comunicação FlexRay (CONSORTIUM, Copyright 2004) desenvolvido pelo consórcio FlexRay, formado por um forte grupo de industrias da área automotiva.

Uma característica comum destes protocolos é o comportamento estático de troca de mensagens no segmento de controle - faixa de tempo destinada às transmissões de mensagens que possuem restrições temporais -, constituídas por informações periódicas. O arbitramento no barramento de comunicação (método de controle de acesso ao meio físico de comunicação) é essencialmente baseado em TDMA, onde os tempos de transmissão de todas as mensagens devem ser conhecidos com antecedência. Neste caso cada estação ou nó (também denominada como ECU - *Electronic control Unit* ou Unidade de Controle Eletrônico) da rede de comunicação possui um instante, fixado em tempo de projeto, para a transmissão de cada mensagem periódica. Esta estratégia de comunicação é denominada *time-triggered*. O protocolo FlexRay, em particular, combina a estratégia *time-triggered* e outra denominada *event-triggered*. Esta última permite a transmissão de mensagens disparadas por eventos assíncronos, sem ter os intantes de transmissão previamente definidos e utiliza a estratégia denominada FTDMA (*flexible time-division multiple-access*) como método de acesso ao meio físico de comunicação.

A união dos paradigmas *time-* e *event-triggered* fornece um grau de flexibilidade ao protocolo FlexRay ao contrário do protocolo TTP/C que é puramente estático. Esta característica é importante para muitos sistemas de controle digitais modernos, principalmente quando o objeto controlado não é completamente conhecido na fase de projeto, exigindo, assim, um sistema de controle que possa ser adaptável. Um SDTR com um alto grau de flexibilidade pode prover uma melhor utilização de recursos de hardware como processamento e memória; e com uma capacidade de integração de novas funções em resposta a demandas de requisitos funcionais, ou seja, adaptativo. O autor em (ALMEIDA, 2003) apresenta uma avaliação detalhada sobre as principais vantagens de prover flexibilidade em um SDTR.

Outro protocolo digno de destaque, embora não disponível comercialmente, é o FTT-CAN (ALMEIDA; PEDREIRAS; FONSECA, 2002) (Flexible Time-Triggered Communication on CAN) que consiste em uma plataforma de comunicação concebida com foco em flexibilidade na troca de mensagens periódicas, mantendo ainda as garantias temporais. Este protocolo combina um segmento dinâmico de transmissão de mensagens periódicas e outro de mensagens baseadas em eventos. Tais segmentos são temporalmente isolados no acesso ao barramento de comunicação CAN. Ou seja, suporta ambos os paradigmas: *time-* e *event-triggered*. Em oposição aos protocolos TTP/C e FlexRay, que possuem um controle estático do instante de transmissão de mensagens periódicas, no FTT-CAN toda a carga de comunicação no segmento *time-triggered* pode ser dinamicamente controlada em tempo de execução.

## 2 PROTOCOLOS DE COMUNICAÇÃO PARA APLICAÇÕES AUTOMOTIVAS

Um SDTR é composto por um conjunto de nós interconectados que se comunicam por um canal físico com algumas regras impostas pelo protocolo de comunicação. O sistema de comunicação é um dos componentes principais de um sistema de controle distribuído e, em alguns casos, é um componente crítico onde são requeridas técnicas de tolerância a falhas como, por exemplo, redundâncias. Com o crescente aumento de aplicações de segurança crítica nas novas gerações de sistemas embarcados automotivos como, por exemplo, aplicações *x-by-wire*, um comportamento determinístico nos serviços de comunicação é necessário. Tal exigência garante a previsibilidade da taxa de amostragem do processo de controle e do processo de atuação destas aplicações mesmo em altas condições de carga de trabalho no barramento de comunicação.

Os protocolos de comunicação adequados para sistemas automotivos de segurança critica são divididos em duas categorias já previamente discutidas: protocolos event-triggered ( ET) e time-triggered (TT). Uma variedade de protocolos já foi proposta para esta aplicação seguindo ambos ou um paradigma de comunicação. O protocolo CAN é um dos mais antigos, sendo o mais conhecido e amplamente utilizado. Normalmente, existem duas redes CAN distintas nesse tipo de aplicação: uma com alta velocidade denominada *High Speed* CAN ou puramente C-CAN que é empregada em aplicações de controle de motores, por exemplo; enquanto a outra de baixa velocidade denominada *Low Speed* CAN ou B-CAN é usada em aplicações de interior e conforto nos automóveis. A taxa máxima de transferência da rede CAN é 1 Mbit/s, mas ambas as redes operam em 500 e 50kbits/s ou 500 e 125kbits/s respectivamente. A baixa taxa transferência de dados comparada com o potencial máximo do CAN é escolhida a fim de garantir uma maior imunidade a ruídos. A *Low Speed* CAN é tolerante à ausência de um dos fios do barramento CAN, sendo capaz de manter a comunicação confiável mesmo em um único fio. Isto é possível através da utiliza-

ção de *transceivers* - a combinação de transmissor e receptor em um mesmo dispositivo ou encapsulamento - específicos, como por exemplo o Philips 1054.

O protocolo LIN (*Local Interconnect Network*) (CONSORTIUM, 2006) tem como foco o baixo custo de infra-estrutura eletrônica, com implementações baseadas em UART com baixas taxas de transmissão (20kbit/sec). O LIN é frequentemente usado em aplicações internas, como controle de fechamento de portas e outros sensores simples, sem requisitos de segurança crítica ou de pontualidade. Já na subárea denominada infotainment, no qual aplicações nas áreas de sistemas de informação (como sistemas de navegação via GPS) e de entretenimento (videos e músicas) existem os protocolos D2B (USA, 2004) e MOST (GRZEMBA, 2007), desenvolvido para aplicações multimídia.

## 2.1 Classificação SAE para os protocolos de comunicação automotiva

Sobre as aplicações de redes de comunicação automotivas, a Sociedade de Engenheiros Automotivos SAE (*Society of Automotive Engineers*) definiu três categorias básicas de aplicação de redes para SDTR em automóveis seguindo características relacionadas a determinismo temporal e requisitos de segurança. Abaixo é apresentado um resumo conceitual de cada categoria:

CLASSE A - Basicamente, esta categoria compreende aplicações sem rígidas restrições temporais, ou seja, aplicação não tempo real. As exigências de largura de banda são baixas e esta categoria de redes automotivas não é empregada em sistemas de segurança crítica. Exemplos de aplicações dessa categoria são: sistemas de iluminação interna e externa, posicionamento e controle de assentos, espelhos retrovisores e vidros. Uma característica comum nestes exemplos de aplicações da Classe A SAE é que a carga útil de dados das mensagens e sua taxa de atualização são relativamente baixas. LIN (CONSORTIUM, 2006) e TTP/A (KOPETZ, 1997) são exemplos de protocolos desta categoria.

CLASSE B - Está presente em aplicações como sistema de controle automático de ar condicionado (por exemplo, sistema *dual-temp*), sistema de computador de bordo com informações de consumo instantâneo, médio, autonomia, tempo e distância percorrida. Tais aplicações demandam uma largura de banda maior que a categoria anterior, isso devido às taxas de atualizações das variáveis envolvidas nestes sistemas. A Classe A e Classe B não são adequadas para aplicações de segurança crítica. O protocolo CAN é um grande exemplo desta categoria, e é largamente utilizado na indústria automotiva.

CLASSE C - Esta categoria inclui aplicações que são intrinsecamente envolvidas com a dinâmica do veículo e controle de direção, como, por exemplo, sistemas eletrônicos ativos de direção e freios e sistema automático de controle de câmbio. Nestes sistemas, a taxa de atualização de informações é na ordem de 1 a 10ms e a carga útil das mensagens transmitidas e recebidas nestes sistemas é maior se comparado às categorias anteriores, A e B. Além disso, aplicações da Classe C são ditas como de segurança crítica porque uma queda do sistema pode levar a consequências desastrosas e por este motivo estas demandam alta previsibilidade, confiabilidade e baixa latência. A fim de cumprir estes requisitos, as plataformas de comunicação com técnicas de tolerância a falhas são necessárias. Atualmente, TTP/C (TTA-GROUP, 2003) e FlexRay (CONSORTIUM, Copyright 2004) são exemplos de protocolos para aplicações Classe C.

Considerando os paradigmas apresentados anteriormente, o ET e TT, existem alguns protocolos presentes na indústria ou mesmo ainda em meio acadêmico que favorecem um ou outro, ou permite a coexistência de ambos. Nas subseções seguintes são apresentados alguns destes protocolos.

## 2.2 LIN

LIN (Local Interconnect Network) é um protocolo mestre escravo *time-triggered*. Seu meio físico consiste de um único canal com taxa máxima de transmissão de 20Kbit/s. Sua aplicação principal está nos dispositivos discretos como, por exemplo, controle de posição dos bancos dos tripulantes, sistema de travamento de portas, sensores de chuva, e controle das luzes internas. Com baixa taxa de transmissão de dados, meio físico de um canal, o LIN possibilita um baixo custo de interconexão de sensores e atuadores comparado a uma rede CAN, por exemplo. Audi, BMW, DaimlerChrysler, Motorola, Vulcano, Volvo, e Volkswagen propuseram e desenvolveram este protocolo de padrão aberto de baixo custo.

LIN possui uma auto sincronização sem a necessidade de um cristal para sincronização de relógio nos nós escravo, levando a uma redução de custo significante de plataforma de hardware. Devido ao seu método de acesso ao meio baseado em mestre-escravo, o protocolo não necessita de arbitramento. Um mestre pode controlar até 16 escravos em uma distância máxima de 40 metros, cabendo a ele o controle e o início do tráfego da mensagem no barramento. Com isso, o escravo não transmite até a autorização do mestre, o que proporciona uma latência fixa da mensagem. Tais características reduzem a complexidade da implementação do sistema. O protocolo LIN utiliza o conceito de tarefa. Em cada nó, todas as tarefas executadas, inclusive as tarefas de comunicação dos escravos, são alternadas entre tarefas de envio e de recepção, coordenadas pelo nó mestre. O formato frame é fixo, sendo composto por um cabeçalho (*header*) e um campo de

resposta (*response*). O frame pode possuir entre zero e oito bytes de comprimento, seguidos de um byte de *checksum*.

## 2.3 CAN

O CAN (*Controller Area Network*) é um protocolo de comunicação serial intrinsecamente *event-triggered*. Foi concebido originalmente para o setor automotivo pela empresa Bosch na Alemanha no início da década de 80, tendo sido adotado pela grande maioria dos fabricantes de automóveis, tornando-se um dos protocolos mais difundidos. Atualmente o uso do protocolo não se restringe apenas para aplicações no setor automotivo, uma vez que existem variantes deste protocolo com foco em automação industrial. Exemplos destes protocolos são CANOpen e DeviceNet (CIA, 2001b). Na indústria automotiva este protocolo é usado em unidades de controle de motor, quadro de instrumentos, e outros. Por outro lado, seu baixo custo de desenvolvimento de hardware e software possibilita seu emprego em componentes relacionado à *body electronics* e ítens de conforto veicular - iluminação interna, controle de vidros e bloqueio de portas - em substituição aos sistemas de chicotes tradicionais (CIA, 2001a). O CAN foi internacionalmente padronizado em 1993 como ISO 11898-1 e inclui a camada de enlace de dados do modelo de referencia de sete camadas ISO/OSI. Além do CAN, foram desenvolvidos outros protocolos para o uso na indústria automotiva como o ABUS da Volkswagen, o VAN da Peugeot e Renault e o J1850 da Chrysler, General Motors e Ford. Estes protocolos diferem fundamentalmente ao nível da taxa de transmissão, formato das mensagens, detecção de erros e seu tratamento. O CAN é um barramento do tipo *broadcast*, isto quer dizer que todos os nós escutam todas as transmissões. Não existe forma de enviar uma mensagem exclusivamente para um determinado nó, pois todos os nós ouvem essa mensagem. O hardware do CAN, no entanto, pode conter filtros locais de forma que cada nó somente reaja a mensagens que sejam de seu interesse. A transmissão de mensagens é realizada de forma assíncrona, somente através de requisições da aplicação, através de escrita direta nos *buffers* de mensagem e nos registradores do controlador CAN. Cada mensagem deve ter um identificador próprio que é usado para atribuir uma identificação do conteúdo da mensagem e também define a prioridade da mesma. O identificador com valor menor possui a prioridade mais alta. O protocolo de CAN define dois estados lógicos do canal de comunicação: dominante e recessivo. O estado dominante corresponde ao valor "0", e este sobrescreve o outro estado recessivo que por sua vez representa o valor lógico "1". A mensagem com maior quantidade de bits dominantes em posições mais significativas de seu campo identificador, que corresponde àquela mensagem com menor valor no identificador - ganhará o processo de arbitramento e transmitirá seus bytes de dados. Um nó transmissor deve monitorar o canal para determinar se seu bit recessivo foi sobrescrito por um bit dominante de outro nó transmissor da mesma rede. Um nó que perde o processo de arbitramento imediatamente cessa sua transmissão e continua como receptor da mensagem em curso de transmissão.

O protocolo CAN possui quatro tipos de frames distintos: frame de dados, frame remoto, frame de erro e um frame de sobrecarga. Além dos frames citados, existe um espaço ente frames que é responsável por separar os frames do tipo dados ou remoto de qualquer outro tipo que os precedem. Isto não acontece com os outros tipos de frames (erro e de sobrecarga).

Na camada física o barramento CAN é classificado como “par trançado diferencial”. Este conceito atenua fortemente os efeitos causados por interferências eletromagnéticas, uma vez que qualquer ação sobre um dos fios será sentida também pelo outro, causando flutuação em ambos os sinais para o mesmo sentido e com a mesma intensidade.

Os níveis lógicos no barramento CAN, dominante e recessivo, são criados em função da condição presente nos fios CAN_H e CAN_L que compõe o meio físico da rede. Na camada física, o protocolo usa codificação de NRZ (*Non Return to Zero*) com o método de *bit stuffing.* O formato de mensagem CAN contém 47 bits de informações de controle do protocolo (o ID, CRC, ACK e bits de sincronização, etc.) A transmissão de mensagens utiliza o mecanismo denominado de bit stuffing , o qual insere um bit após cada cinco bits consecutivos de mesmo valor. Os segmentos do frame de dados "start of frame", "identifier", "control field", "data field"e "CRC sequence"são codificadas pelo método bit stuffing. Os demais segmentos do frame de dados e remoto (delimitador CRC, ACK, e o end of frame) são fixados de forma a não receber os bit stuffing. Também os frames de sobrecarga e de erro são fixos e inalterados no instante da transmissão (CIA, 2001a).

## 2.4 TTCAN

TTCAN (RAHUL SHAH, 2002), (Time-Triggered CAN) é uma extensão time-triggered para o protocol CAN. O principal objetivo deste protocolo é evitar o *jitter* resultante da comunicação baseado em CAN nativo e garantir uma comunicação determinística para aplicações em SDTR.

No protocolo TTCAN a regra de comunicação segue uma mensagem de referência enviada por um nó mestre. Esta mensagem é a referência de tempo global da rede, e baseada neste tempo algumas mensagens são atribuídas para transmissão. Uma estratégia semelhante também é empregada nos protocolos FTT (FTT-CAN e FTT-Ethernet) para obrigar uma referência de tempo para transmissão de mensa-

gem, que será apresentada mais adiante. A especificação ISO 11898 foi estendida ganhando uma nova versão com suporte a comunicação TT que recebe o novo código ISO 11898-4 em dois níveis: o primeiro garante a comunicação TT por meio da mensagem de referência de um nó mestre; no outro nível uma base de tempo globalmente sincronizado é fornecida e uma correção de desvio de tempo global entre os nós é estabelecida a fim de manter um sincronismo temporal. A mensagem de referência tem um único identificador para ser reconhecida pelos nós presentes na rede. Esta possui um byte de informações de controle e os demais bytes podem ser usados para transferência de dados. No segundo nível do protocolo, a mensagem de referência transporta uma informação de tempo global do mestre. São 4 bytes para informações de controle e os demais bytes são usados para transferência de dados como nível 1 do protocolo. O ciclo de comunicação - denominado ciclo básico - é o intervalo de tempo entre duas mensagens consecutivas de referência. O ciclo básico é composto de algumas janelas de tempo com tipos e tamanhos diferentes para transmissão de mensagens. A primeira janela, denominada de janela exclusiva, é usada para transmissões de mensagens TT. Uma outra janela de tempo, denominada janela de arbitragem, é preenchida com as transmissões ET, que concorrem através do mecanismo de arbitramento seguindo as prioridades de acordo com o valor presente no campo identificador nas mensagens seguindo o protocolo CAN nativo. Retransmissões não são permitidas em ambas as janelas de tempo a fim de garantir a referência de tempo de cada janela, instante de inicio e fim. A última é a janela de tempo livre, que é um tempo reservado para extensões futuras da rede. O protocolo TTCAN permite alterar uma janela de tempo livre para uma de arbitragem ou exclusiva em caso de necessidades adicionais de largura de banda. Como a tabela de mensagem é inteiramente definida de forma estática e em tempo de projeto, nenhum conflito acontece no sistema se a construção da tabela foi feita usando ferramentas de análise adequada. Em caso de qualquer distúrbio - devido a um nó intruso - a arbitragem do protocolo CAN nativo é usado na ordenação das transmissões, porém as mensagens envolvidas estarão sujeitas a *jitter* de acordo com a carga intrusiva presente.

## 2.5 TTP/C

O TTP/C (TTA-GROUP, 2003) é uma plataforma de comunicação baseado no TTP (*Time-Triggerred Protocol*) e desenvolvido com o objetivo de atender todos os requisitos das aplicações SAE classe C. Como citado anteriormente, essa classe de aplicação é categorizada como de segurança crítica, exigindo determinismo e previsibilidade temporal além de tolerante a falhas. A origem deste protocolo foi no projeto MARS (*Maintainable Real-time System*), na Universidade de Tecnologia de Viena na Áustria.

Entre o processador e o controlador de comunicação existe uma interface de rede de comunicação (*communication network interface* - CNI). A CNI é a interface entre o controlador TTP/C e o processador dentro de uma ECU, basicamente é região de memória *dual-port* mapeada. É também denominada como uma interface de base de mensagens. Ela fornece ao processador uma área de memória para submissão e recebimento de mensagens e informações de status do controlador de comunicação TTP/C. A Figura 1 apresenta os componentes da estrutura de uma ECU TTP/C.

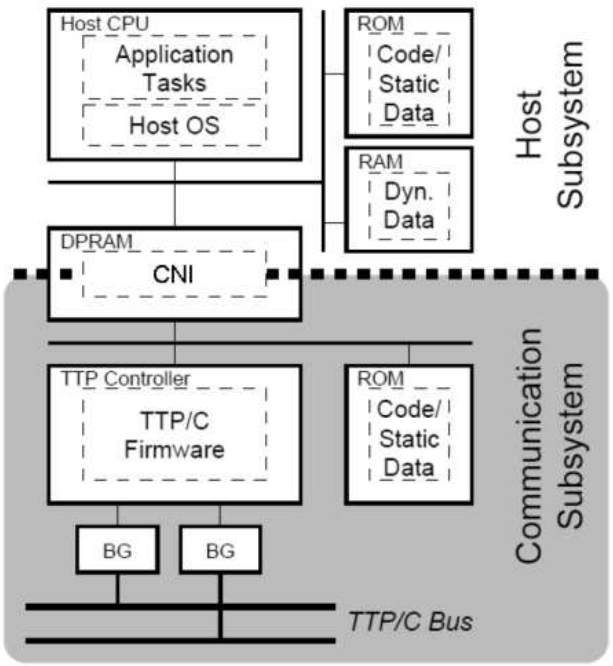


Figura 1: Estrutura de uma ECU TTP/C

O controlador TTP/C é suportado por dois guardiões de barramento (*bus-guardians*). Cada canal físico de comunicação é protegido por este dispositivo, os quais impedem o barramento de serem monopolizados por falhas em nós. Na *Time-Triggerred Architecture* (KOPETZ; BAUER, 2001) todas informações sobre o comportamento do sistema, tal como, nós transmissores e receptores em um determinado instante no tempo são definidas a priori, ou seja, em fase de projeto. Dois tipos de frames são definidos no protocolo. O primeiro, denominado *I-Frame* (*Initialization frames*), é usado para inicializar o sistema. Ele contém o estado interno do controlador TTP/C em seu campo de dados. Este permite integrar nós para participar do protocolo quando recebem um *I-Frame*. *I-Frame* são enviados pelo sistema de comunicação TTP/C durante a fase de inicialização do protocolo (*cold start*), e em intervalos predefinidos durante a operação normal do protocolo para facilitar a reintegração de nós em estado de falha.

Por fim, *N-Frames* (*Normal frames*) são usados durante a operação normal e contém dados do software de aplicação. O byte de cabeçalho de um *N-Frame* contém três campos, o primeiro bit identifica o tipo da mensagem, os três bits seguintes são usados para requisição de troca de modo de operação do sistema como todo, e os demais quatro bits são usados como informação de reconhecimento sobre a recebimento da mensagem. O controle de acesso ao barramento é controlado por um esquema estático TDMA ("Time Division Multiple Access"), exigindo uma sincronização de relógio entre os nós em uma rede TTP/C. Cada nó tem permissão de transmissão somente durante uma janela de tempo determinada, denominada de janela TDMA. Com relação à duração das janelas TDMA e à sequência de envio dos nós, todas janelas TDMA são iguais. Entretanto, o comprimento e conteúdo das mensagens se diferem a cada ciclo, de acordo com o software de aplicação. Os atributos das mensagens enviadas e recebidas pelo protocolo são descritas em uma estrutura de dados estática, denominada de *Message Descriptor List* (MEDL), que reside em memória ROM dentro do subsistema de comunicação. De acordo com esta lista o controlador TTP/C, periodicamente e autonomamente, lê da CNI as mensagens a serem transmitidas e escreve as mensagens recebidas também na CNI. Um dos mais importantes dados presentes na MEDL é, entretanto, o endereço de cada mensagem dentro da CNI e o comprimento da mesma.

## 2.6 FlexRay

O FlexRay (CONSORTIUM, Copyright 2004) é uma plataforma de comunicação completa, com alta taxa de transmissão, determinístico e ainda suporta técnicas de tolerância a falhas. É aplicável em sistemas de segurança critica em automóveis. Foi desenvolvido em um consorcio de grandes empresas da área automotiva e empresas da área de semicondutores. As principais empresas participantes do *core partners* são: BMW, DaimlerChrysler, General Motors, Ford, Volkswagen, Bosch, Motorola e Philips.

O FlexRay não substitui os demais protocolos de aplicação automotiva, ao contrário, ele opera coexistindo com protocolos existentes desempenhando sua função especifica, como o CAN, LIN e MOST.

A troca de dados entre numerosos dispositivos de controle, sensores e atuadores nas aplicações SDTR em automóveis é, atualmente, realizada através do protocolo CAN. Entretanto, a introdução dos novos conceitos, como os sistemas *x-by-wire*, resultaram no aumento dos requisitos, especialmente com relação à tolerância a falhas, erros e determinismo temporal nas transmissões de mensagens. O FlexRay comporta estes requisitos através de transmissão de mensagens em janelas fixas e pelas técnicas de tolerância a falhas e redundância nas transmissões em dois canais físicos. O FlexRay trabalha com o principio TDMA, onde as mensagens têm suas janelas de transmissão fixas nas quais cada nó tem exclusivo acesso ao barramento neste instante. As janelas são repetidas por ciclos. O tempo no qual cada mensagem é transmitida é previsível, consequentemente é um meio de comunicação determinístico. Entretanto, as janelas de transmissões fixas para cada mensagem têm desvantagem, que é a não exploração eficiente do recurso físico, como citado das seções anteriores. Por esta razão, o FlexRay subdivide o ciclo em dois segmentos, um estático e outro dinâmico. As janelas fixas são situadas no segmento estático no inicio de cada ciclo de transmissão. No segmento dinâmico as janelas são utilizadas de forma dinâmica por cada nó. Exclusivo acesso ao barramento é somente permitido em um curto tempo (denominado *mini-slots*). O tempo das janelas, neste segmento, é somente estendido para o tempo requerido em projeto se, somente se, ocorrer transmissão dentro do *mini-slot*. Desta forma, FlexRay proporciona uma melhor utilização do recurso, sendo usado somente com necessidades de transmissão. A plataforma de comunicação do FlexRay possui dois canais físicos separados, com taxa de transmissão de até 10 Mbit/s cada. Os dois canais são usados redundantemente, mas podem também transmitir mensagens diferentes, em tal caso a vazão de informação (*throughput*) é dobrada. Para o suporte a funções síncronas e otimização de largura de banda por meio de pequenas distâncias entre mensagens, o protocolo implementa uma base de dados comum (*global time*).

A sincronização de relógio é realizada com a transmissão de uma mensagem no segmento estático do ciclo, e, com base instante de recebimento desta mensagemos relógios locais de cada nó da rede FlexRay são sincronizados. Uma ECU FlexRay consiste em um processador, o controlador de comunicação (CC) e o guardião de barramento (*bus guardian* - BG). O processador processa dados do software de aplicação e fornece-os para o controlador de comunicação que, por sua vez, se encarrega de transmiti-los. Os guardiões de barramento monitoram o acesso ao barramento de comunicação. O processador informa ao BG em qual janela de transmissão o CC do respectivo nó está alocado. O BG, então, permite ao CC transmitir a mensagem somente nesta janela. O recebimento de dados é sempre permitido.

# 3 O PROTOCOLO FTT-CAN

*Flexible Time-Triggered on Controller Area Network* (FTT-CAN) foi proposto com o objetivo de cobrir o requisito de flexibilidade em sistemas de tempo-real. Tal requisito não é presente em protocolos essencialmente TT. Basicamente, o protocolo faz uso do conceito de ciclo elementar com duas fases de transmissão a fim de combinar ambos os paradigmas TT e ET com um isolamento temporal entre eles. Além disso, o tráfego TT é escalonado *on-line* por um nó particular

denominado de nó mestre, que provê um controle de admissão de mensagem no segmento TT sem prejudicar os requisitos temporais das mensagens já presentes no sistema (ALMEIDA; PEDREIRAS; FONSECA, 2002). O controle de admissão é realizado por um escalonador central, executado pelo nó mestre. Este pode receber de forma dinâmica, ou seja, em tempo de execução, novos pedidos de transmissão de mensagens TT não antes previstas. Estas requisições são processadas e avaliadas por um algoritmo que verifica a factibilidade da nova inserção. Com controle de admissão online, o protocolo suporta o tráfego de TT em um modo flexível, sob a garantia dos requisitos temporais (baseado no modelo de escalonamento dinâmico). O FTT-CAN aproveita-se do nativo controle de acesso ao meio do protocolo CAN para eliminar a necessidade de informações de controle no protocolo a fim de controlar a comunicação TT, reduzindo, desta forma, o overhead do protocolo. O nó mestre ativa as transmissões nos nós escravos seguindo um modelo denominado mestre-escravo flexível (traduzido do termo em inglês *relaxed master-slave method*), que requisita de forma simultânea as mensagens a serem transmitidas em um dado segmento TT de um ciclo elementar. Uma mensagem específica - denominada de *Trigger Message* (TM) - é transmitida pelo nó mestre a fim de ativar o início de um ciclo elementar ou *Elementary Cycle* (EC) dentro de cada nó escravo que por sua vez transmitirá mensagens nos segmentos TT e ET. A mensagem TM transporta informações que indicam quais mensagens serão transmitidas no segmento TT. A Figura 2 apresenta um ciclo EC com seus respectivos segmentos, e exemplifica a codificação de uma mensagem TM.

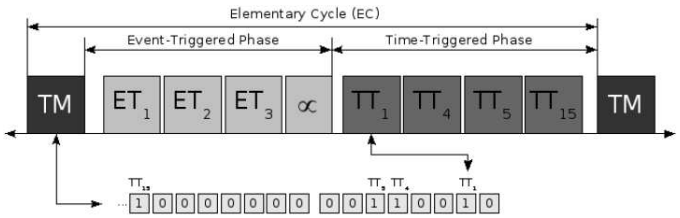


Figura 2: Ciclo elementar (EC) do protocolo FTT-CAN

Em cada EC o protocolo define duas fases sucessivas, assíncronas e síncronas, que corresponde a dois segmentos separados. O primeiro é utilizado para comunicação ET e chamado assíncrono porque os pedidos de transmissão podem ser emitidos a qualquer momento, gerados, por exemplo, devido a uma troca de estado de algum sensor monitorado. O posterior é usado para comunicação TT, chamado síncrono, porque as transmissões são realizada de forma síncrona em relação a mensagem TM enviada pelo nó mestre em cada EC. O segmento síncrono ou TT do EC tem uma duração $lsw(n)$ e é ajustada de acordo com o tráfego escalonado para o segmento no respectivo EC. O segmento assíncrono ou ET tem uma duração $law(n)$ igual ao restante de tempo entre a mensagem TM e o segmento TT. Ou seja, o segmento TT tem seu $lsw(n)$ variável, de acordo com o número de mensagens no respectivo EC. O protocolo permite estabelecer uma duração de máximo para o segmento TT e correspondentemente uma largura de banda máxima para este tipo de tráfego. O restante é destinado para o segmento ET. Não existem requisições explícitas por mensagem do mestre para os nós escravos, desta forma cada nó escravo trabalha de forma independente seguindo somente a mensagem TM no inicio de cada ciclo EC. As eventuais colisões entre mensagens dos nós escravos em ambos os segmentos são resolvidas pelo nativo mecanismo de arbitramento do protocolo CAN. Vale ressaltar que o segmento TT possui somente os instantes de inicio e fim sem as divisões de tempo ou janelas de transmissão como existe em outros protocolos. Desta forma, todas as mensagens TT são transmitidas no mesmo instante temporal no início do segmento e ordenadas através do mecanismo de arbitramento. No segmento ET os nós com transmissões pendentes podem iniciar as transmissões imediatamente no inicio do segmento. Para garantir as restrições temporais das mensagens do segmento TT o protocolo FTT-CAN o protege de interferências oriundas de requisições assíncronas dentro do segmento TT. Para isto, um isolamento temporal entre ambos os segmentos é forçado, prevenindo a tentativa de transmissões que não possam ser completadas dentro do segmento ET. Este isolamento é alcançado removendo do *buffer* de transmissão do controlador de rede qualquer pedido pendente que não possa ser realizado até a conclusão daquele segmento, mantendo-os em filas de transmissão para o próximo segmento ET. Deste modo, uma pequena quantia de tempo de inatividade pode surgir no fim do segmento ET. Por outro lado, no fim do segmento TT, outra pequena quantia de tempo de inatividade surge devido às variações geradas pelo mecanismo de *bit stuffing* usado na codificação física do protocolo CAN. Relacionado ao efeito do mecanismo de *bit stuffing*, em (ALMEIDA; PEDREIRAS; FONSECA, 2002) considera-se o pior caso de ocorrência. Isto negligencia uma fonte de *jitter* no segmento TT. A seção 4 discute este problema que afeta o comportamento temporal deste segmento.

O protocolo FTT-CAN possui dois serviços de comunicação, um para cada segmento. *Synchronous Messaging System* (SMS) e *Asynchronous Messaging System* (AMS). O serviço de SMS segue o modelo produtor-consumidor enquanto o AMS oferece somente os serviços básicos de transmissão e recepção usados quando a camada de aplicações possui mensagens aperiódicas para transmissão. Os nós escravos com mensagens aperiódicas pendentes podem transmitir somente durante o segmento ET que é o tempo restante e que não é usado pelo segmento TT no ciclo EC. As transmissões de mensagens periódicas são realizadas de maneira autônoma, do ponto de vista do software de aplicação. Isto é, o protocolo é responsável pela transmissão de todas as mensagens dentro do segmento TT, de forma que as tarefas do software

de aplicação não necessitam invocar os serviços de envio e recepção de mensagens. No segmento ET, porém, mensagens são transmitidas em resposta para pedidos de explícito da camada de aplicações.

## 3.1 Escalonamento de tarefas em um sistema FTT-CAN

Da mesma forma que o escalonamento de mensagens TT é realizado no protocolo FTT-CAN, as tarefas também podem ser despachadas (CALHA; FONSECA, 2002). Deste modo o nó mestre assume o controle do sistema distribuído de forma global, ativando mensagens e tarefas no barramento através da mensagem TM. Esta se torna um mecanismo confiável, provendo um sincronismo global entre nós presentes na rede. Este mecanismo garante a viabilidade do propósito de escalonamento de tarefas no sistema sem um consumo significativo de recursos computacionais.

Em SDTR tarefas podem produzir ou consumir uma mensagem, ou em alguns casos ambos. Uma tarefa que gera algum dado é chamada uma tarefa produtora e por outro lado uma tarefa que usa dados para qualquer propósito é chamada de tarefa consumidora, que constitui a estratégia produtor-consumidor. As demais tarefas do sistema que não interagem com outras, são, deste modo, chamadas de tarefas independentes ou *stand-alone*. As interações entre tarefas podem ser representadas como grafos de precedência, que mostram as relações de dependências entre tarefas e mensagens, e também mostram o fluxo de dados entre esses. O nó mestre pode controlar a execução de tarefas e mensagens associadas com o fluxo de dados do produtor até as tarefas consumidoras. Deste modo o paradigma FTT, quer seja sobre CAN ou Ethernet, pode garantir o comportamento de tempo real da execução de tarefas e transmissão de mensagens de forma integrada. Em (CALHA; FONSECA, 2002) e (CALHA; SILVA; FONSECA, 2006) considera-se tarefas TT escalonadas juntas a mensagens TT. Para este propósito o campo de dados da mensagem TM deve acomodar duas áreas de flags de ativação, uma para tarefas e outra para mensagens. Como os flags de mensagens TT, cada bit da área separada para codificação de tarefas TT indica se uma tarefa deve ser despachada no ciclo EC corrente ou não. É importante definir o deslocamento de fase relativa, *Ph*, de cada tarefa e mensagem.

Uma restrição imposta pela mensagem TM é que todo período, $P$, e fase, *Ph*, para ambas as tarefas e mensagens, devem ser arredondados para um valor múltiplo do ciclo EC. A duração de ciclo é a unidade básica de tempo para o sistema.

Agora, o nó mestre necessita negociar mensagens e tarefas, então o *STR* deve ser atualizado com os atributos das tarefas TT. Uma tarefa TT produtora ou consumidora tem o mesmo período de sua respectiva mensagem. Um conjunto destas tarefas tem alguns atributos.

$$SRT \equiv \{ST_i(N_i, MP_i, MC_i, C_i, Ph_i, P_i, D_i, Pr_i), i = 1...N_{st}\}$$

Onde $ST_i$ representa uma tarefa TT, $C$ é o tempo de execução no pior caso, $P$ o período, $D$ o *deadline* medido relativo ao instante de liberação, *Pr* a prioridade, $N$ o nó onde a tarefa é executada, *Ph* o deslocamento de fase relativa que determina o primeiro instante de liberação após a partida do sistema. Para tarefas interativas existem ainda dois atributos adicionais, *MP*, a mensagem produzida e *MC*, a mensagem consumida. Em (CALHA; SILVA; FONSECA, 2006) considera-se o segmento de execução como o intervalo entre a liberação e o *deadline* da tarefa. No conjunto de atributos da mensagem TT dois novos atributos são necessárias.

$$SRT \equiv \{SM_i(PT, CTL_{i,j}, DLC_i, C_i, Ph_i, P_i, D_i, Pr_i), i = 1...N_{sm}\}$$

O primeiro é o *PT*, que representa a tarefa produtora e outro é *CTL*, que consiste em uma lista de tarefas consumidoras. Esta lista é necessária porque mais de uma tarefa pode consumir a saída de uma tarefa produtora. Também em (CALHA; SILVA; FONSECA, 2006) o segmento de transmissão é considerado como o intervalo entre a liberação e o deadline da mensagem. Atualmente, em literaturas sobre sistemas FTT existem duas abordagens para escalonamento de tarefas. Uma abordagem denominada de *Net-Centric*, onde mensagens impõem restrições sobre o conjunto de tarefas, e outra denominada *Node-Centric* onde tarefas impõem restrições sobre o conjunto de mensagens. O principal fator é o uso dos recursos de sistema, a rede para transmissão de mensagem e os nós para execução de tarefa. De acordo com estes fatores, quatro combinações podem então ocorrer:

- Baixa carga na rede e baixa carga computacional no nó, onde qualquer abordagem pode ser usada;
- Alta carga na rede e baixa carga computacional no nó, correspondem a abordagem Net-Centric;
- Baixa carga na rede e alta carga computacional no nó, correspondem a abordagem Node-Centric;
- Alta carga na rede e alta carga computacional no nó, onde ambas as abordagens deveriam ser consideradas a fim de selecionar-se a com melhor desempenho.

A respeito de escalonamento holístico (*holistic scheduling*), isto é, o escalonamento conjunto de tarefas e mensagens o ponto de partida é verificar o fluxo de dados do sistema. Após

a construção do grafo de relação de precedência a fim de definir *Ph* para mensagens sem considerar tarefas, o macro ciclo pode ser calculado. O macro ciclo é um intervalo que compreende um ou mais ciclos EC, com um conjunto de tarefas e mensagens, que será indefinidamente repetido, até o fim de uma tarefa ou a ocorrência de um erro (CALHA; FONSECA, 2002)..

Para o propósito de escalonamento, qualquer algoritmo pode ser usado. Existem duas abordagens para escalonamento de tarefas e mensagens, que são: escalonamento independente e dependente. O escalonamento pode ser simultaneamente realizado para tarefas e mensagens considerando escalonamento independente ou o escalonamento de mensagens é realizado antes do escalonamento de tarefas na abordagem dependente.

## 4 LIMITAÇÕES DO PROTOCOLO FTT-CAN

Alguns problemas que têm impacto sobre o desempenho de sistema de controle via rede baseado em protocolo FTT-CAN foram identificados durante a implementação original do protocolo FTT-CAN proposto em (ALMEIDA; PEDREIRAS; FONSECA, 2002). Estas desvantagens são apresentados nas seções a seguir. Inicialmente é apresentado a transmissão síncrona de mensagem, onde são citados dois aspectos negligenciados que geram *jitter* neste tráfego. Por fim, é apresentado o disparo de tarefas síncronas, onde foi identificado deficiência na sua execução de tarefas síncronas. Estes aspectos ainda não foram abordados na literatura.

### 4.1 Tráfego de mensagens Time-Triggered

Apesar dos problemas identificados afetarem tanto fase TT como a ET, o foco deste trabalho é sobre o tempo de resposta na fase TT (tráfego síncrono), porque este exige um elevado grau de previsibilidade quando aplicado em SDTR. Em sistemas onde a execução de uma tarefa antecede a transmissão de mensagens, qualquer variação na liberação da tarefa e tempo de execução produz atrasos nas mensagens. Fontes de variação do tempo de liberação de tarefas são por exemplo: preempção por tarefas de mais alta prioridade, variação na execução do escalonador ou na latência de interrupção e até mesmo devido à variação na própria execução da tarefa. Tais problemas induzem defasagens no tempo na execução de tarefas que são causados pela variação de tempo de resposta das tarefas produtoras de mensagens em sistemas sem qualquer meio de sincronismo. Esta defasagem é denominada como *phasing*. Nolte (NOLTE; HANSSON; NORSTRÖM, 2002) aborda este efeito analisando o tempo de resposta de mensagem em SDTR sobre o protocolo CAN. Os atrasos também podem ser introduzidos pela estratégia de controle de acesso ao meio de comunicação do protocolo utilizado. Outra fonte de variabilidade temporal na comunicação é relacionado ao mecanismo de *bit stuffing* do protocolo CAN.. O número de *stuff* bits inseridos depende do padrão de bits de uma mensagem em questão, por exemplo: uma mensagem CAN com 8 bytes de dados e 47 bits de controle pode ser transmitida com 0 a 19 *stuff* bits. Isso torna difícil uma análise precisa do tempo de resposta no protocolo CAN, e, por esta razão, alguns trabalhos não consideram esta incidência (TINDELL; BURNS; WELLINGS, 1995). O impacto do *bit stuffing* aumenta quando a rede possui baixa velocidade de transmissão. Quanto menor for a taxa de transmissão da rede, maior será o impacto na ordem de unidades de tempo. Em (NOLTE; HANSSON; NORSTRÖM, 2003) é apresentado uma análise probabilística de pior caso de tempo de transmissão baseada na utilização da distribuições de *bit stuffing* em vez de um valor de incidência máxima no pior caso. A Figura 3 ilustra este problema no FTT-CAN, considerando uma situação hipotética de uma rede com quatro nós sendo que cada um com uma mensagem TT com prioridades diferentes. O processo de liberação das mensagens, que é responsável pela transmissão das mensagens TT em cada nó pode ser afetado por variações no serviço de interrupção e no escalonador do RTOS (Real Time Operating System) em uso. Esta situação pode gerar uma condição de bloqueio devido à defasagem temporal do processo de liberação de mensagem, ilustrado na Figura 3. A influência do método *bit stuffing* também é representado na Figura 3. Este conduz a uma maior latência que gera um *jitter* na transmissão de mensagem subsequente.

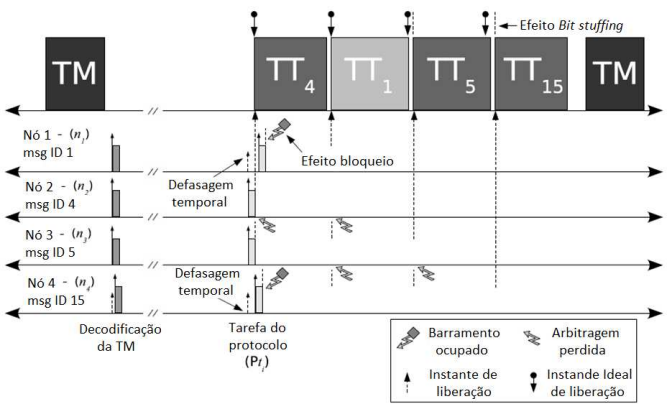


Figura 3: Impacto do *bit stuffing* e efeito bloqueio

Quanto maior o número de mensagens a serem transmitidas na fase TT, maior será este impacto. Mensagens escalonadas para transmissão no fim da fase TT sofrem um maior *jitter*. Uma vez que esse atraso é medido em bits, quanto menor for a taxa de transmissão maior será o impacto.

## 4.2 Execução de tarefas Time-Triggered

Em relação ao escalonamento holístico no protocolo FTT-CAN, em (CALHA; FONSECA, 2002) e (CALHA; SILVA; FONSECA, 2006) não é especificada uma janela para as tarefas síncronas (tarefas TT). Em (CALHA; SILVA; FONSECA, 2006) considera-se a janela de execução de uma tarefa síncrona como o intervalo entre o instante de liberação e o *deadline* da tarefa. No entanto, os autores não definem um instante para a liberação de uma determinada tarefa TT em um dado EC. Além disso, em (CALHA; FONSECA, 2002) a janela de transmissão é definida como o intervalo entre o instante de liberação e o *deadline* da mensagem. Mas, neste caso, todas as mensagens TT são liberadas no ponto inicial da fase TT. De acordo com (CALHA; FONSECA, 2002) uma tarefa TT é liberada assim que a TM é decodificada, onde cada nó verifica se há alguma tarefa para execução no respectivo ciclo EC. Deste modo, todas as tarefas TT terão a execução na fase ET. Esta abordagem degrada o tráfego ET com execuções de tarefa não-assíncrona, gerando assim interferências uma vez que as tarefas ET têm prioridade menor do que as tarefas TT. Outro problema na utilização desta abordagem é a variação no processo de decodificação da TM, que é inevitável, porque podem ocorrer TMs com mais tarefas/mensagens decodificadas do que outras, por isso o tempo de computação é variável.

Considerando as situações apresentadas acima, a Figura 4 apresenta um exemplo desta abordagem. Como pode ser visto na figura, tarefa TT executando fora de sua respectiva fase lesa a execução de tarefas ET. O tempo de execução de uma tarefa TT irá intervir no tempo de resposta a eventos externos que podem ocorrer durante o tempo de sua execução. Afinal, a fase ET foi destinada a serviços assíncronos. Por exemplo, no $EC_i$ da Figura 4, não existe qualquer mensagem ET pendente para transmissão, por isso a mensagem $ET_1$ é transmitida somente após a tarefa $Ts_1$ terminar sua execução e a tarefa $Ta_1$ processar seu respectivo evento externo que ocorreu durante a execução de $Ts_1$. No próximo EC, $EC_{i+1}$, existem duas mensagens ET ($ET_3$ e $ET_4$) pendentes desde a última EC, que são transmitidas corretamente em sua fase sem interferências, pois segundo o protocolo FTT-CAN todas as mensagens ET pendentes são liberadas no meio de transmissão da mensagem TM como ilustrado na Figura 4. Deste modo, somente após as tarefas $Ts_2$ and $Ts_3$ terminarem sua execução a tarefa ET $Ta_2$, resultante de outro evento externo, poderá ser executada. No entanto, após a execução de $Ta_2$ não há tempo para a transmissão da mensagem ET resultante da tarefa, e esta será colocado em uma fila de espera para o próximo ciclo EC. Consequentemente, com estas evidências, podemos concluir que executar tarefas TT na fase ET pode gerar interferências nas tarefas e mensagens ET, provocando um *dead time* nesta fase (região sombreada na Figura 4).

# 5 NOVA ABORDAGEM PROPOSTA

## 5.1 Tráfego de mensagens Time-Triggered

Para superar os inconvenientes apresentados na transmissão de mensagem TT, um método baseado em *offset* é apresentado para forçar a ordenação correta das mensagens TT de forma a reduzir o *jitter* inerente do efeito bloqueio e do método *bit stuffing*. O trabalho original sobre este método *offset* foi realizado por Liu e Sun (SUN; LIU, 1996), que é um protocolo de sincronização para garantir a relação de precedência entre as tarefas periódicas pela a inclusão de deslocamentos no tempo (*offset*). Busca-se atribuir um *offset* para toda tarefa de protocolo responsável pela liberação das mensagens TT em cada EC, ($OP_{ti}$), de modo que sua liberação será sempre superior ao pior caso do tempo de transmissão do conjunto de mensagens TT ($C_{TTm}$). Com isto, reduz-se o pessimismo gerado pelo *bit stuffing* e garante-se a eliminação do efeito de bloqueio na fase TT do protocolo FTT-CAN. Todavia, é necessário considerar-se o pior caso de ocorrências de *bit stuffing* e incluir o espaço *inter-frame* para cálculo do $C_{TTm}$. O *offset* de sistema ($O_{sys}$) é um valor fixo de poucos bits, que é ajustado em tempo de projeto assim como o $C_{TTm}$. O $O_{sys}$ tem a finalidade de assegurar uma lacuna temporal entre o final de uma mensagem TT e o início de outra. Esta lacuna varia devido às ocorrências de *bit stuffing*. O cálculo de *offset* para cada mensagem TT é dado pela equação:

$$O_{P_{t_i}} = (C_{TT_m} + O_{sys}) m_{indexEC}$$

Onde $m_{indexEC}$ é a posição da mensagem no respectivo ciclo EC que está em conformidade com a prioridade, e este deve ser $\geq$ *0*. Assim, a primeira mensagem no respectivo EC terá $O_{Pti=0}$ = *0*.

Aplicando este método, cada tarefa de protocolo terá um deslocamento $O_P$ no tempo, promovendo a liberação de mensagem num instante de barramento ocioso após a transmissão da mensagem anterior. Além disso, este método torna possível a utilização de mensagens TT com tamanho de dados diferentes, o que não era possível na implementação original do protocolo FTT-CAN. O custo dessa nova abordagem é uma perda de largura de banda quando ocorrer mensagens com tempo de transmissão menor do que $C_{TTm}$.

## 5.2 Execução de tarefas Time-Triggered

É notório que interferências na fase ET na implementação original do FTT-CAN podem ser geradas através da execução de tarefas TT. Para tentar contornar este problema, uma

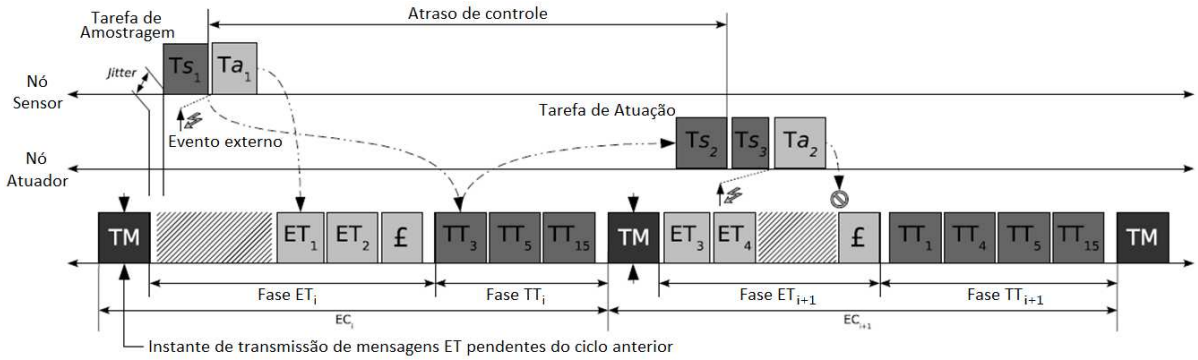


Figura 4: Disparo de tarefas no FTT-CAN

proposta de janelas de execução é apresentada. O objetivo é restringir a execução de tarefas TT dentro de sua respectiva fase ou o mais próximo possível, reduzindo as interferências na fase ET. Assim, um tarefa TT é liberada dentro de uma janela de tarefa definida. A duração dessa janela de tempo é fixa e definida considerando os valores máximos WCET do conjunto de tarefa TT, adicionados a um fator *jitter*. Este fator deve ser o pior caso do *jitter* de execução da tarefa e do serviço de interrupção. A janela da tarefa produtora sempre antecede a janela da mensagem a ser produzida, e a janela de uma tarefa consumidora é sempre realizada no próximo EC da janela da mensagem a ser consumida. A Figura 5 mostra esta abordagem proposta.

A janela de tarefa da primeira mensagem a ser produzida na fase TT do ciclo EC será lançada na parte final da fase ET. A fim de considerar-se a possibilidade de mensagens com o mesmo período em um mesmo nó, define-se $EC_{period}$=*minorPeriodOfMessage/2*. Desta forma, não se pode aplicar deslocamentos entre mensagens TT em diferentes EC permitindo que a janela da tarefa produtora preceda a janela da mensagem. A atribuição das janelas de tarefas é feita *on-line* durante cada EC enquanto o processo de decodificação da TM está sendo executado. Entretanto, a garantia de execução das tarefas TT é realizada através de escalonamento a priori do conjunto de tarefas possível para o sistema.

# 6 IMPLEMENTAÇÃO E VALIDAÇÃO EXPERIMENTAL

A fim de validar as extensões propostas ao protocolo FTT-CAN, uma arquitetura eletrônica veicular hipotética foi desenvolvida contendo seis ECUs. A idéia principal era criar uma arquitetura automotiva *drive-by-wire* que poderia ser usada como plataforma eletro-eletrônica para futuros trabalhos no Departamento de Engenharia Elétrica da UFRGS. A arquitetura proposta inclui algumas funções veículares, tais como: *steer-by-wire*; assistência a estacionamento dianteiro e traseiro; indicação de velocidade do veículo; indicação de nível de combustível e indicação de temperatura motor. Com exceção da primeira função, todas as funções estão presentes no painel de instrumentos que é responsável por apresentar as respectivas informações para o condutor. A seguir as funções distribuídas entre os nós são apresentadas:

ECU1 - leitura do estado do sistema de assistência de estacionamento dianteiro e temperatura do motor;

ECU2 - leitura da velocidade do veículo, ângulo das rodas e controle dos ângulo das rodas de acordo com o ângulo do volante;

ECU3 - FTT-CAN mestre, gera a mensagem TM;

ECU4 - leitura do ângulo do volante e controle de força ( *force-feedback)*;

ECU5 - leitura de todas as informações: velocidade do veículo, temperatura do motor, nível do tanque de combustível, e estado do sistema de assistência de estacionamento;

ECU6 - leitura do estado do sistema de assistência de estacionamento traseiro e nível do tanque de combustível.

## 6.1 Plataforma de Hardware

Duas plataformas de hardware distintas forma utilizadas: uma para as ECUs, que são componentes presentes no corpo e sistema eletrônico do veículo - sistema de assistência de estacionamento e *steer-by-wire*, respectivamente e outra plataforma de hardware para o painel de instrumentos. Cada ECU é composta por um hardware baseado em microproces-

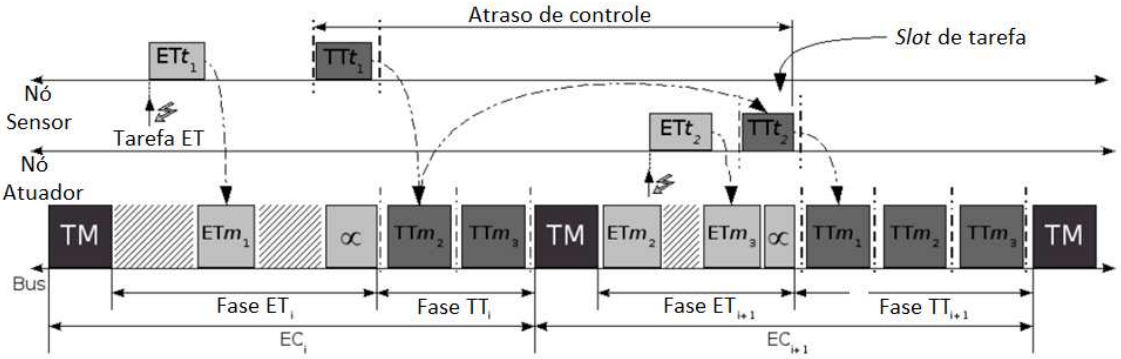


Figura 5: Nova abordagem para disparo de tarefas

sadores Freescale Coldfire 5282 66MHz , que tem um controlador CAN embutido, 16MB de memória RAM e 2MB de memória Flash. O painel de instrumentos, por sua vez, é uma plataforma de hardware com processador Freescale 5200 PowerPC. Esta placa tem 64MB de memória RAM e 32MB de memória flash, o processador executa a 400Mhz. Os principais dispositivos do processador são: CAN, PCI (Peripheral Component Interconnect) e controladores Ethernet embutido. Montado no slot PCI está uma placa de vídeo, a qual é conectada a um LCD de 8" de largura de tela touchscreen.

## 6.2 Plataforma de Software

A estrutura de software das ECUs que compoem o corpo eletrônico do veículo é composta do sistema operacional $\mu$Clinux (DIONNE; DURRANT, 2002) com RTAI (*Real-Time Application Interface*) (MANTEGAZZA, 2001). A implementação do FTT-CAN com as novas abordagens foi implementada nesta estrutura de software. A Figura 6 apresenta os blocos implementados – em branco – e os blocos existente na estrutura de software selecionada.

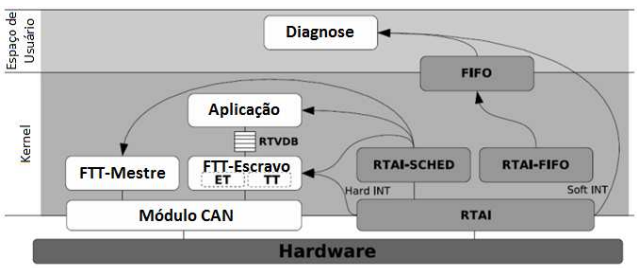


Figura 6: Arquitetura de sofware do FTT-CAN com RTAI sob $\mu$Clinux

O painel de instrumentos possui uma arquitetura de software baseada no Linux com a extensão RTAI, porém todas as funções do painel de instrumentos são executadas em modo usuário normal (na espaço de usuário). O protocolo FTT-CAN deste componente foi implementado usando os recursos RTAI e se comunicando com as aplicações usando IPC (*Inter Process Communication*) e compartilhando principalmente memória FIFOs.

A Figura 7 apresenta a arquitetura de software usado no painel de instrumentos. Para construção da interface homem-máquina foi utilizado o software Lintouch. Este possui código fonte aberto e é utilizado principalmente como interface para processos de automação industriais. Para utilizar Lintouch no projeto foi necessário criar-se um *plugin* para FTT-CAN, para obter todas as informações de telemetria de outros nós da rede.

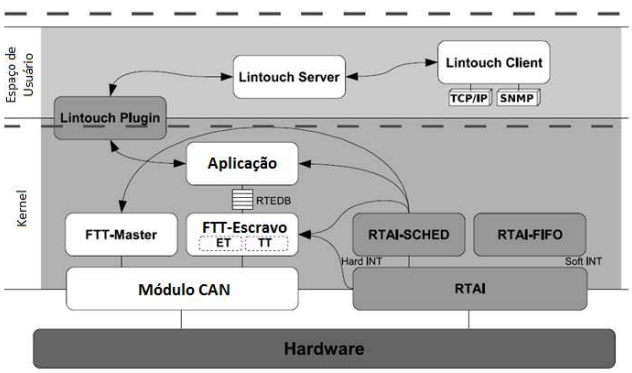


Figura 7: Arquitetura de software do painel de instrumentos

## 6.3 Resultados

O conjunto de mensagens e tarefas TT da aplicação são apresentadas nas Tabelas 1 e 2, respectivamente. A prioridade de

mensagem, $P_i$, na Tabela 1 e a identificação de tarefa, $Id_i$, na Tabela 2 são diretamente mapeados para *flags* de mensagem e tarefa na mensagem TM enviada pelo nó FTT-CAN mestre. A última coluna da Tabela 2 apresenta as mensagens consumidas ($C$) e produzidas ($P$) pelas respectivas tarefas TT da aplicação. O ciclo elementar (EC) do FTT-CAN para esta aplicação é de 2,5 ms para garantir a integração progressiva das mensagens e tarefas de mesmo período. O menor período de mensagem e de tarefa é de 5ms. Todas as medidas foram feitas por um conjunto de 3000 amostras de mensagens e tarefas, com uma rede CAN com taxa de 250kbps e 4$\mu$s de resolução de *timer*.

Tabela 1: Conjunto de mensagens TT

| Msg | Descrição | ECU | $P_i$ | $T_i$ | $Ph_i$ |
|---|---|---|---|---|---|
| 1 | Ângulo de atuação volante | 4 | 7 | 5 | 0 |
| 2 | Velocidade do veículo | 2 | 6 | 5 | 2,5 |
| 3 | Ângulo das rodas | 2 | 5 | 5 | 2,5 |
| 4 | Temperatura do motor | 1 | 4 | 500 | 5 |
| 5 | Nível de combustível | 6 | 3 | 500 | 10 |
| 6 | Sensor colisão frontal | 1 | 2 | 200 | 15 |
| 7 | Sensor colisão traseiro | 6 | 1 | 200 | 20 |

### 6.3.1 Tráfego de mensagens TT

O *jitter* mensurado é apresentado para a mensagem TM e mensagens 1, 2, 5, 7. De acordo com os parâmetros da Tabela 1, as mensagens 3, 5 e 7 são sempre as primeiras da fase TT do EC devido às suas prioridades $P_i$ e defasagem $Ph_i$, e por isso estas sofrem maior impacto do efeito bloqueio. As mensagens 1 e 2 estão no fim da fase TT, no entanto, estas sofrem com o pessimismo do método *bit stuffing*. As medidas sem a abordagem com *offset* atingiu um alto grau de bloqueio, sendo 11,6% e 10,2% do conjunto amostrado para as mensagens 5 e 7, respectivamente. Além disso, um release *jitter* entre 28$\mu$s ($J_{max}$) e -28$\mu$s ($J_{min}$) para as mensagens 1 e 2 geradas pelo efeito *bit stuffing*. A Tabela 3 apresenta os resultados da abordagem proposta. $J_{max}$ e $J_{min}$ são as máximas e mínimas variações (*jitter*), $\sigma$ é o desvio padrão e % é a porcentagem de ocorrências do efeito bloqueio. A adoção da abordagem proposta leva a uma melhoria do *jitter* imposto pelos *stuff* bits e elimina completamente as situações de bloqueio.

Execução de tarefas TT

A Tabela 4 apresenta o *jitter* de liberação da tarefa produtora *Id 0*, e a tarefa consumidora/produtora *Id 1* usando abordagem com janelas de execução.

Tabela 2: Conjunto de tarefas TT

| $Id_i$ | Descrição | ECU | $T_i$ | $Ph_i$ | C/P Msg |
|---|---|---|---|---|---|
| 0 | Amostragem do ângulo do volante | 4 | 5 | 0 | P M1 |
| 1 | Controle do ângulo do volante | 2 | 5 | 2,5 | C M1 P M3 |
| 2 | Amostragem da velocidade do veículo | 1 | 5 | 2,5 | P M2 |
| 3 | Controle feedback do volante | 4 | 5 | 5 | C M2 C M3 |
| 4 | Amostragem temperatura motor | 1 | 500 | 5 | P M4 |
| 5 | Amostragem nível de combustível | 6 | 500 | 10 | P M5 |
| 6 | Amostragem sensor de colisão frontal | 1 | 200 | 15 | P M6 |
| 7 | Amostragem sensor de colisão traseiro | 6 | 200 | 20 | P M7 |
| 8 | Indicação temperatura motor | 5 | 500 | 7,5 | C M4 |
| 9 | Indicação nível de combustível | 5 | 500 | 12,5 | C M5 |
| 10 | Indicação sensor colisão frontal | 5 | 200 | 17,5 | C M6 |
| 11 | Indicação sensor colisão traseiro | 5 | 200 | 22,5 | C M7 |

Os resultados foram similares para outras tarefas. A Figura 8 apresenta uma imagem de um ciclo EC em um osciloscópio. O canal 1 mostra a execução do processo do protocolo FTT-CAN, isto é, o processo decodificador da mensagem TM, o processo de liberação de mensagem TT e a execução de uma tarefa TT produtora dentro da janela. O canal 2 mostra o sinal do barramento CAN com o ciclo EC, contendo a mensagem TM, as fases ET e TT.

A proposta permite algumas vantagens em relação a abordagem original do protocolo FTT-CAN, que são: (i) introdução da capacidade de um sistema ser integrado junto a outros sem perda e danos nos requisitos temporais de aplicação (*composability*) e com maior resolução que a proposta original do protocolo, sendo possível com os instantes fixos de

Tabela 3: Resultado com método offset

| Msg | $J_max$ | $J_min$ | $\sigma$ |
|---|---|---|---|
| 1 | 16 | -16 | 3,58 |
| 2 | 16 | -16 | 3,53 |
| 5 | 4 | -4 | 2,80 |
| 7 | 8 | -4 | 2,53 |
| TM | 4 | -4 | 2,02 |

Tabela 4: Resultado com método slot de tarefa

| $Task(Id_i)$ | $J_max$ | $J_min$ | $\sigma$ |
|---|---|---|---|
| 0 | 12 | -8 | 3,56 |
| 1 | 12 | -8 | 3,59 |

transmissão e execução de mensagens e tarefas; (ii) utilização de mensagens TT com comprimento de dados diferentes sem degradação do tempo de resposta de outras mensagens do sistema; (iii) garantia de tempo de resposta de mensagens TT mesmo em condições de falhas de outros nós e a restrição de execução de tarefas TT dentro de sua respectiva fase (ou próximo) reduzindo o impacto no tempo de resposta de eventos.

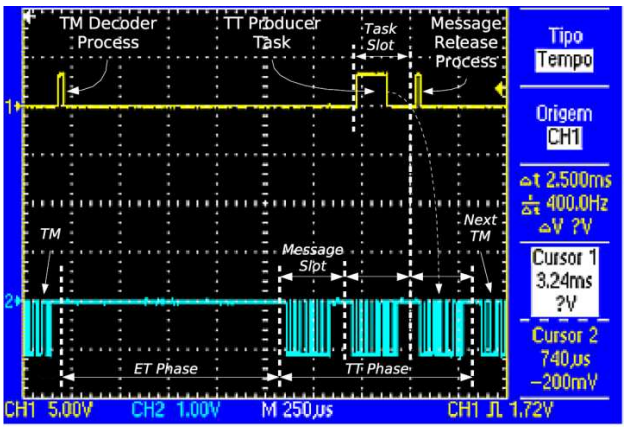


Figura 8: Resultado da implementação, ciclo EC com execução de uma tarefa T

## 7 CONCLUSÃO

Este artigo apresentou uma visão geral sobre alguns protocolos de comunicação empregados em aplicações embarcadas veiculares. Especialmente, o protocolo FTT-CAN foi alvo do trabalho realizado, uma vez que este protocolo permite combinar formas de comunicação temporizadas e orientadas a eventos. Algumas limitações do protocolo FTT-CAN foram apresentadas e soluções para melhorias do desempenho do protocolo foram propostas. O primeiro problema verificado no protocolo é quando há uma execução de tarefa precedendo a transmissão de uma mensagem, onde qualquer variação em função da execução ou da liberação da tarefa induz atrasos na transmissão da respectiva mensagem. Como apresentado, esse problema pode gerar uma defasagem temporal na liberação dessas tarefas, que, por consequência gera variação no tempo de resposta da mesma e, assim, pode gerar condições de bloqueio do ponto inicial da fase síncrona de transmissão de mensagens. Outro aspecto negligenciado no protocolo original, em relação à transmissão de mensagens, foi a variabilidade do tempo de transmissão gerado pelo método *bit-stuffing* nativo do protocolo CAN que aumenta o comprimento das mensagens gerando atrasos. Estes atrasos se somam quando, em uma mesma fase síncrona de um dado ciclo EC, existem várias mensagens. Assim, a última mensagem terá o maior impacto, ou seja, sempre terá o maior *jitter*. Outro aspecto identificado foi relacionado ao escalonamento de tarefas, onde não existia uma janela de execução especifica para tarefas síncronas. Assim, uma tarefa síncrona era liberada logo em seguida à codificação da mensagem TM no inicio do ciclo EC e acima da fase assíncrona. Esta situação levava a interferências nas tarefas e mensagens assíncronas e consequentemente prejuízos nos seus respectivos tempos de resposta. No ponto de vista de sistemas de controle via rede, as consequências destes aspectos identificados podem ser mais severas, porque pode afetar a dinâmica do processo pelo aumento ou grande variações do atraso de controle. As soluções propostas foram avaliadas experimentalmente e os resultados obtidos comprovaram a eficiência da proposta apresentada.

## REFERÊNCIAS BIBLIOGRÁFICAS


TTA-GROUP. Time-Triggered Protocol TTP/C High-Level Specification Document: Protocol Version 1.1. [S.l.]: TTA-Group, 2003.

CONSORTIUM, F. FlexRay Communications System Protocol Specification Version 2.0. [S.l.]: FlexRay Consortium, Copyright 2004.

ALMEIDA, L. A word for operational flexibility in distributed safety-critical systems. $8^{th}$ IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, [S.l.], p.177–184, Jan. 2003.

ALMEIDA, L.; PEDREIRAS, P.; FONSECA, J. The FTT-CAN Protocol - Why and How. IEEE Transactions on Industrial Electronics, [S.l.], v.49, n.6, p.1189– 1201, Dec. 2002.

CALHA, M.; FONSECA, J. Adapting FTT-CAN for the joint dispatching of tasks and messages. Factory Com-

munication Systems 4th IEEE InternationalWorkshop, [S.l.], p.117– 124, 2002.

CALHA, M.; SILVA, V.; FONSECA, J. Kernel design for FTT-CAN systems. 6th IEEE InternationalWorkshop on Factory Communication Systems, [S.l.], 2006.

CONSORTIUM, L. LIN Specification Package Revision 2.1. [S.l.: s.n.], 2006.

USA, M.-B. Domestic Digital Bus (D2B). http://www.mercedestechstore.com/.

GRZEMBA, P. D. I. A. MOST - The Automotive Multimedia Network. [S.l.]: Franzis, ISBN 978-3-7723-5316-1, 2007.

KOPETZ, H.; BAUER, G. The time-triggered architecture. In: SPECIAL ISSUE ONMODELING AND DESIGN OF EMBEDDED SOFTWARE, [S.l.], v.91, n.1, p.112–126, 2001.

KOPETZ, H. Real-Time Systems - Design Principles for Distributed Embedded Applications. [S.l.]: Kluwer Academic Publishers, 1997.

TTA-GROUP - Time-triggered protocol TTP/C high-level specification document protocol version 1.1. [S.l.]: TTA-Group, 2003.

CIA. CAN physical layer. http://www.can-cia.org/can/physical-layer/, 2001.

CIA. CAN in Automation (CiA). http://www.can-cia.org/., 2001

DIONNE, J.; DURRANT, M. Embedded Linux Microcontroller Project. http://www.uclinux.org.

NOLTE, T.; HANSSON, H.; NORSTRÖM, C. Effects of varying phasings of message queuings in CAN based systems. Proceedings of the 8th International Conference on Real-Time Computing Systems and Applications (RTCSA'02), Tokyo, Japan, p.261–266, March 2002.

NOLTE, T.; HANSSON, H.; NORSTRÖM, C. Probabilistic Worst-Case Response-Time Analysis for the Controller Area Network. Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03), Washington, DC, USA, p.200–207, May 2003.

MANTEGAZZA, P. RTAI project. http://www.rtai.org/.

RAHUL SHAH, X. D. An Introduction to TTCAN (Time Triggered Controller Area Network). http://cs.uni-salzburg.at/ck/teaching.

SUN, J.; LIU, J. Synchronization Protocols in Distributed Real-Time Systems. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 1996. Anais. . . [S.l.: s.n.], 1996. p.38–45.

TINDELL, K.; BURNS, A.; WELLINGS, A. Calculating Controller Area Network message response times. Control Engineering Practice, [S.l.], p.vol. 3, no. 8, pp, 1995.