

Especificação, Modelação e Projecto de Sistemas Embutidos

Tópicos sobre redes de comunicação para SE

Paulo Pedreiras

pbrp@ua.pt



Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

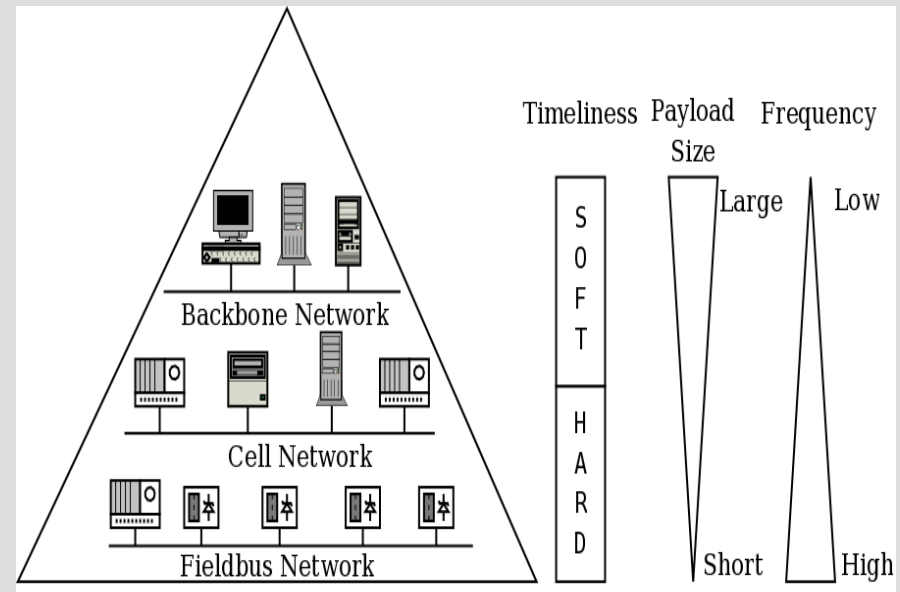
Redes de comunicação para SE

- Ao longo dos últimos anos tem havido uma crescente tendência para o emprego de **arquitecturas distribuídas** em SE
- Esta tendência nasceu no domínio da automação industrial, tendo por motivação a **redução** e **simplificação** da cablagem, simplificação da **manutenção** e **detecção de avarias**, ...
- Entretanto esta tendência **alargou-se** muitos dos domínios de aplicação dos SE
- Há diversas **especificidades** associadas a este domínio de aplicação, levando a que tenham sido desenvolvidos **protocolos específicos**

Requisitos

Estas redes possuem **requisitos distintos** das redes de dados “vulgares”

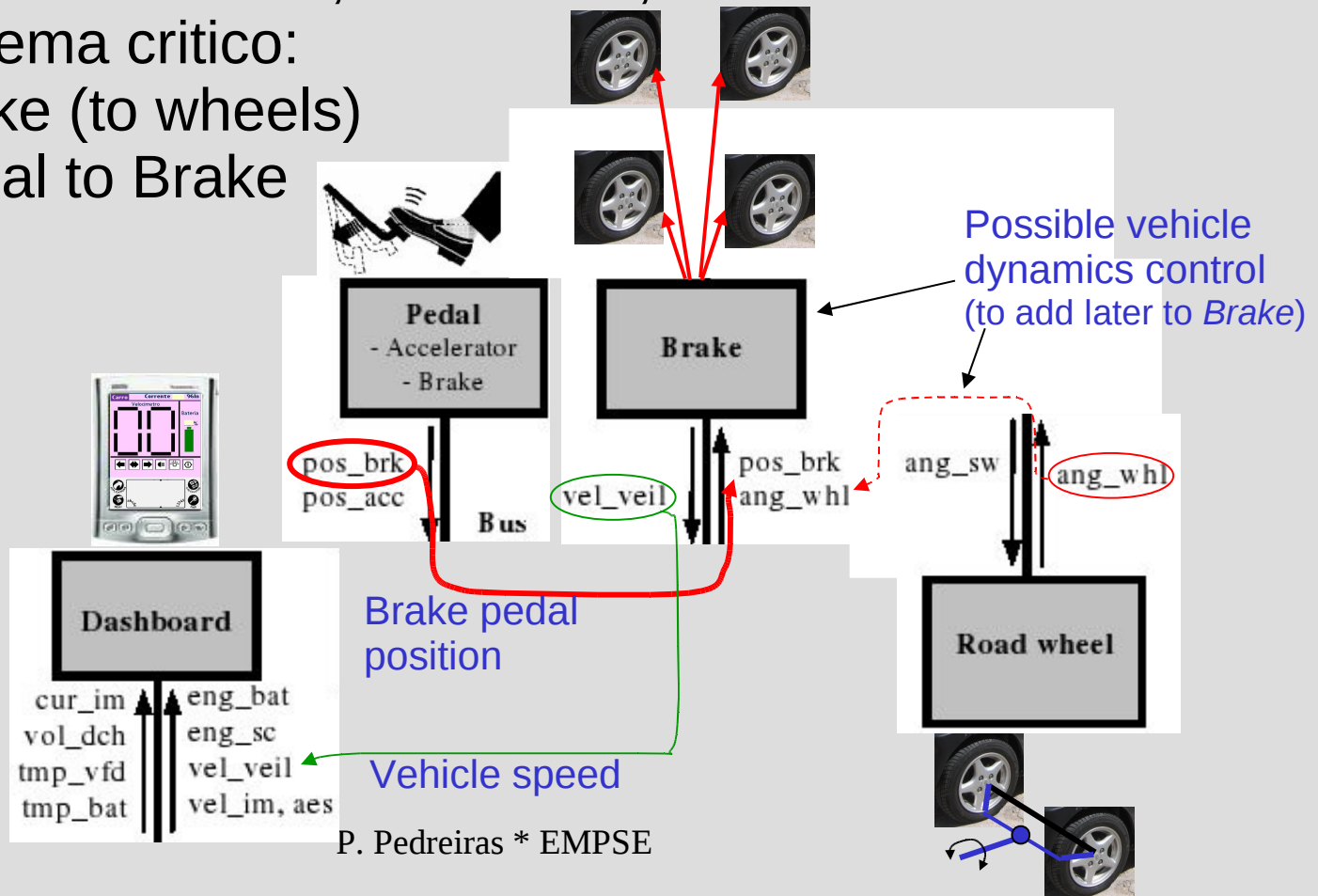
- **Padrão de tráfego**
 - Pacotes “pequenos” e muito frequentes
- **Baixo peso computacional**
 - *Stacks* simplificadas, típica/ 3 níveis (Phy/DL/App)
- **Comportamento tempo-real**
 - Latência e *jitter* determinísticos
- **Robustez e tolerância a falhas**
- Facilidade de **manutenção e diagnóstico**
- **Segurança, ...**



Exemplo: subsistema de travagem de um veículo eléctrico

- Exemplo: *Brake-by-wire*

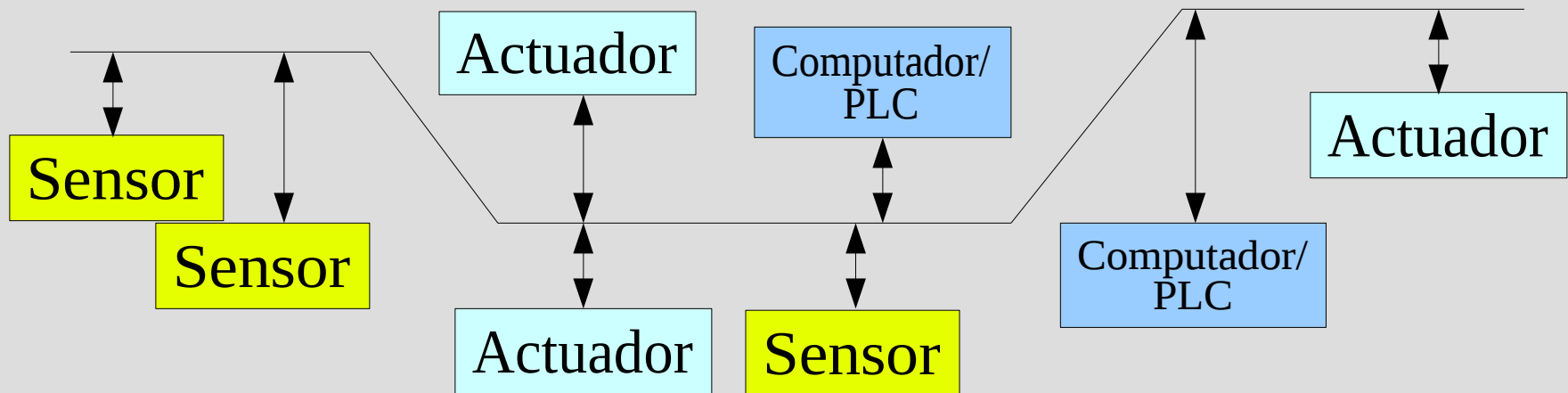
- **VEIL** – Veículo Eléctrico Isento de Licença de condução
- Parceria entre ISEC, IEETA/UA, IPCB
- Subsistema crítico:
 - Brake (to wheels)
 - Pedal to Brake



Arquitectura

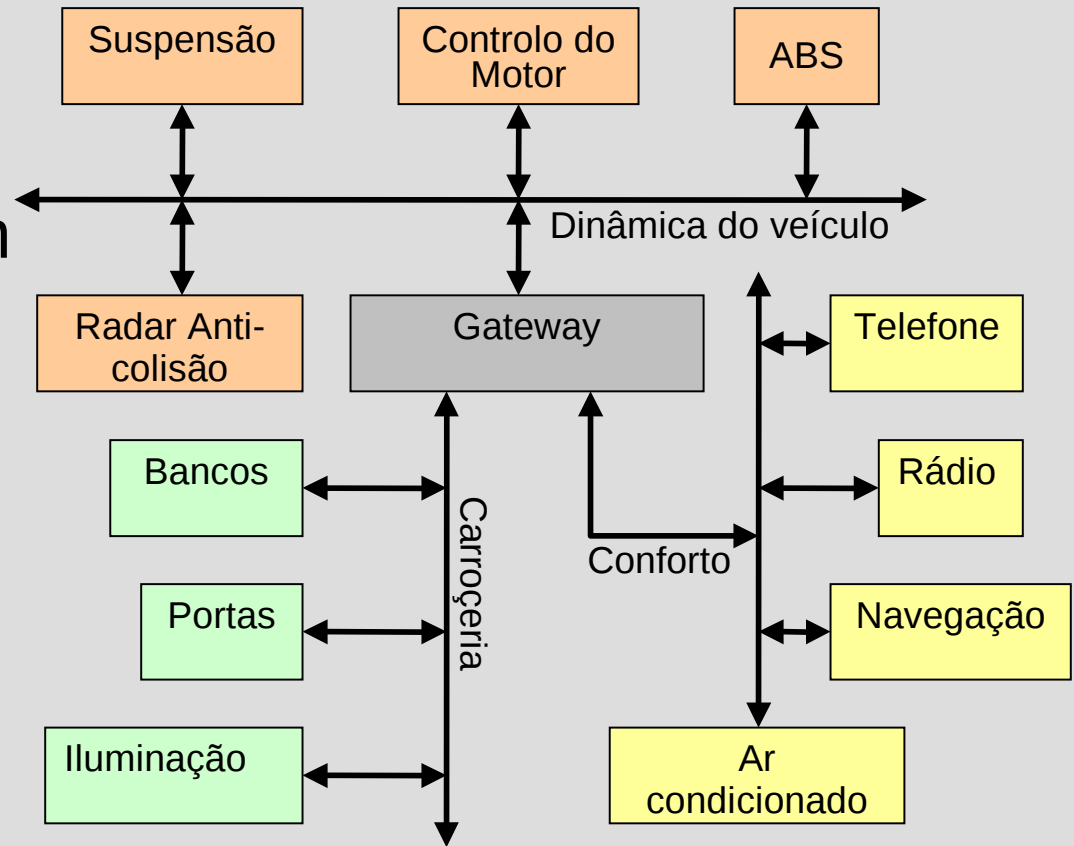
São possíveis **Arquitecturas Centralizadas** (uma única unidade de processamento) ou **Distribuídas** (múltiplas unidades de processamento)

- **Simplificação** e **redução de cablagens**
- **Dados** dos sensores podem ser **partilhados** (arq. dist.)
- **Processamento** pode ser **distribuído** por vários nós
 - E.g. os sensores podem efectuar localmente filtragem/ condicionamento de sinal; uma tarefa complexa pode ser separada em sub-tarefas alocadas a diferentes nós



Exemplo

Arquitectura “típica”
do sub-sistema de
comunicação de um
veículo automóvel
actual



Nota: é **frequente** a utilização de **diferentes protocolos** (e.g. CAN, LIN, FlexRay) no **mesmo veículo!**

CAN

CAN – Controller Area Network

- Protocolo criado pela Bosch, GmbH, para uso na **indústria automóvel**
- Standards ISO 11519 (94) e 11898 (95)
- **Uso generalizou-se**, estendendo-se a controlo de processos, automação, e aplicações embutidas em geral, ...
- Define apenas a **camada física** e de **ligação de dados**
- Diversas camadas de aplicação
 - CANOpen, DeviceNet, ...

CAN

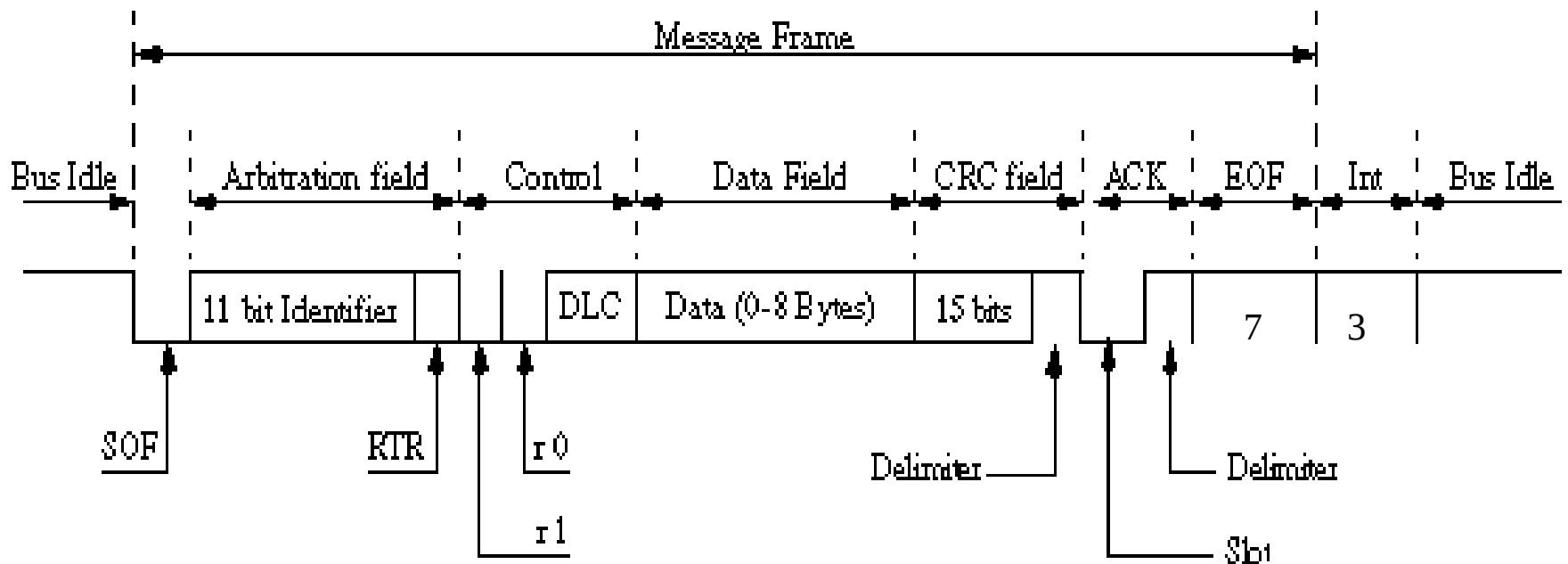
- Algumas **propriedades**
 - Barramento **série**, multi-master, *broadcast*
 - Taxas de transmissão entre 5 Kb/s e 1 Mb/s
 - **Tamanho** máximo do bus **depende da velocidade**
 - aprox. 40m @ 1Mbit/s, 1Km @ 50Kbit/s)
 - **Número** máximo de **nós** num segmento **depende dos transceivers** (32, 64, 128...)
 - **Detecção de erros** por CRC com 15bit
 - **Confinamento de erros**
 - Nós entram em “bus off” após N erros
 - **Recuperação de erros**
 - Retransmissão de mensagens

CAN

- Acesso **assíncrono** ao bus
- Trama pode conter entre **0 e 8 bytes de dados!**
- *Source-addressing*
 - **Identificador** associado à **mensagem** e não ao transmissor/receptor
 - Identificador com 11/29 bit (versão A/B resp.)
- Mecanismo de **arbitragem não destrutivo** e prioritizado, baseado nos identificadores
- **Resolução** de colisões bit-a-bit (*bit-wise*) **determinística**
 - CSMA/BA (NBA?,CA?,DCR?)

CAN

Formato da trama CAN



SOF: start of frame
RTR: Remote transmission request
r0,r1: reserved bits

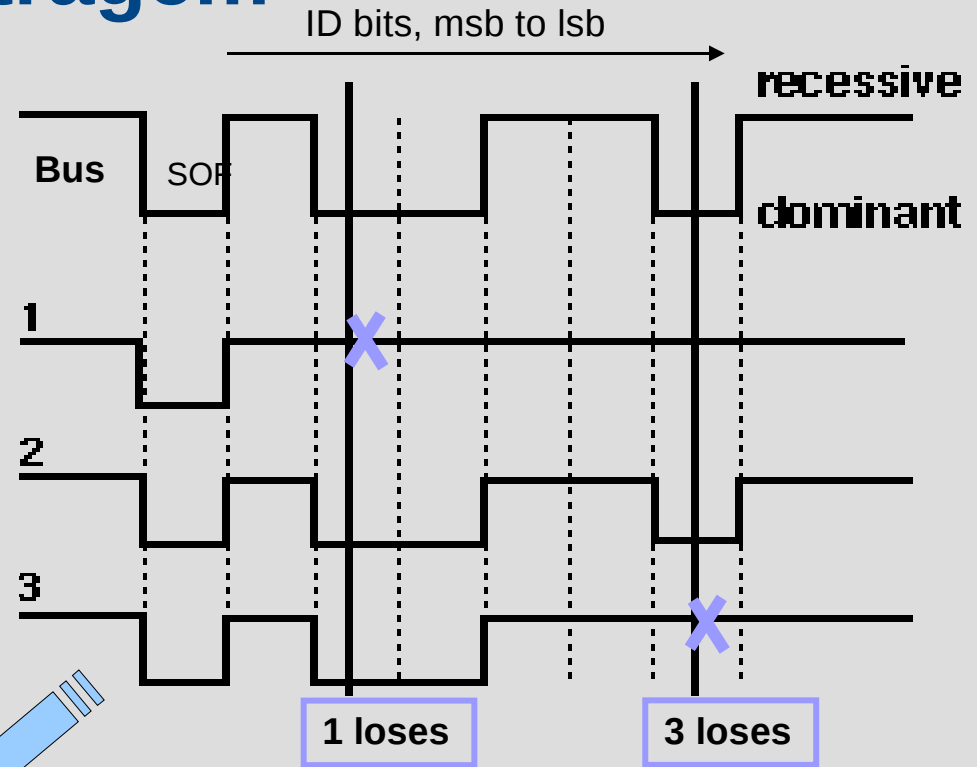
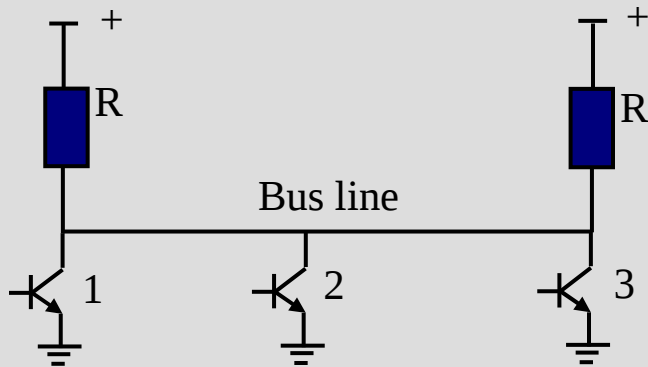
ACK: acknowledgement field
EOF: end of frame
Int: intermission gap

CAN

Mecanismo de **arbitragem**

Wired-AND implementado a nível dos drivers

- '0' – Nível dominante
- '1' – Nível recessivo



- Todos os nós que **transmitem observam** o bus em paralelo
- Se **transmitem** um bit **recessivo** e **lêem** um bit **dominante**
>>> **Arbitragem perdida, retiram-se do processo**
- **IDs** das mensagens têm de ser **diferentes!**



FlexRay

- Desenvolvido por um **consórcio de empresas**: “FlexRay consortium”, BMW, Ford, Bosch,...
- **Combinação** de conceitos da **TTA** (Kopetz) com **Byteflight** (protocolo desenvolvido pela BMW)
- Especificado em SDL
- Objectivos:
 - **Determinismo** temporal e **tolerância a erros**
 - **Largura de banda >> protocolo CAN**
 - » Inicialmente 10Mbit/sec, maiores velocidades possíveis;
 - Baseado em TDMA (Time Division Multiple Access)
 - » **Slots fixos** com **acesso exclusivo** ao bus
 - Estrutura cíclica baseada em **duas fases** (segmentos), uma **estática** e outra **dinâmica**

FlexRay: ciclo de comunicação

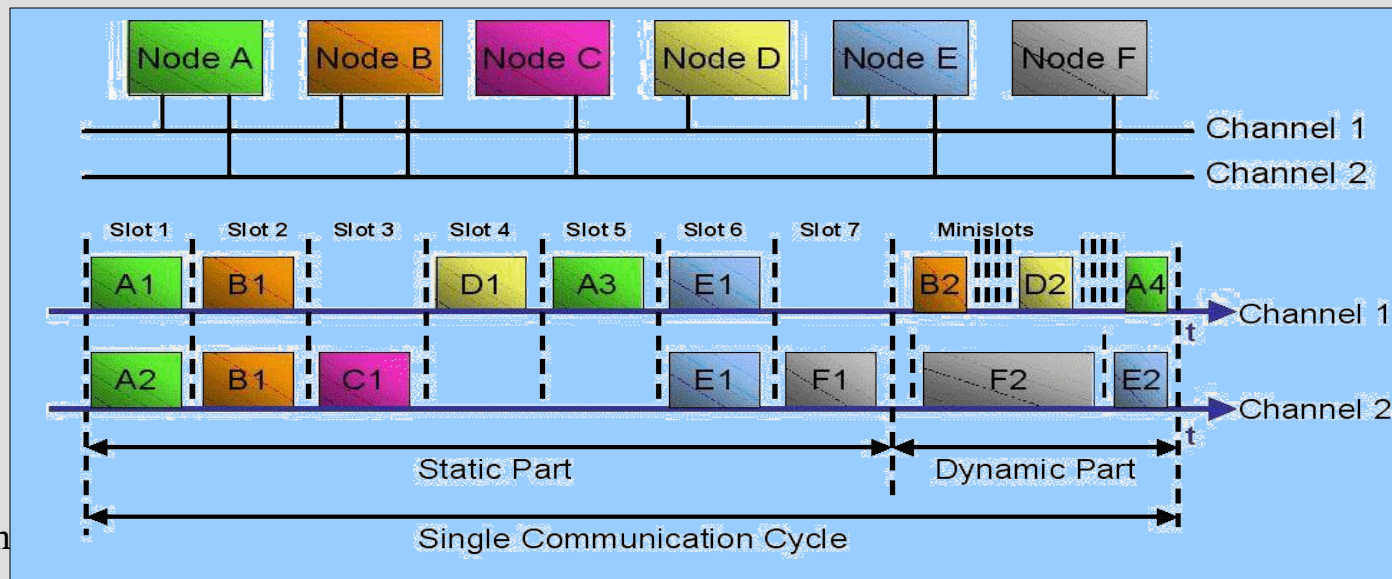


Fase estática:

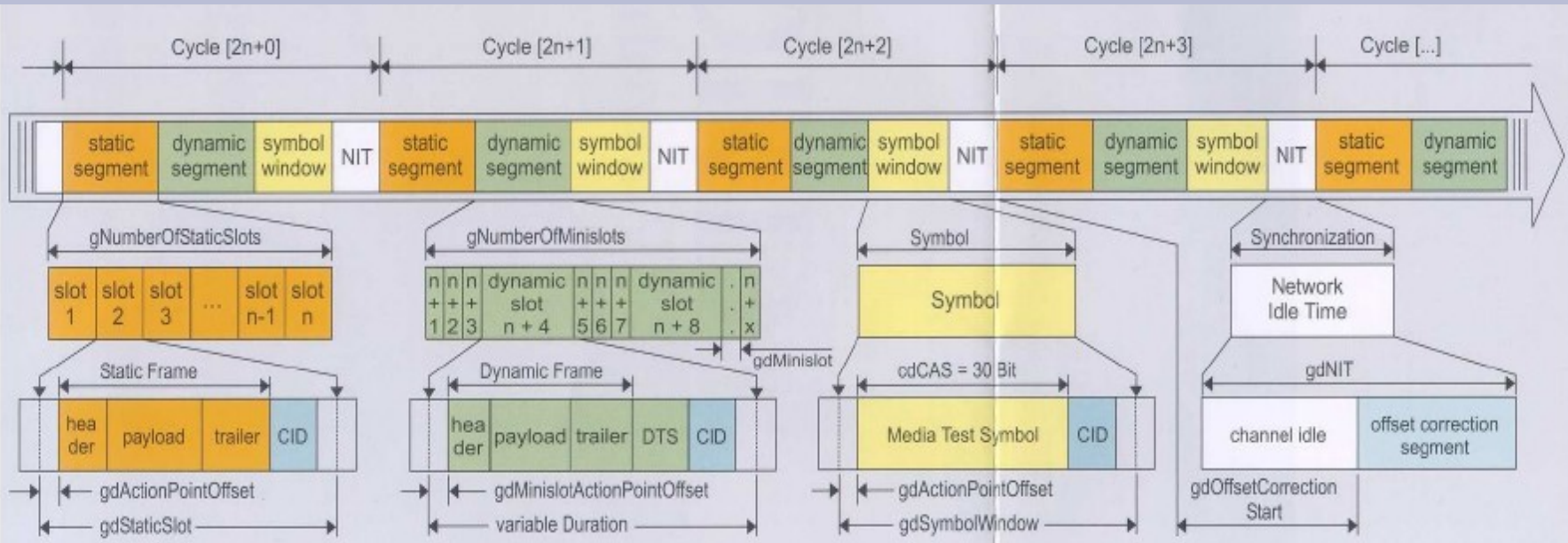
- **Acesso exclusivo** ao bus. **Escalonamento** definido em *pre-runtime*. **Slots de tamanho fixo**. **Redundância** (mesma mensagem pode ser opcional/ tx. em ambos os canais)

Parte dinâmica:

- **Segmentos** com **tamanho dinâmico**, **arbitragem** baseada em *mini-slotting*, **transmissão assíncrona** (transmissão apenas quando explicitamente requerido pela aplicação)



Intervalos temporais em Flexray



Quelle: Vector Informatik GmbH

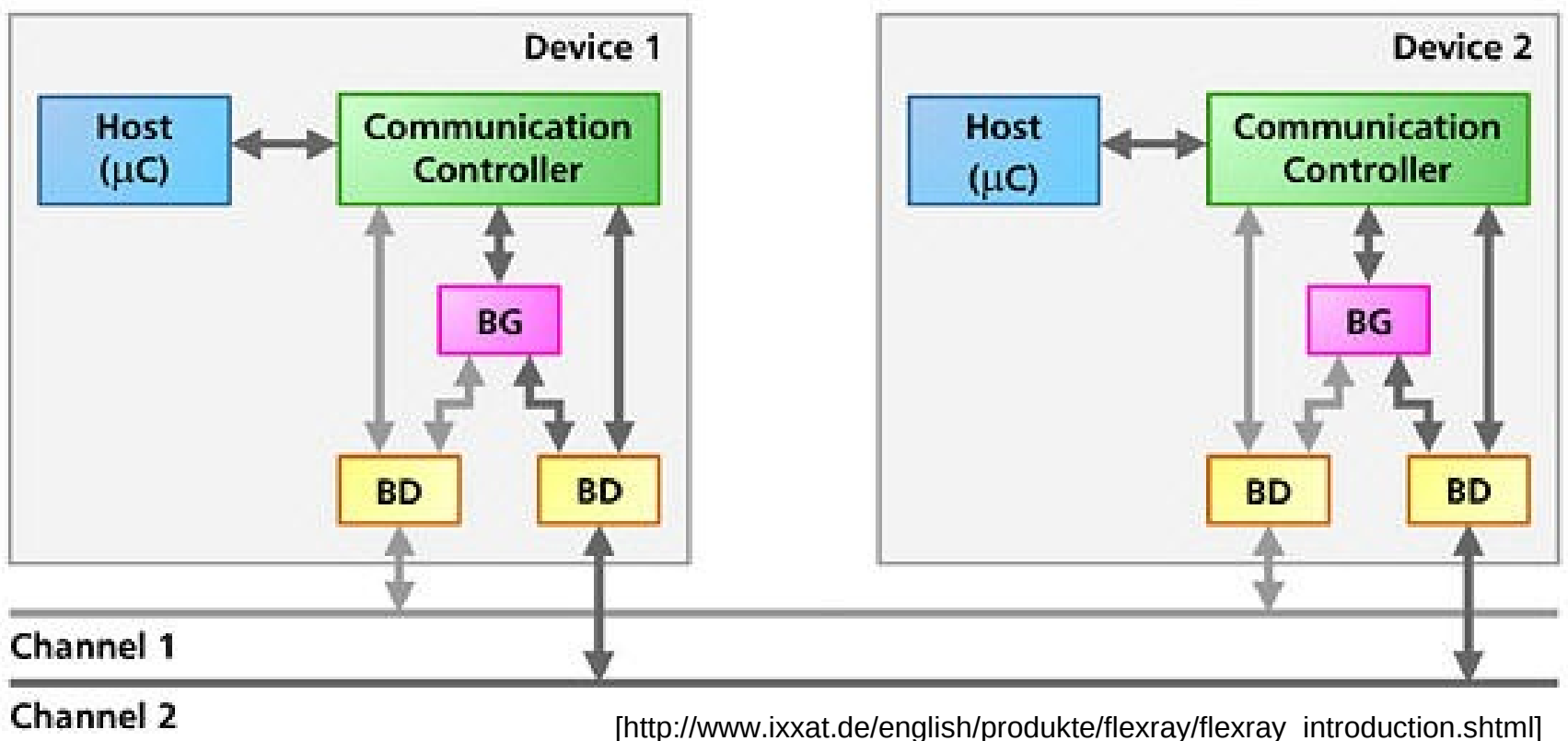
- **Microtick (μt)** = período de relógio local (pode variar entre nós do sistema)
- **Macrotick (mt)** = unidade básica de tempo, sincronizada entre os diversos nós do sistema ($=r_i \times \mu t$, r_i varia entre os nós)
- **Slot** = Intervalo alocado por transmissor no segmento estático ($=p \times mt$, p : fixo em runtime mas configurável)
- **Minislot** = Intervalo alocado por transmissão na fase dinâmica ($=q \times mt$, q : variável) *mini-slot* curto quando não há transmissão
- **Cycle** = Segmento estático + segmento dinâmico + tempo *idle*

Estrutura das redes Flexray



São utilizados “Bus guardians” que **protegem** o sistema de **falhas no domínio temporal**

– e.g. “babbling idiots”



Sumário

- Redes de comunicação para SE
 - Requisitos
 - Caracterização geral
 - Arquitecturas
 - Exemplos:
 - CAN
 - FlexRay