Aula 17

- O barramento CAN (*Controller Area Network*)
- Características fundamentais
- Aplicações
- Topologia da rede e codificação
- Tipos de tramas
- Deteção de erros
- Filtros de aceitação de mensagens
- Arbitragem

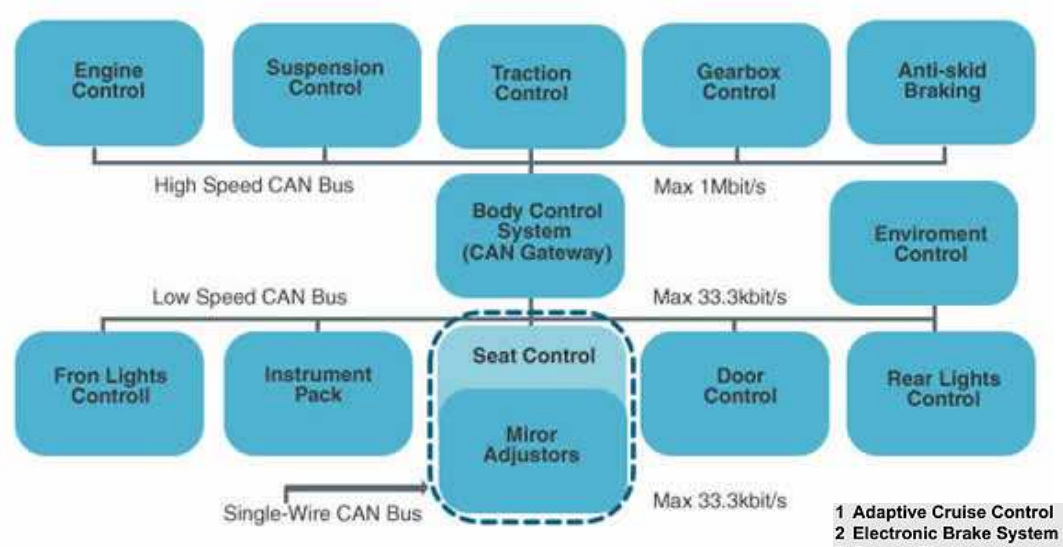
José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

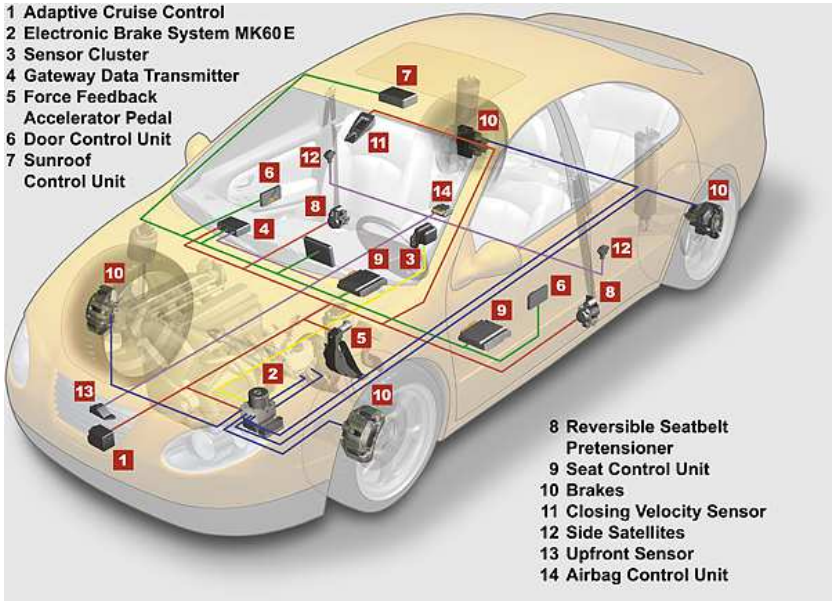
Introdução

- Desenvolvido em 1991 (versão 2.0) pela Bosch para simplificar as cablagens nos automóveis

  (http://esd.cs.ucr.edu/webres/can20.pdf)
- Utiliza comunicação diferencial em par entrançado
- Taxas de transmissão até 1 Mbit/s
- Adequado a aplicações de segurança crítica; elevada robustez
  - Tolerância a interferência eletromagnética
  - Capacidade de detetar diferentes tipos de erros
  - Baixa probabilidade de não deteção de um erro de transmissão (4.7×10^{-11})
- Atualmente usado num leque muito variado de aplicações
  - Comunicação entre subsistemas de um automóvel
  - Aviónica, Aplicações industriais, Domótica, Robótica
  - Equipamentos médicos, ...

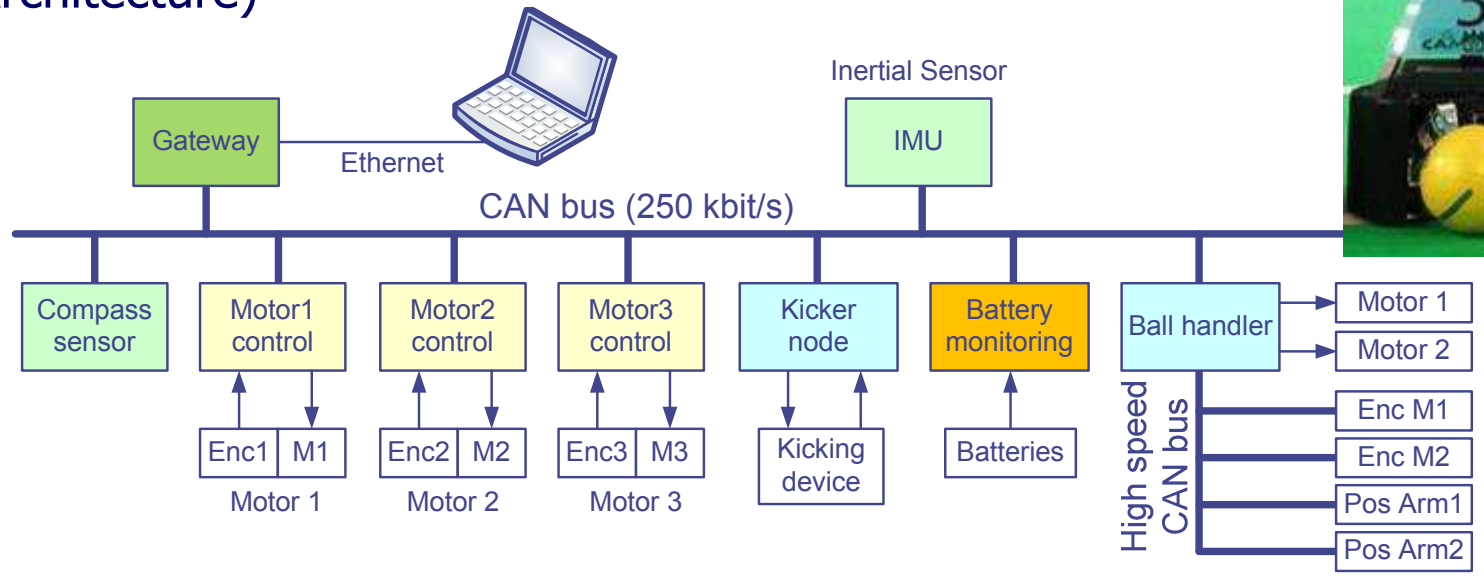
Exemplos de aplicação





Exemplos de aplicação

- Infraestrutura sensorial e de atuação dos robots da equipa de futebol robótico do DETI: CAMBADA (**C**ooperative **A**utonomous **M**obile ro**B**ots with **A**dvanced **D**istributed **A**rchitecture)



- Arquitetura distribuída em que cada nó desempenha uma tarefa ou conjunto de tarefas relacionadas
- O sistema é facilmente alterável; por exemplo, acrescentar um novo sensor não implica qualquer alteração na estrutura existente (basta ligar o novo nó ao barramento CAN)

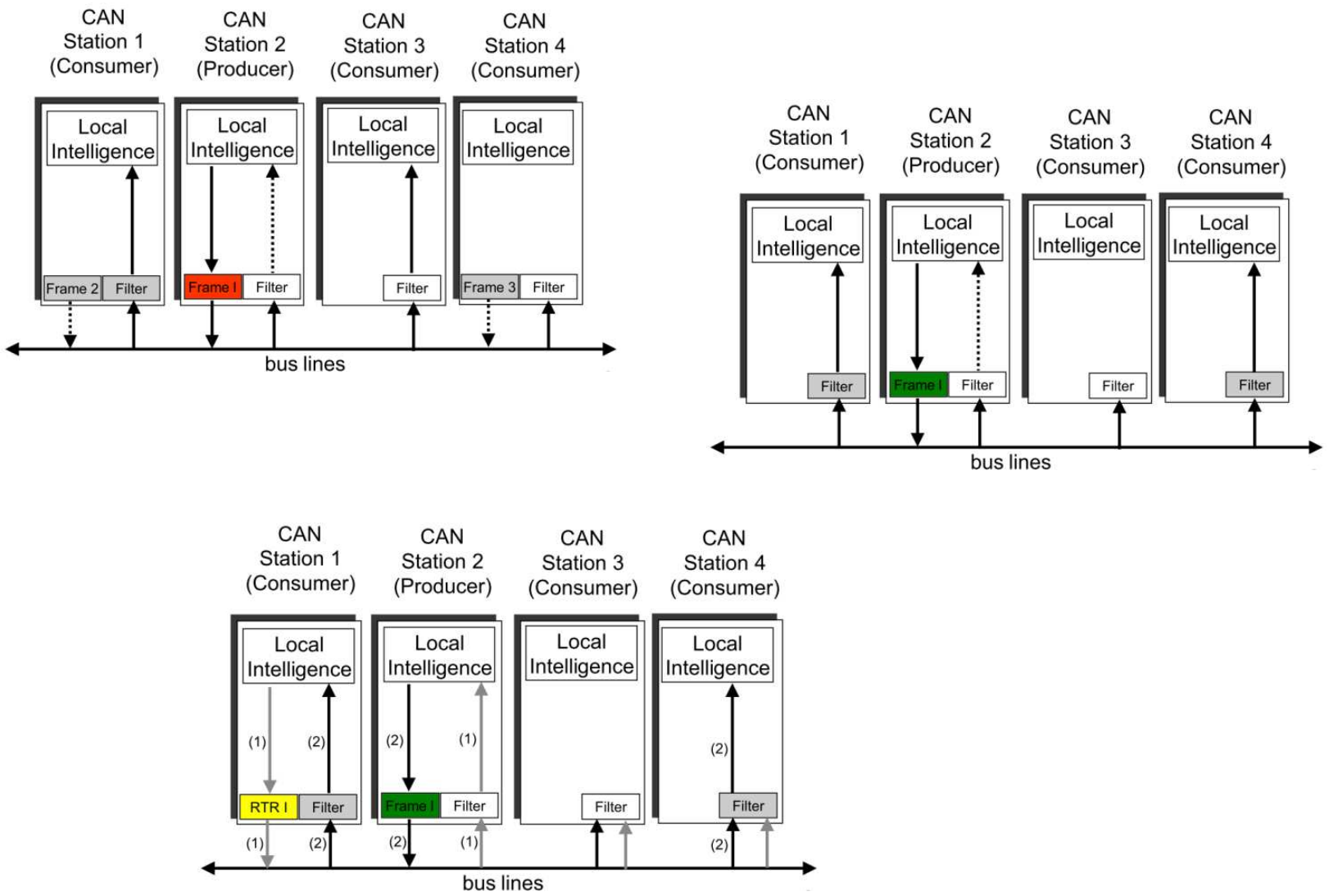
Características fundamentais

- Transmissão orientada ao bit: a informação é encapsulada em tramas estruturadas (mensagens), que incluem um identificador (ID), um campo de dados, bits de controlo e de verificação de erro
- As mensagens no CAN são difundidas para todos os nós na rede; cada nó pode decidir aceitar ou ignorar determinadas mensagens com base no seu identificador (ID)
- O barramento CAN permite comunicação bidirecional, mas apenas um nó pode transmitir de cada vez
- O CAN opera como um barramento "multi-master", i.e., qualquer nó do barramento pode iniciar uma transmissão
- Uma vez que dois ou mais nós podem querer aceder simultaneamente ao barramento para transmitir, o CAN implementa um mecanismo de arbitragem, que evita colisões e perda de informação
- Cada mensagem tem um ID único que identifica a natureza do seu conteúdo; esse ID determina também a prioridade da mensagem e, consequentemente, a prioridade no acesso ao barramento

Características fundamentais

- As mensagens não têm qualquer referência ao nó emissor ou recetor; os nós filtram as mensagens com base no identificador da trama
- Paradigma produtor-consumidor / Transmissão em "broadcast"
  - Os nós que geram mensagens (produtores) enviam-nas sem um destinatário específico
  - Todos os nós na rede CAN recebem essas mensagens ("Broadcast")
  - Apenas os nós interessados na mensagem (consumidores) a processam, ignorando as restantes
- Suporte para pedidos de transmissão remota (Remote Transmission Request): um nó pode pedir a transmissão de uma mensagem sem enviar dados
- Mecanismo avançado de deteção de erros, incluindo verificação de CRC e retransmissão automática de mensagens recebidas com erro
- **Relógio implícito**: comunicação assíncrona, sem transmissão explícita do relógio:
  - O transmissor e o recetor têm relógios locais independentes
  - A sincronização dos nós é feita através da deteção de transições na linha de dados

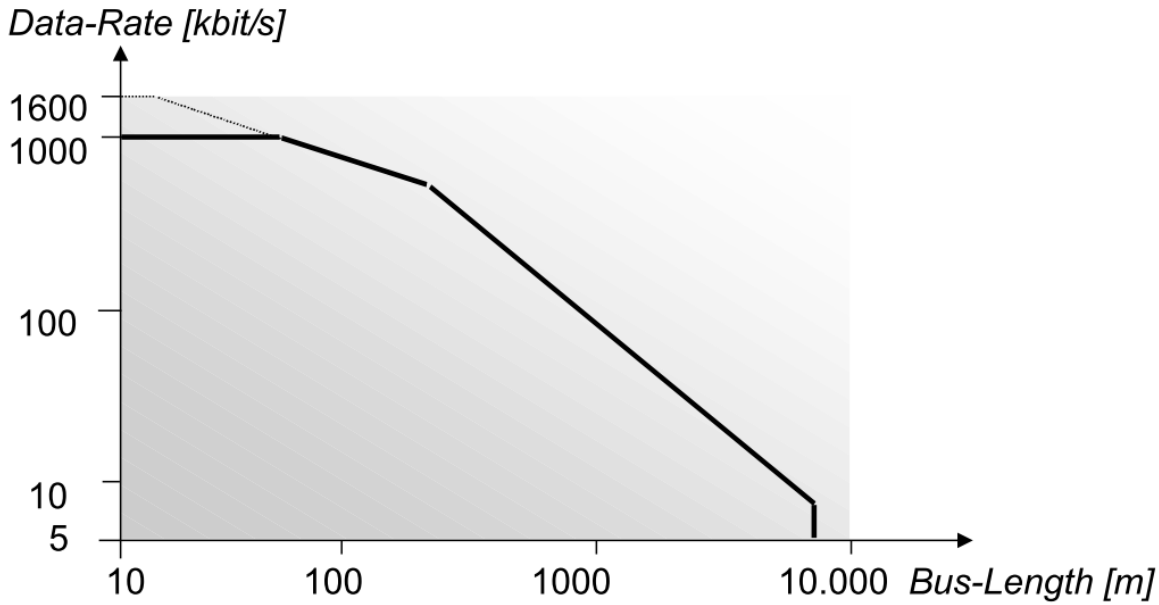
Características fundamentais



Comprimento máximo do barramento

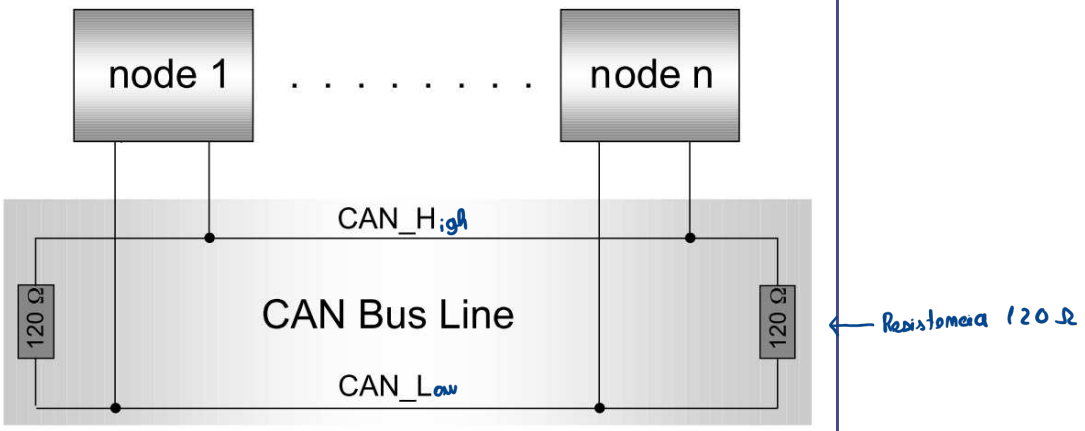
- O comprimento máximo do barramento (sem regeneração do sinal), depende da taxa de transmissão, do tipo de cabo usado e do ambiente elétrico (interferência eletromagnética)
- A tabela apresenta valores indicativos

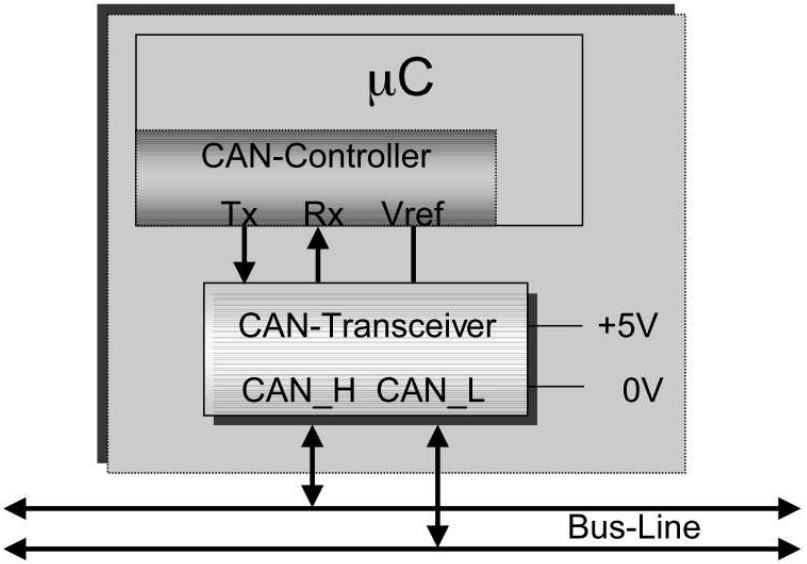
Bit Rate	Bus Length	Nominal Bit-Time
1 Mbit/s	30 m	1 µs
800 kbit/s	50 m	1,25 µs
500 kbit/s	100 m	2 µs
250 kbit/s	250 m	4 µs
125 kbit/s	500 m	8 µs
62,5 kbit/s	1000 m	20 µs
20 kbit/s	2500 m	50 µs
10 kbit/s	5000 m	100 µs



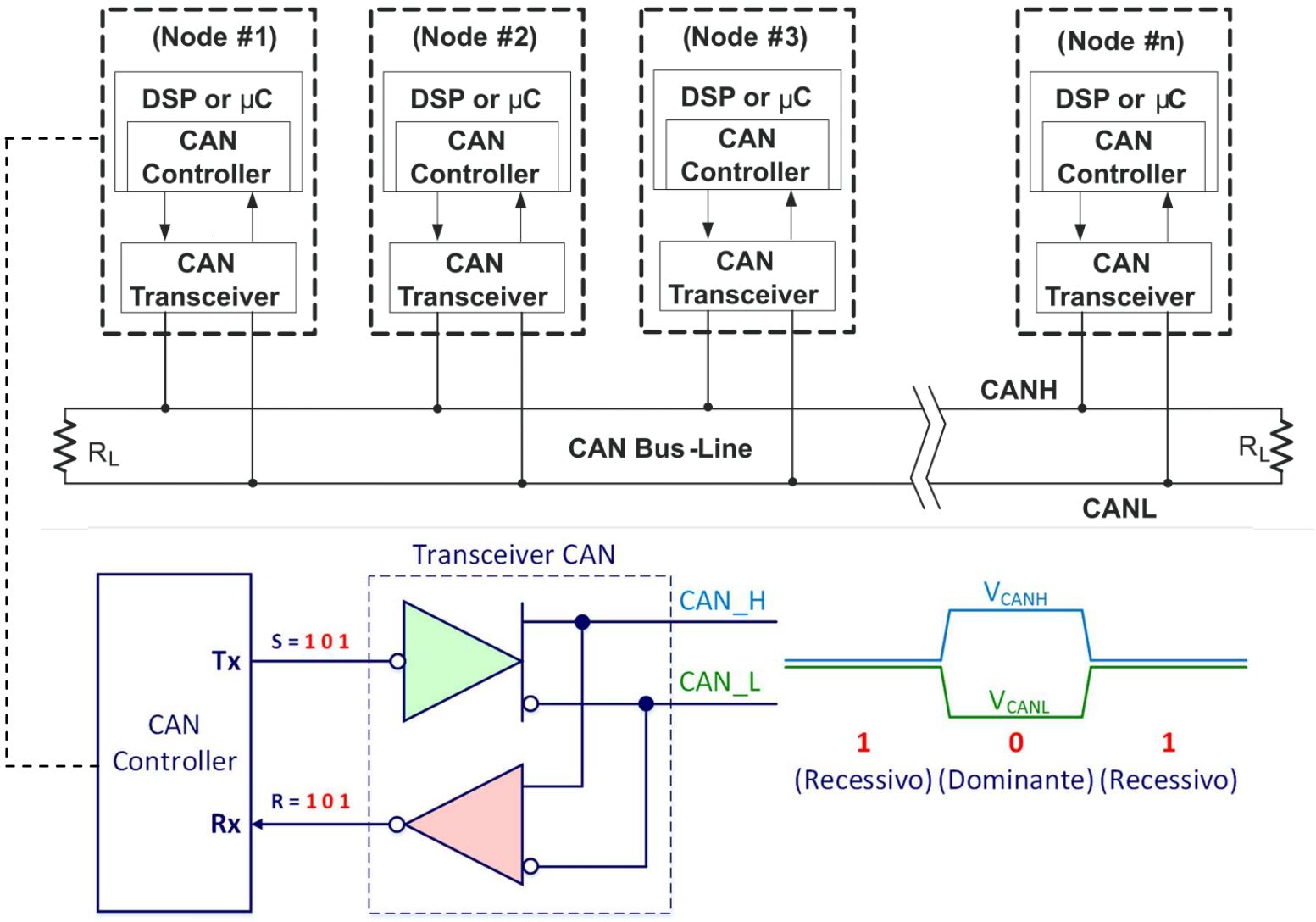
Topologia da rede e estrutura de um nó

- Comunicação **diferencial**, **par entrançado**
- Na transmissão, o "transceiver" transforma o nível lógico presente na linha Tx em duas tensões e coloca-as nas linhas **CAN_H** e **CAN_L**
- Na receção, o "transceiver" discrimina o nível lógico pela **diferença de tensão entre CAN_H e CAN_L** e o resultado é enviado através da linha Rx para o controlador CAN

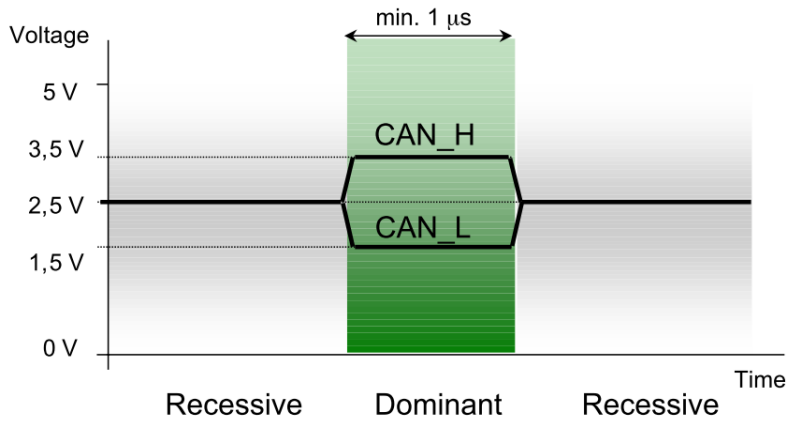


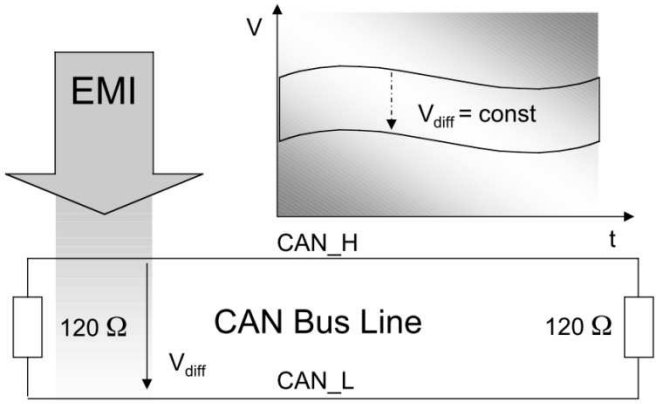


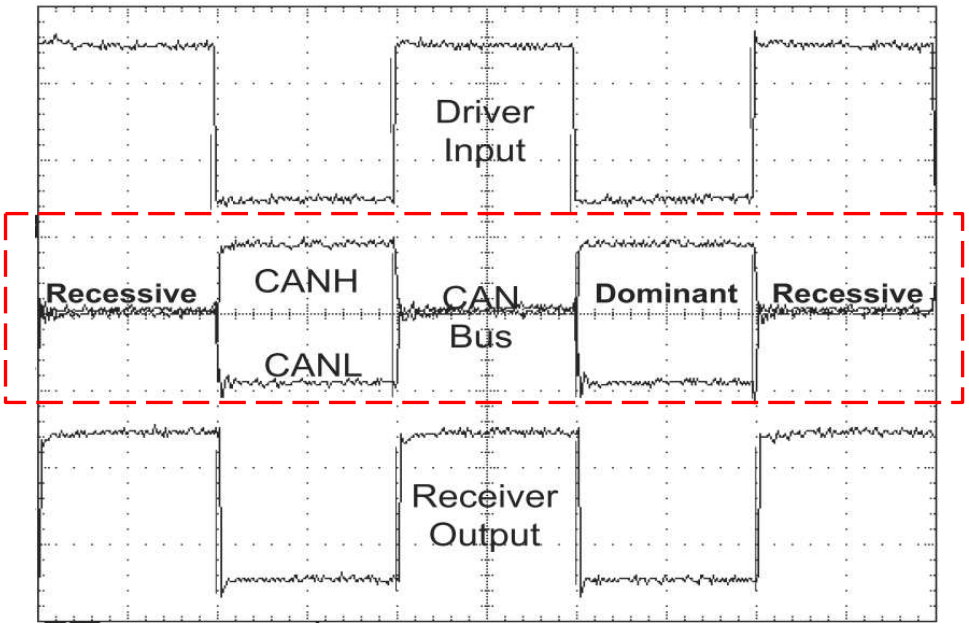
Topologia da rede e estrutura de um nó



Transmissão diferencial



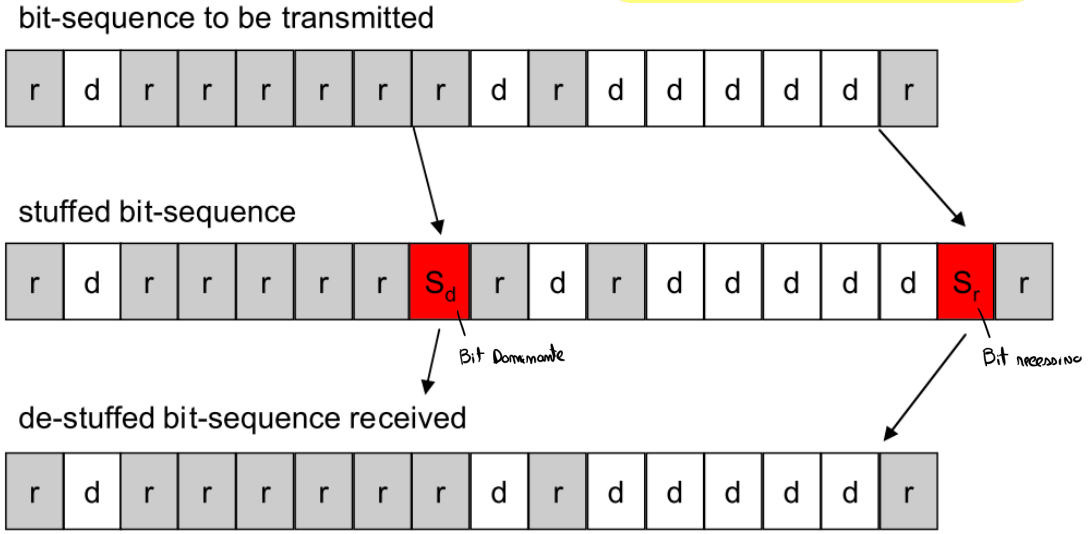




Codificação

- Codificação Non-Return-to-Zero - bit recessivo ('1')/dominante ('0')

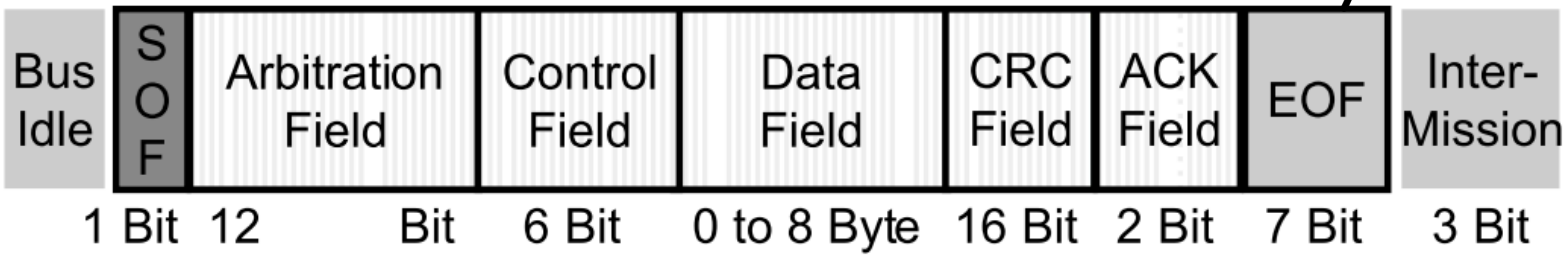
- **"Bit-stuffing"**
  - Por cada 5 bits iguais é inserido 1 bit de polaridade oposta



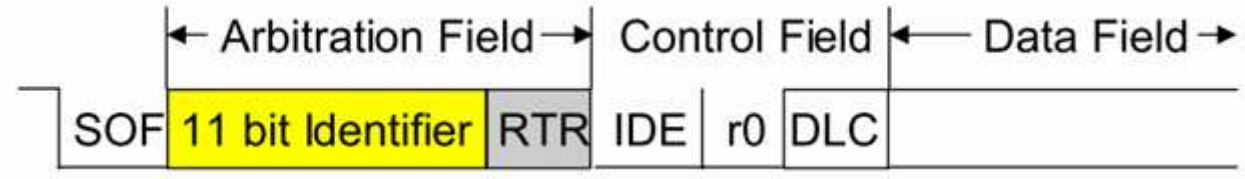
  - Garante um tempo máximo entre transições da linha de dados, assegurando que há transições suficientes para manter sincronizados os instantes de amostragem de dados em cada um dos nós

Formato da trama de dados (CAN 2.0A)

End of Frame

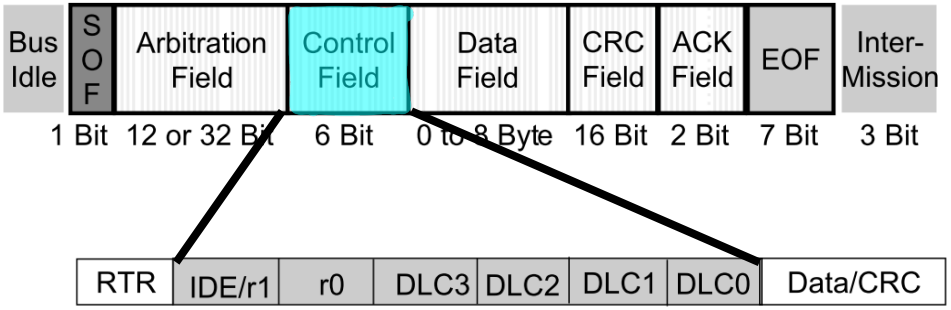


- **SOF** (Start of Frame)
  - Bit dominante ('0') indica o início da trama
  - Usado para sincronização dos instantes de amostragem dos nós recetores
- **Arbitration** – 12 bits (11+1)
  - **Identifier** (11 bits) – identificador da mensagem que também serve para arbitragem entre diferentes *masters* que podem iniciar a transmissão das suas tramas em simultâneo (id mais baixo, maior prioridade)
  - **RTR** (1 bit) Remote Transmission Request - dominante numa trama de dados
- Standard Frame Format (CAN 2.0A):



Formato da trama de dados (CAN 2.0A)

- **IDE** (identifier extension)
  - Bit dominante ('0') significa trama standard (CAN 2.0A, 11-bit identifier)
  - Bit recessivo ('1') significa trama CAN 2.0B (com identificador estendido de 29 bits)
- **r0** – reservado
- **DLC3 – DLC0**
  - Número de bytes de dados (0 a 8)
- **Data** (Campo de dados)
  - 0 a 8 bytes (0 a 64 bits)
  - MSBit first (/byte)



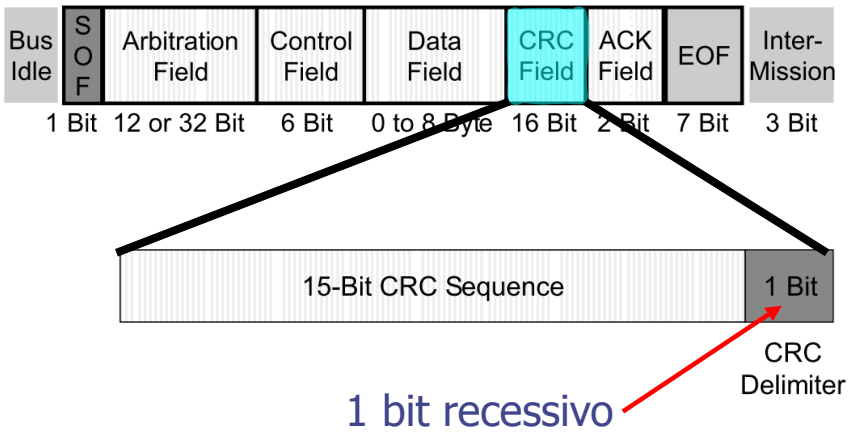
No. of Data Bytes	Data Length Code (DLC)			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d/r	d/r	d/r

min. length of Data Field = 0 Byte

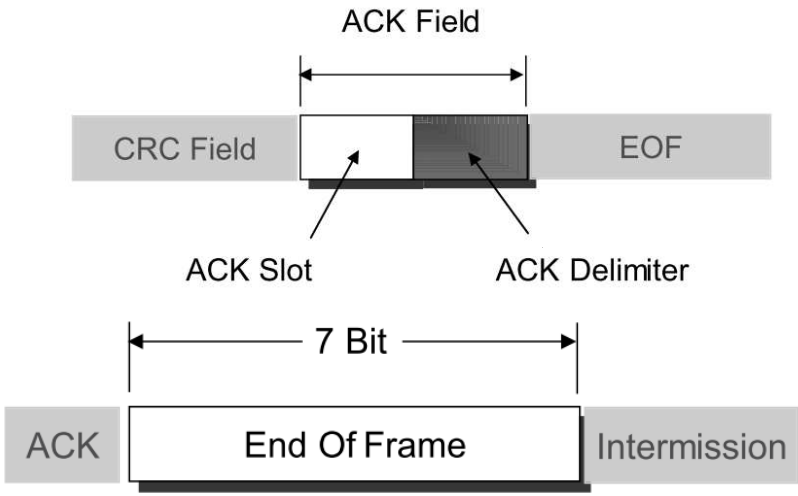
max. length of Data Field = 8 Byte

Formato da trama de dados (CAN 2.0A)

- **CRC** (Cyclic Redundancy Check)
  - Deteção de erros
  - Produtor e consumidor calculam a sequência de CRC com base nos bits transmitidos/recebidos
  - Produtor transmite a sequência CRC calculada
  - Consumidor compara a sequência CRC calculada localmente com a recebida do produtor
- **ACK** (Acknowledge)
  - Validação da trama (ACK Slot)
  - Recessivo (produtor)
  - Dominante (1+ consumidores)
- **EOF** (End of Frame)
  - Terminação da trama (7 bits recessivos)
- **IFS** (interframe/intermission)
  - Mínimo de 3 bits recessivos



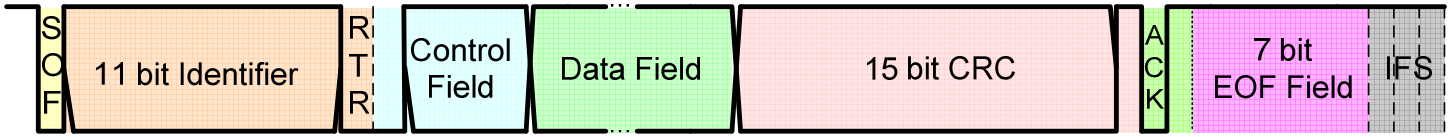
Remark: *The CRC Delimiter is a fixed formatted recessive bit.*



Tempo mínimo para poder haver outra mensagem

Tipos de tramas

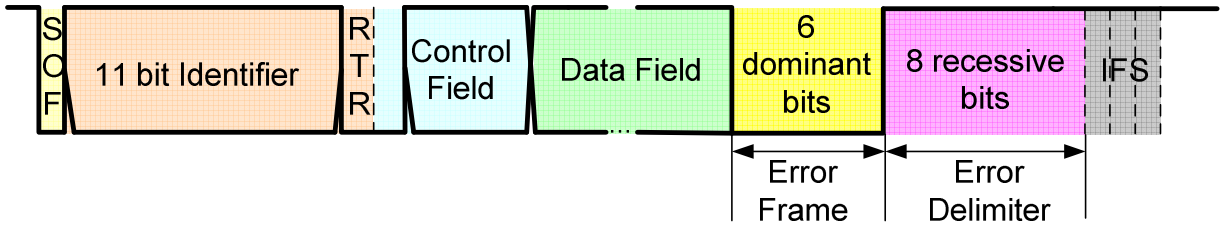
- **Data Frame**
  - Usada no envio de dados de um nó produtor para o(s) consumidor(es); numa trama de dados o bit RTR está a '0' (dominante)



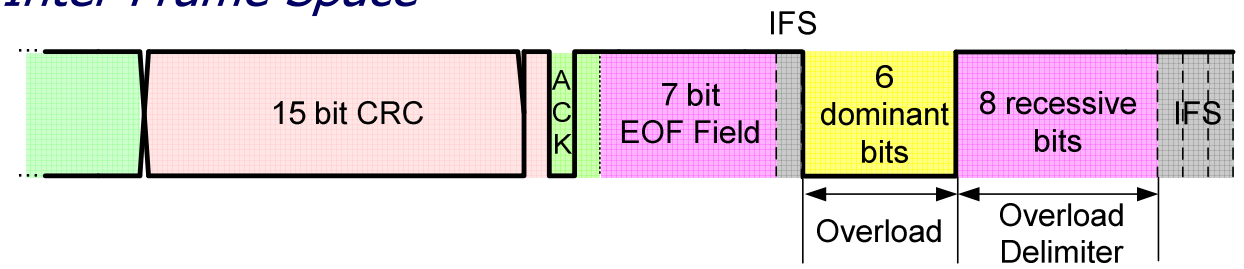
- **Remote Transmission Request Frame**
  - Enviada por um nó consumidor a solicitar (ao produtor) a transmissão de uma trama de dados específica (trama tem o campo RTR a '1' – recessivo, o que a diferencia de uma trama de dados)

Tipos de tramas

- **Error Frame**
  - Usada para reportar um erro detetado (a trama de erro sobrepõe-se a qualquer comunicação invalidando uma transmissão em curso



- **Overload Frame**
  - Usada para atrasar o envio da próxima trama (enviada por um nó em situação de sobrecarga que não teve tempo para processar a última trama enviada). Deve iniciar-se durante os dois primeiros bits do *Inter Frame Space*

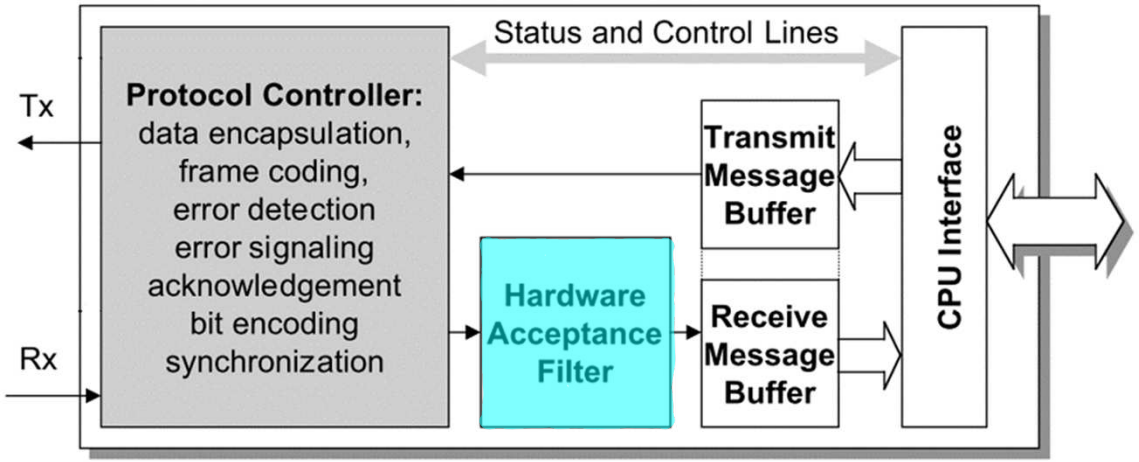


Deteção de erros de comunicação

S

- São usados vários métodos de deteção de erros. Se a receção de uma trama falha em qualquer um deles essa trama não é aceite e é gerada uma trama de erro que força o produtor a reenviar
- **CRC Error** – o CRC calculado não coincide com o CRC recebido
- **Acknowledge Error** – o produtor não recebe um bit dominante ('0') no campo ACK, o que significa que a mensagem não foi recebida por nenhum nó da rede (todos os nós fazem o "acknowledge" da receção da trama)
- **Form Error** – esta verificação analisa campos da mensagem que devem ter sempre o valor ´lógico '1' (recessivo): EOF, delimitador do ACK e delimitador do CRC; se for detetado um bit dominante em qualquer destes campos é gerado um erro
- **Bit Error** – cada bit transmitido é analisado pelo produtor da mensagem: se o produtor lê um valor que é o oposto do que escreveu gera um erro (exceções: identificador, ACK)
- **Stuffing Error** – se, após 5 bits consecutivos com o mesmo nível lógico não for recebido um de polaridade oposta, é gerado um erro

Arquitetura típica de um controlador CAN

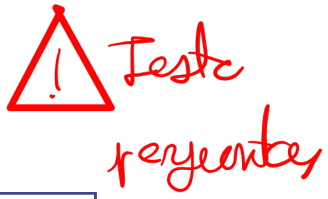


- O controlador CAN implementa o protocolo em hardware
- O "CPU interface" assegura, tipicamente, a comunicação com o CPU de um microcontrolador (registos de controlo, estado e dados – buffers)
- O "**hardware acceptance filter**" filtra as mensagens recebidas com base no seu ID; por programação é possível especificar quais os IDs das mensagens que serão copiadas para o "Receive Message Buffer" (i.e., que serão disponibilizadas ao microcontrolador)
- Este mecanismo de filtragem ao descartar mensagens não desejadas, reduz a carga computacional no microcontrolador

Filtros de aceitação de mensagens e máscaras

- O CAN é um barramento de tipo **"broadcast"**, ou seja, uma mensagem transmitida por um nó é recebida por todos os nós da rede
- O controlador CAN de cada nó lê todas as mensagens que circulam no barramento e coloca-as num buffer de receção temporário designado por "Message Assembly Buffer" (MAB)
- Logo que uma mensagem válida é recebida no MAB, é aplicado um **mecanismo de filtragem** que permite que apenas as mensagens de interesse para o nó sejam copiadas para o buffer de receção (as restantes são descartadas)
- A filtragem é feita por verificação dos bits do identificador da mensagem

Filtros de aceitação de mensagens e máscaras



- O mecanismo de filtragem é constituído por um conjunto de **filtros** e **máscaras:** na sua forma mais simples, a mensagem só é copiada para o buffer de receção se o identificador da mensagem igualar um dos filtros de aceitação (previamente configurados por software)
- As máscaras fornecem flexibilidade adicional ao permitir definir quais os bits do identificador que têm que ser iguais aos definidos nos filtros e quais os que são aceites incondicionalmente

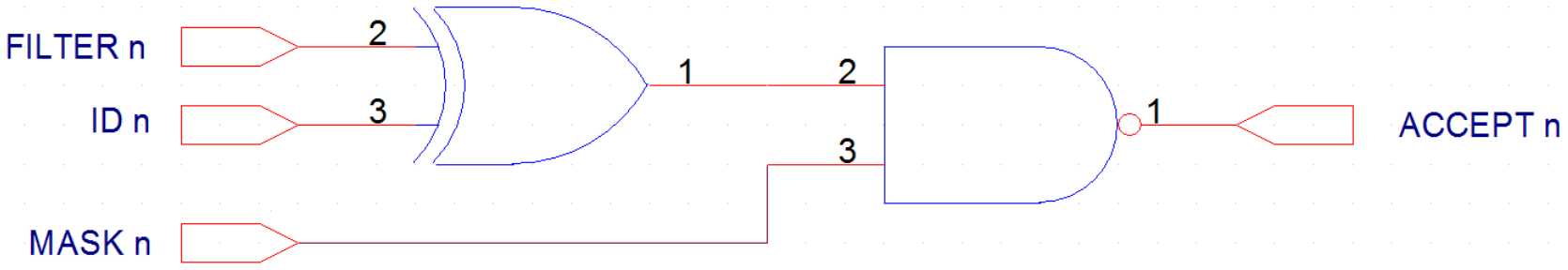
Mask bit n	Filter bit n	Message Identifier bit n	Accept/Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Dont care

Bit interessa

- $ACCEPT = ACCEPT_{10} \,.\, ACCEPT_9 \,.\, \ldots \,.\, ACCEPT_0$
- Se ACCEPT=1, a mensagem é copiada para o buffer de receção

Filtros de aceitação de mensagens e máscaras



- ACCEPT = $ACCEPT_{10}$. $ACCEPT_9$. … . $ACCEPT_0$
- Se ACCEPT=1, a mensagem é copiada para o buffer de receção

- Exemplos (ID de 11 bits):
  - Máscara com o valor 0x000: todas as mensagens são aceites
  - Máscara com o valor 0x7FF, filtro com o valor 0x1F4: apenas a mensagem com o ID 0x1F4 é aceite
  - Máscara com o valor 0x7FC, filtro com o valor 0x230: são aceites as mensagens com os Ids 0x230, 0x231, 0x232 e 0x233

Máscara a Ø, todas as mensagem são aceites (dont care)

1. Máscara com o valor 0x7FF, filtro com o valor 0x1F4: apenas a mensagem com o ID 0x1F4 é aceite

Mascara 0x 7FF 0b 111 1111 1111

Filtro 0x 1F4 0b 001 1111 0100 ← A msg tem de ter exata com o filtro

ID = 0x 1F4

2. Máscara com o valor 0x7FC, filtro com o valor 0x230: são aceites as mensagens com os Ids 0x230, 0x231, 0x232 e 0x233

Bit's Dont Care

Mascara 0x 7FC ⇒ 0b 111 1111 1100

Filtro 0x 230 ⇒ 0b 010 0011 0000

00 } 0x 230
01 } 0x 231
10 } 0x 232
11 } 0x 233

3.

Mascara 0x 7FF 0b 111 1110 1111

1 9 4

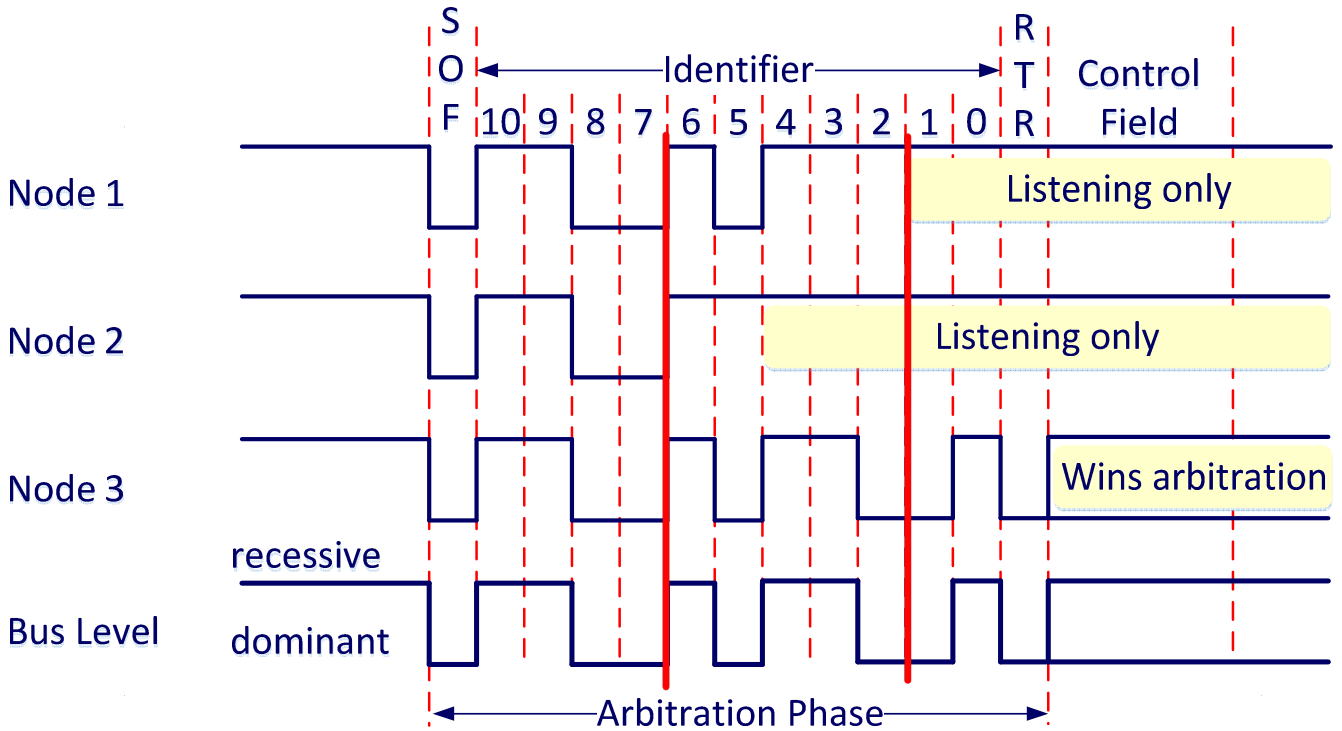
Filtro 0x 1F4 0b 001 1001 0100

001 1000 0100 — 0x 184

001 1001 0100 — 0x 194

Controlo de acesso ao meio – Arbitragem

- Realizada durante os campos ID e RTR das tramas (*arbitration field*)
- Baseada em bit recessivo / bit dominante



- O nó produtor da mensagem com o identificador de menor valor binário ganha o processo de arbitragem e transmite os seus dados (um identificador com todos os bits a '0' tem a mais alta prioridade)