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, Arnaldo Oliveira, Tomás Silva, Bernardo Cunha

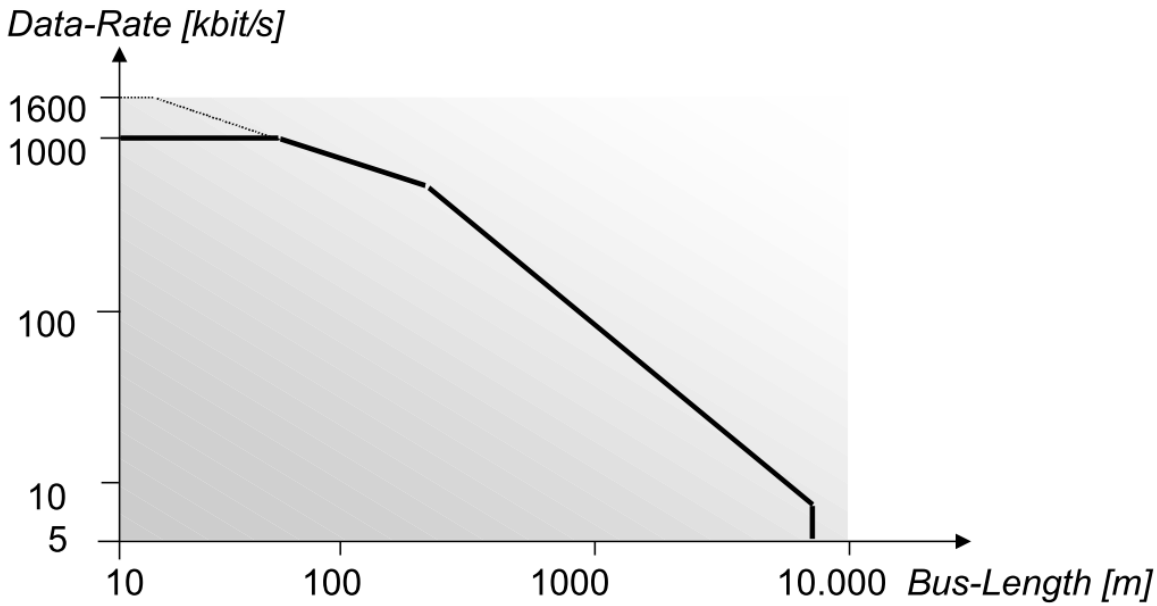
Introdução

- Desenvolvido em 1991 (versão 2.0) pela Bosch para simplificar as cablagens nos automóveis
- Utiliza comunicação diferencial em par entrançado
- Taxas de transmissão até 1 Mbit/s
- Número máximo de nós no barramento: 40
- Adequado a aplicações de segurança crítica; elevada robustez
  - Capacidade de detetar diferentes tipos de erros
  - Tolerância a interferência eletromagnética
  - 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
  - Aplicações industriais, Domótica, Robótica
  - Equipamentos médicos, ...

Introdução

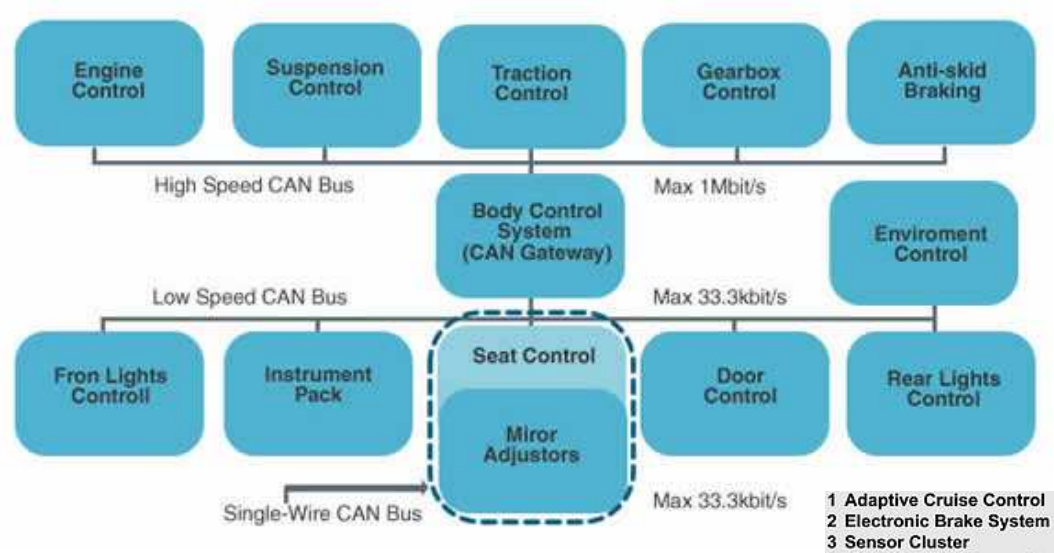
- O CAN é um barramento "multi-master": qualquer nó do barramento pode produzir informação e iniciar uma transmissão
- Uma vez que dois nós podem querer aceder simultaneamente ao barramento para transmitir, tem que haver uma forma de arbitrar o acesso
- Comunicação bidirecional "half-duplex"
- A informação produzida é encapsulada em tramas
- Transmissão em "broadcast": um transmissor pode enviar informação para todos os nós ao mesmo tempo
- No CAN cada mensagem tem um ID único; esse ID determina a prioridade da mensagem e, consequentemente, a prioridade no acesso ao barramento

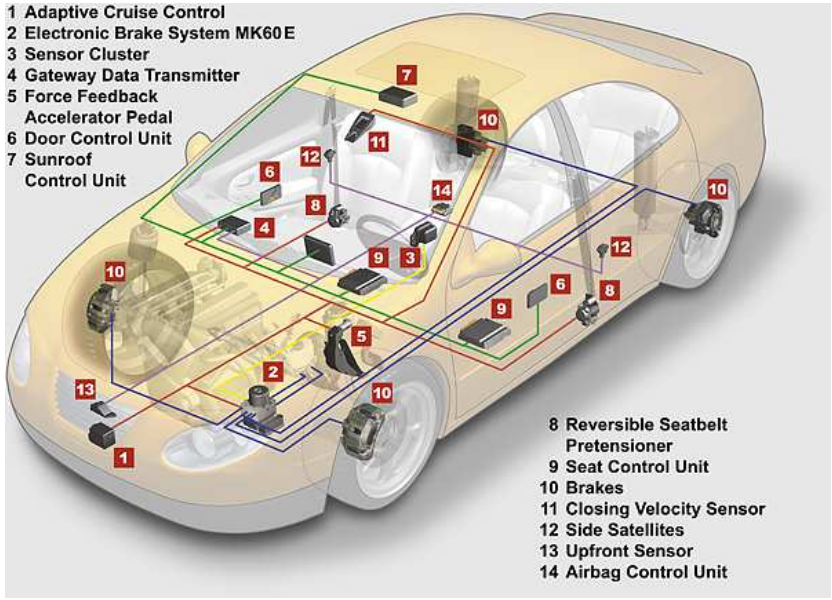
Comprimento máximo do barramento



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

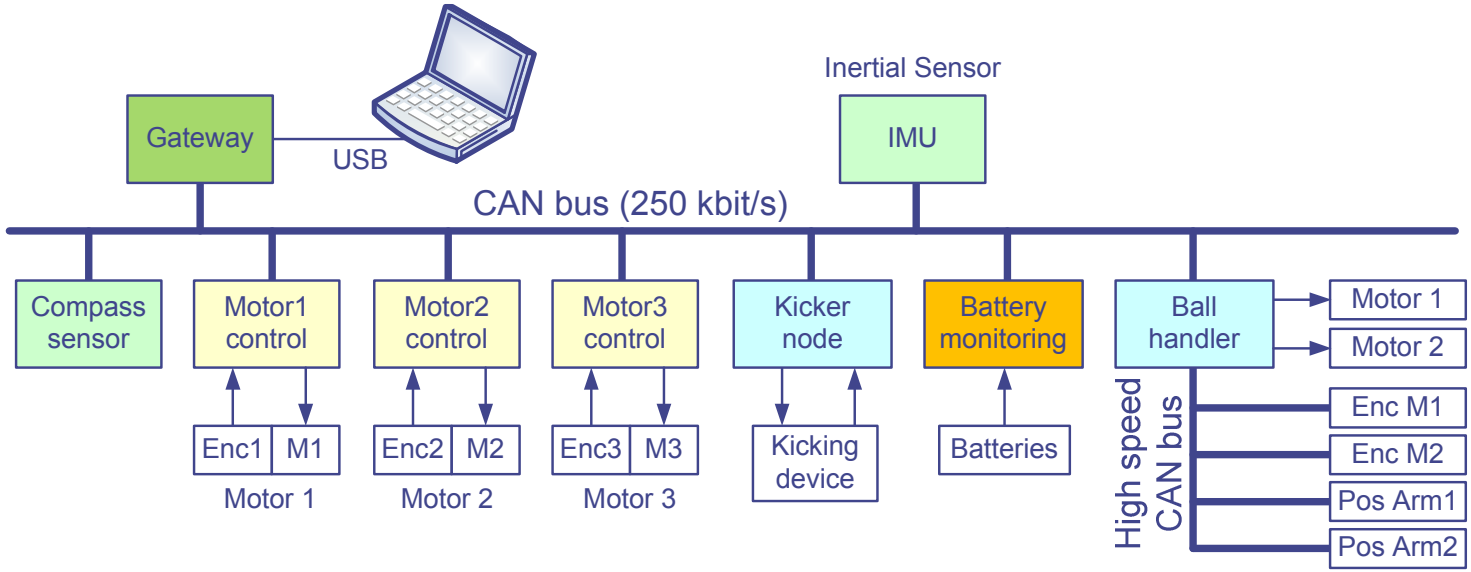
Exemplos de aplicação





Exemplos de aplicação

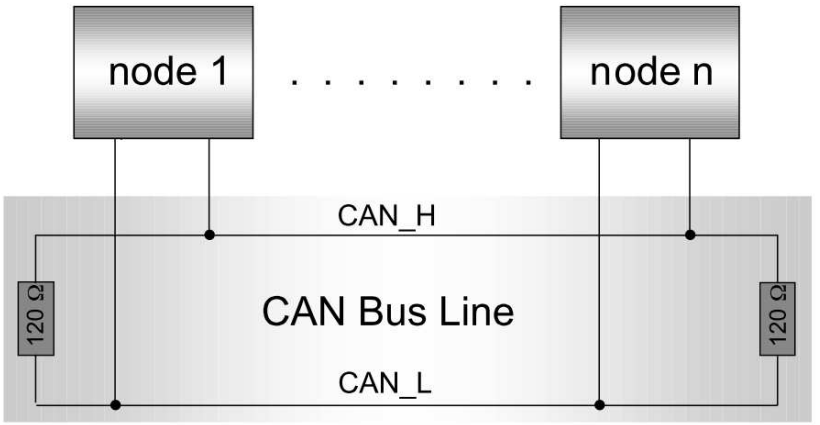
- Infraestrutura sensorial e de atuação dos robots da equipa de futebol robótico do DETI: CAMBADA (**C**ooperative **A**utonomous **M**obile ro**B**ots with **A**dvanced **D**istributed **A**rchitecture)

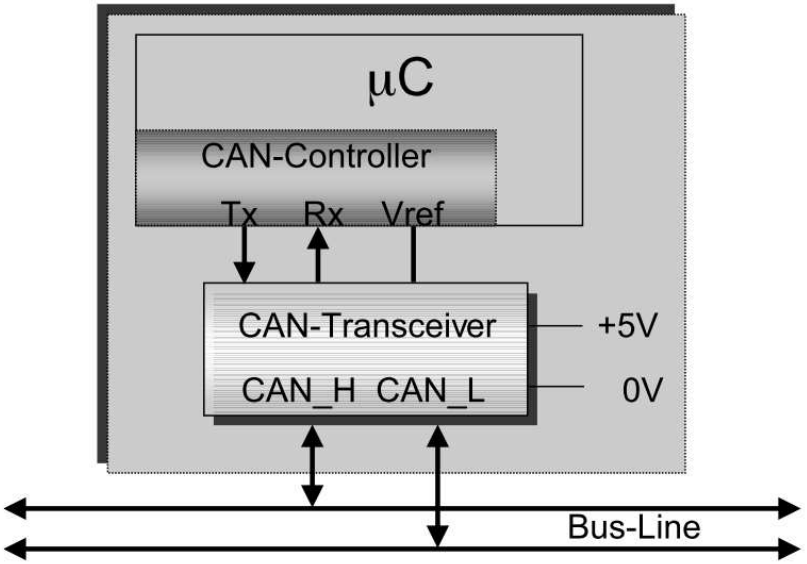


- Arquitetura distribuída em que cada nó desempenha uma tarefa ou conjunto de tarefas relacionadas
- O sistema é facilmente alterável; por exemplo, acrescentar um novo sensor não implica qualquer alteração na estrutura existente (basta ligar o novo nó ao barramento CAN)

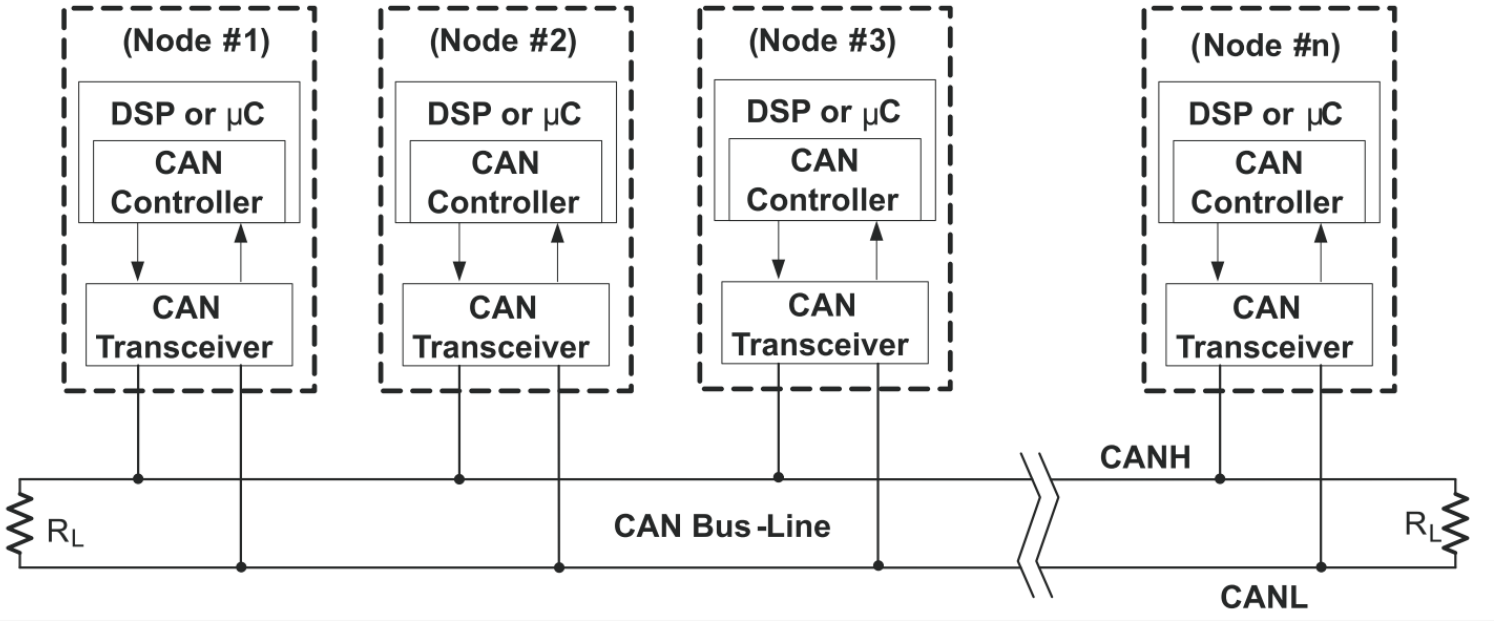
Topologia da rede e estrutura de um nó

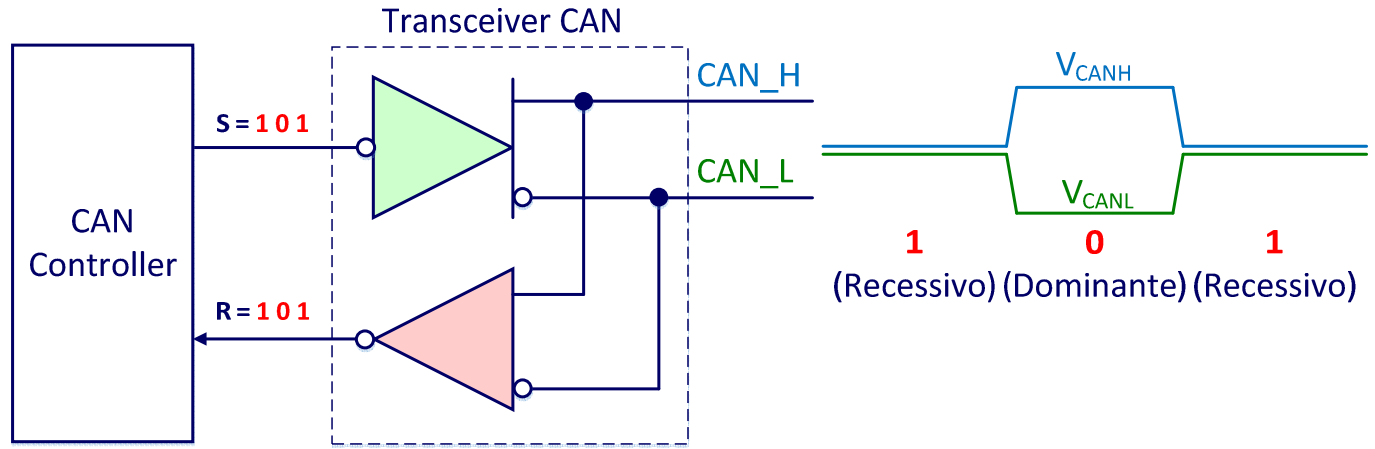
- Comunicação **diferencial**, **par entrançado**
- Na transmissão, o "transceiver" transforma o nível lógico presente na linha Tx em duas tensões e coloca-as nas linhas **CAN_H** e **CAN_L**
- Na receção, o "transceiver" discrimina o nível lógico pela **diferença de tensão entre CAN_H e CAN_L** e o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





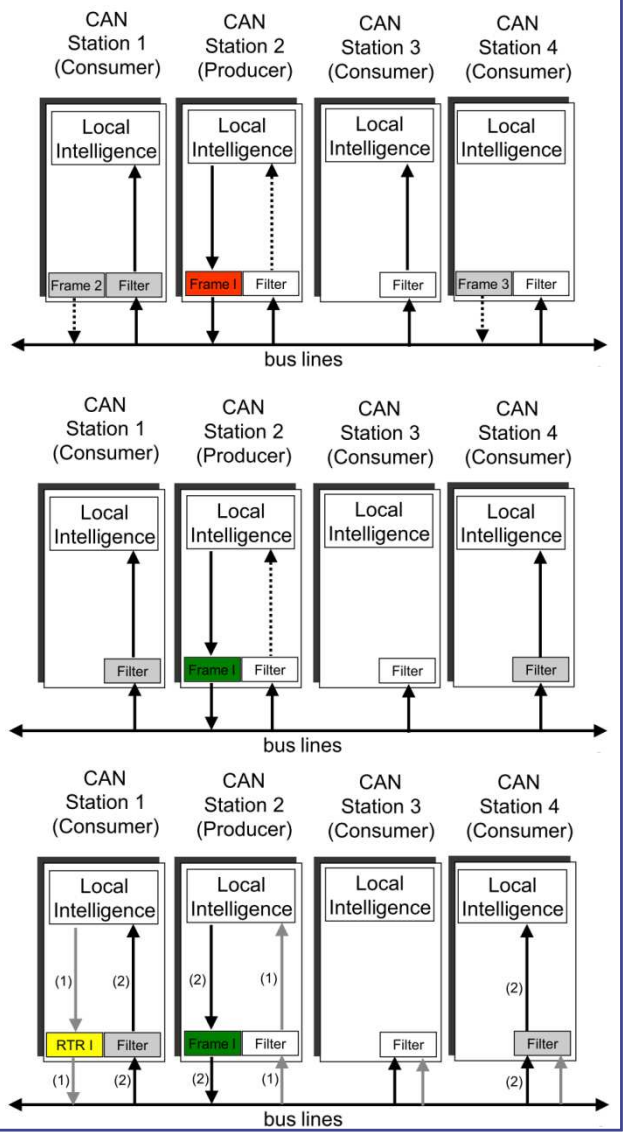
Topologia da rede e estrutura de um nó





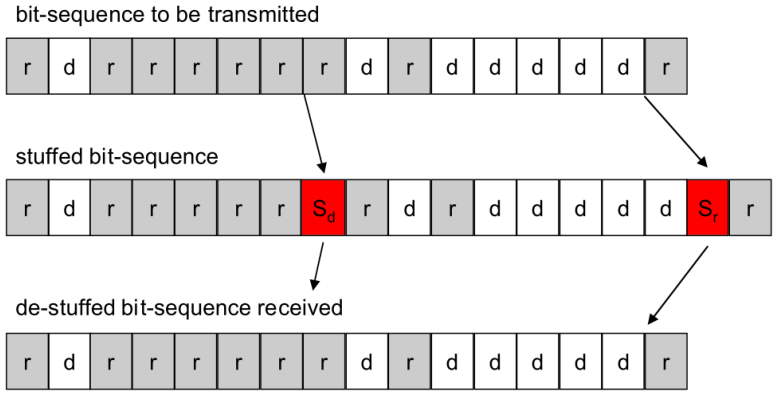
Características fundamentais

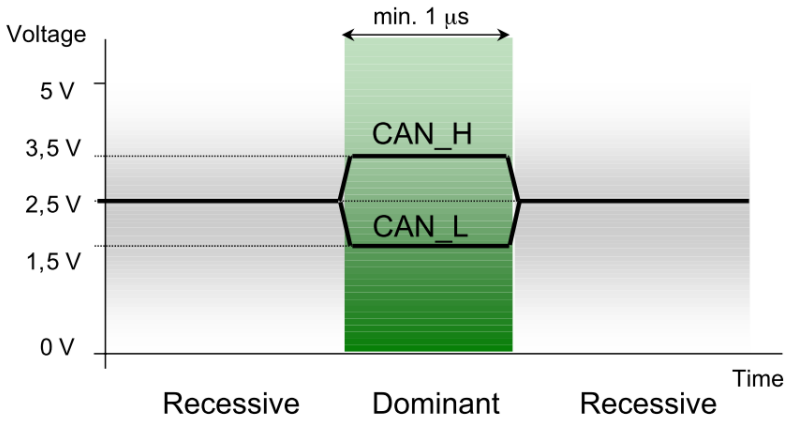
- Sincronização de relógio:
  - **Relógio implícito** (comunicação assíncrona, i.e. não há transmissão do relógio - o transmissor e o recetor têm relógios locais independentes)
- Transmissão orientada ao bit
- Barramento série "multi-master"
  - Diversos nós trocam mensagens encapsuladas em tramas
- Paradigma produtor-consumidor / Transmissão em "broadcast"
  - Identificação do conteúdo da mensagem (não existe identificação do nó de origem ou de destino)
- Capacidade de Remote Transmission Request
- Correção de erros baseada em retransmissão

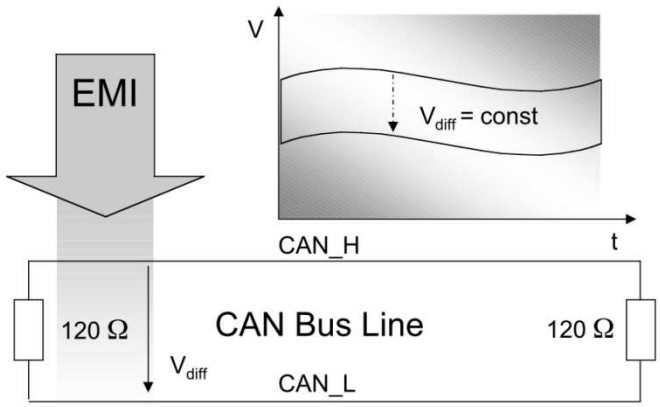


Codificação

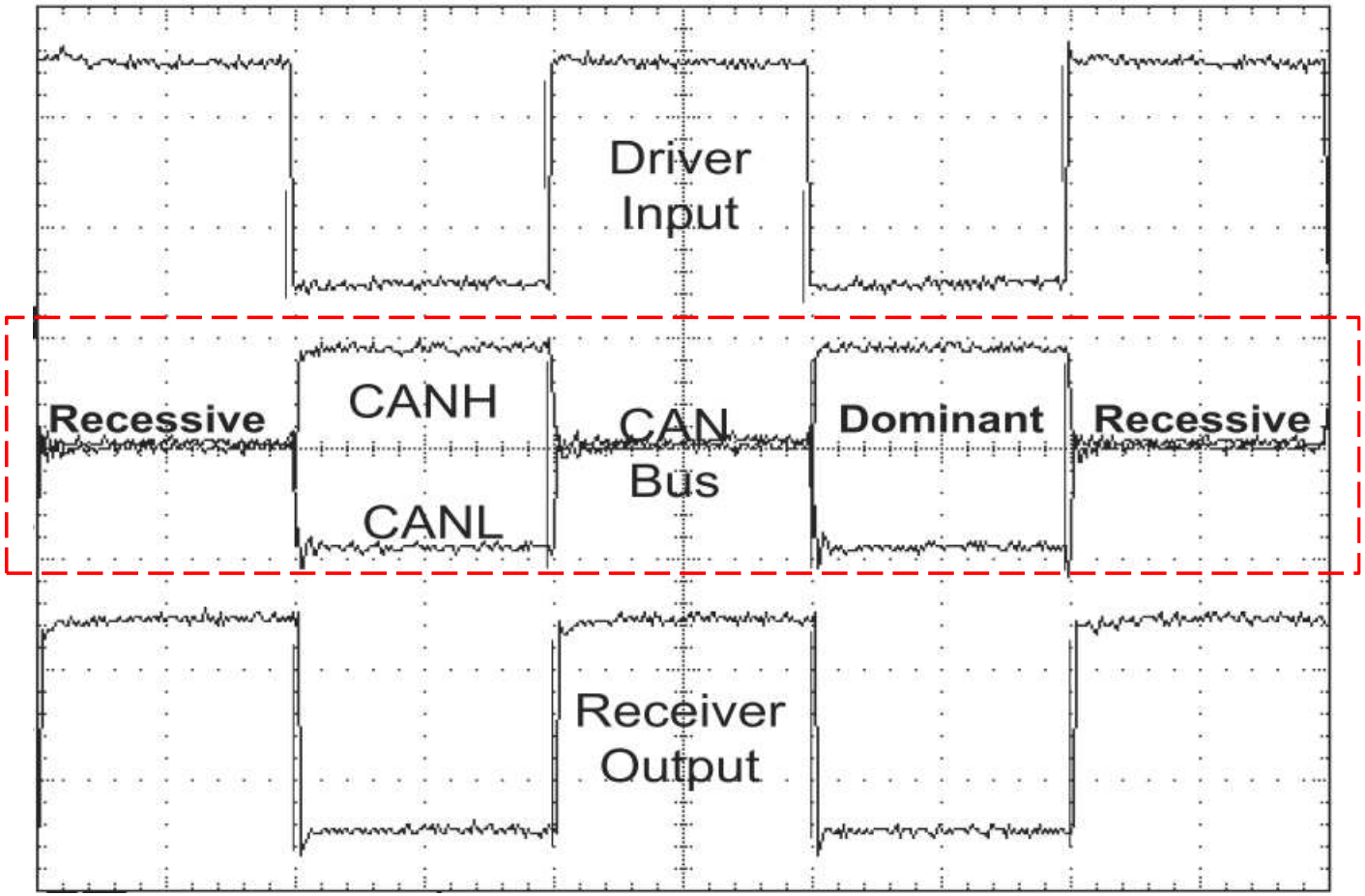
- Codificação Non-Return-to-Zero
  - Bit recessivo ('1')/dominante ('0')
- "**Bit-stuffing**"
  - Por cada 5 bits iguais é inserido 1 de polaridade oposta
  - Garante tempo máximo entre transições da linha de dados, assegurando que há transições suficientes para manter os relógios dos nós sincronizados
- Transmissão diferencial



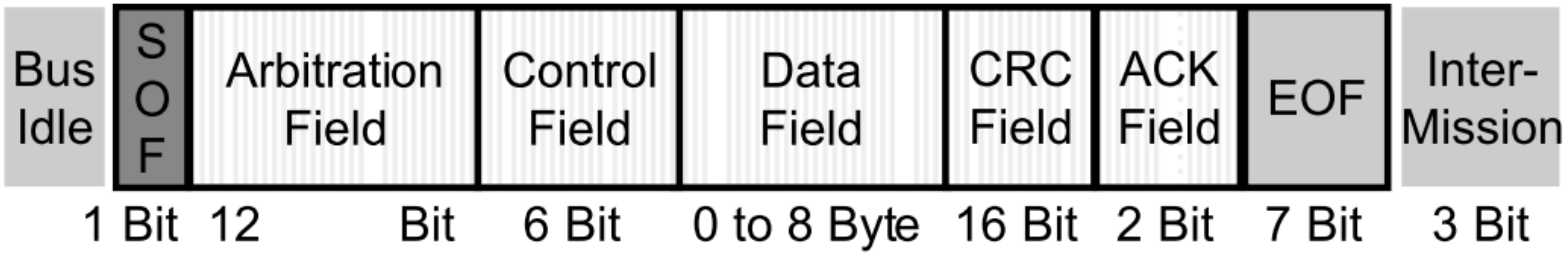




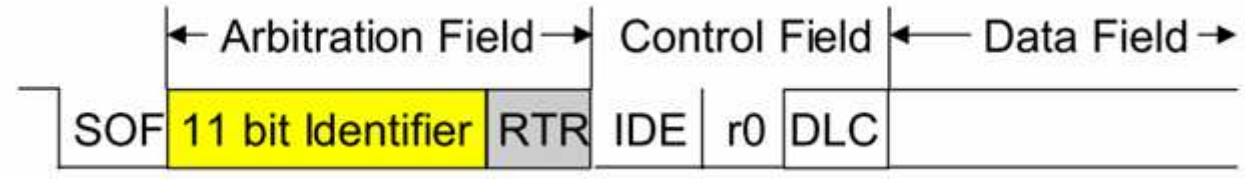
Exemplo de tráfego no barramento



Formato da trama de dados (CAN 2.0A)

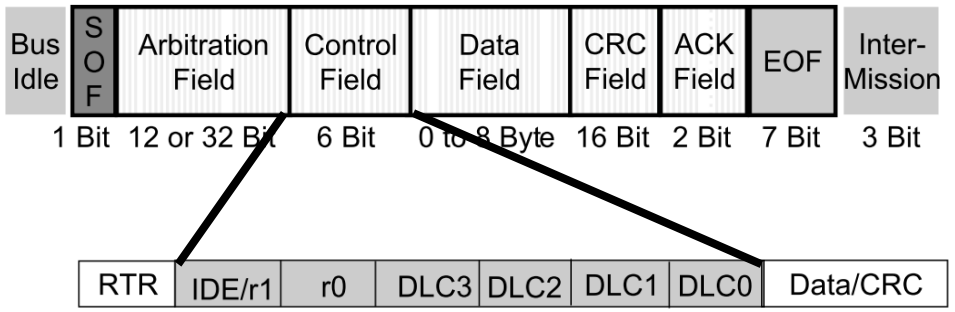


- **SOF** (Start of Frame)
  - Bit dominante ('0') indica o início da trama
  - Usado para sincronização do relógio dos nós recetores
- **Arbitration**
  - **Identifier** (11 bits) – identificador da mensagem que também serve para arbitragem entre diferentes *masters* que podem iniciar a transmissão das suas tramas em simultâneo (id mais baixo, maior prioridade)
  - **RTR** (Remote Transmission Request – 1 bit)
- 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 extendido 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)



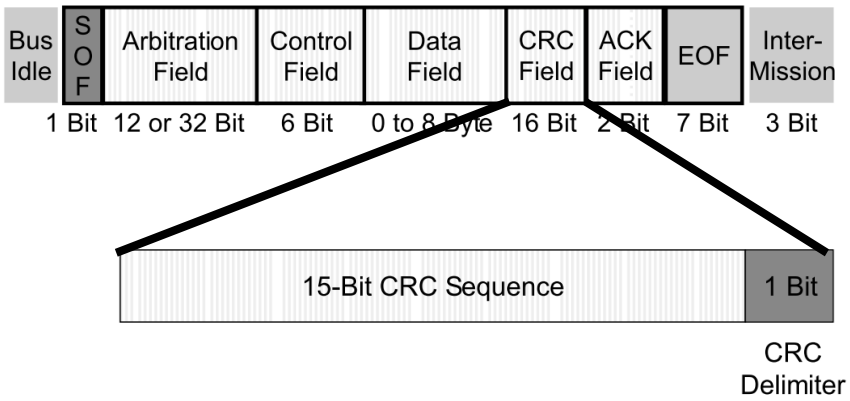
No. of Data Bytes	Data Length Code (DLC)			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d/r	d/r	d/r

min. length of Data Field = 0 Byte

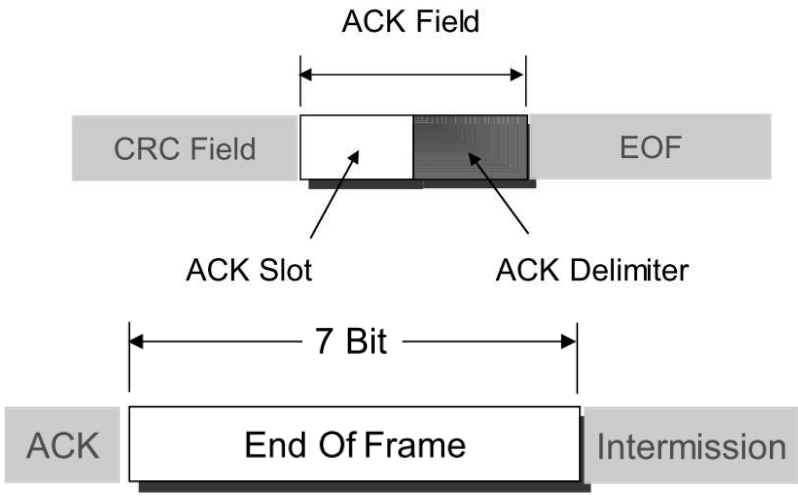
max. length of Data Field = 8 Byte

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



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



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)
- **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)

Deteção de erros de comunicação

- 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

Filtros de aceitação de mensagens e máscaras

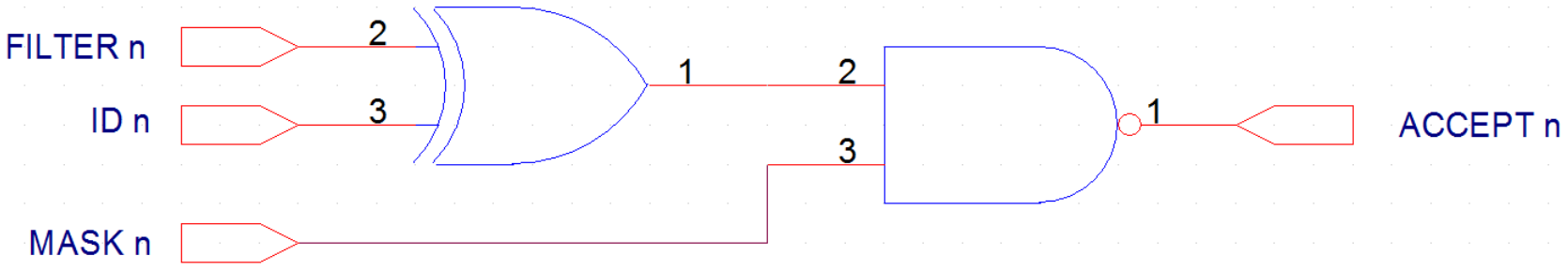
- O CAN é um barramento de tipo "**broadcast**", ou seja, uma mensagem transmitida por um nó é recebida por todos os nós da rede
- O controlador CAN de cada nó lê todas as mensagens que circulam no barramento e coloca-as num registo temporário designado por "Message Assembly Buffer" (MAB)
- Logo que 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

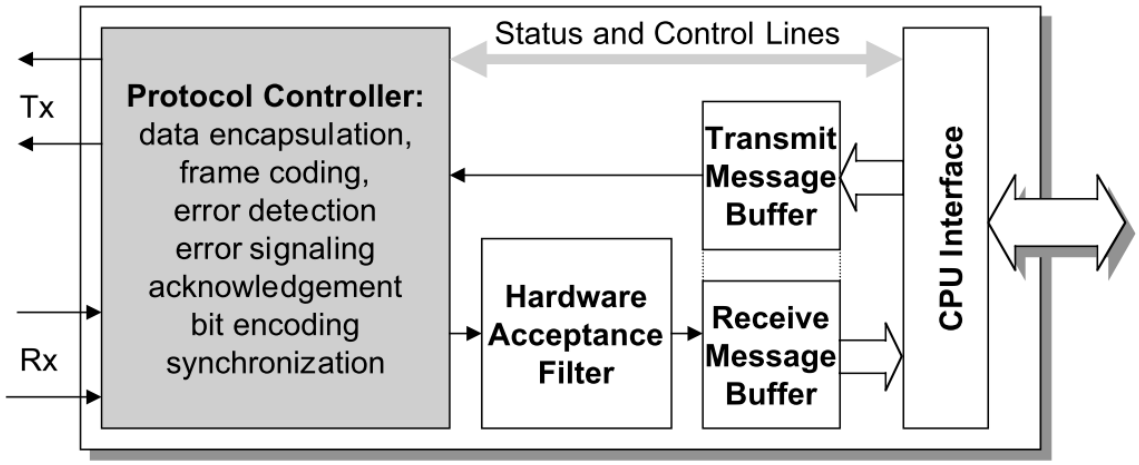
Mask bit n	Filter bit n	Message Identifier bit n	Accept/Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

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

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

Controlo de acesso ao meio – Arbitragem

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