Controlled Area Network Bus

Table of Contents

[CAN Protocol 2](#_Toc106752977)

[Features 4](#_Toc106752978)

[Basic Configuration 4](#_Toc106752979)

[Electromagnetic Interference 5](#_Toc106752980)

[Physical Layer 5](#_Toc106752981)

[Collision Avoidance 6](#_Toc106752982)

[Transmission Speeds 8](#_Toc106752983)

[Frame Types 9](#_Toc106752984)

[Data Frame 9](#_Toc106752985)

[Remote Frame 10](#_Toc106752986)

[Error Frame 10](#_Toc106752987)

[Overload Frame 10](#_Toc106752988)

**Controlled Area Network Bus** is a protocol used in **Large Scale Embedded Systems**. It is primarily used in **vehicles** to allow microcontrollers and devices to communicate without a host.

Typically, sensors, actuators and other control devices use CAN. These devices are **not directly connected** to the bus, but go through a host processor and a CAN controller.

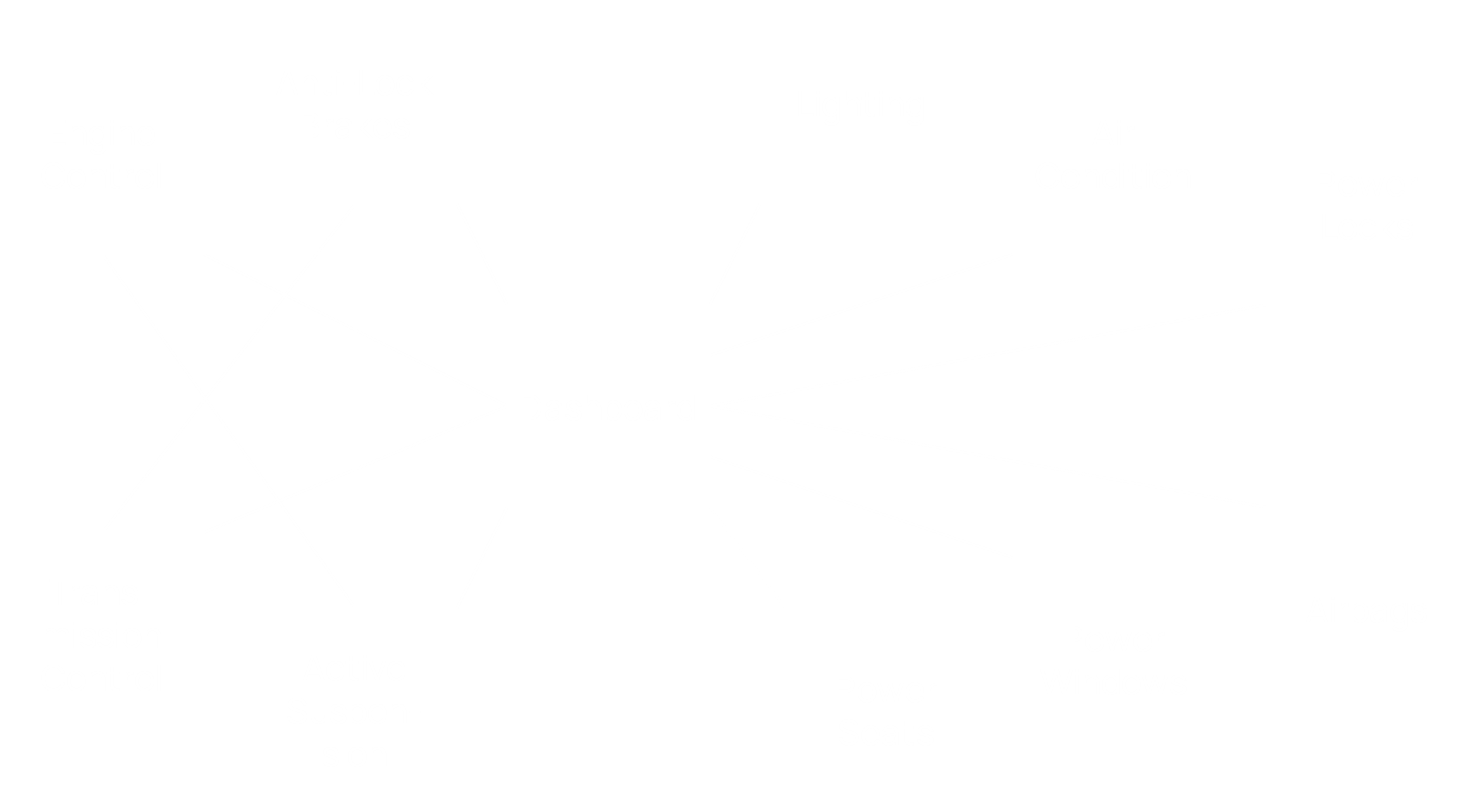
## CAN Protocol

The CAN Protocol uses three layers:

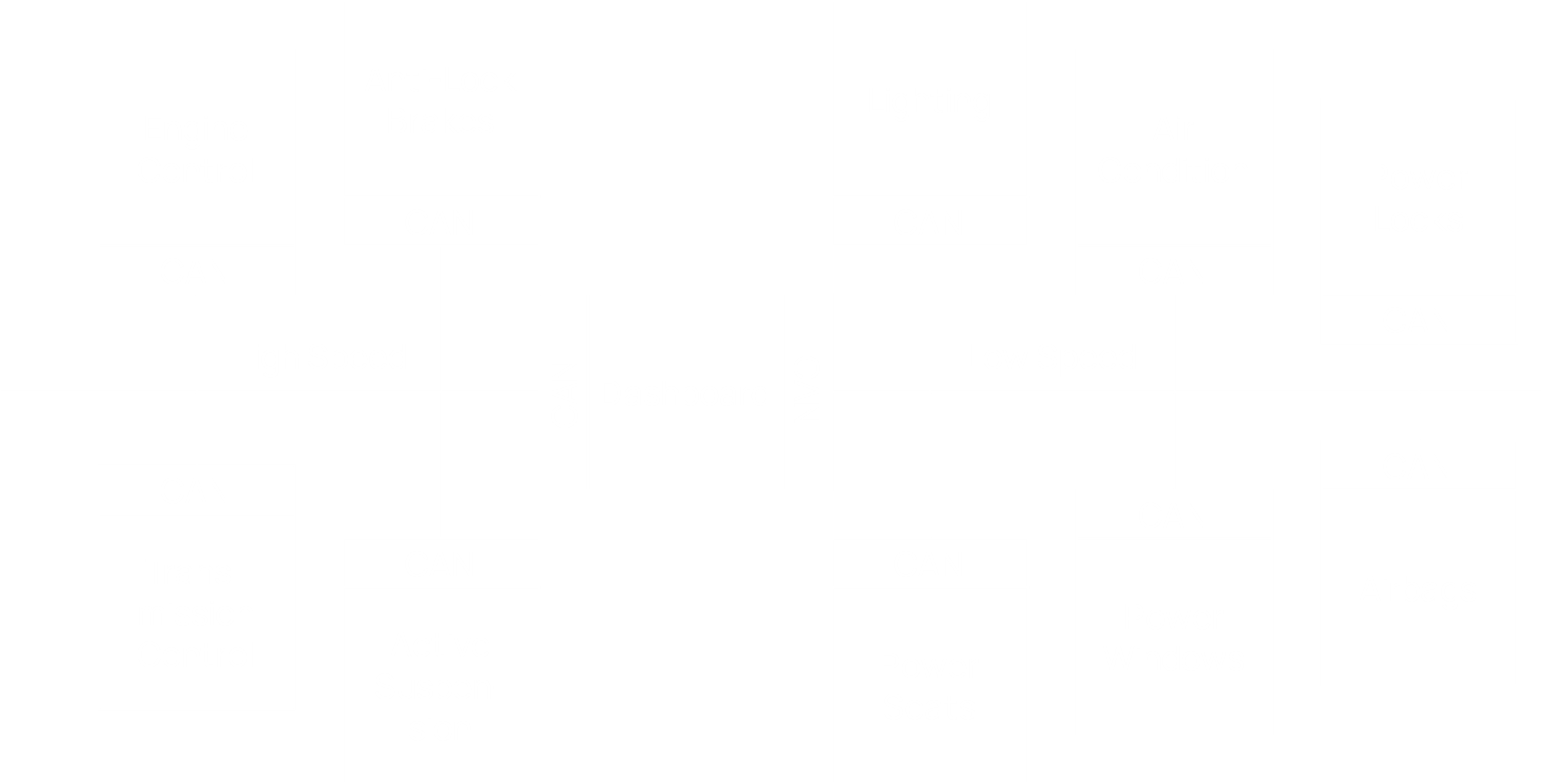
1. Physical Layer
2. Protocol Layer
3. Message Filtering Layer (with add-on protocols)

These layers work in a manner like that in TCP/IP from networking.

Before the development of CAN, all the components of a vehicle were interconnected.



CAN reduces these interconnections by introducing **serial bus systems**. The only overhead is that some **CAN-specific hardware** had to be added to each component that would provide the protocol to communicate via the bus.

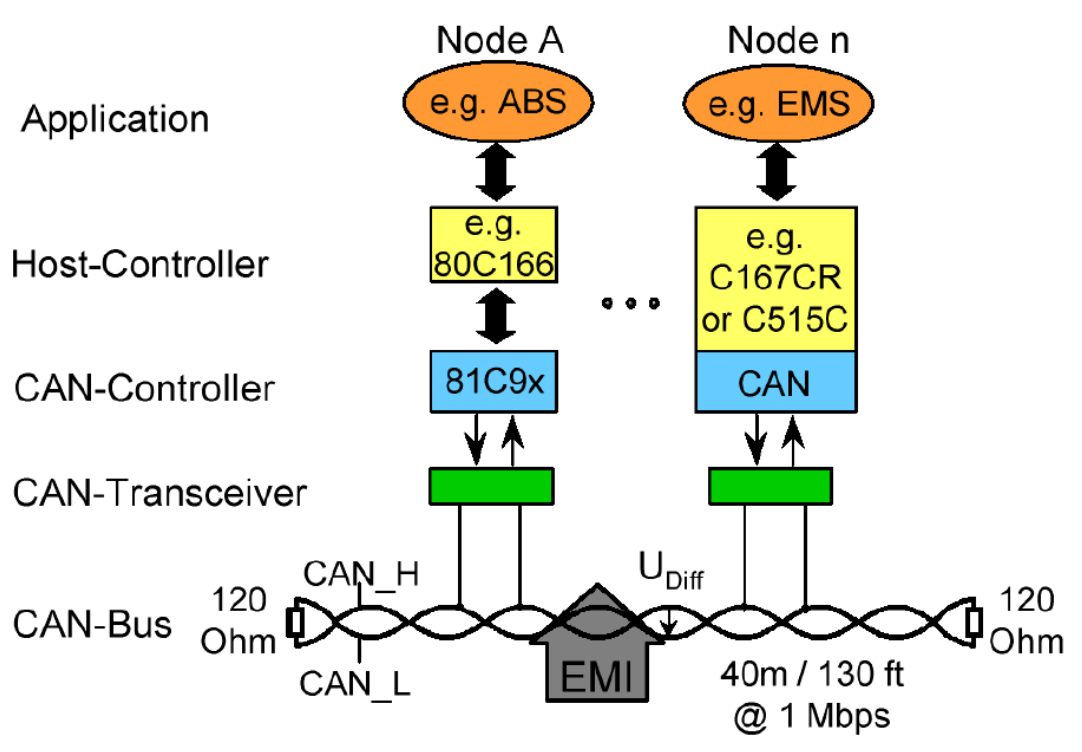


## Features

* CAN **broadcasts** messages, meaning all the nodes on the bus can hear all transmissions. The hardware at each node filters out the messages that are not relevant to the node.
* It is **easy to connect and disconnect** nodes.
* It uses the **multi-master concept**, which we will examine later.
* The number of nodes is **unlimited**.
* There is no **node addressing**.

## Basic Configuration

The typical structure of the CAN hardware is provided below.



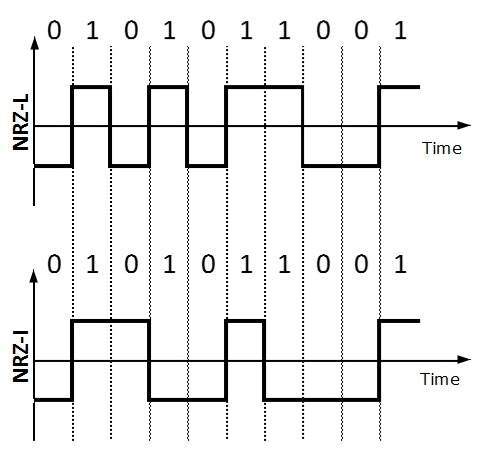
## Electromagnetic Interference

CAN is **insensitive to electromagnetic interference**. It uses a **twisted-pair cable**, which has two lines twisted together. One line carries the data while the other line carries noise. As long as the data signal has a significantly higher strength than the noise signal, the data will be understandable.

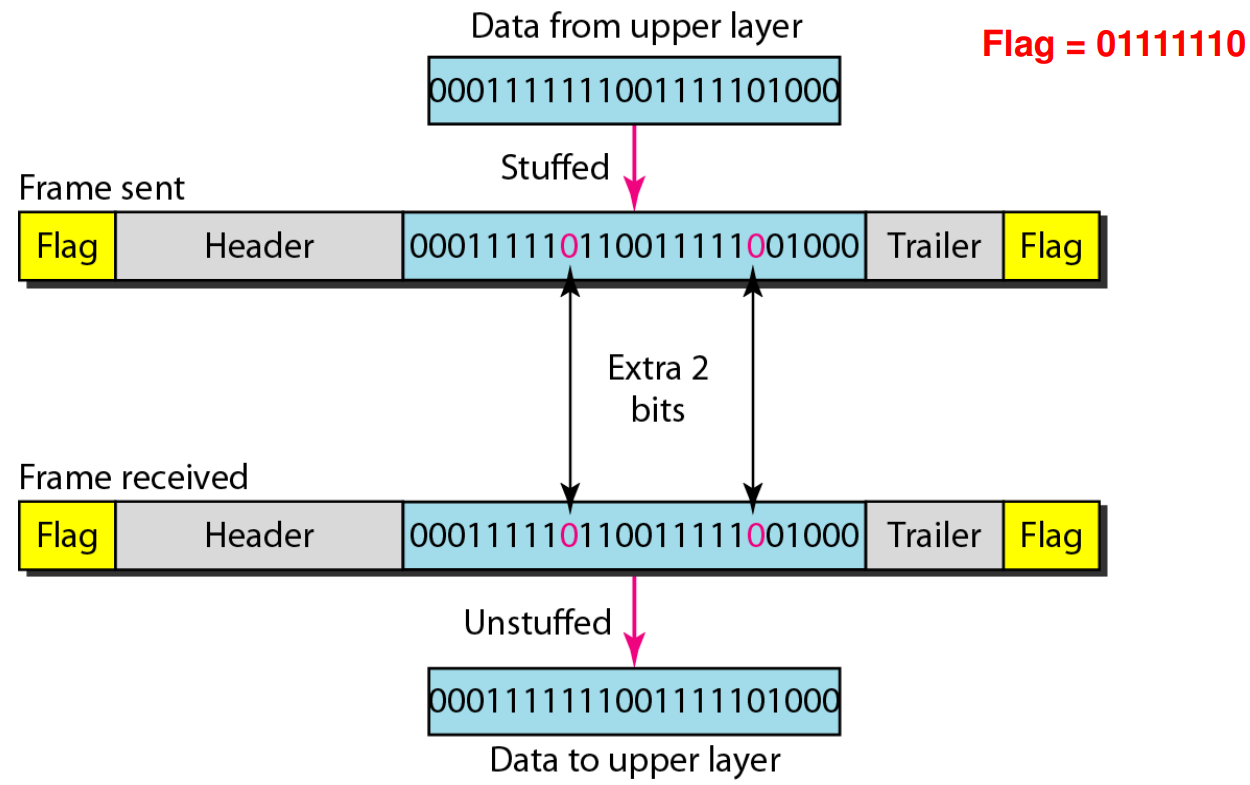
On top of this, **shielding** can also be used on the transmission lines, which reduces electromagnetic interference from outside getting in and also stops electromagnetic interference from inside getting out.

## Physical Layer

At the physical layer, **Non-Return to Zero** (NRZ) modulation is used, which is a technique to turn digital data into a digital signals. There are two forms of NRZ, NRZ-L and NRZ-I, but it is not specified which form is used in the CAN bus.



Additionally, **bit stuffing** is also used. Each frame being transmitted on the CAN bus consists of a start flag, a header, the data, a trailer and finally an end flag. The start and end flags use the pattern 01111110. However, it is possible that the data will also have this pattern, which might cause the receiver to think the frame has ended earlier than it actually does. To prevent this, whenever there are **5 consecutive 1s**, then the CAN bus adds a 0 after them on the sender end. On the receiver end, this 0 is removed.



## Collision Avoidance

One of the basic principles of networking is the concept of **Carrier Sense Multiple Access** (CSMA), which essentially says that no station should use the channel if the channel is already busy. There are several mechanisms that are used to ensure this. The specific one that CAN bus uses is called the **Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration**.

The CAN bus is said to be an **AND bus**. This means that for multiple stations, only if all the stations give a 1 bit, will the bus itself be in the 1 state.

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **Bus** |
| D | D | D | D |
| D | D | R | D |
| D | R | D | D |
| D | R | R | D |
| R | D | D | D |
| R | D | R | D |
| R | R | D | D |
| R | R | R | R |

In the table above, 1s are represented as R for recessive and 0s are represented as D for dominant. This is because even a single 0 can cause the entire bus to become 0, meaning the 0 is dominant over the 1.

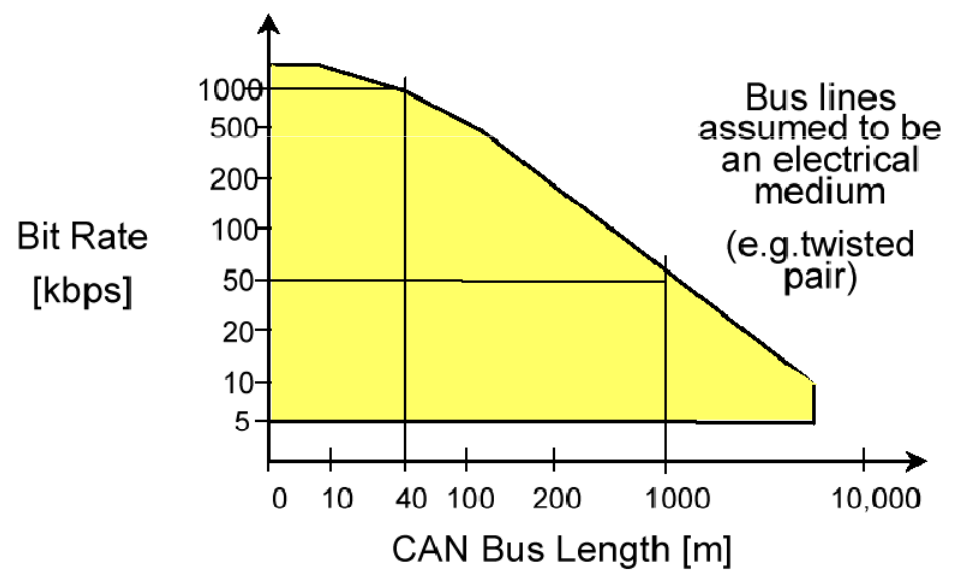
Non-destructive arbitration uses this feature of the CAN bus. Each station is assigned an **11-bit identifier**, which is used to decide which station should be allowed to transmit if multiple stations are trying to transmit at the same time.

Suppose we have three stations, A, B and C, which have the identifiers 00000001000, 00000001001 and 00000001100 respectively. The first 8 bits are the same for all of these stations (00000001), so nothing happens. On the 6th bit, C is the only station with a 1. This causes C to back off (since 1 is recessive), but since both A and B have a dominant bit, we must continue with those two. The 7th bit is 0 for both A and B, however, on the 8th bit, B has a 1 while A has a 0. This causes B to back off, meaning A wins the right to transmit.

These identifiers are assigned on a **first-come, first-serve basis** and, as can be seen above, the lowest number has the highest priority.

## Transmission Speeds

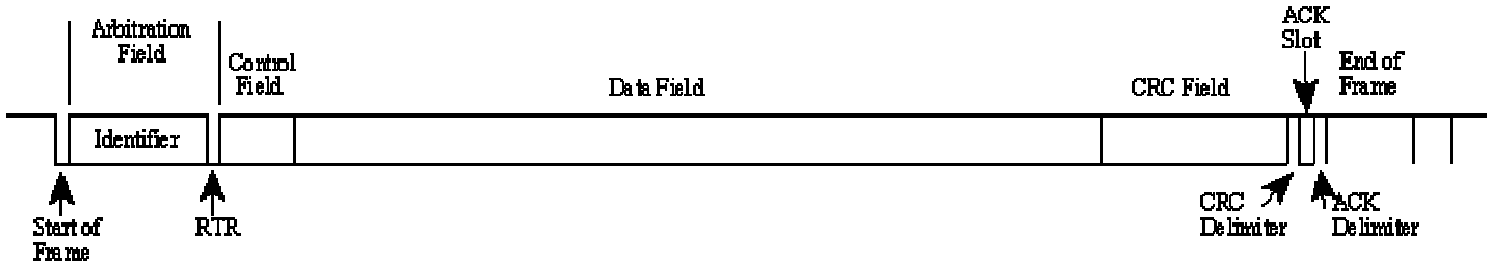
The maximum distance that a CAN bus should cover is **40 meters**. Beyond this, the transmission speeds begin to degrade.



## Frame Types

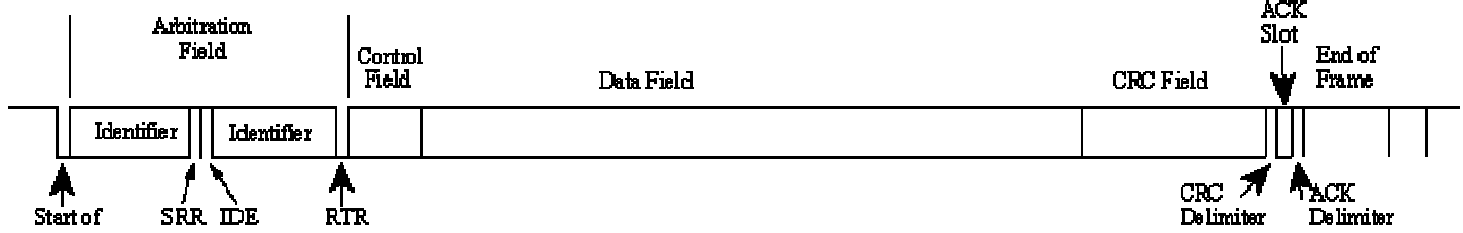
The CAN standard defines **four frame types**, data frames, remote frames, error frames and overload frames.

### Data Frame



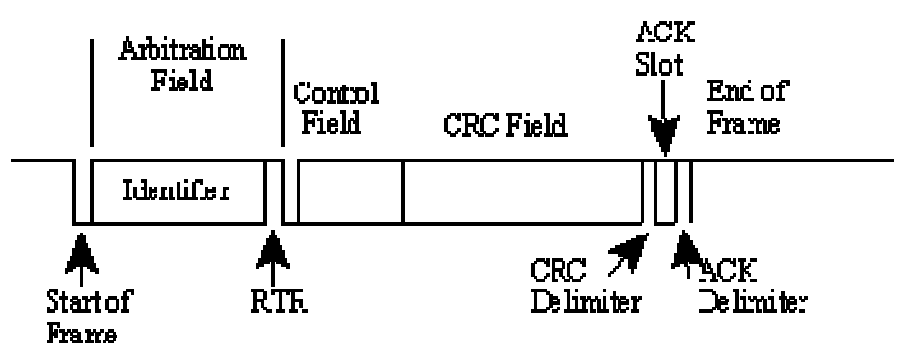
The diagram above shows the CAN **data frame**. It consists of a start of frame byte, the 11-bit identifier, a control field, a data field, a CRC field, an ACK slot and finally the end of frame byte. Note that an ACK only specifies that some node has received the data correctly. This may not be the intended node.

More specifically, this shows the **CAN 2.0A** format of the data frame. There is another format called the **CAN 2.0B**, shown below. The only difference is that the identifier is of 29 bits here instead.



### Remote Frame

Normally, a source sends data and a destination receives it. However, we can have a situation where the destination requests that the source send some data. This is done using a **remote frame**. A remote frame is exactly the same as a data frame except that it does not have any **body**.



### Error Frame

The **error frame** has two fields. The first field is the superposition of the error flags contributed from different stations. The second field is the **error delimiter**, given by 8 recessive bits.

### Overload Frame

The **Overload Frame** essentially tells the sender that the receiver is busy right now. This frame is not used by modern CAN controllers, since they are clever enough to avoid that situation entirely.