Control Area Network (CAN) Bus

Course Teacher:

Md. Obaidur Rahman, Ph.D.
Professor
Department of Computer Science and Engineering (CSE)
Dhaka University of Engineering & Technology (DUET), Gazipur.

Course ID: CSE - 4619
Course Title: Peripherals, Interfacing and Embedded Systems
Department of Computer Science and Engineering (CSE),
Islamic University of Technology (IUT), Gazipur.

Lecture References:

- Book:
  - *Embedded System Design*, **Author:** P. Marwedel
  - *Embedded System Design: An Introduction to Processes, Tools and Techniques*, **Author:** Arnold Berger, Arnold S. Berger
- Lecture Materials:
  - *Microprocessor Engineering*, Sheffield Halam University.

Introduction to CAN

- **CAN bus** (for **controller area network**) is an important embedded protocol for vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer.
- Primarily automotive, but now also used in other areas such as aerospace, maritime, industrial automation and medical equipment.
- Development of Controller Area Network bus started originally in 1983 at Robert Bosch.
- The first CAN controller chips, produced by Intel and Philips, came on the market in 1987.

Introduction to CAN

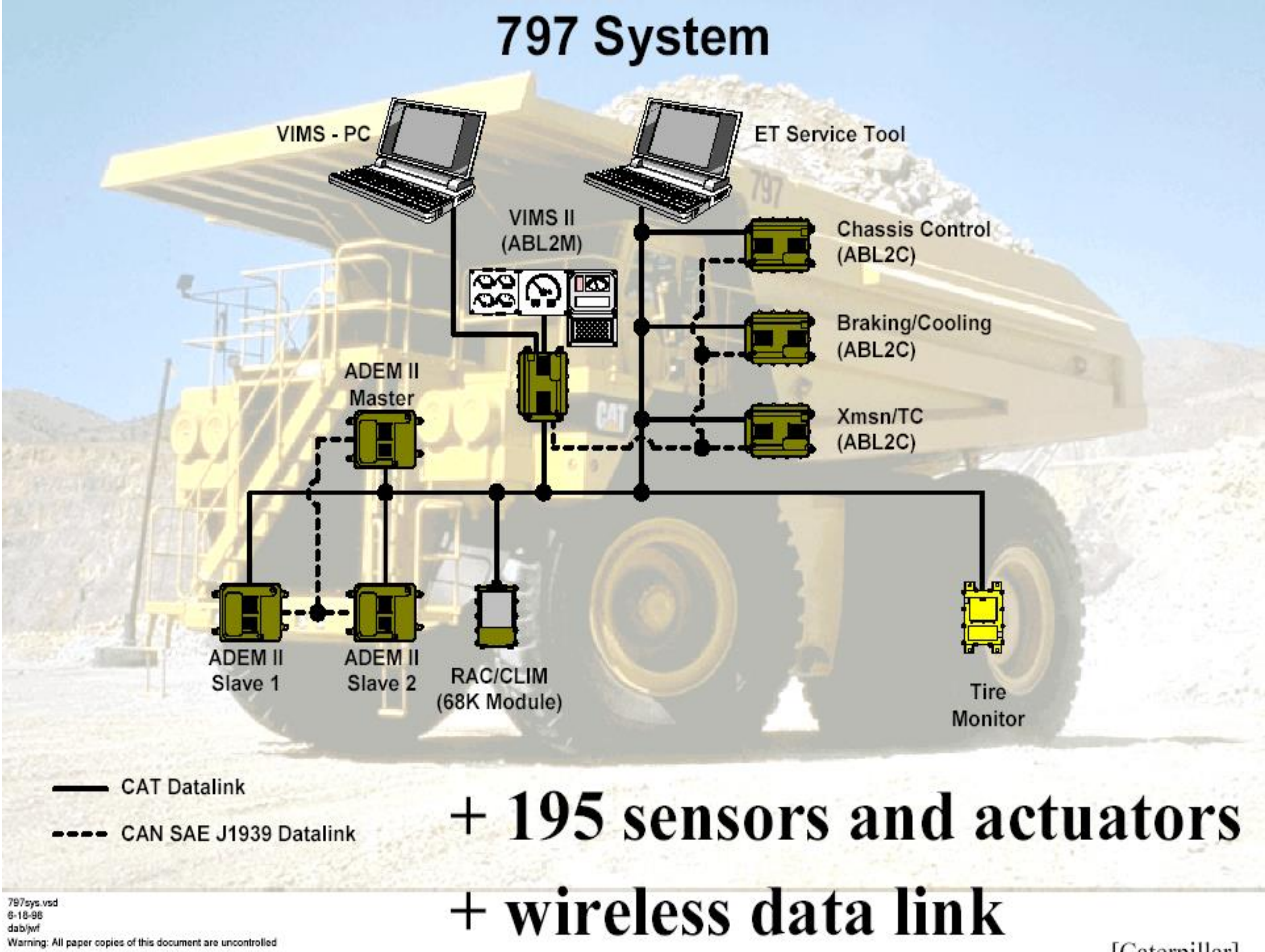
- The devices that are connected by a CAN network are typically sensors, actuators, and other control devices.
- These devices are not connected directly to the bus, but through a host processor and a CAN controller.
- CAN specifies:
  - Physical layer
  - Protocol layer
  - Message filtering layer (with add-on protocols)
- **Important Note about CAN**
  - How message prioritization achieved
  - How "small" nodes can be kept from overloading with received messages

The Development of CAN

The development of CAN began when more and more electronic devices were implemented into **modern motor vehicles**. Examples of such devices include *engine management systems, active suspension, ABS, gear control, lighting control, air conditioning, airbags and central locking*. All this means more safety and more comfort for the driver and of course a reduction of fuel consumption and exhaust emissions.

To improve the behavior of the vehicle even further, it was necessary for the different **control systems (and their sensors) to exchange information**. This was usually done by discrete interconnection of the different systems (i.e. ***point to point wiring***). The requirement for information exchange has then grown to such an extent that a cable network with a length of up to several miles and many connectors was required. This produced growing problems concerning material cost, production time and reliability.

CAN (SAE J1939) Example: Caterpillar 797

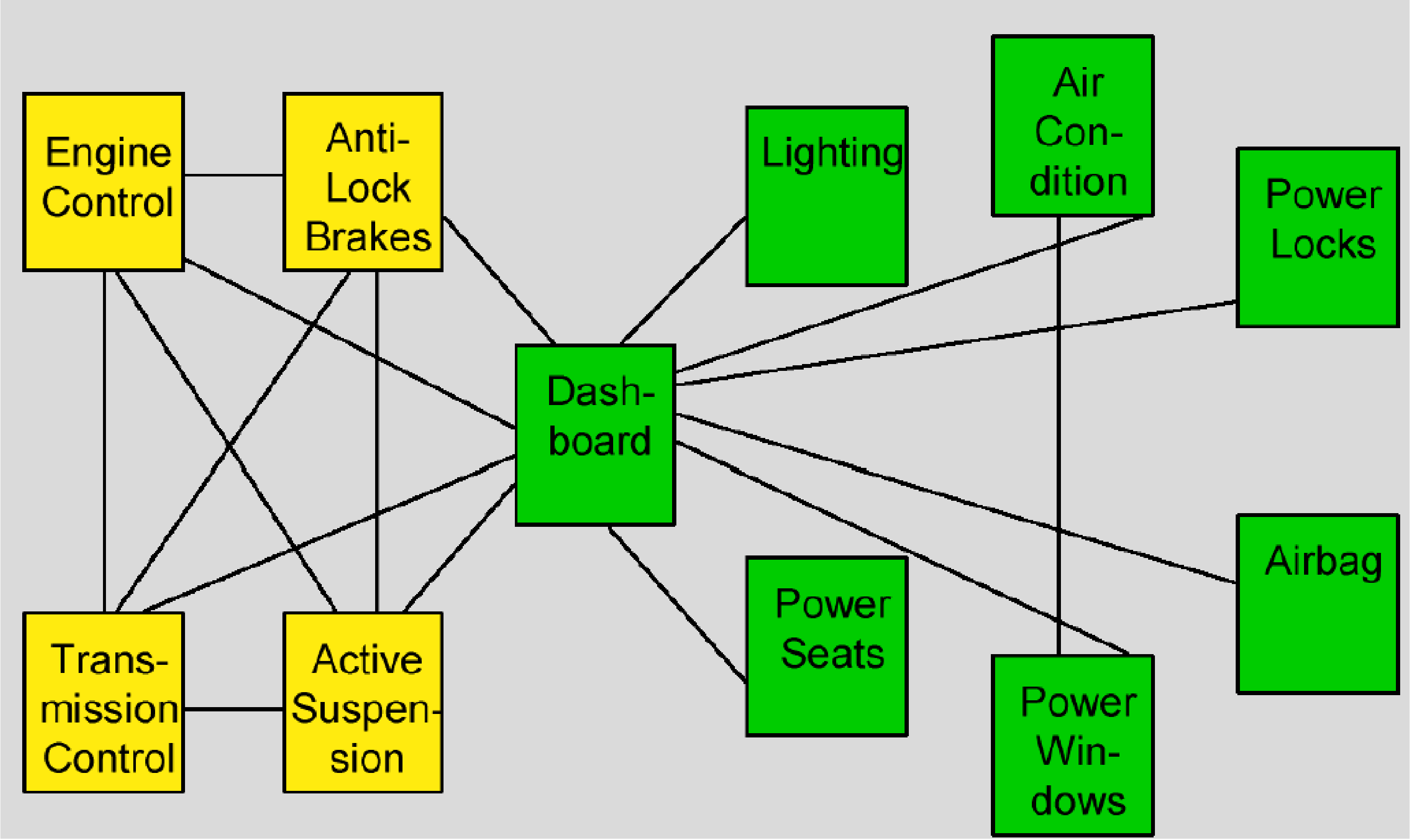


+ 195 sensors and actuators

+ wireless data link

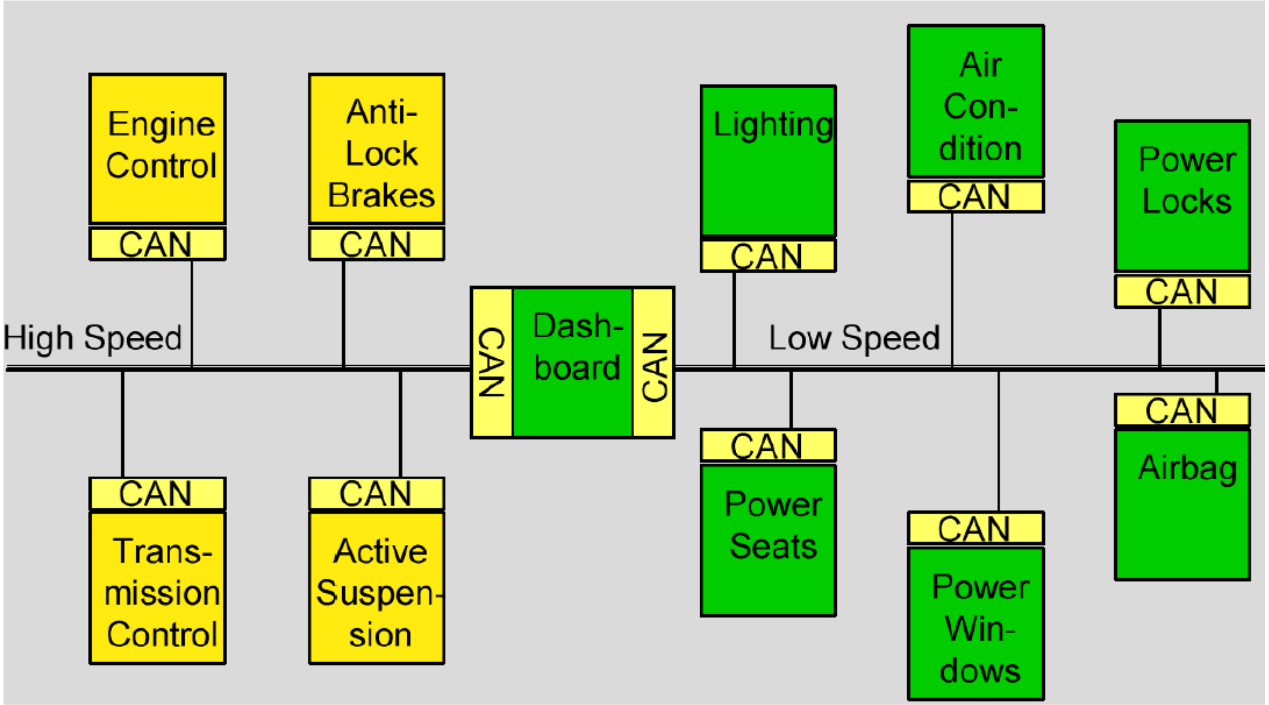
[Caterpillar]

Before CAN

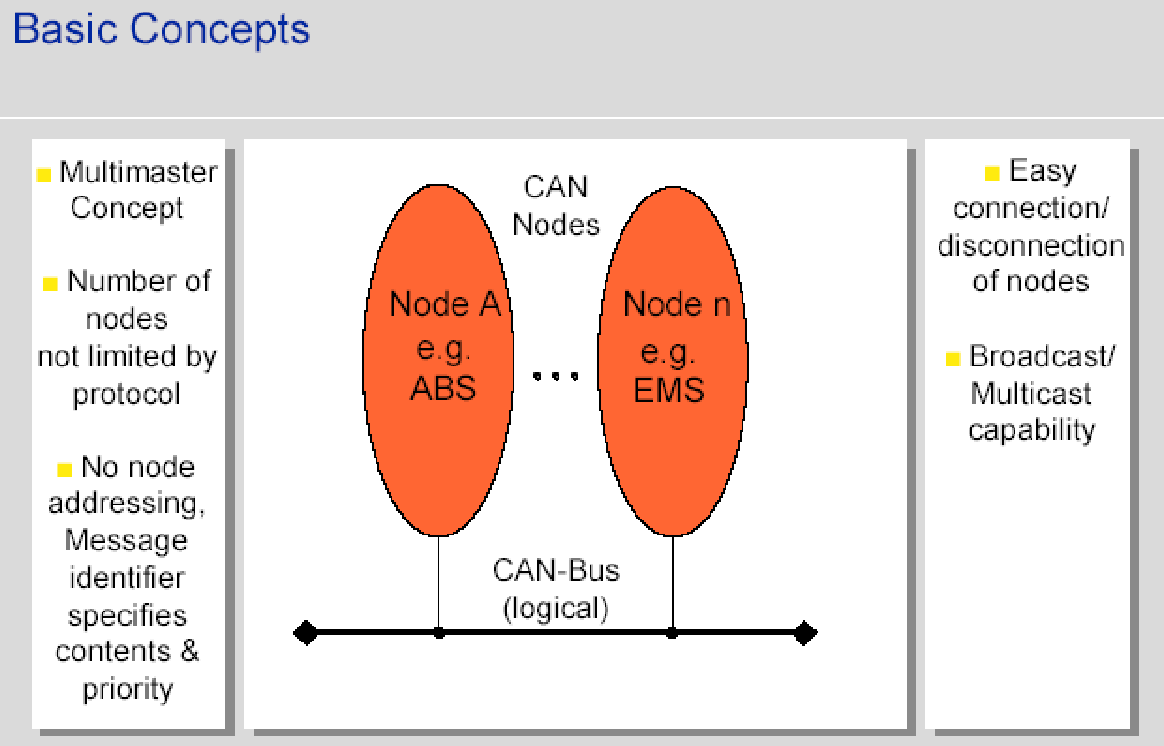

Engine Control
Anti-Lock Brakes
Lighting
Air Con-dition
Power Locks
Dash-board
Trans-mission Control
Active Suspen-sion
Power Seats
Power Win-dows
Airbag

With CAN

The solution to this problem was the connection of the control systems via a serial bus system. This bus had to fulfill some special requirements due to its usage in a vehicle. With the use of CAN, point-to-point wiring is replaced by one serial bus connecting all control systems. This is accomplished by adding some CAN-specific hardware to each control unit that provides the "rules" or the protocol for transmitting and receiving information via the bus.



The CAN bus



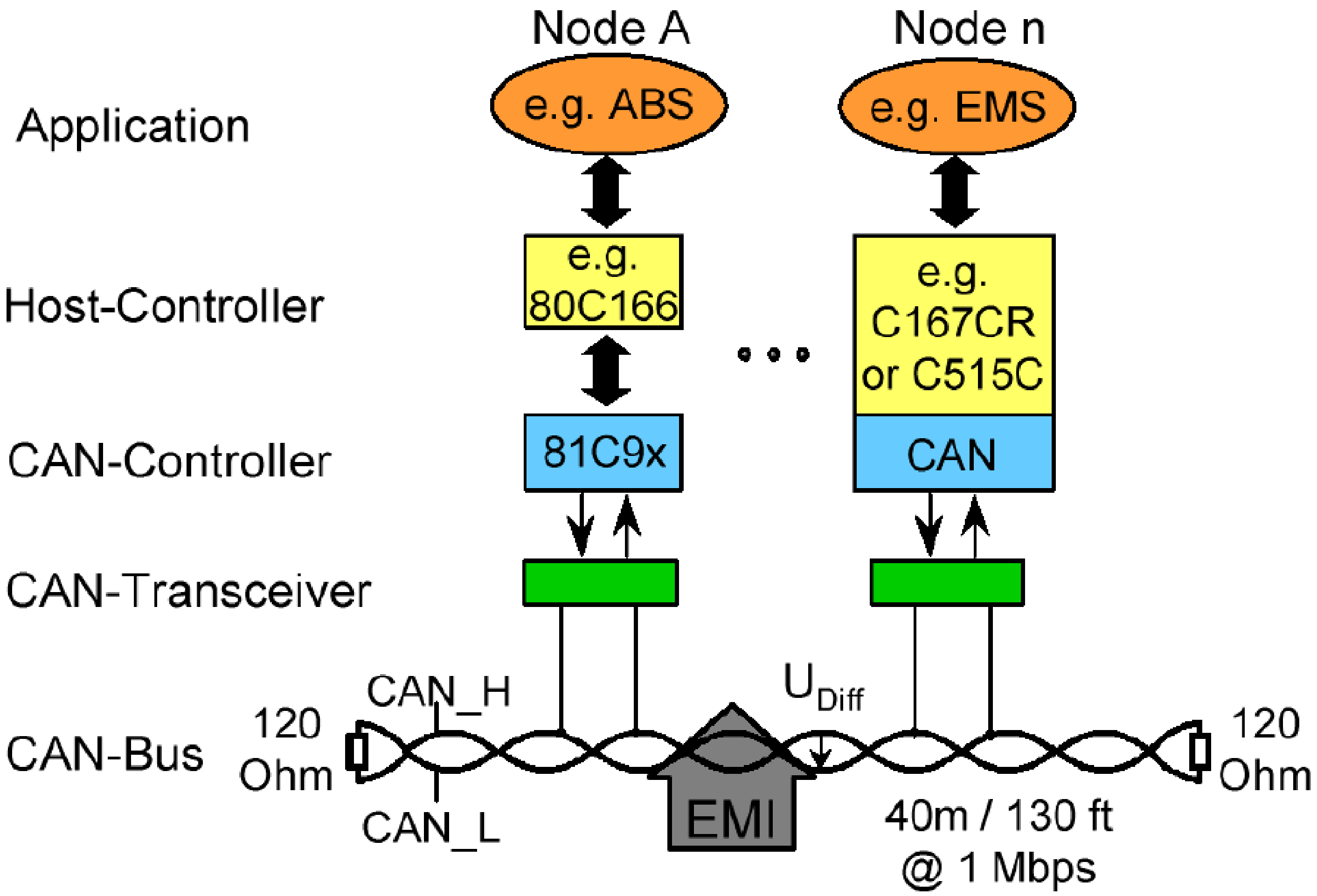
Basic Concepts

- Multimaster Concept
- Number of nodes not limited by protocol
- No node addressing, Message identifier specifies contents & priority

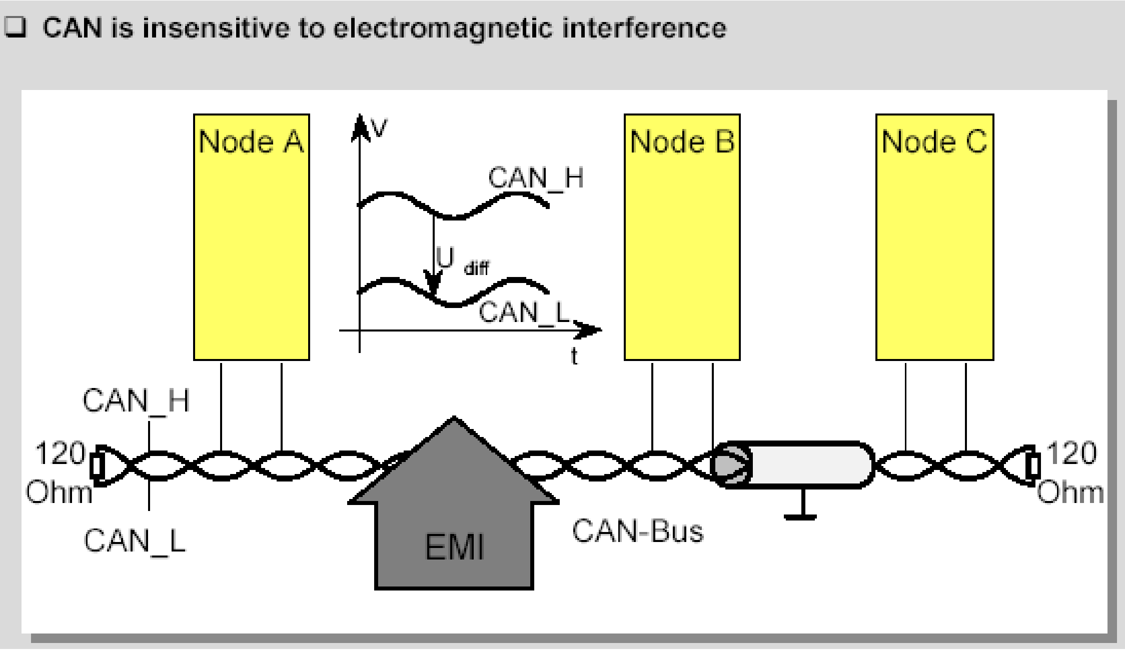
- Easy connection/ disconnection of nodes
- Broadcast/ Multicast capability

- CAN is a broadcast type of bus.
  - This means that all nodes can "hear" all transmissions. There is no way to send a message to just a specific node; all nodes will invariably pick up all traffic. The CAN hardware, however, provides local filtering so that each node may react only on the "interesting" messages.

Basic Configuration



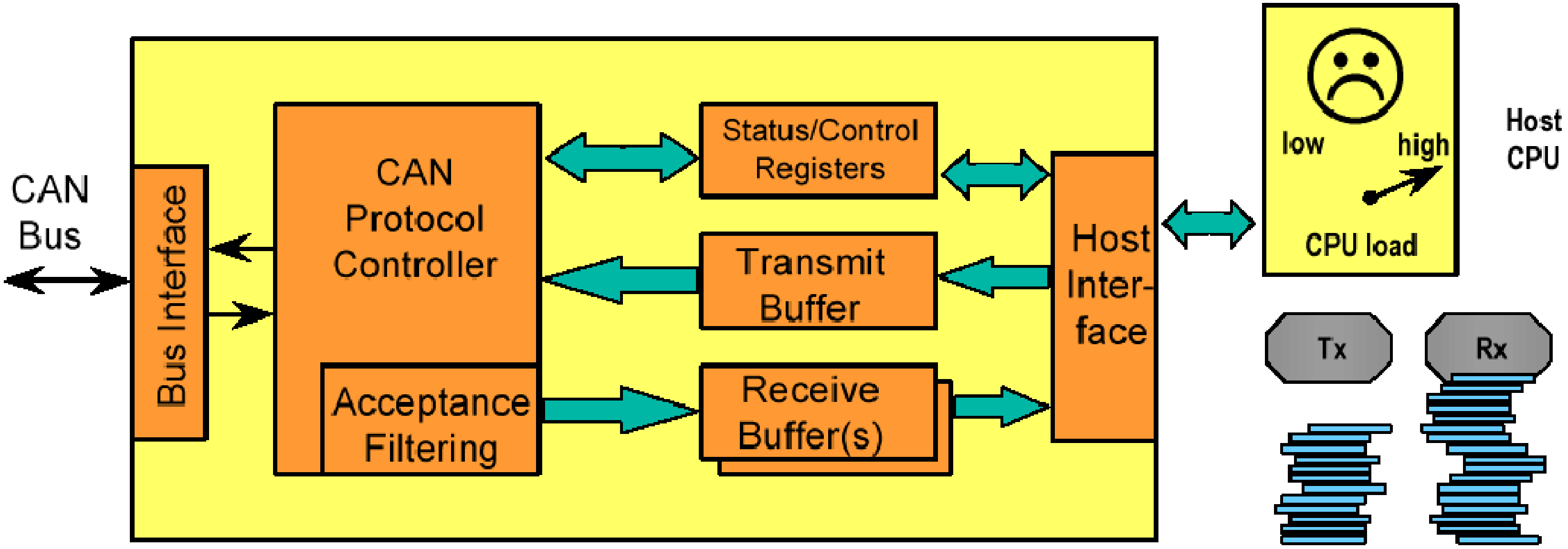
CAN and EMI



❑ CAN is insensitive to electromagnetic interference

Due to the differential nature of transmission CAN is insensitive to electromagnetic interference, because both bus lines are affected in the same way which leaves the differential signal unaffected.
To reduce the sensitivity against electromagnetic interference even more, the bus lines can additionally be shielded. This also reduces the electromagnetic emission of the bus itself, especially at high baudrates.

A Basic CAN controller



- Cheap CAN controller – CPU could get overrun with messages even if it didn't need them.

A Basic CAN controller

- **Host processor**
  - The host processor decides what received messages mean and which messages it wants to transmit itself.
  - Sensors, actuators and control devices can be connected to the host processor.
- **CAN controller** (hardware with a synchronous clock)..
  - *Receiving*: the CAN controller stores received bits serially from the bus until an entire message is available, which can then be fetched by the host processor (usually after the CAN controller has triggered an interrupt).
  - *Sending*: the host processor stores its transmit messages to a CAN controller, which transmits the bits serially onto the bus.

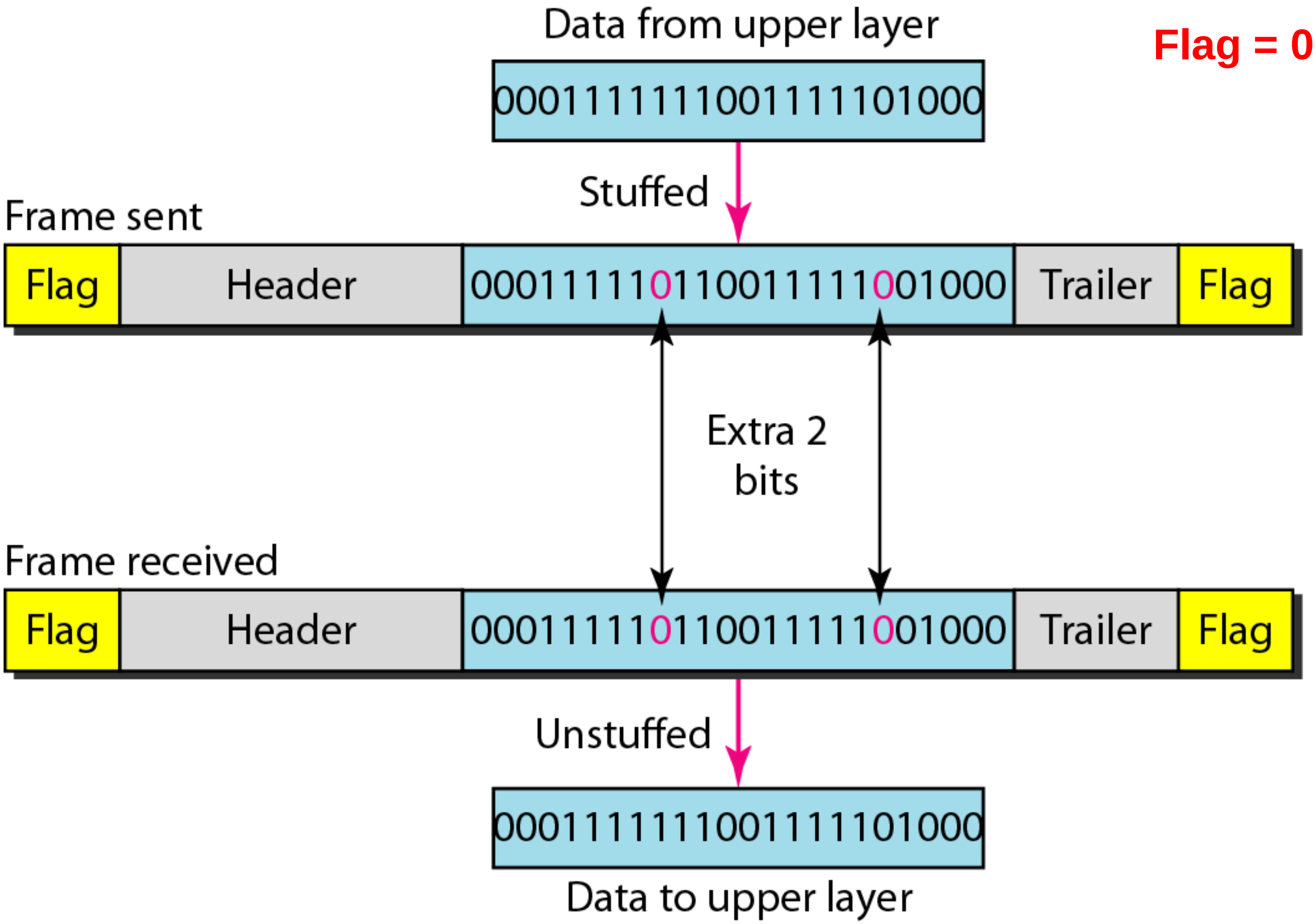
A Basic CAN controller

- **Transceiver**
  - *Receiving*: it adapts signal levels from the bus to levels that the CAN controller expects and has protective circuitry that protects the CAN controller.
  - *Transmitting*: it converts the transmit-bit signal received from the CAN controller into a signal that is sent onto the bus.

CAN Bus Features

- The physical layer uses differential transmission on a twisted pair wire. The bus uses **Non-Return To Zero (NRZ)** with **bit-stuffing**.
- The nodes are connected to the bus in a ***wired-and*** fashion: if just one node is driving the bus to a logical 0, then the whole bus is in that state regardless of the number of nodes transmitting a logical 1.
- Max. transfer rate of 1000 kilobits per second at a maximum bus length of 40 meters or 130 feet when using a twisted wire pair which is the most common bus medium used for CAN.
- Message length is short with a maximum of 8 data bytes per message and there is a low latency between transmission request and start of transmission. The messages are protected by a **CRC** type checksum

CAN Bus Features: Basic Bit Encoding



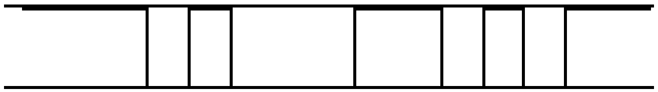
Flag = 01111110

CAN Bus Feature

- The bus access is handled via the advanced serial communications protocol **Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration.** This means that collision of messages is avoided by bitwise arbitration without loss of time.
- There is no explicit address in the messages, instead, **each message carries a numeric value** which controls its **priority on the bus**, and may also serve as an identification of the *contents* of the message.
- An elaborate error handling scheme that results in retransmitted messages when they are not properly received.
- There are effective means for isolating faults and removing faulty nodes from the bus.

CAN Bus Characterstics

Two logic states
possible on the bus:
“1” = recessive
“0” = dominant



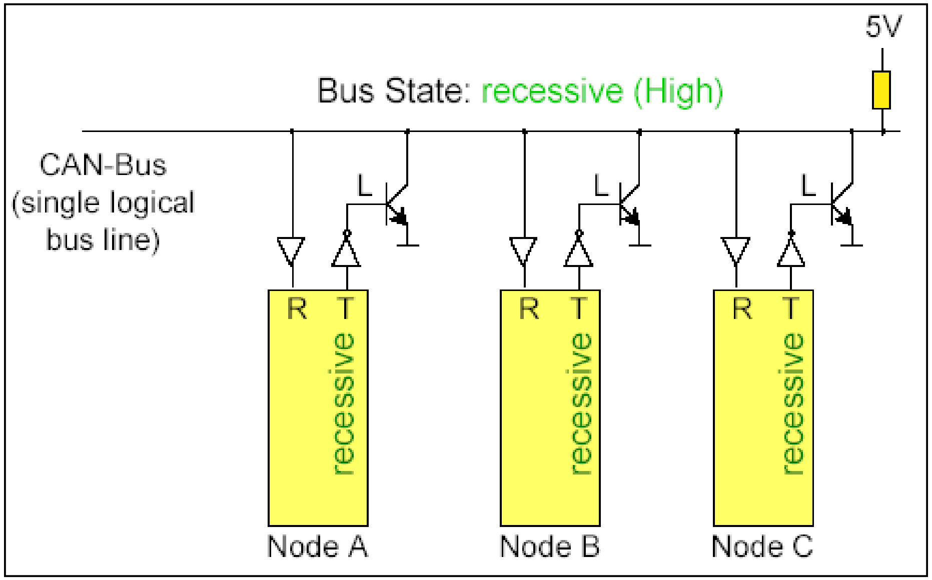
“1” = recessive
“0” = dominant

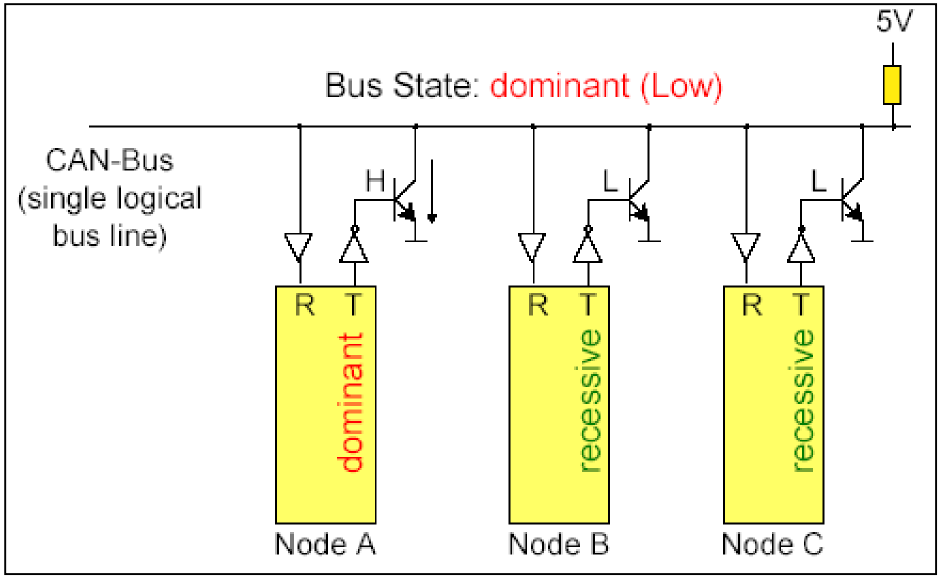
A	B	C	BUS
D	D	D	D
D	D	R	D
D	R	D	D
D	R	R	D
R	D	D	D
R	D	R	D
R	R	D	D
R	R	R	R

As soon as one node nodes transmits
a dominant bit (zero):
Bus is in the dominant state.

Only if all nodes transmit
recessive bits (ones):
Bus is in the recessive state.

Bus Characteristics – Wired AND





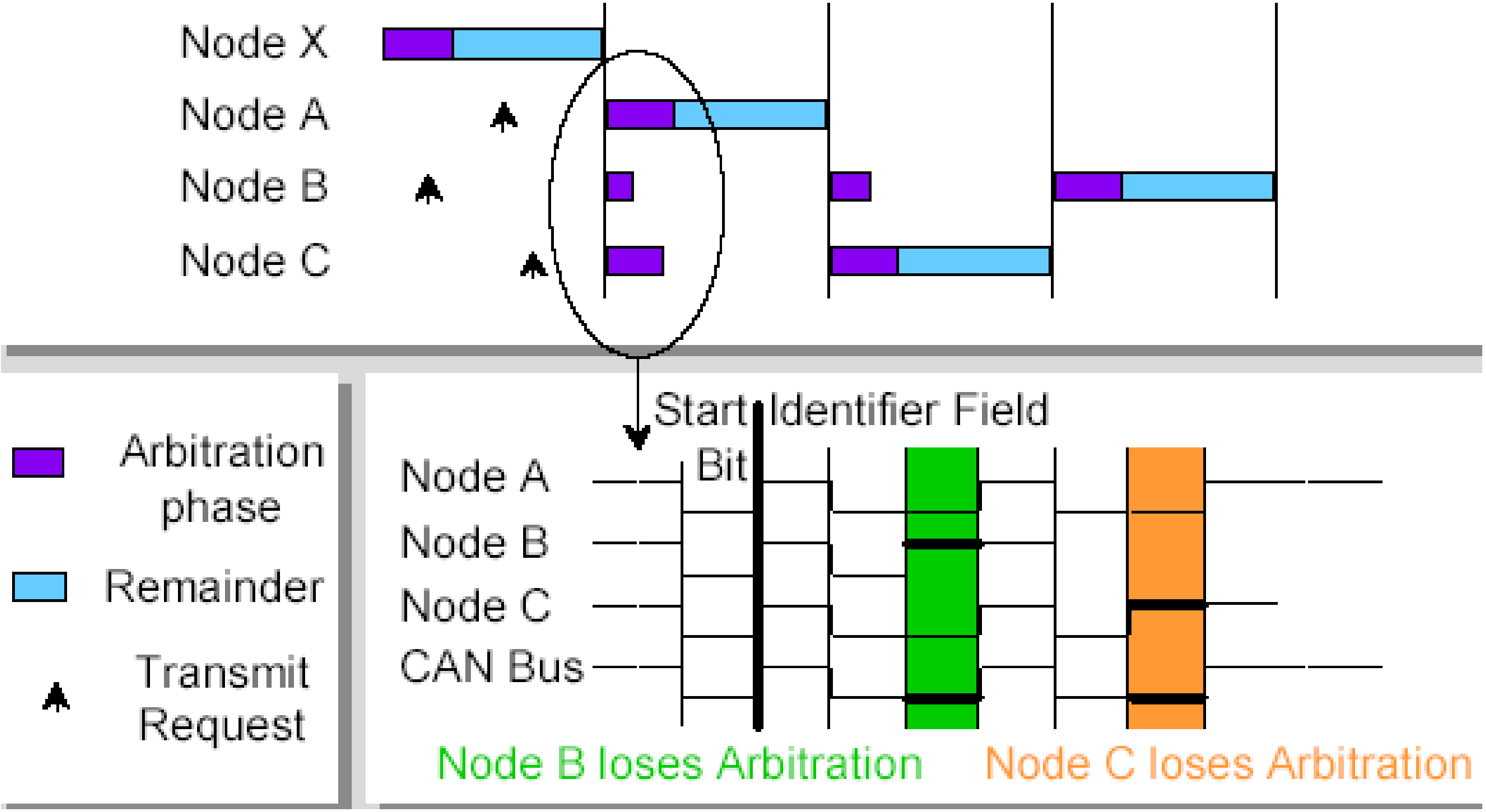
Only if all nodes transmit recessive bits (ones), the Bus is in the recessive state.

If any one node transmits a dominant bit (zero), the bus is in the dominant state.

T is Transmitter, R is receiver. Note nodes can therefore check the line while transmitting. This is important particularly during arbitration.

Bus Access and Arbitration – CSMA/CD NDA

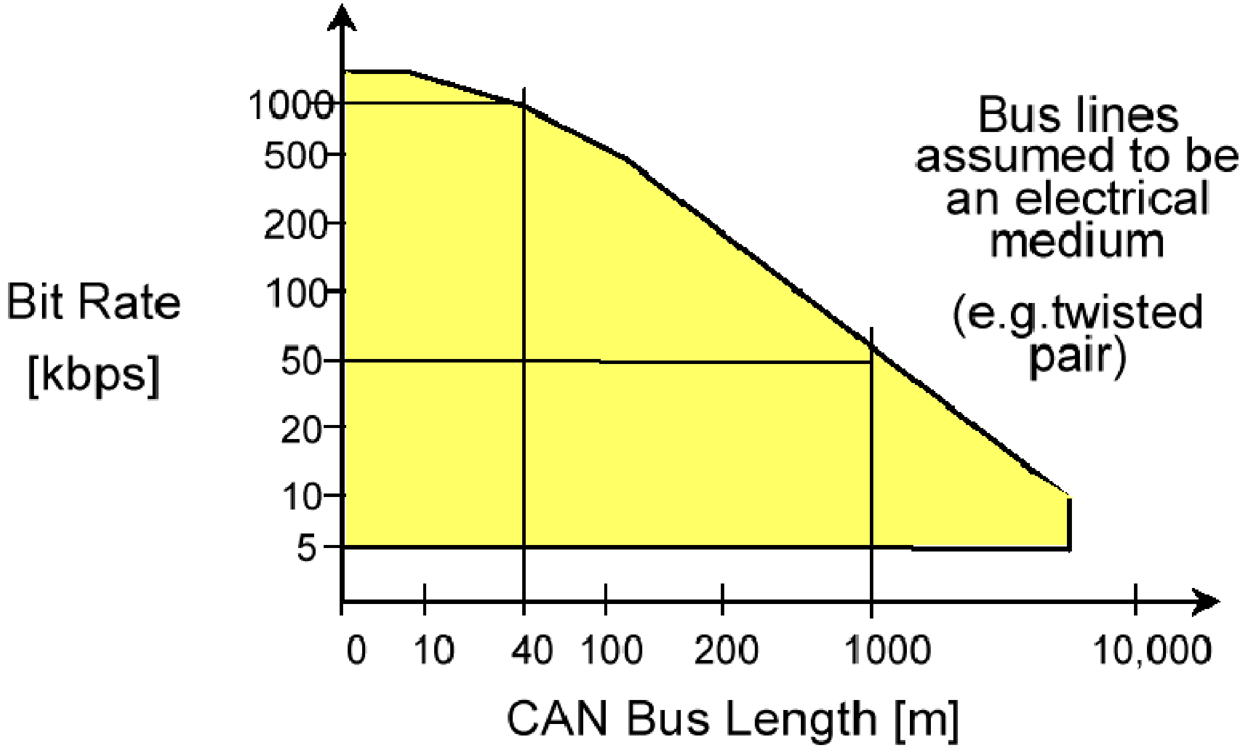
CSMA/CD NDA – Carrier Sense Multiple Access/Collision Detection by Non Destructive arbitration



Bus Transmission Speed

Arbitration limits bus speed. Maximum speed = 2 x t_{pd}

t_{pd} = propagation delay of electrical medium



The Can Protocol

- Specifies how small packets of data may be transported from point A to point B using a shared communications medium.
- It (quite naturally) contains nothing on topics such as
  - flow control
  - transportation of data larger than can fit in a 8-byte message
  - node addresses
  - establishment of communication, etc.

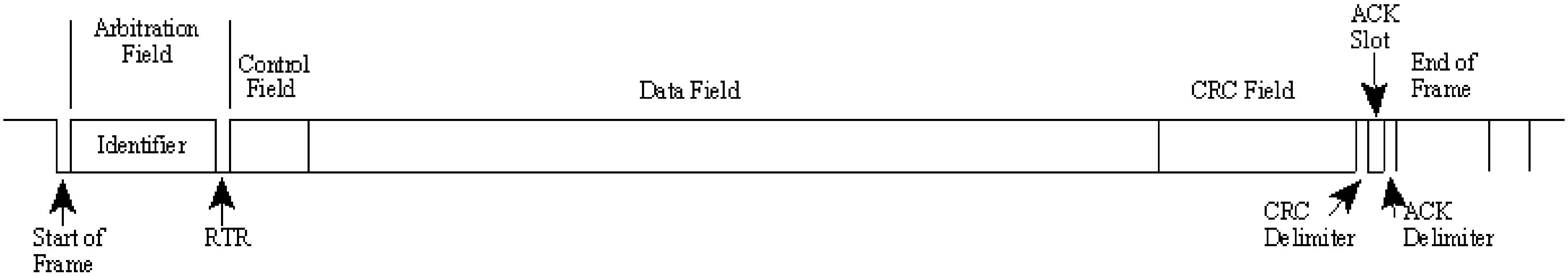
The CAN Standard

- The CAN standard defines four message types
  - Data Frame – the predominantly used message type
  - Remote Frame
  - Error Frame
  - Overload Frame
- The messages uses a clever scheme of bit-wise arbitration to control access to the bus, and each message is tagged with a priority.
- The CAN standard also defines an elaborate scheme for error handling and confinement.
- CAN may implemented using different physical layers, and there are also a number of different connector types in use.

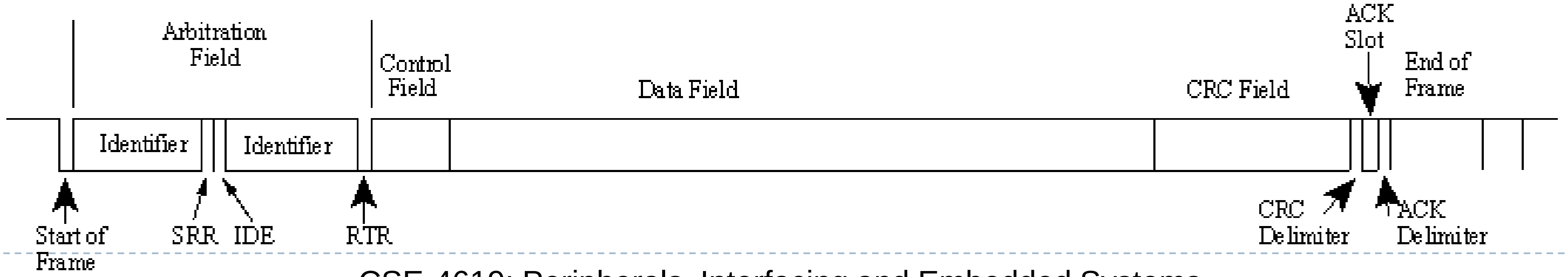
CAN Data Frames

Note 1: It is worth noting that the presence of an Acknowledgement Bit on the bus does not mean that any of the *intended* addressees has received the message. The only thing we know is that *one or more* nodes on the bus has received it correctly
Note 2: The Identifier in the Arbitration Field is not, despite of its name, necessarily identifying the contents of the message.

- CAN 2.0A ("standard CAN" 11-bit ID) Data Frame.

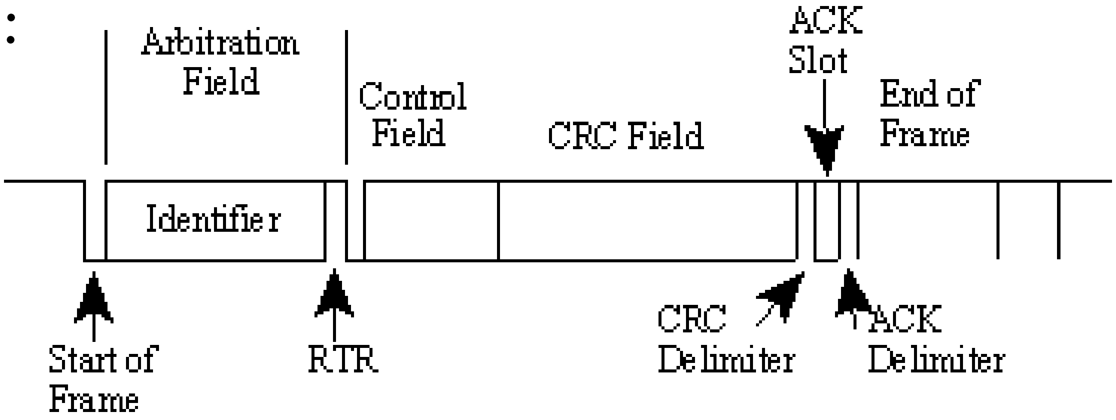


- CAN 2.0B ("extended CAN" 29-bit ID) Data Frame.



Remote Frame (contd.)

- Generally data transmission is performed on an autonomous basis with the data source node (e.g., a sensor) sending out a Data Frame. It is also possible, however, for a destination node to request the data from the source by sending a Remote Frame.
- Sometimes it is claimed that the node responding to the Remote Frame is starting its transmission as soon as the identifier is recognized.
- A Remote Frame (2.0A type):



Error Frame

- The error frame consists of two different fields:
  - The first field is given by the superposition of ERROR FLAGS (6–12 dominant/recessive bits) contributed from different stations.
  - The following second field is the ERROR DELIMITER (8 recessive bits).

Overload Frame

Summary: "I'm a very busy little 82526 device, could you please wait for a moment?"

- The Overload Frame is mentioned here just for completeness. It is very similar to the Error Frame with regard to the format and it is transmitted by a node that becomes too busy. The Overload Frame is not used very often, as today's CAN controllers are clever enough not to use it. In fact, the only controller that will generate Overload Frames is the now obsolete 82526

Thank You !!