Contents

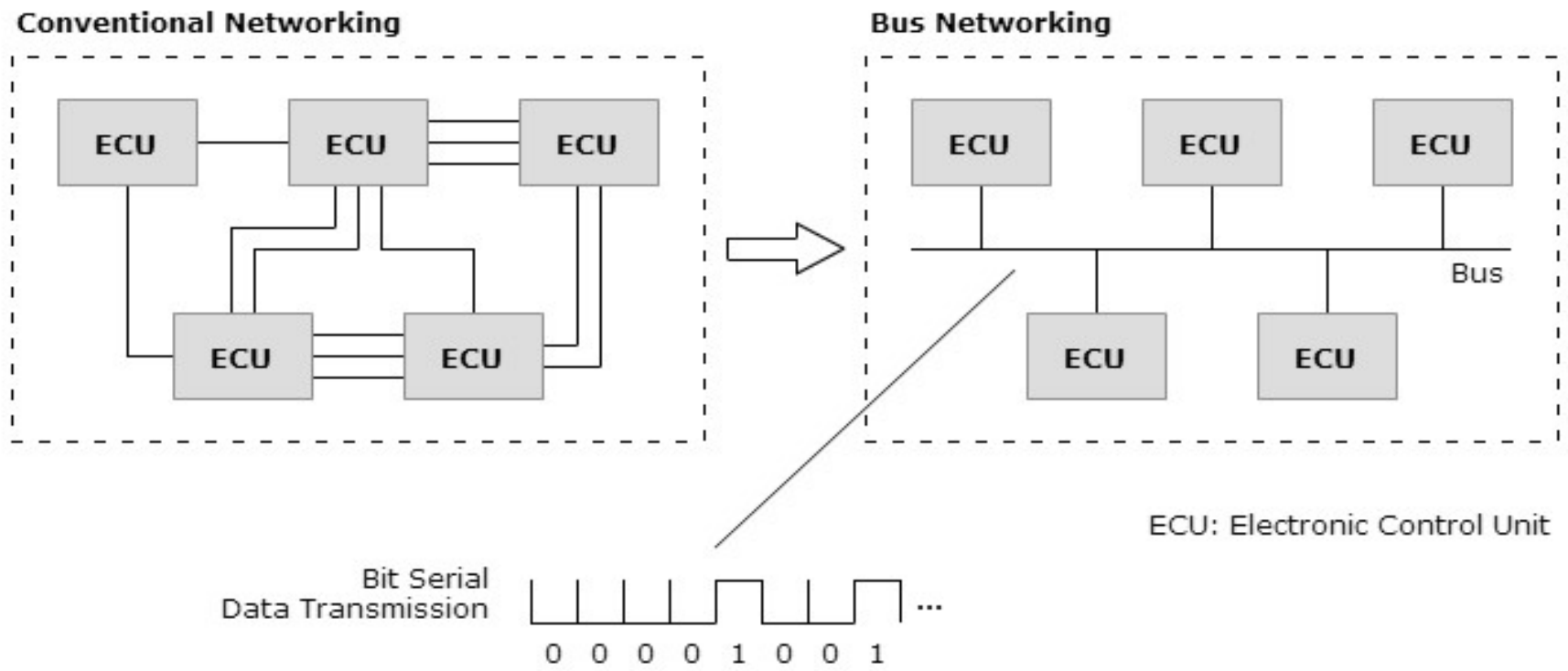
Why CAN?

- Motivation
- Bus Networking
- Communication Model
- Three Layer Model
- Architecture of Serial Bus System

Motivation

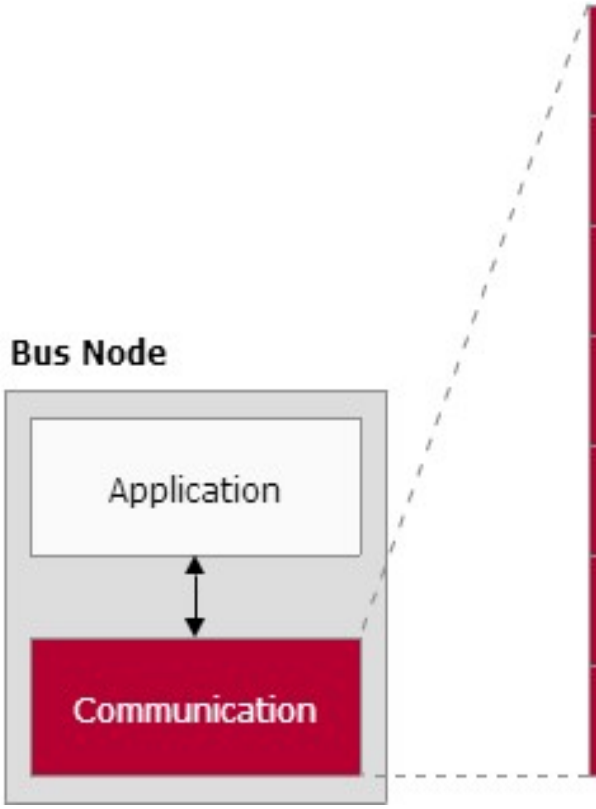
- The recent history of the automobile is characterized by intensive **electronification**.
- Over the course of time, electronic functions found their way into the automobile that could not be implemented without highly active **data exchange** between electronic ECUs.
- Electronic systems were requiring more and more intensive networking to implement them, which drove **wiring expense** sky high.
- A way out of this dilemma soon appeared in the form of bit-serial exchange of data via a **communication channel (Bus)** jointly used by a number of electronic control units.

Bus Networking



Communication Model/OSI Model

The OSI communication model (Open System Interconnection) published by ISO (International Standardization Organization) in 1983 is a **reference architecture** for implementing CAN communication.

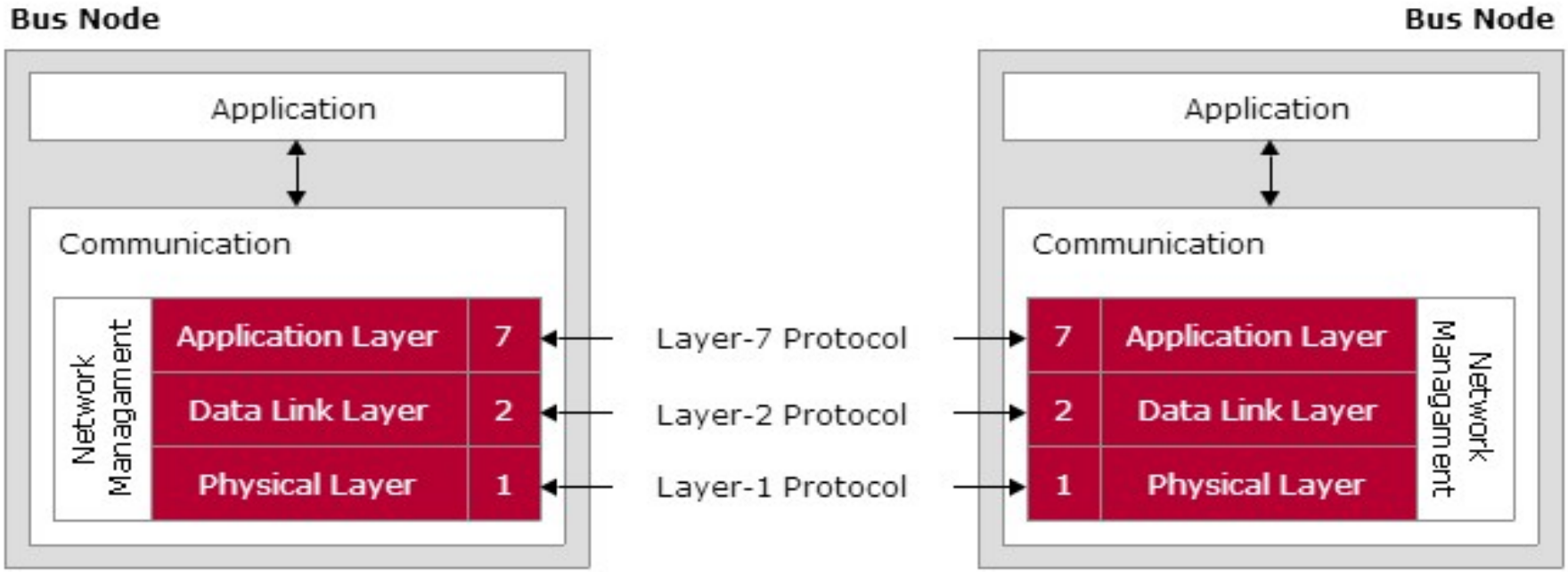


7	Application Layer	access to the communication system, decoupling application from communication
6	Presentation Layer	semantics, compression, encryption
5	Session Layer	handling of longer sessions, definition of synchronization points
4	Transport Layer	connecting and disconnecting, segmenting, sequencing, assembling
3	Network Layer	routing (selection of transmission path between two bus nodes)
2	Data Link Layer	data protection, message building, bus access
1	Physical Layer	physical bus interface, plug connector, transmission medium, line coding

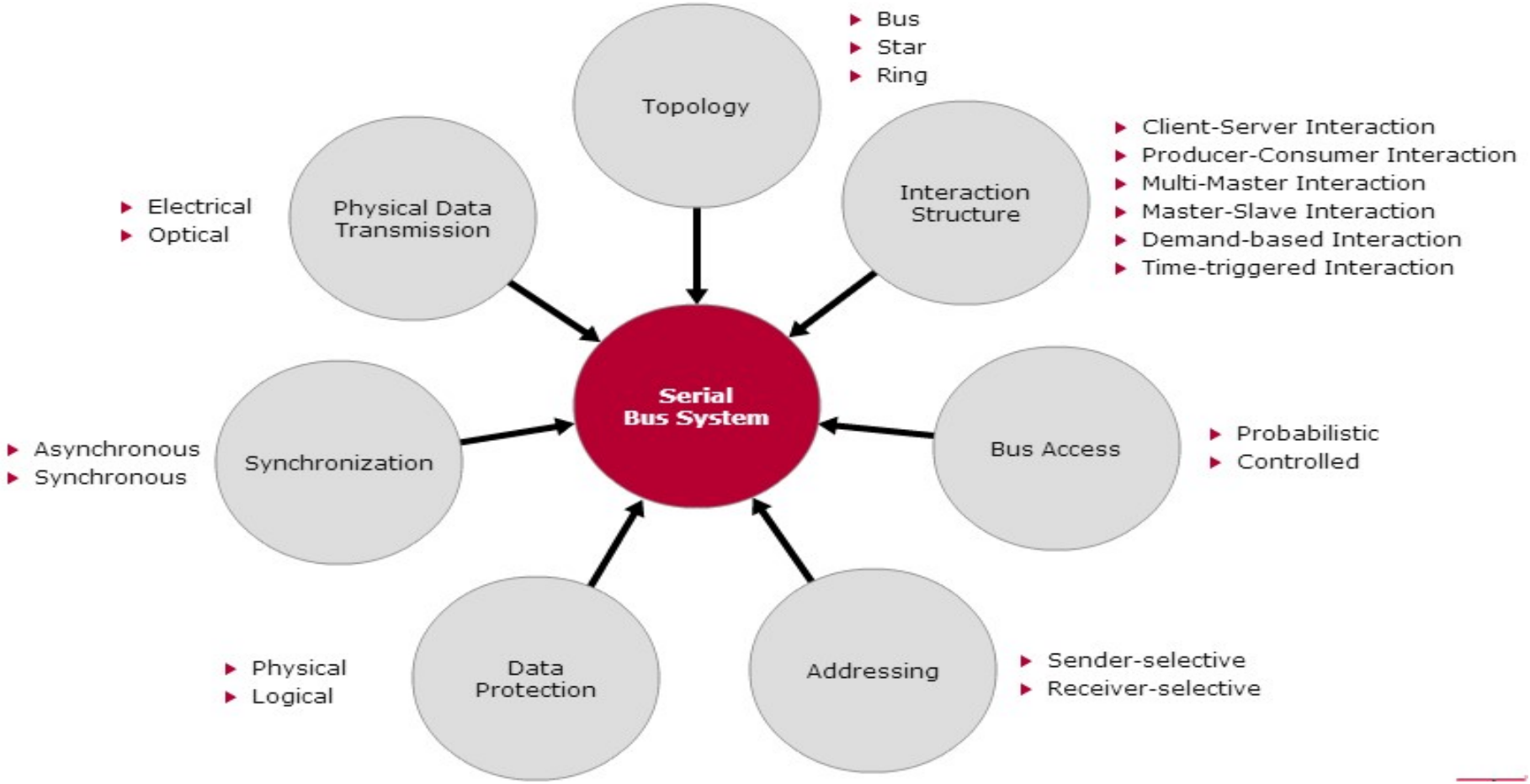
Three Layer Model

- For data exchange between electronic control units in the motor vehicle, not all of the communication functions, defined in the ISO/OSI model, are necessary.

- Essentially, only the two lowermost layers are relevant to serial data communication in the motor vehicle. These layers are the **Data Link Layer** and the **Physical Layer**.



Architecture of Serial Bus System



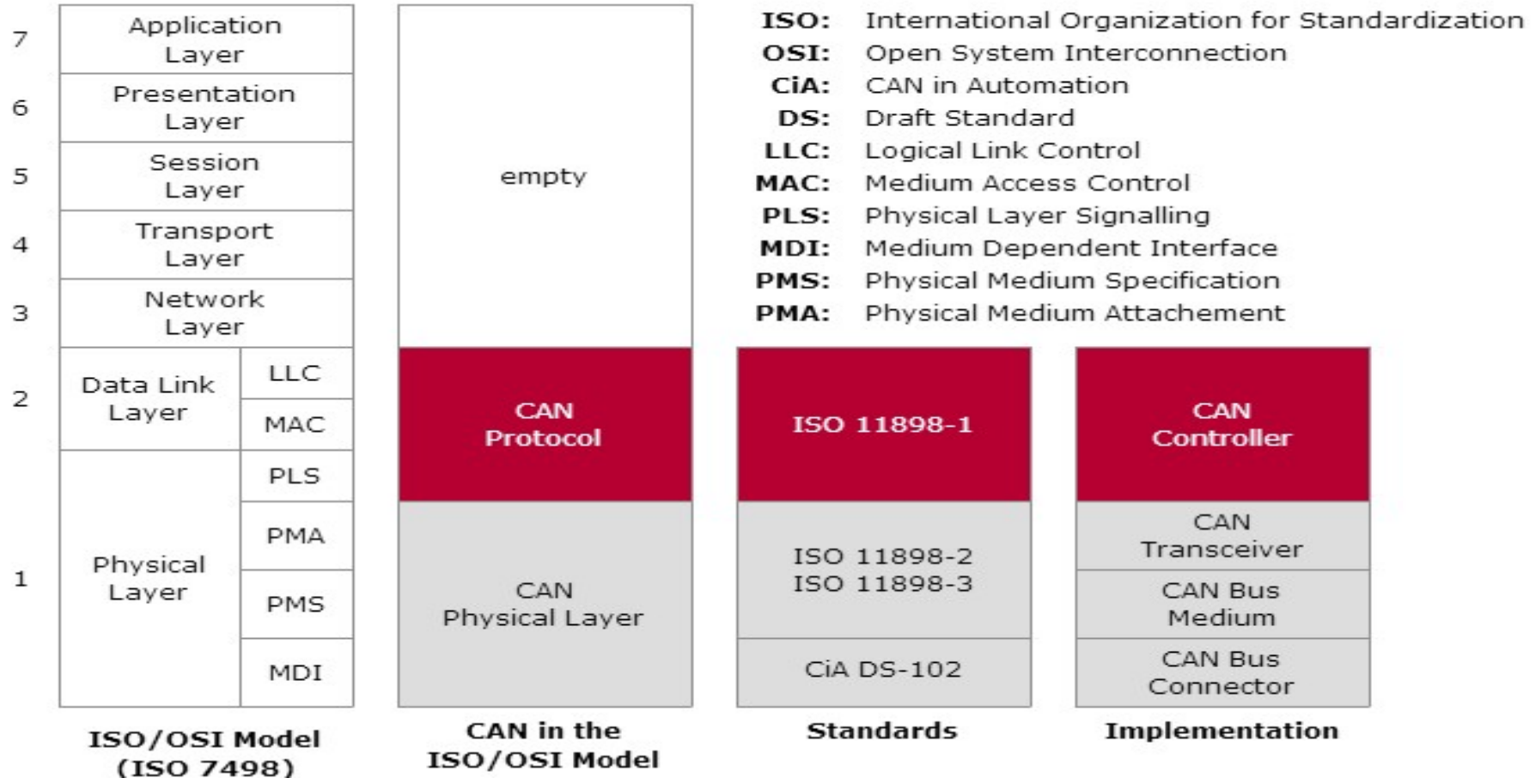
CAN Introduction

- CAN Standard
- CAN Network
- CAN Communication Principle
- CAN Bus Levels

CAN Standard

- CAN has been standardized since 1993 and is available as an ISO standard (International Standardization Organization): **ISO 11898**.
- While it initially consisted of three parts, today it has five parts.
- The **first** part describes the **event-driven communication protocol**.
- A time-triggered extension can be found in the **fourth** part.
- The **second** and **third** parts cover information on the bus interface and physical data transmission: A distinction is made here between the **High-Speed variant** (data rates up to 1 MBit/s) and **Low-Speed variant** (data rates up to 125 KBit/s).
- The **fifth** part describes the behavior of a CAN node in the High-Speed network in “Low Power Mode”.
- A maximum network extension of about 40 meters is allowed. At the ends of the CAN network, **bus termination resistors** contribute to preventing transient phenomena (reflections).

CAN Standard (Continued)

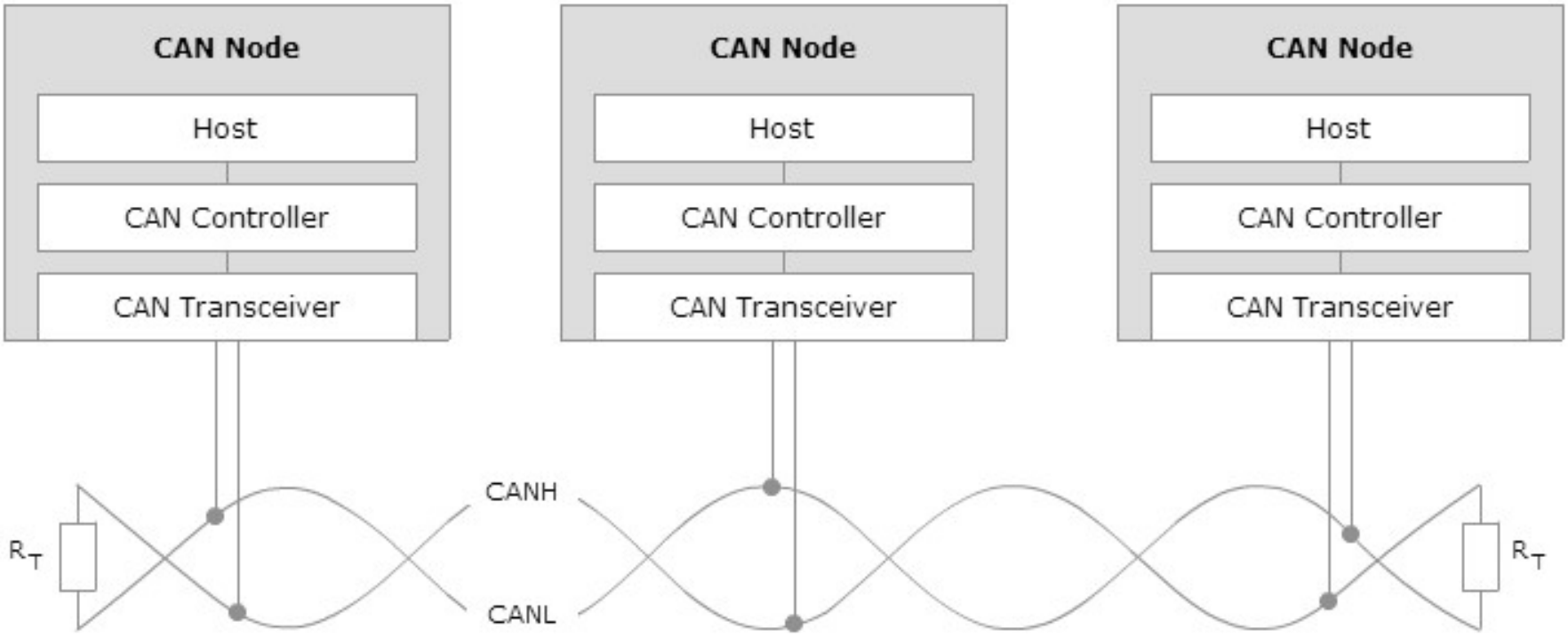


ISO: International Organization for Standardization
OSI: Open System Interconnection
CiA: CAN in Automation
DS: Draft Standard
LLC: Logical Link Control
MAC: Medium Access Control
PLS: Physical Layer Signalling
MDI: Medium Dependent Interface
PMS: Physical Medium Specification
PMA: Physical Medium Attachement

CAN Network

- A CAN network is a system network made up of **CAN nodes** (electronic control units with a CAN interface), which exchange data with one another over their individual **CAN interfaces** and a **transmission medium** (**CAN bus**) that interconnects all of the CAN interfaces.
- A CAN interface is made up of two parts: the communication software and communication hardware.
- While the communication software is made up of higher level communication services, the fundamental communication functions are implemented in hardware.
- The **CAN controller** provides for uniform handling of the **CAN communication protocols**.
- The **CAN transceiver** serves to interface the CAN controller to the CAN bus.
- The transmission medium that is generally used is a **twisted pair line**.
- The **symmetrical signal transmission** that occurs over this line is very insensitive to external interference.

CAN Network Diagram



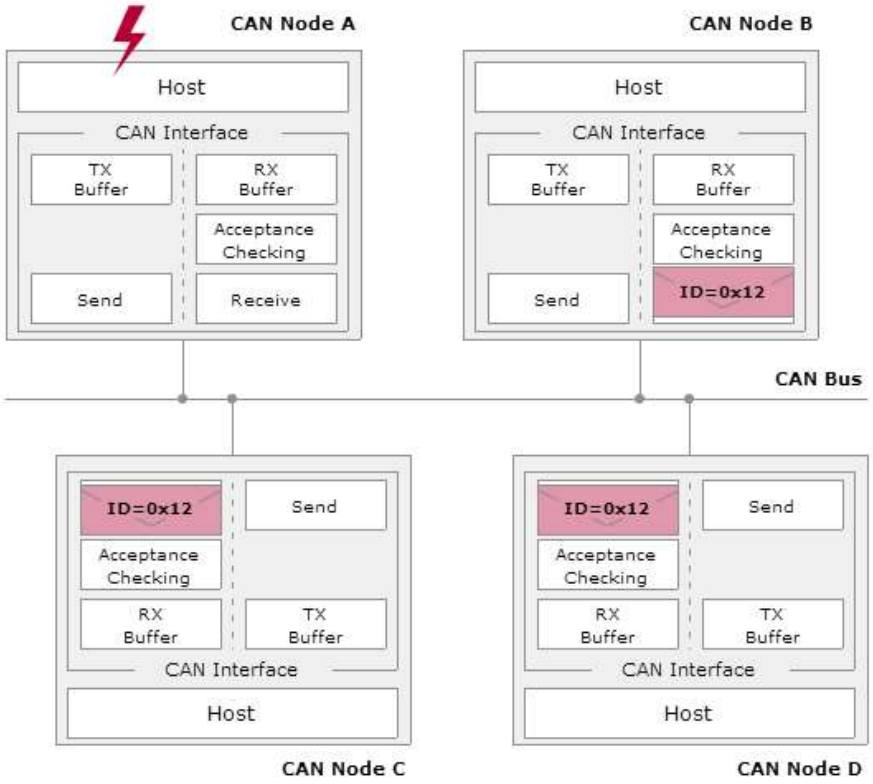
CANH: CAN High Line
CANL: CAN Low Line
R_T: Termination

CAN Communication Principle

- In a CAN network, the transmitted data frames and their sequence are not a function of the progression of time, rather they depend on the occurrence of special events.
- In principle, each CAN node is authorized to access the CAN bus immediately after an event occurs. In conjunction with the relatively short message length of a maximum of 130 bits in standard format, and the high data transmission rate of up to 1 MBit/s, the method enables **quick reactions to asynchronous processes**.
- This is an important prerequisite for **real-time data transmission capability** in the low milliseconds range, which is primarily demanded by applications in the powertrain and chassis areas.

- To guarantee real-time communication despite **random bus access**, bus access is based on the **CSMA/CA method** (**Carrier Sense Multiple Access/Collision Avoidance**).
- First, the CSMA/CA method ensures that CAN nodes wishing to send do not access the CAN bus until it is available.
- Second, in simultaneous bus accesses occur, the CSMA/CA method ensures that the CAN node with the highest priority data frame prevails.

CAN Communication Principle



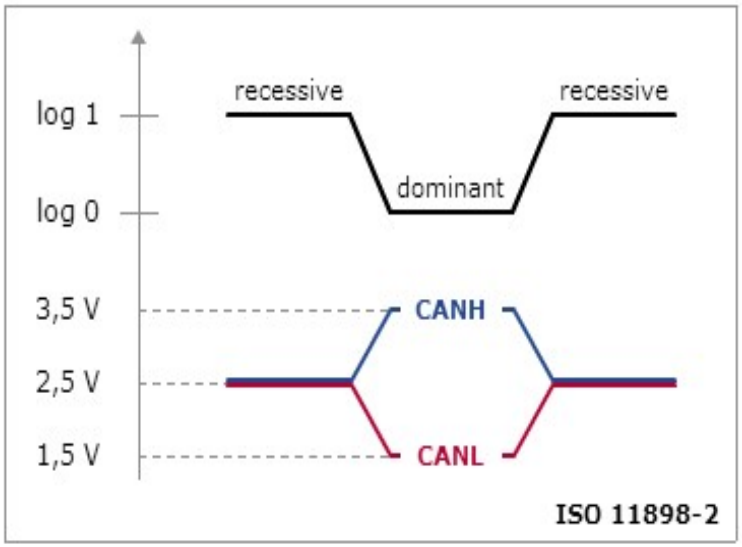
Communication Matrix

Data Frame	CAN Node A	CAN Node B	CAN Node C	CAN Node D
ID=0x12	Sender	Receiver		
ID=0x34		Sender	Receiver	Receiver
ID=0x52		Receiver		Sender
ID=0x67	Receiver	Receiver	Sender	Receiver
ID=0xB4	Receiver		Sender	
ID=0x3A5	Sender	Receiver	Receiver	Receiver

CAN Bus Levels

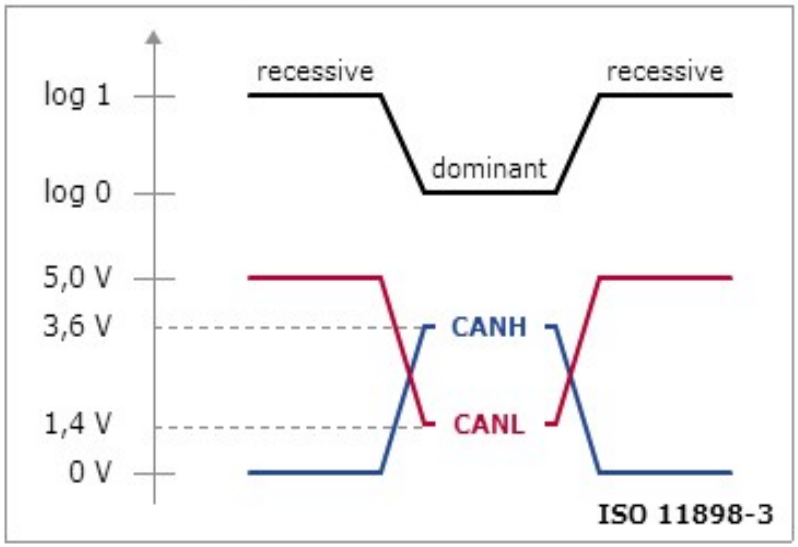
CAN Communication

High-Speed CAN Bus Levels



CAN Communication

Low-Speed CAN Bus Levels



- ISO 11898-2 High Speed CAN assigns logical “1” to a typical differential voltage of 0 Volt. The logical “0” is assigned with a typical differential voltage of 2 Volt.
- ISO 11898-3 Low Speed CAN assigns a typical differential voltage of 5 Volt to logical “1”, and a typical differential voltage of 2 Volt corresponds to logical “0”.

CAN Frames Format

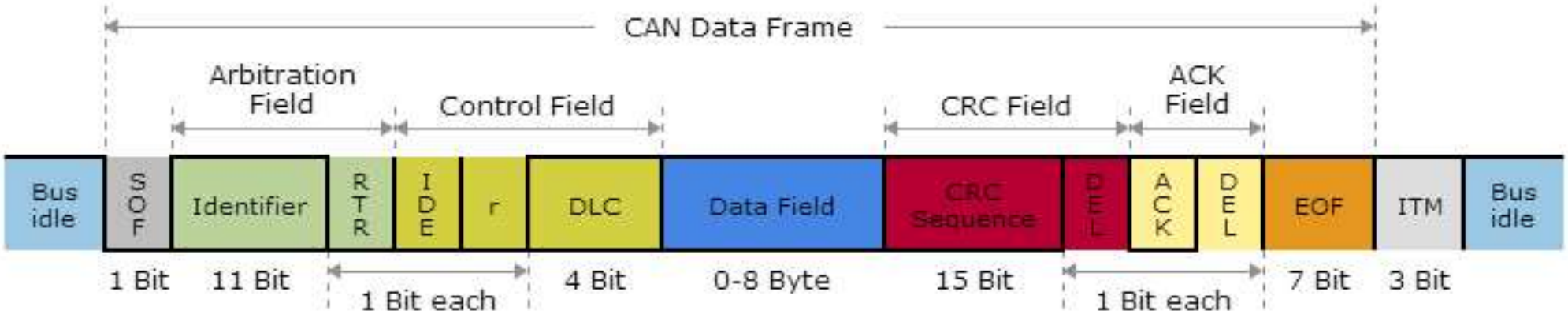
- CAN Data Frame
- CAN Remote Frame
- CAN Error Frame
- CAN Standard and Extended Frame
- CAN Frame – Acknowledgement
- CAN Frame – Bit Stuffing

Type and Role of Each Frame

Frame	Roles of frame	User settings
Data frame	This frame is used by the transmit unit to send a message to the receive unit.	Necessary
Remote frame	This frame is used by the receive unit to request transmission of a message that has the same ID from the transmit unit.	Necessary
Error frame	When an error is detected, this frame is used to notify other units of the detected error.	Unnecessary
Overload frame	This frame is used by the receive unit to notify that it has not been prepared to receive frames yet.	Unnecessary
Interframe space	This frame is used to separate a data or remote frame from a preceding frame.	Unnecessary

CAN Data Frame

- Transmission of a data frame begins with a start symbol (**Start of Frame: SOF**). It is used by the CAN nodes to synchronize to the sender.
- During transmission, the CAN nodes maintain accurate timing by evaluating each edge transition and adjusting their clocks as necessary.
- The **bit stuffing method** guarantees that an edge transition occurs after five homogeneous bits at most.
- An end symbol (**End Of Frame: EOF**) marks the end of a data frame.



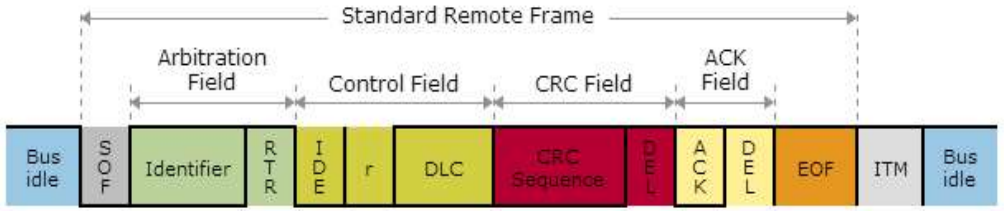
CAN Data Transmission

- Data in the CAN network is transmitted using message frames, the so-called **CAN data frame**.
- Useful data up to **eight bytes** in length can be transmitted in the data field of a data frame.
- The precise number of useful bytes is indicated by the **DLC** (**Data Length Code**).
- Every data frame is available for receiving by every CAN node **(broadcasting)**.
- The useful data is protected by the **CRC method** (**Cyclic Redundancy Check**): The sender appends a **checksum** (**CRC sequence**) to the useful data, which is evaluated by the receiver by applying the CRC algorithm.
- Depending on the result of the evaluation, the receiver acknowledges either positively or negatively in the **ACK slot** that follows the CRC sequence.
- Each data frame is identified by an **identifier**. In **standard format** it is made up of 11 bits, in **extended format** it is 29 bits.

Remote Frame and Error Frame

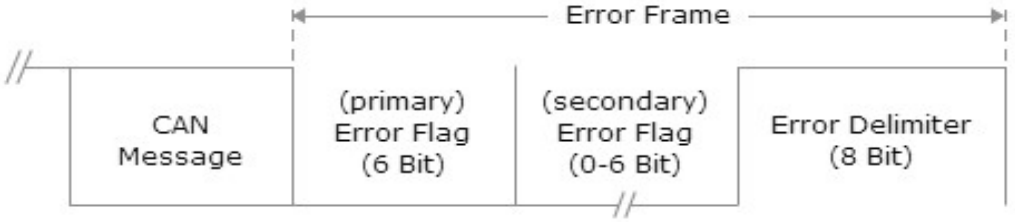
CAN Framing

Standard Remote Frame



CAN Framing

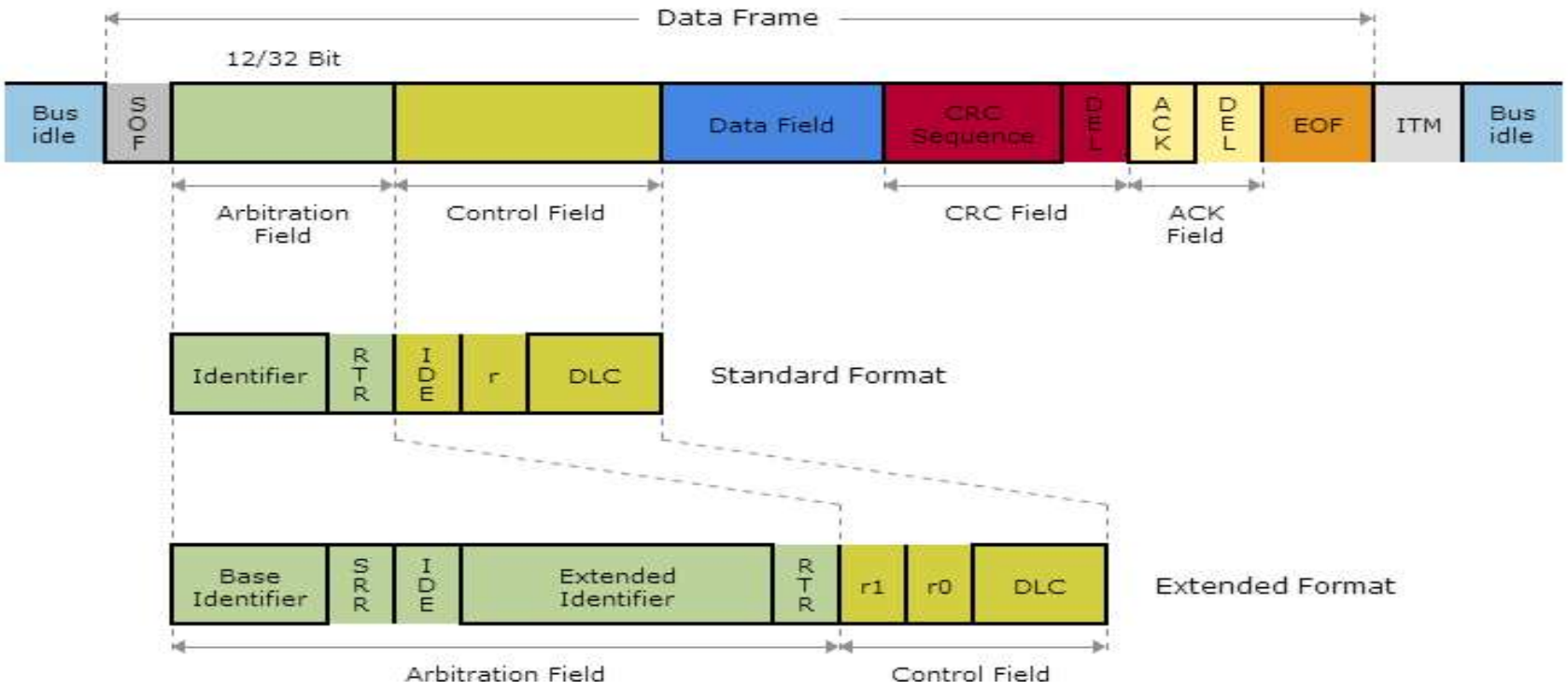
Error Frame



Standard and Extended Frame

CAN Framing

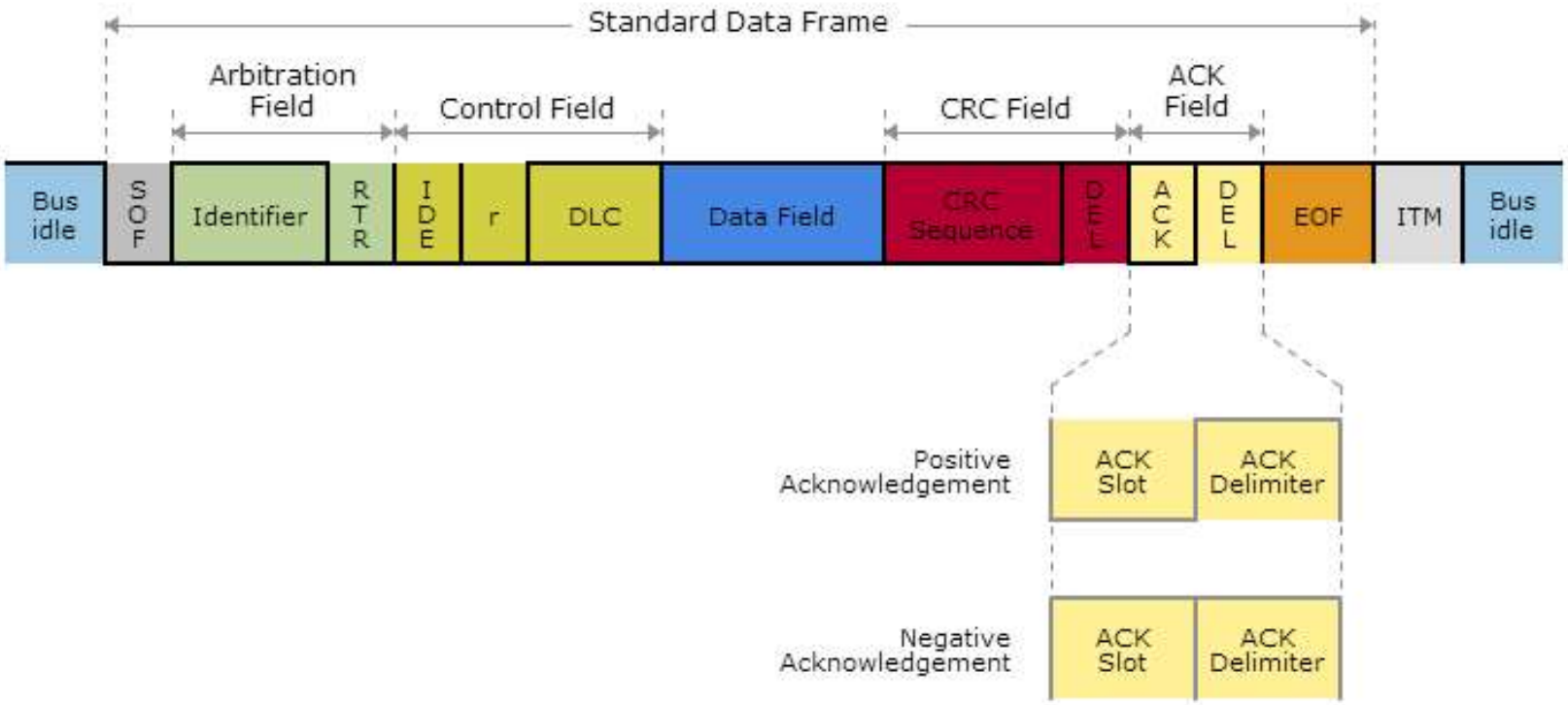
Data Frame in Standard and Extended Format



CAN Framing - Acknowledgement

CAN Framing

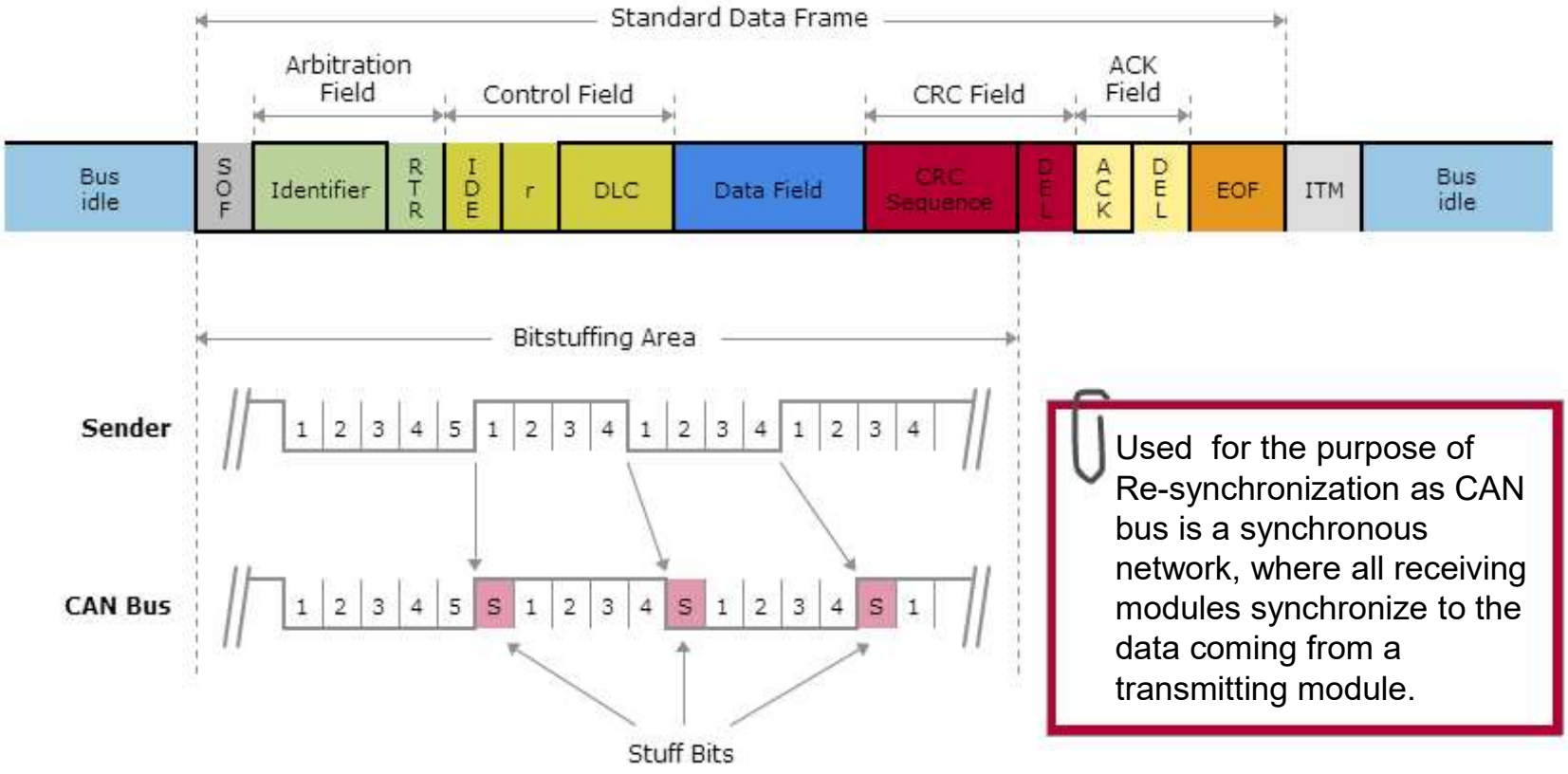
Acknowledgement



CAN Framing – Bit Stuffing

CAN Framing

Bit Stuffing



Used for the purpose of Re-synchronization as CAN bus is a synchronous network, where all receiving modules synchronize to the data coming from a transmitting module.

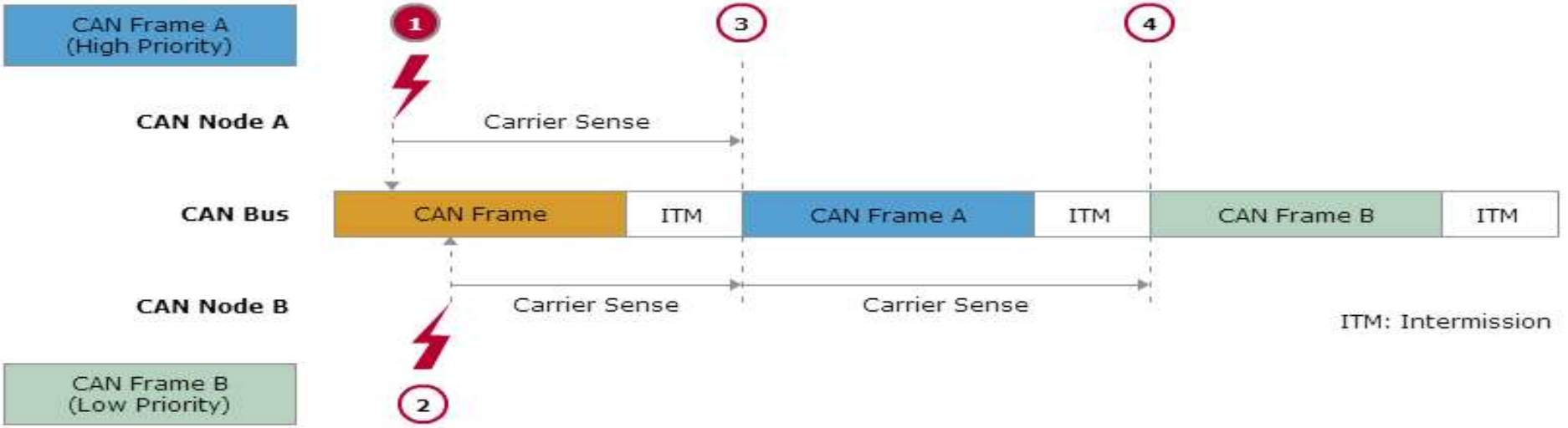
CAN Bus Access

- Bus Access in CAN Network
- Bitwise Bus Arbitration
- Prioritization

CAN Bus Access

- ISO 11898-1 defines a **multi-master architecture** to assure high availability and event-driven data transmission.
- Each node in the CAN network has the right to access the CAN bus without requiring permission and without prior coordination with other CAN nodes.
- Although bus access based on an **event-driven** approach enables very quick reactions to events, there is the inherent risk that several CAN nodes might want to access the CAN bus at the same time, which would lead to undesirable overlaps of data on the CAN bus.
- To preserve the communication system's **real-time capability**, ISO 11898-1 provides for a bus access that guarantees nondestructive data transport. The so-called CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) method is used here.
- The CSMA/CA method ensures that CAN nodes wishing to send do not access the CAN bus until it is available.
- In case of simultaneous bus access, the CSMA/CA method based on **bitwise bus arbitration** ensures that the highest priority CAN message among the CAN nodes prevails.
- In principle, the higher the priority of a CAN message the sooner it can be transmitted on the CAN bus. In case of poor system design, low priority CAN messages even run the risk of never being transmitted.

Bus Access in the CAN network



Event 1

CAN node A wishes to access the CAN bus to transmit its high-priority CAN message. However, it cannot do so, because the CAN bus is busy.

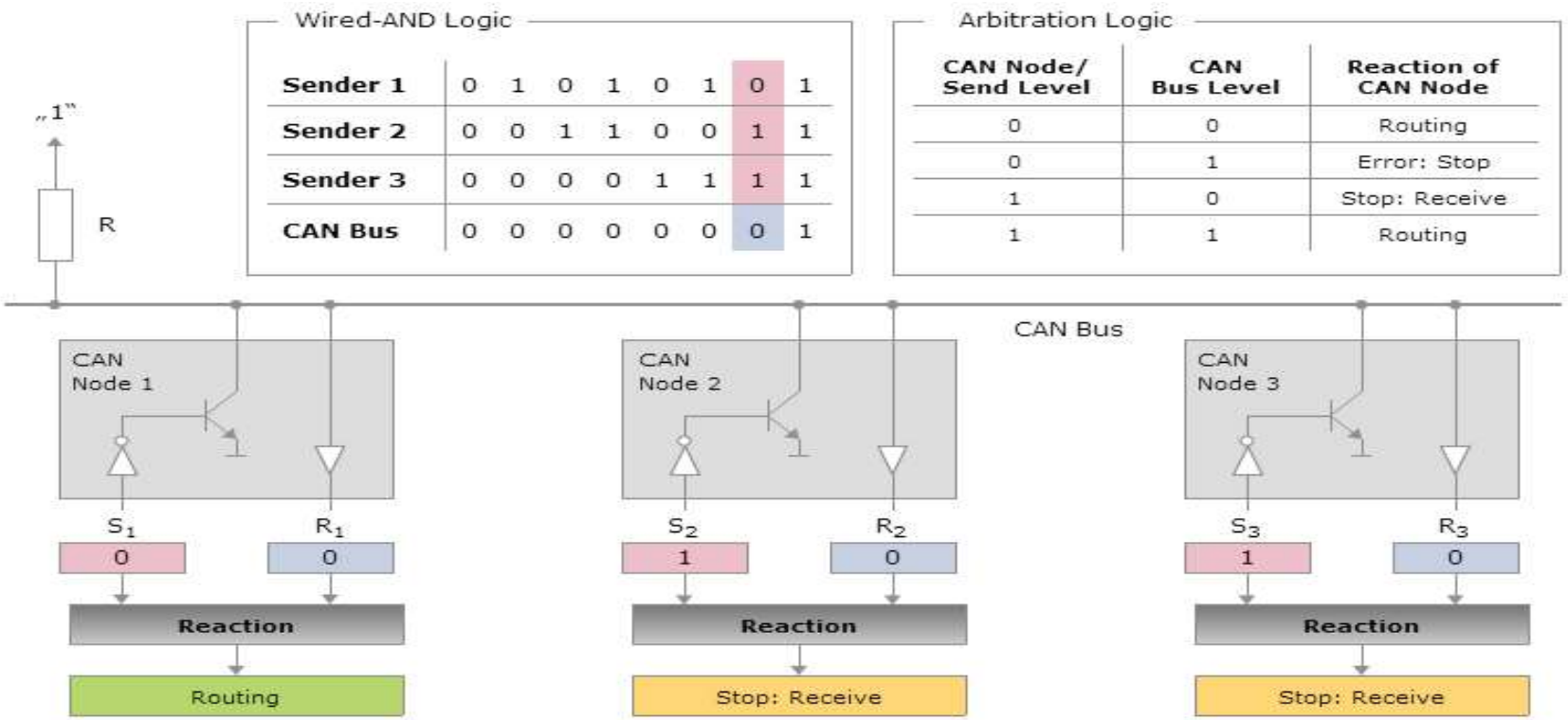
Effect

CAN node A cannot access the CAN bus until it is available. This occurs after the ITM expires. CAN node A monitors the CAN bus (Carrier Sense) to detect this.

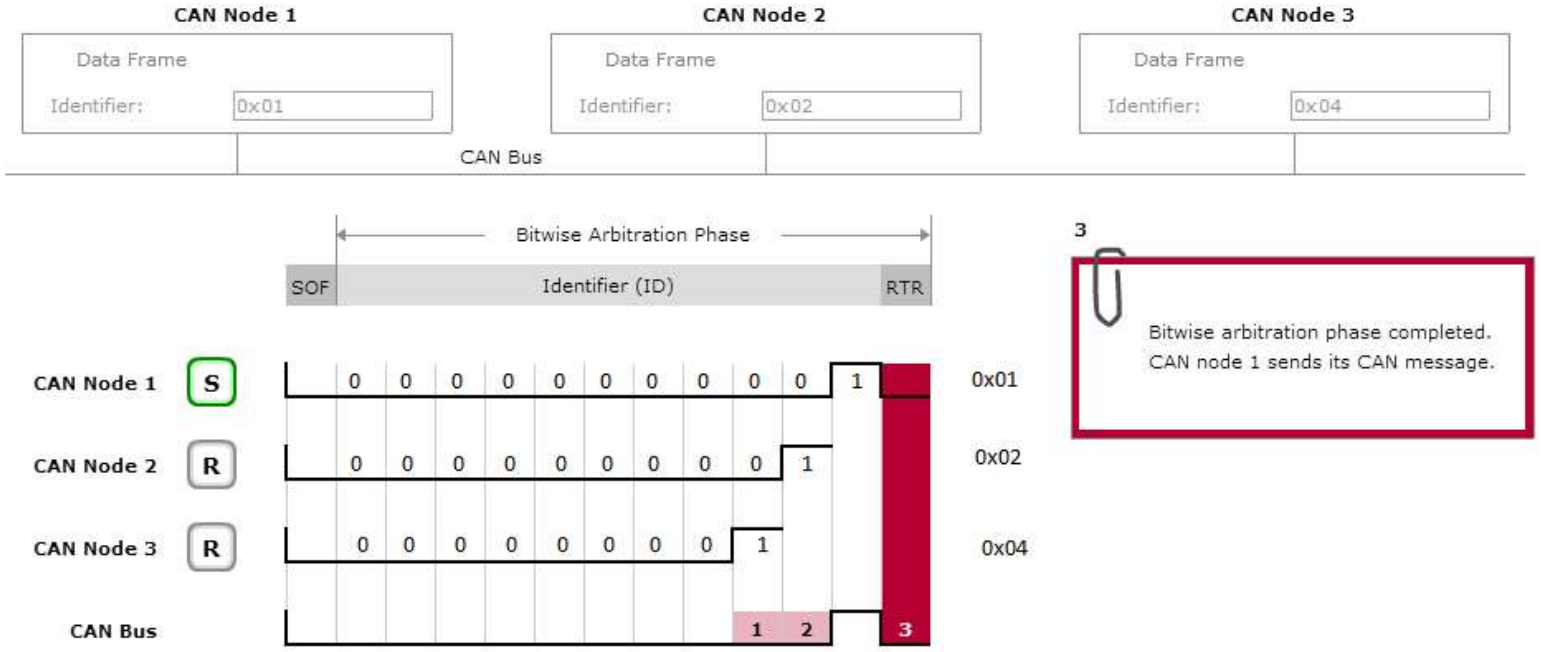
Bitwise Bus Arbitration

- The key component of the bus access method defined by ISO 11898-1 is bitwise bus arbitration. It prevents **collisions** from occurring despite simultaneous bus access.
- After network-wide synchronization, all CAN nodes wishing to send place their identifier of the CAN message bitwise onto the CAN bus, from most significant to least significant bit.
- In this process, the **wired-AND bus logic** upon which the CAN network is based ensures that a clear and distinct bus level results on the bus.
- At the end of the **arbitration phase**, the CAN node transmitting the CAN message with the **lowest ID** gets authorization to send. CAN nodes with lower priority messages switch to the receiving state, later they access the CAN bus for another sending attempt as soon as it is available again.

Wired AND Logic and Arbitration Logic



Example – Bitwise Bus Arbitration



Prioritization

- The **priorities of the CAN messages** are decisive in obtaining bus access in the CAN network. They are encoded via the identifier which is transmitted bitwise from the most significant to the least significant bit.

- **Wired-AND bus logic** and **arbitration logic** ensure that the priority of the CAN message increases with decreasing identifier value: The smaller an identifier is, the higher the priority of the CAN message.

- If the bus load is not too high, this type of random, nondestructive and priority-controlled bus access provides for **fair and very quick bus access**.

- Nonetheless, it must be taken into account that increasing bus load primarily causes delays in lower-priority CAN messages to grow.

- This could impair the **real-time capability** of the CAN communication system. Therefore, in designing the system the priorities of CAN messages should be derived from the urgency of the signals they will transport.

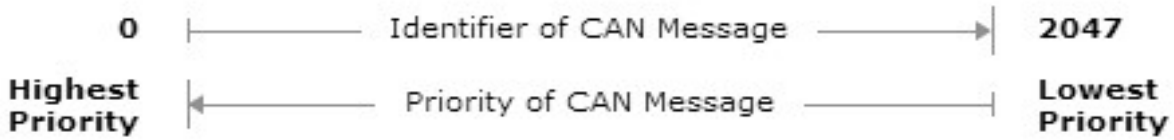
Prioritization of CAN Message

Wired-AND Logic

Sender 1	0	1	0	1	0	1	0	1
Sender 2	0	0	1	1	0	0	1	1
Sender 3	0	0	0	0	1	1	1	1
CAN Bus	0	0	0	0	0	0	0	1

Arbitration Logic

CAN Node/ Send Level	CAN Bus Level	Reaction of CAN node
0	0	Routing
0	1	Error: Stop
1	0	Stop: Receiving
1	1	Routing

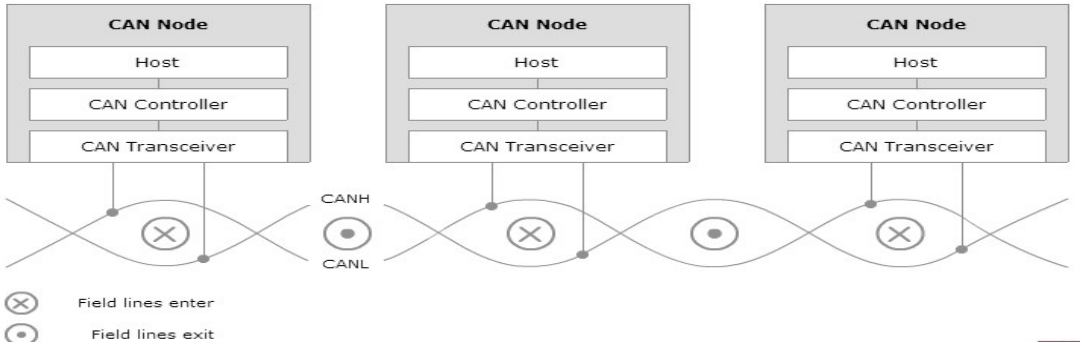


CAN Data Protection

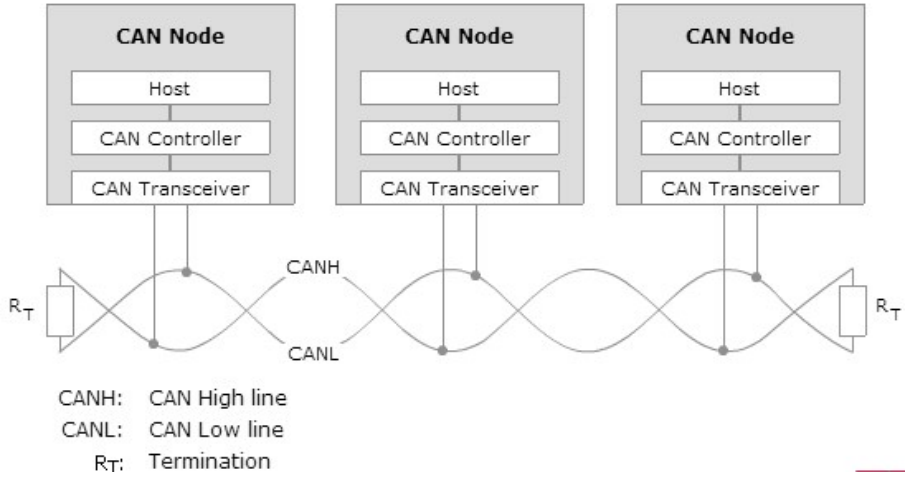
- Physical Error Protection
- Logical Error Detection
- Logical Error Handling
- Error Tracking

CAN Data Protection

- Twisted Pair



- Termination



Logical Error Detection- Bit Monitoring

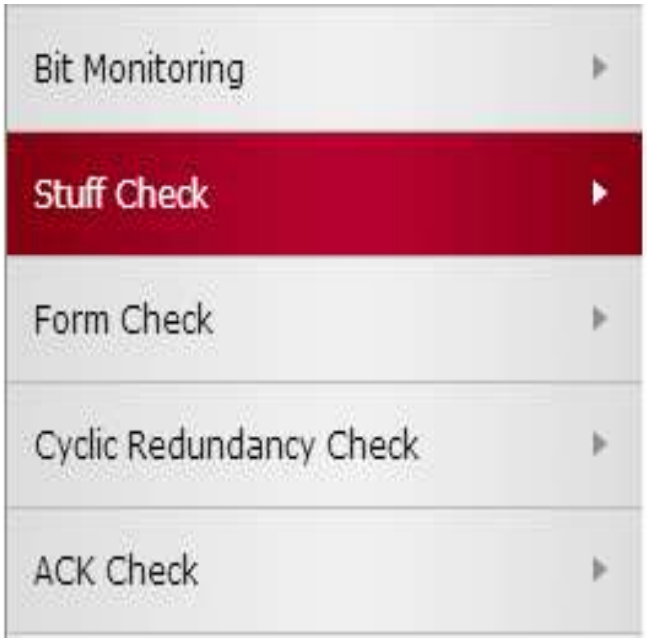
- To detect corrupted messages, the CAN protocol defines **five mechanisms**: bit monitoring, monitoring of the message format (Form Check), monitoring of the bit coding (Stuff Check), evaluation of the acknowledgement (ACK Check) and verifying the checksum (Cyclic Redundancy Check).

Bit Monitoring

- Sender Task
- Compares every bit placed on the CAN bus with the actual bus level
- Discrepancy indicates a bit monitoring error and results in error handling
- Not applied to the Arbitration Field or the ACK Slot

Logical Error Detection- Stuff Check

- The **stuff check** serves to check the bit stream. The CAN protocol specifies that the sender must transmit a complementary bit after five homogeneous bits — for synchronization purposes. There is a stuffing error if more than five homogeneous contiguous bits are received.

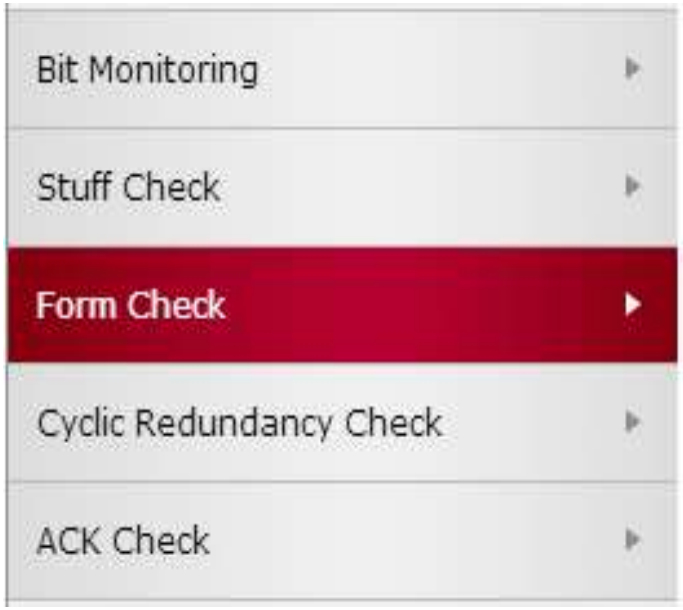


Stuff Check

- Receiver Task
- Compares arriving bit stream for a sequence of six homogeneous bits
- Comparison in the so-called bit stuffing area (from SOF up to and including CRC sequence)
- Detection of a sixth homogeneous bit indicates a bit stuffing error and results in error handling

Logical Error Detection- Form Check

- The **form check** serves to check the format of a CAN message. Each CAN message always exhibits the same bit sequences at certain positions. They are the CRC delimiter, ACK delimiter and EOF. Senders always transmit these message components recessively. A format error exists if a receiver detects a dominant bus level within one of these message components in the Form Check.

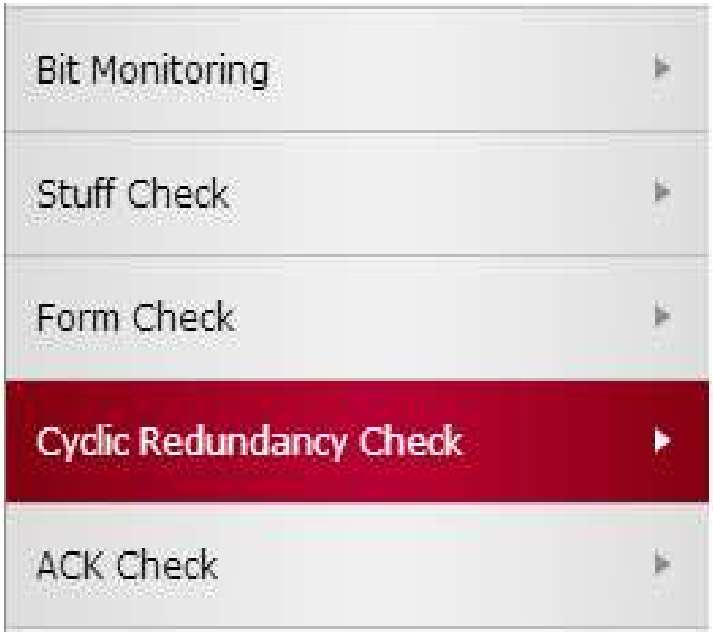


Form Check

- Receiver Task
- Comparison of the arriving bit stream with the message format
- Detection of a dominant delimiter bit (CRC delimiter, ACK delimiter) or a dominant bit within EOF indicates a format error and results in error handling

Logical Error Detection- CRC Check

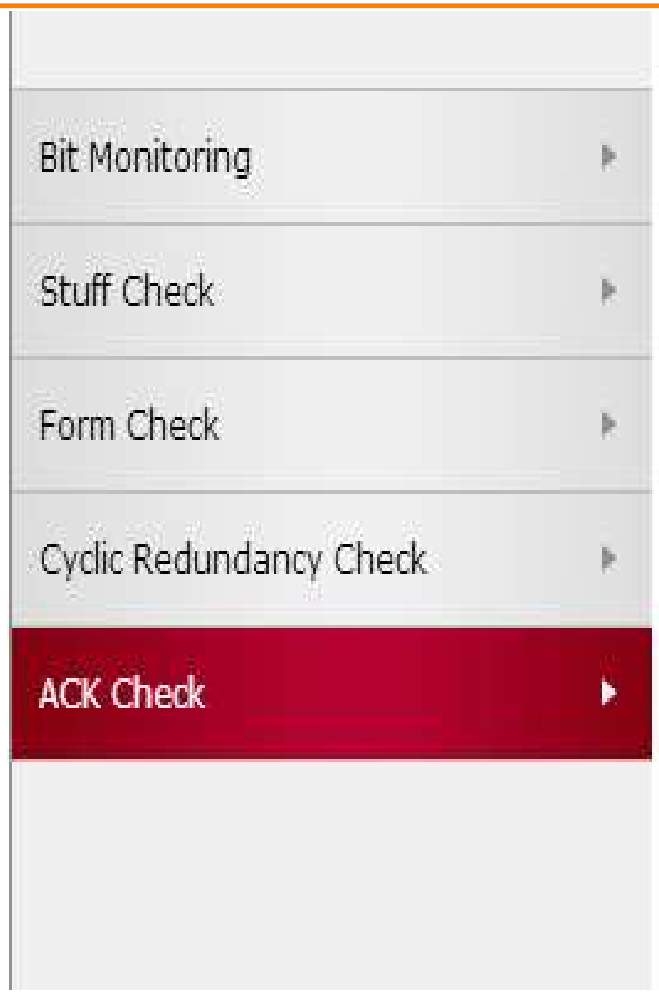
- In the **cyclic redundancy check** (CRC) the polynomial R(x) associated with the arriving data or remote frame should equal a multiple of the generator polynomial G(x) specified by ISO 11898-1. If this is not the case (CRC error), then the data or remote frame was corrupted during its transmission.



Cyclic Redundancy Check

- Receiver Task
- Utilizes the arriving bit stream and generator polynomial for the Cyclic Redundancy Check defined in ISO 11898-1
- Detection of a CRC error results in error handling

Logical Error Detection- ACK Check

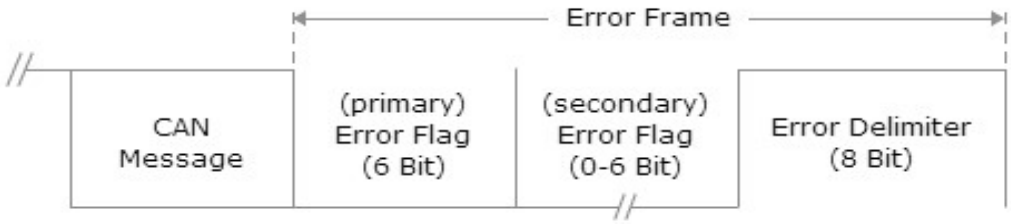


ACK Check

- Sender Task
- Comparison of the recessive bit placed on the CAN bus with the bus level in the ACK slot
- Acknowledge error (ACK error) is detected if the recessive level placed by the sender is not overwritten
- Detection of an ACK error results in error handling

Logical Error Handling

- The CAN protocol prescribes that if the error-detecting CAN node is experiencing a local disturbance, it must inform all CAN nodes connected to the CAN network.
- The error-detecting CAN node transmits an error signal (**error flag**) for this purpose, which is made up of six dominant bits. This is an intentional violation of the bit stuffing rule, and it generates a **bit stuffing error**.
- Transmission of an error flag ensures that all other CAN nodes will also transmit an error flag (secondary error flag) and thereby also terminate the regular data transmission just like the sender of the primary error flag.
- Transmission of an error flag is always terminated by an **error delimiter**. This consists of eight recessive bits.



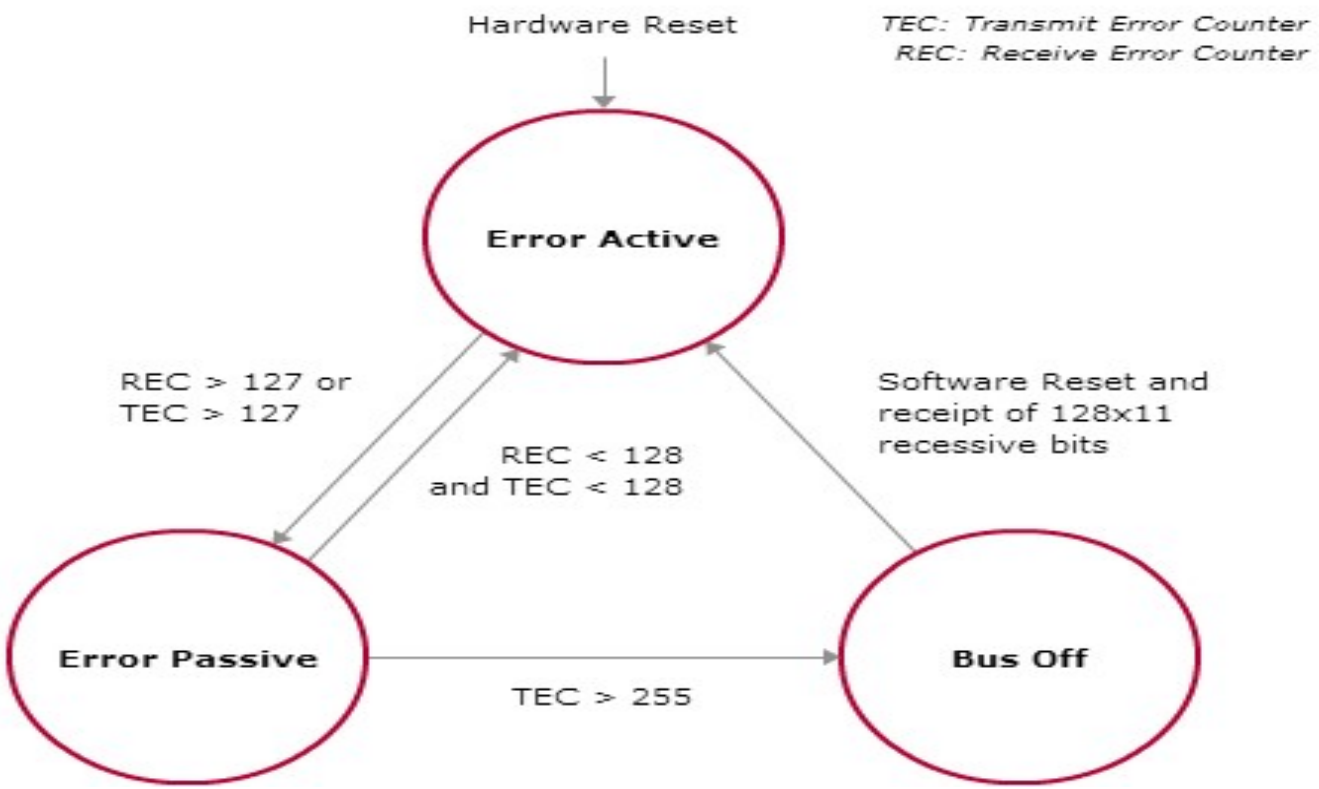
Error Tracking

- To assure network-wide **data consistency**, each node in a CAN network has the right to terminate any CAN message interpreted as faulty. This also applies to a CAN node that erroneously interprets correct CAN messages as faulty.
- To prevent jamming up the transmission medium, the CAN protocol specifies error tracking that allows CAN nodes to distinguish between occasionally occurring disturbances and persistent ones.
- Consequently, each CAN controller has a **TEC (Transmit Error Counter)** and a **REC (Receive Error Counter)**. In case of successful transmission of a data or remote frame, the relevant error counter is decremented (TEC=TEC-1; REC=REC-1).
- Detection and subsequent transmission of a error flag causes the relevant error counter to be incremented according to certain rules.
- For the sender the following rule applies: TEC=TEC+8.
- Error-detecting receivers initially increment their REC by one unit (REC=REC+1).
- For the error causing receiver: REC=REC+8.

Error Tracking

Data Protection in the CAN Network

Error Tracking



Queries?

Thank you