# CAN bus: Reliable Communication in Embedded Networks

## Introduction to CAN Bus

Controller Area Network (CAN) bus is a robust and widely used communication protocol in the realm of embedded systems and automotive applications. Initially developed by Bosch in the mid-1980s, CAN bus was specifically designed to allow microcontrollers and devices within a vehicle to communicate with one another without needing a host computer.

CAN bus facilitates reliable and efficient communication among various electronic control units (ECUs) in vehicles, enabling them to transmit data and control signals swiftly and securely. Its design prioritizes high reliability, real-time capabilities, and fault tolerance, making it a fundamental technology in modern vehicles and many other industrial applications.

The protocol operates on a multi-master serial bus structure, utilizing twisted pair wiring to form a network where multiple ECUs can simultaneously send and receive messages. This communication method allows for a robust and decentralized system, where different components can exchange critical information, such as engine management data, safety system signals, and more, in a highly efficient manner.

## What is CAN Bus?

•Controller Area Network (CAN) bus: a robust communication protocol for embedded systems and automotive applications.

## Key Features:

•Facilitates reliable, real-time communication among electronic control units (ECUs) .

•Multi-master serial bus structure using twisted pair wiring.

•Resilient to electromagnetic interference, ensuring communication integrity.

## Applications:

•Widely used in automobiles for transmitting critical data between ECUs.

•Extends beyond automotive: employed in aerospace, medical devices, industrial automation, and more.

## Significance:

•Fundamental technology enabling efficient and secure data exchange in interconnected systems.

# CAN Bus Architecture:

•**Controller and Interface Controller:** Each device on the CAN network has a controller that manages communication and an interface controller that physically connects to the CAN bus.

•**Twisted Pair Wiring:** It uses twisted pair cables for data transmission, with two wires (CAN-High and CAN-Low) used for sending and receiving information.

**Communication Protocol:**

• **Standardized Frame:** Data is sent in packets called "frames," containing a unique identifier, data, and control bits for error detection and correction.

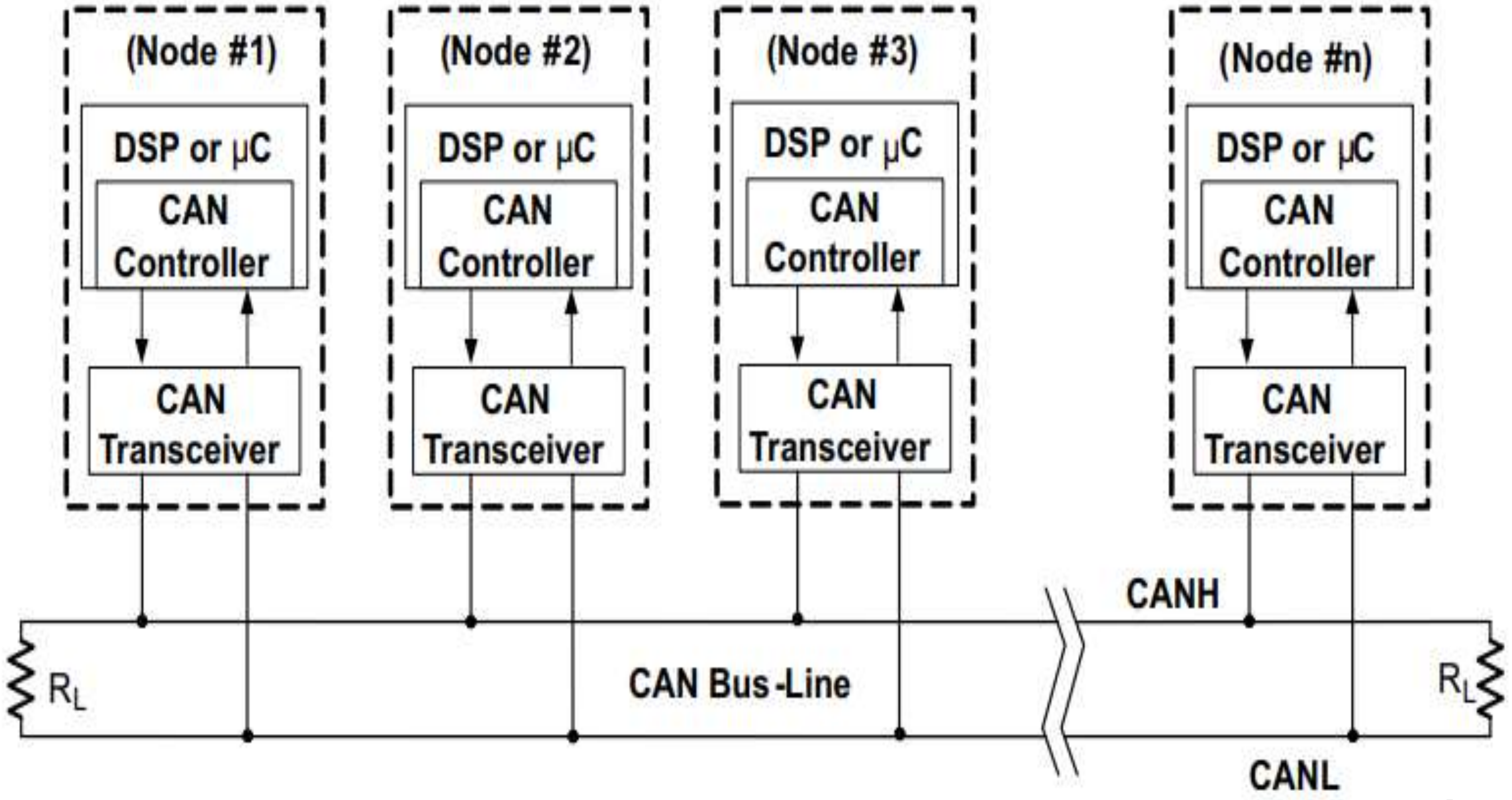
•**Frame Types:** CAN bus employs two main frame types: Data Frames and Remote Frames. Data Frames transmit information, while Remote Frames request data without sending it.

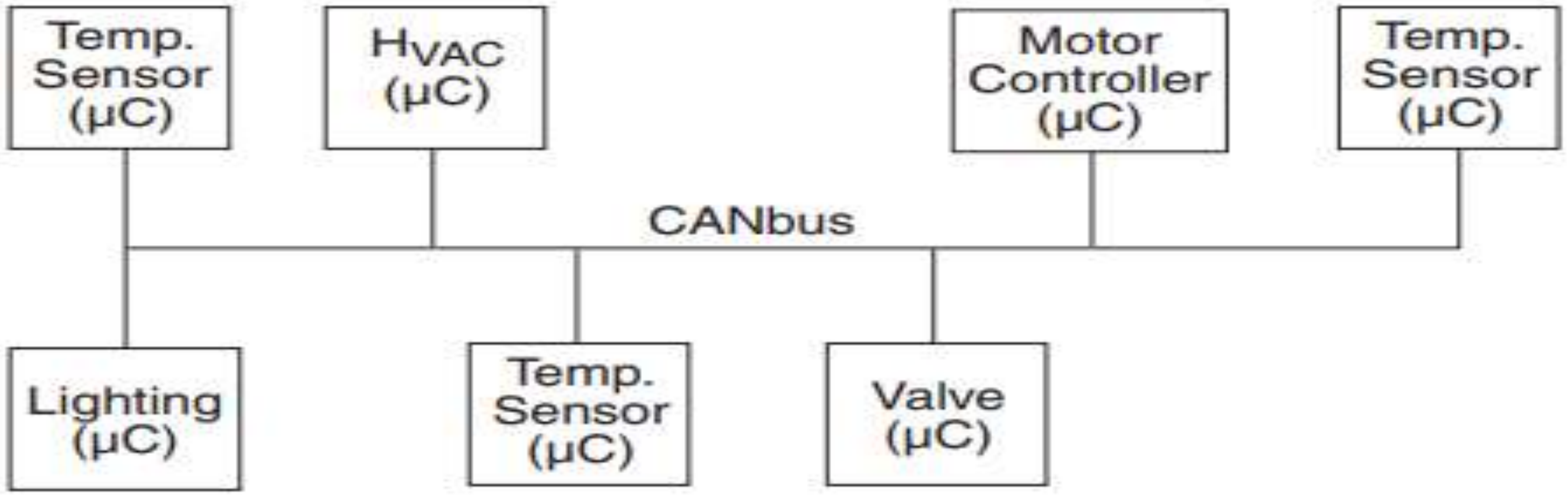
•**Network Topology:**

- **Multi-Master Bus:** CAN network topology allows multiple devices (ECUs) to send and receive data simultaneously.
- **Decentralized Control:** There is no central control coordinating communications. Each device can send data at any time, enabling decentralized communication.

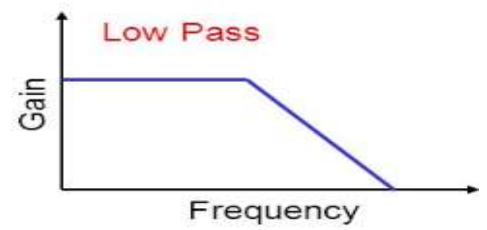
•**Data Transmission:**

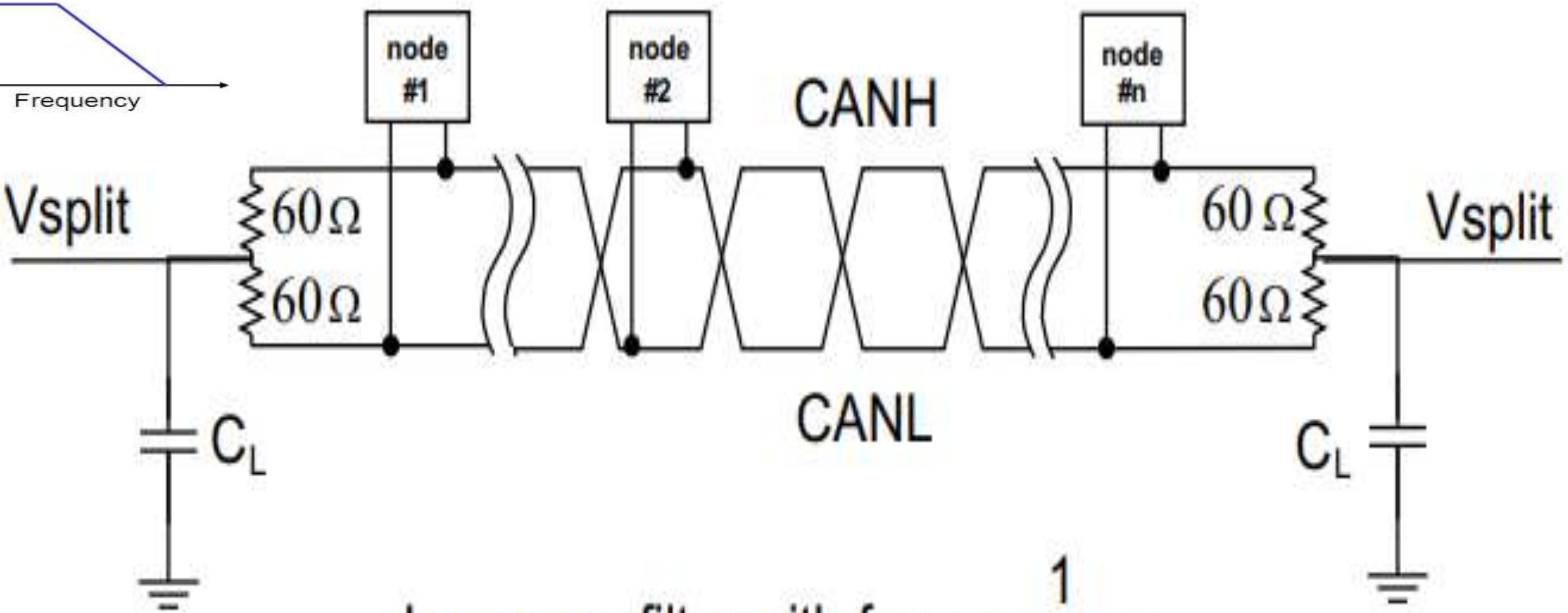
- **Medium Access:** Before transmitting data, devices listen to the bus to avoid collisions. They use a method called "message arbitration" to determine which message has priority on the bus.
- **Error Detection and Retransmission:** CAN bus has error detection and correction mechanisms. If an error is detected, retransmissions are performed to ensure data integrity.


(Node #1)
(Node #2)
(Node #3)
(Node #n)
DSP or μC
CAN Controller
CAN Transceiver
CANH
CANL
CAN Bus -Line
$R_L$
$R_L$



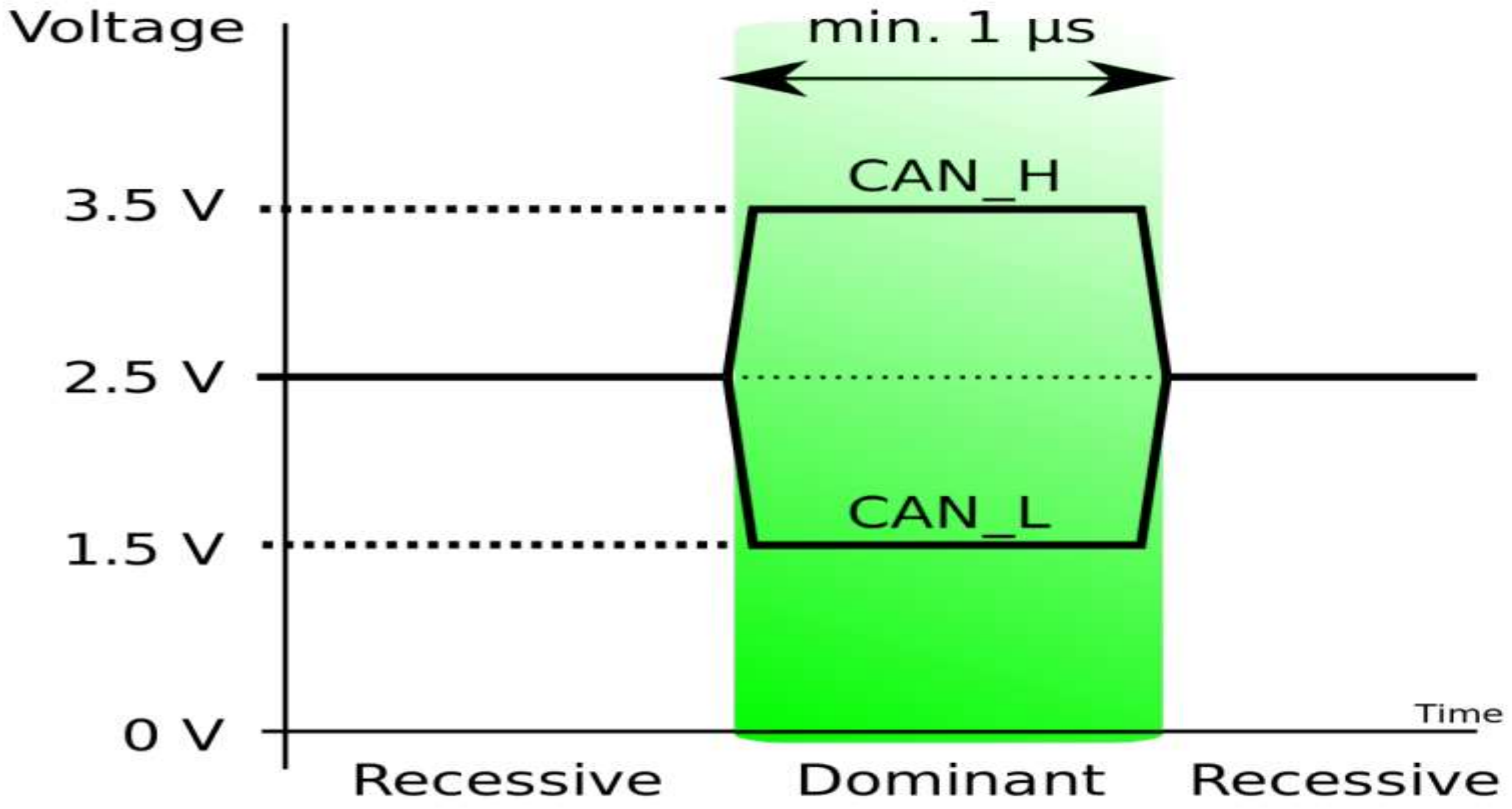
**Distributed Network**



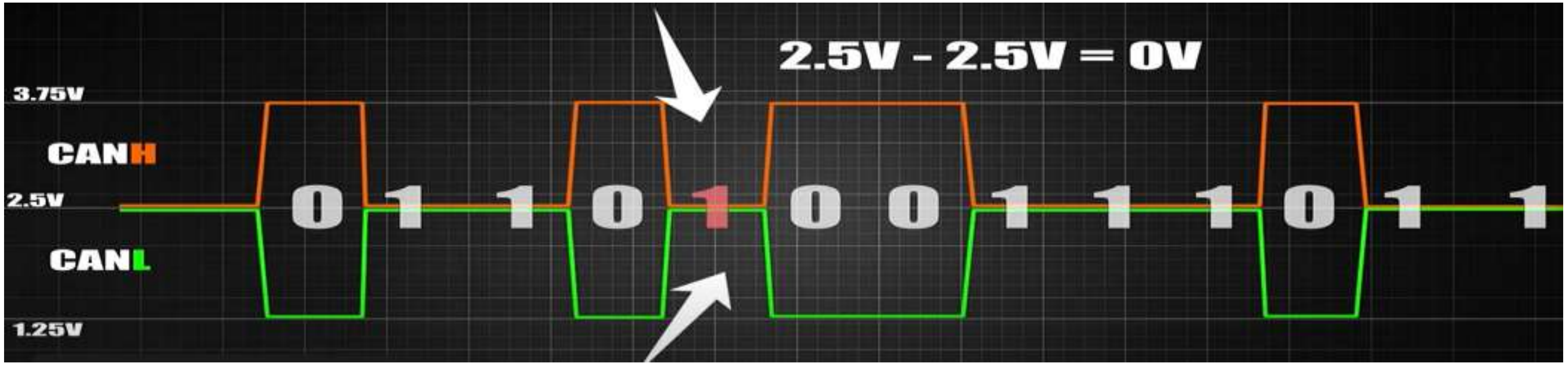


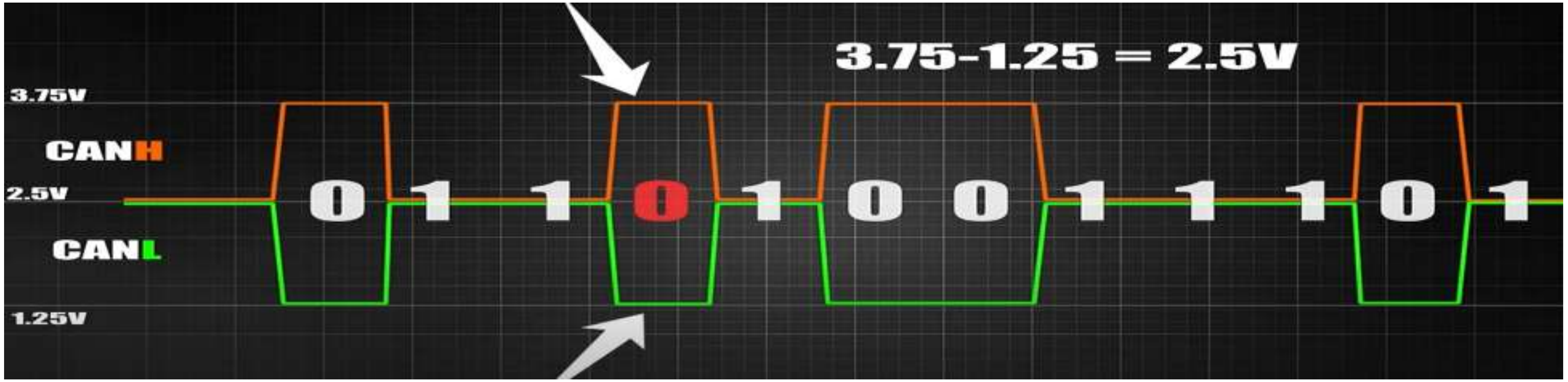
Low-pass filter with $f_C = \dfrac{1}{2\pi R C_L}$

## Tension levels CAN BUS

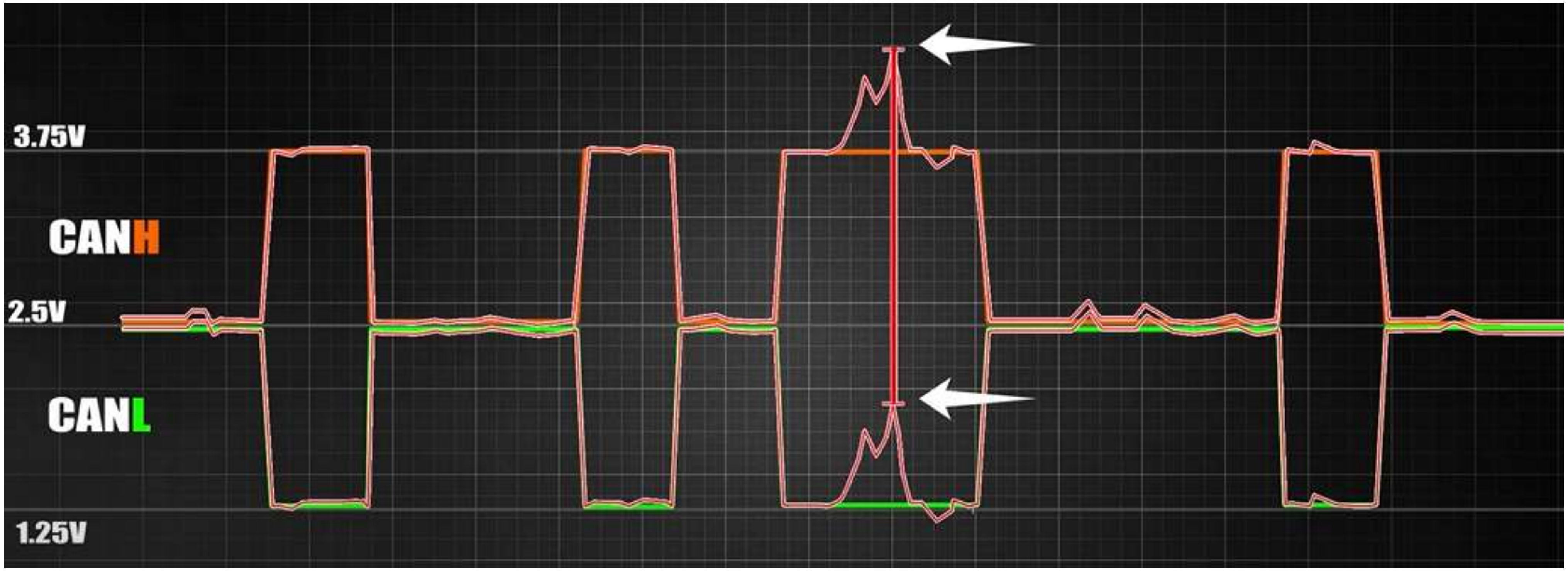


## Normal Signal from a CAN bus line





## Signal with noise from a CAN bus line



## Can Bus Version

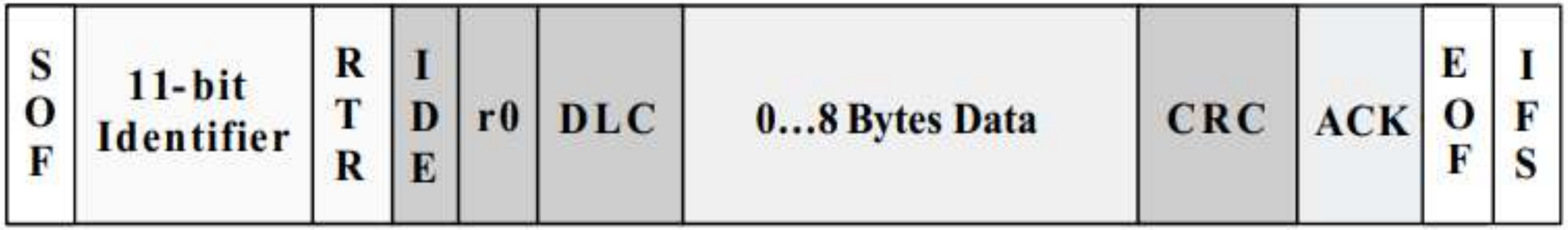
**CAN 1.0:**
•The initial standard of CAN, introduced by Bosch in 1986.
•Offers a data transfer rate of up to 1 Mbps.
•Uses an 11-bit message identifier scheme.
**CAN 2.0:**
It's divided into two sub-versions: CAN 2.0A and CAN 2.0B.
•**CAN 2.0A:**
- Uses 11-bit message identifiers.

•**CAN 2.0B:**
- Allows for extended 29-bit message identifiers.
- Greater device and message identification capacity in larger networks.

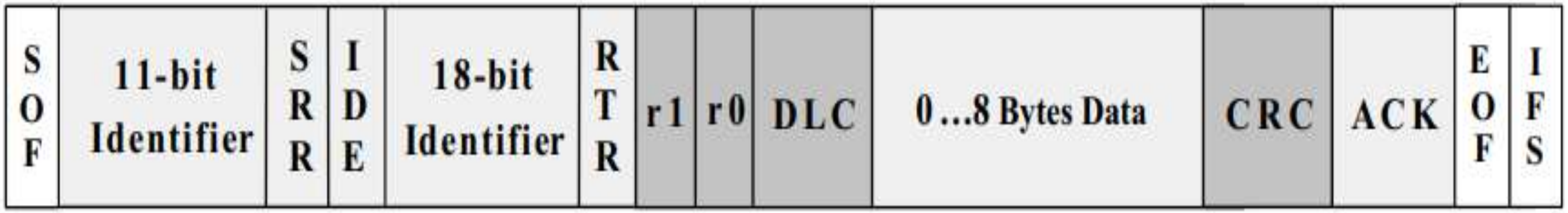
Both sub-versions maintain a data transfer rate of up to 1 Mbps.
**CAN FD (Flexible Data-Rate):**
•Introduced in 2012 to enhance the speed and flexibility capabilities of the CAN standard.
•Allows for higher data transfer rates, reaching up to 8 Mbps.
•Provides a flexible data protocol with larger payload per message and reduced latency.
•Enables dynamic switching between different transmission speeds.

## The Bit Fields of Standard CAN and Extended CAN

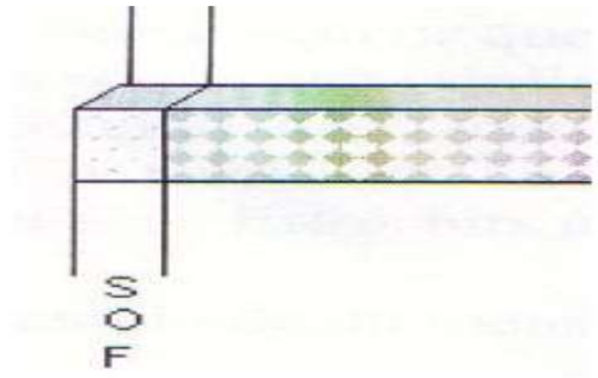
**Standard CAN**



**Extended CAN**

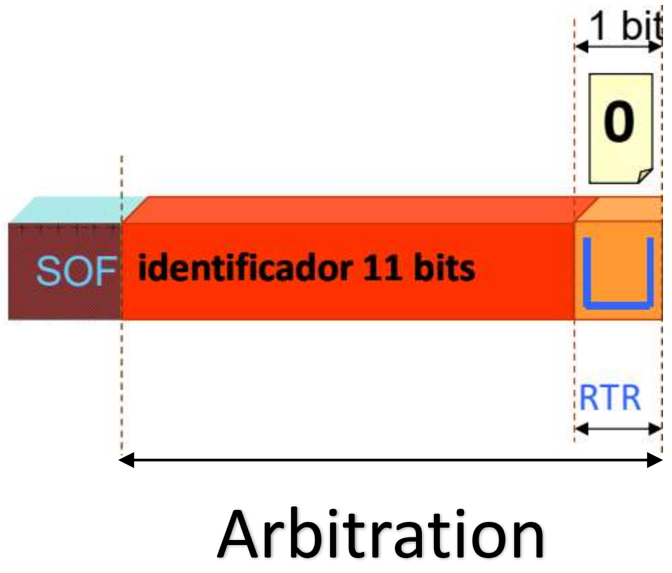


**1.Start of Frame (SOF):** The first bit of the message frame used to synchronize the clocks of devices on the CAN bus.



**2. Identifier (ID):** An 11 or 29-bit field that identifies the message and determines its priority in the CAN network. The identifier bits can be standard or extended and are used to distinguish between different messages on the CAN bus.

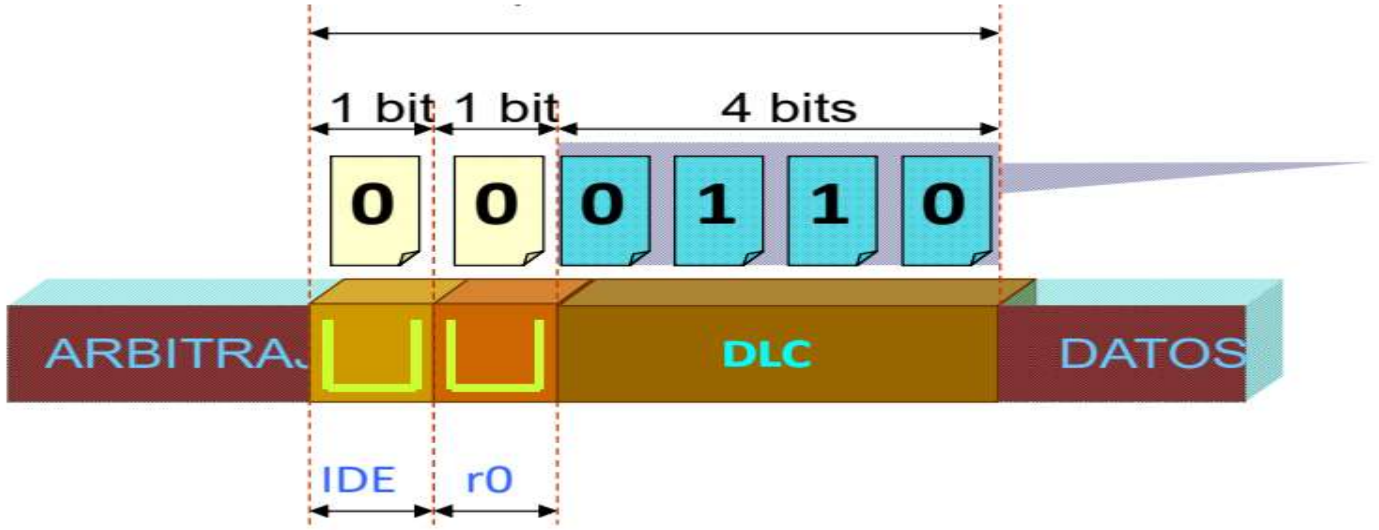
**3. Remote Transmission Request (RTR):** Indicates if the message is a remote transmission request. When set (1), it signals that the receiving node should send the data associated with the message's identifier.



How many identifiers can I create with 11 bit?

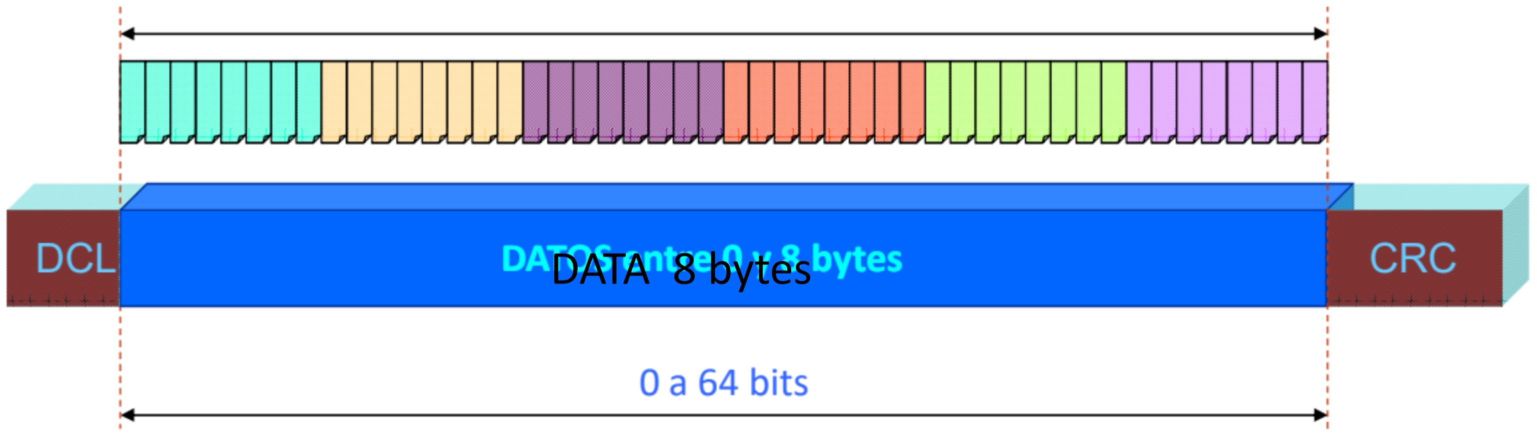
2**11 = 2048

**4. Control (IDE and r0/r1):** The IDE (Identifier Extension) bit, when set (1), indicates that the message contains an extended identifier (29 bits). The r0 and r1 bits are reserved and used to indicate future reserved use.

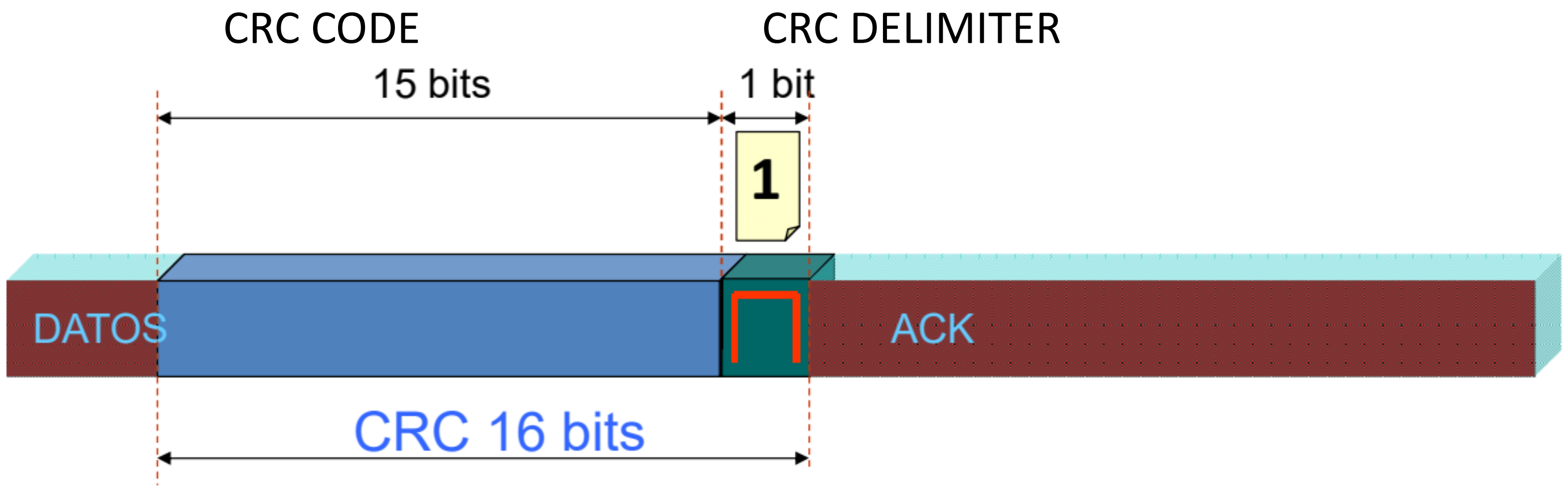


**5. Data Length Code (DLC):** A 4-bit field specifying the number of data bytes being transmitted in the message (ranging from 0 to 8 bytes).

**6. Data Field:** This field can hold up to 8 bytes of data and represents the information being transmitted between devices on the CAN bus.

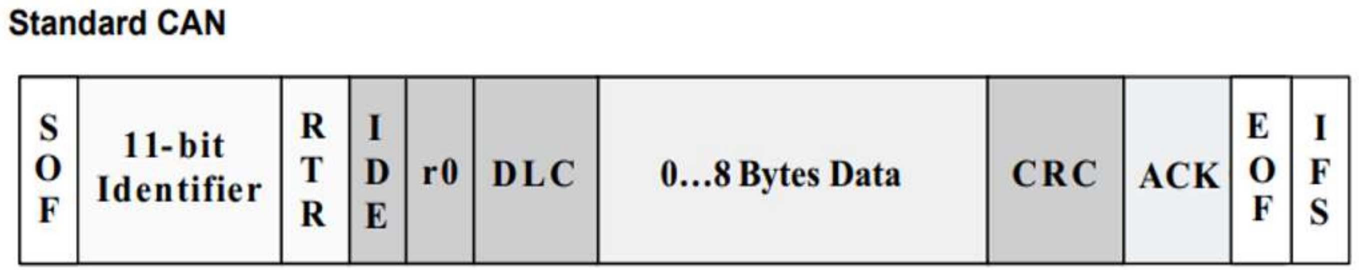


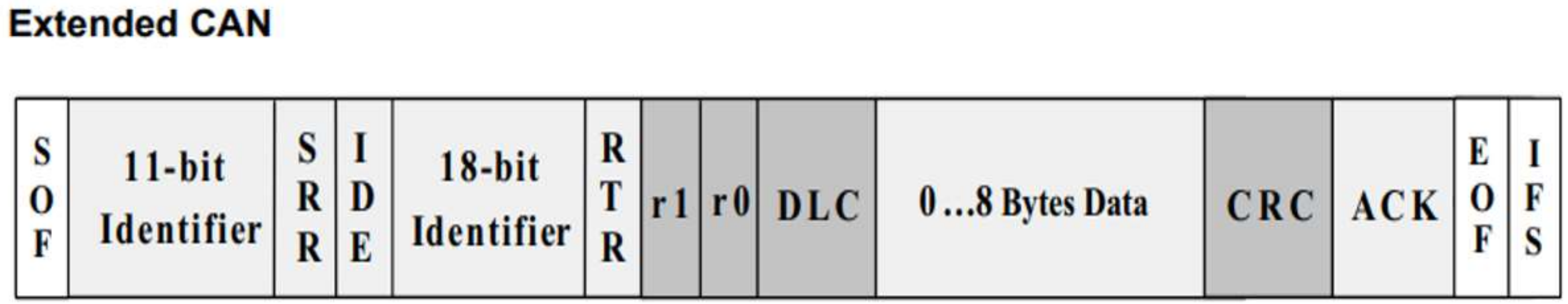
**7. CRC (Cyclic Redundancy Check):** A 15-bit field used for error detection in the data frame, ensuring the integrity of the transmitted data.



**8. Acknowledgment (ACK):** Indicates if the message was received correctly. Receiving nodes send an ACK signal to the transmitting node if the message was received without errors.

**9. Interframe Space (IFS) Field:**
A space between frames to allow other nodes to prepare for the next frame.
**10.End of Frame (EOF) Field:**
Indicates the end of the frame.



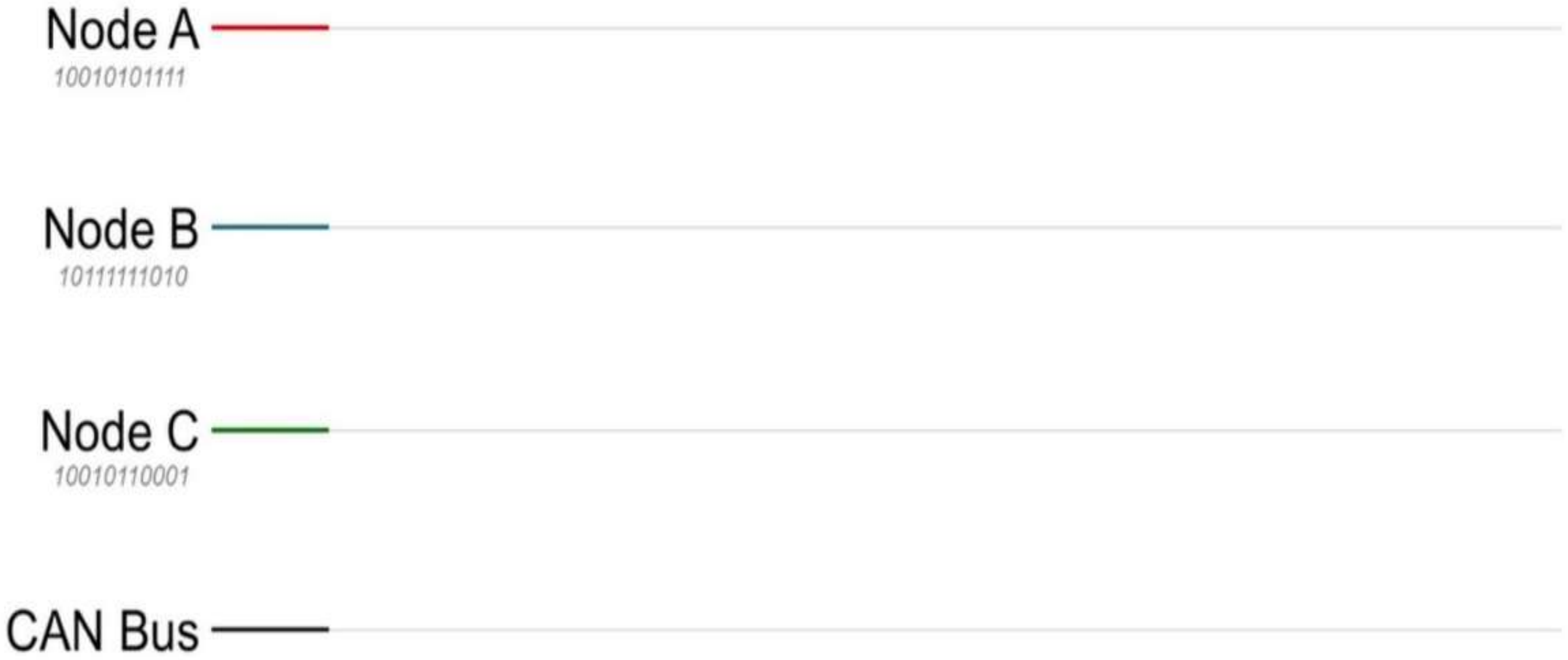


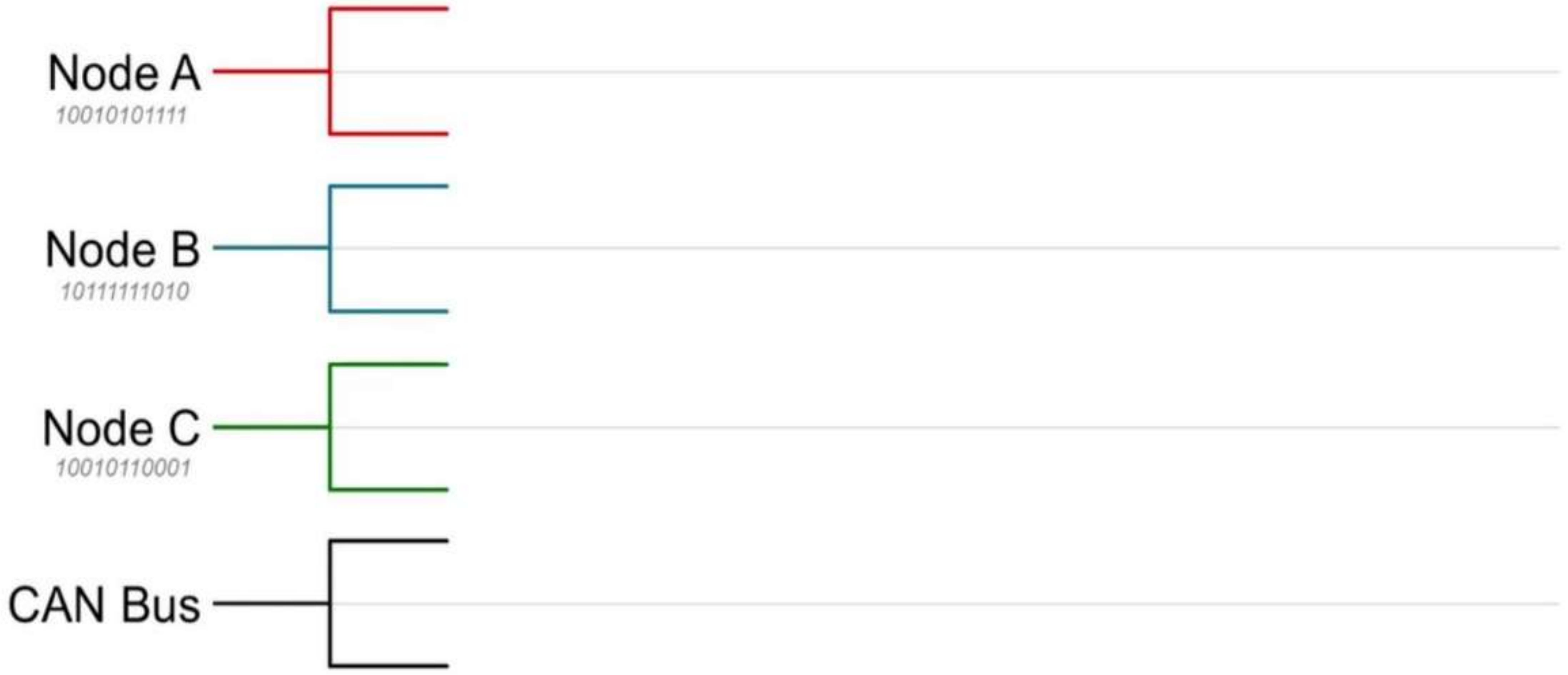
## Optimizing CAN Bus Networks: Understanding Device Limitations and Performance Considerations in CAN 2.0A and CAN 2.0B Standards.

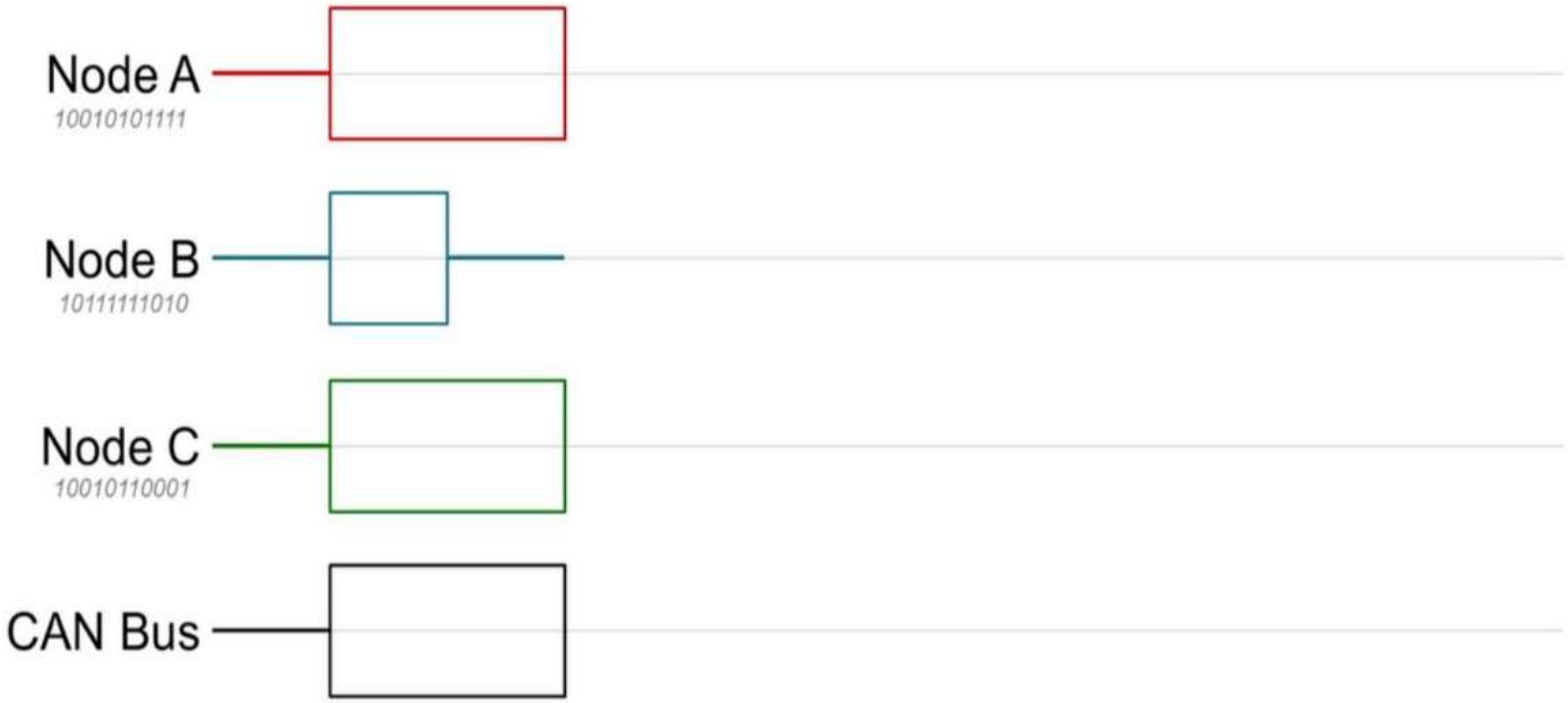
The CAN 2.0A, CAN 2.0B standard allows for up to 127 devices in theory, but the actual number of devices you can effectively connect may vary depending on the specific implementation, network conditions, and performance requirements of your application. It is always advisable to conduct tests and performance analysis to determine the optimal number of devices you can connect to your CAN network without compromising communication efficiency and reliability.

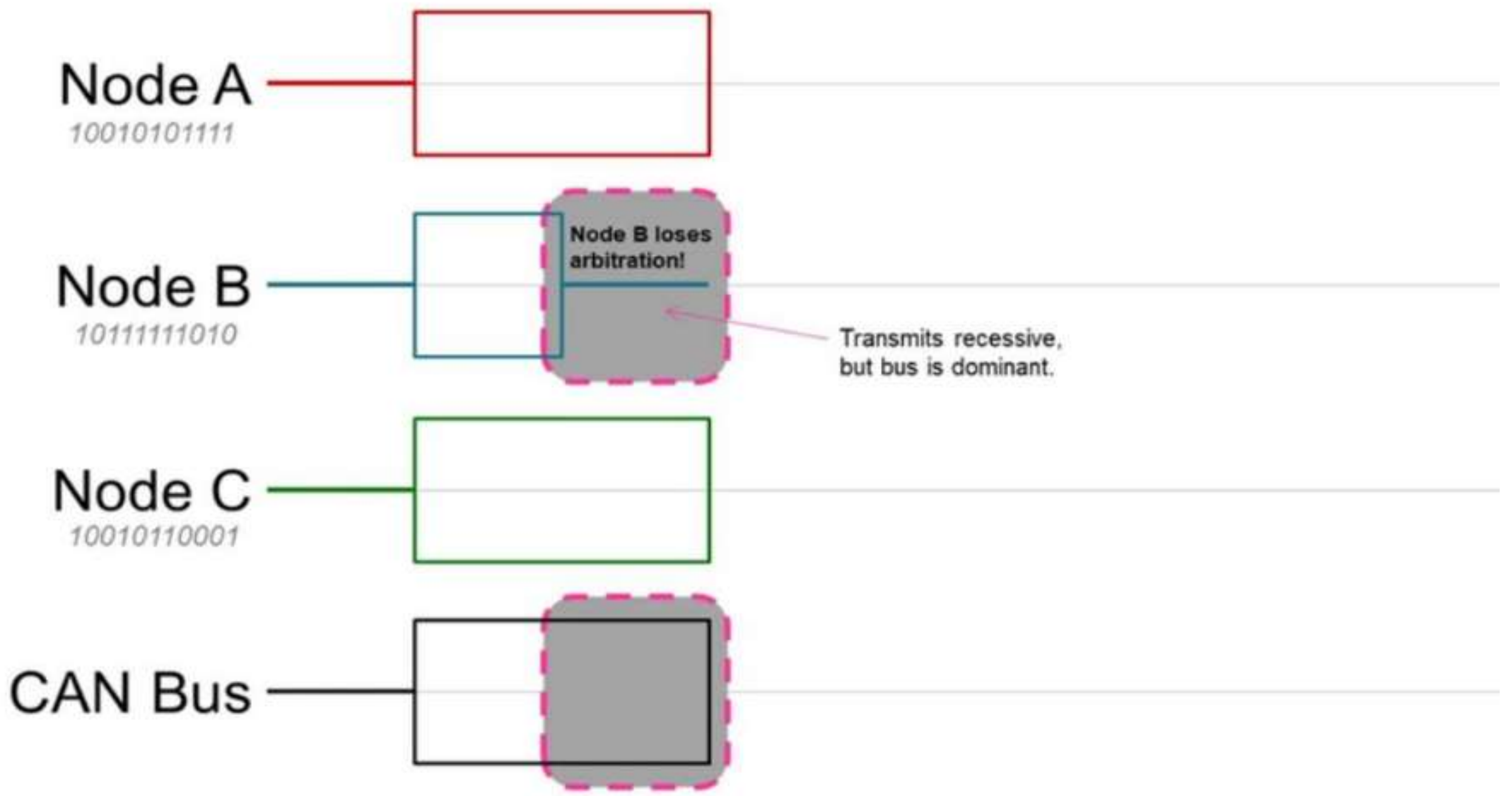
**Arbitration** on a CAN (Controller Area Network) bus is a process that determines which message will be transmitted when multiple nodes on the network attempt to send data simultaneously. CAN is a multi-master communication protocol, meaning that any node can initiate communication. Arbitration is essential for ensuring that messages are transmitted in a collision-free manner.

SNIDO INTELLIGENT SOLUTIONS

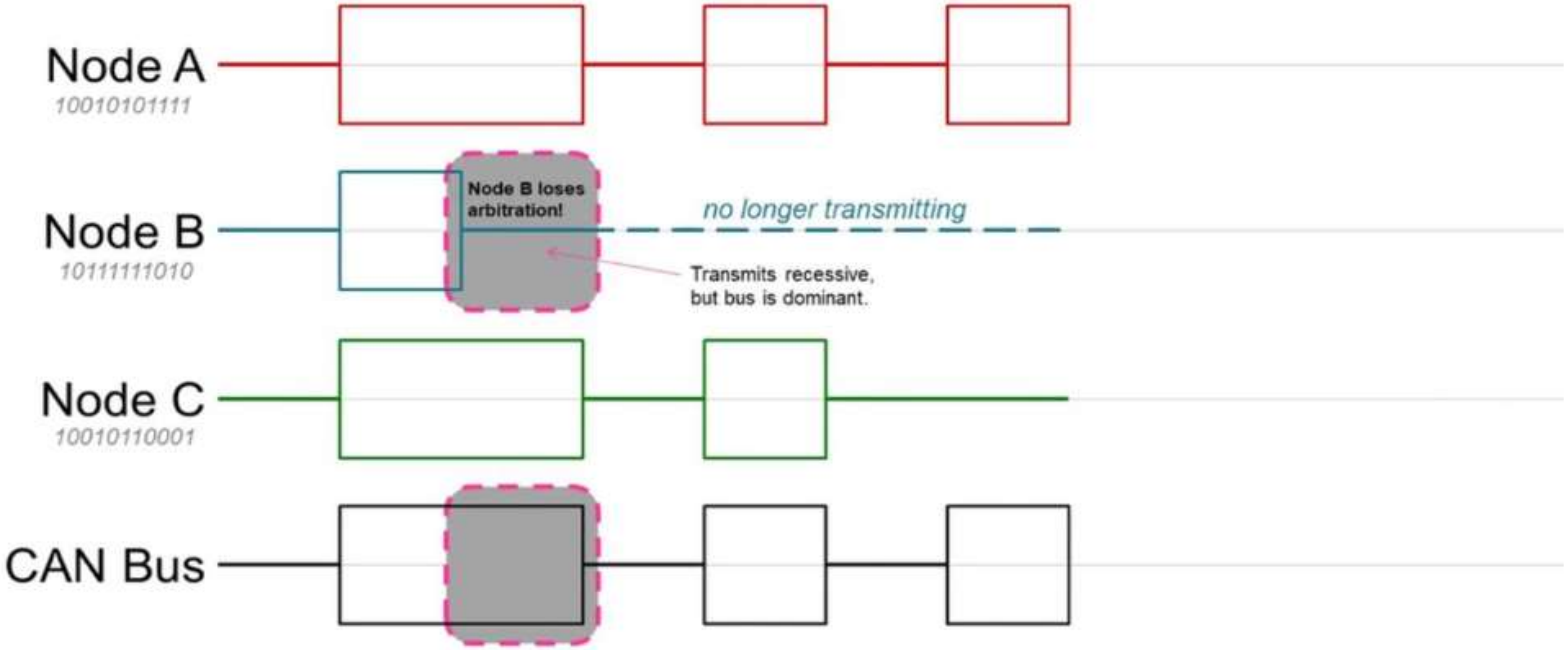

Node A
10010101111
Node B
10111111010
Node C
10010110001
CAN Bus

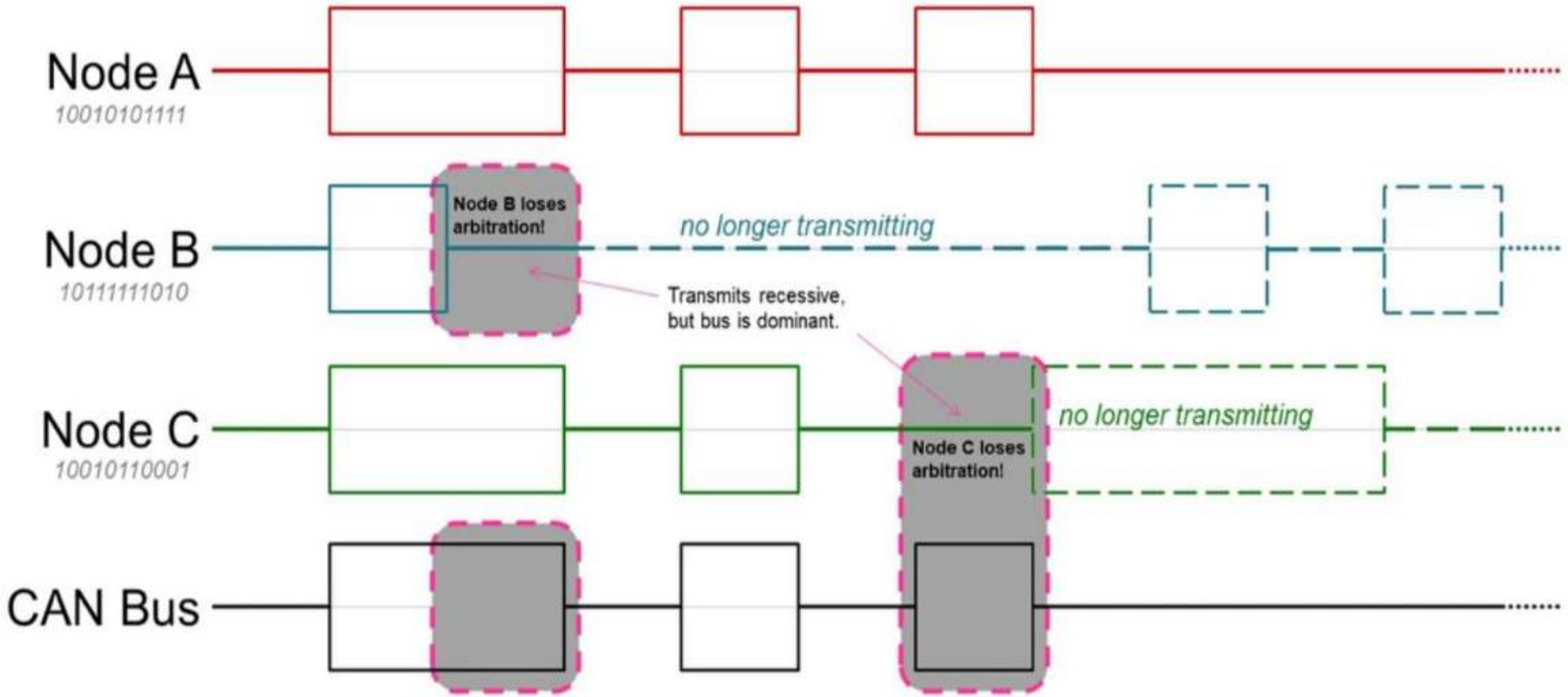

Node A
10010101111
Node B
10111111010
Node C
10010110001
CAN Bus


Node A
10010101111
Node B
10111111010
Node C
10010110001
CAN Bus

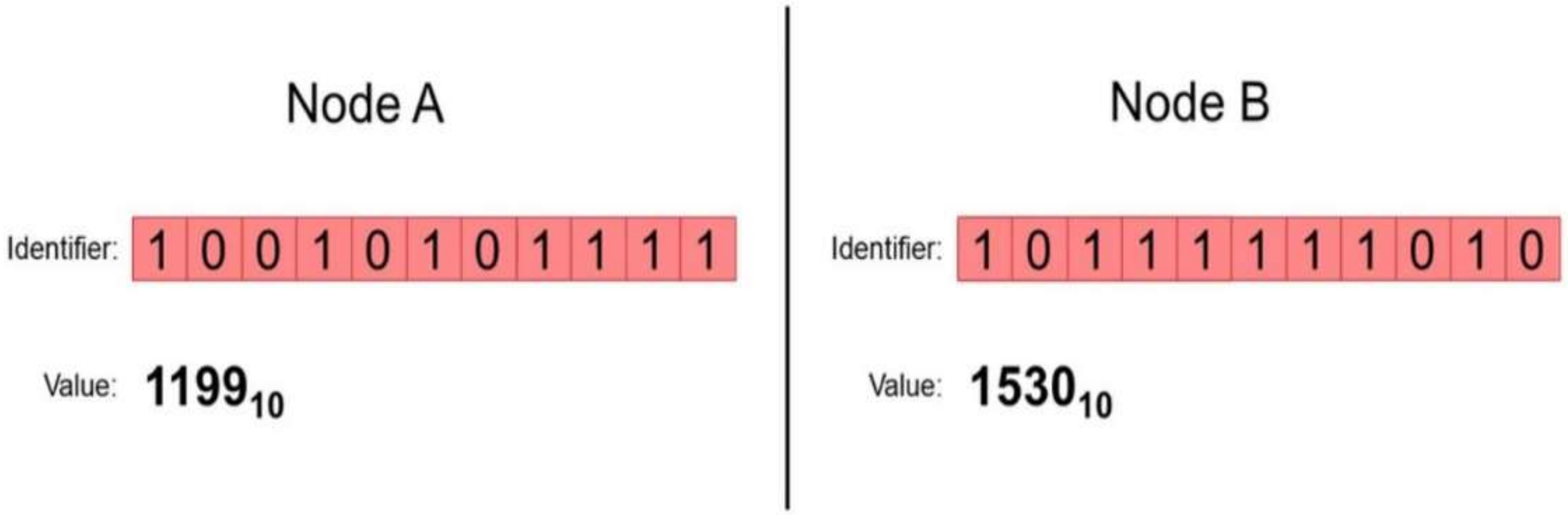

Node A
10010101111
Node B
10111111010
Node B loses arbitration!
Transmits recessive, but bus is dominant.
Node C
10010110001
CAN Bus

INTELLIGENT SOLUTIONS


Node A
10010101111
Node B
10111111010
Node B loses arbitration!
no longer transmitting
Transmits recessive, but bus is dominant.
Node C
10010110001
CAN Bus

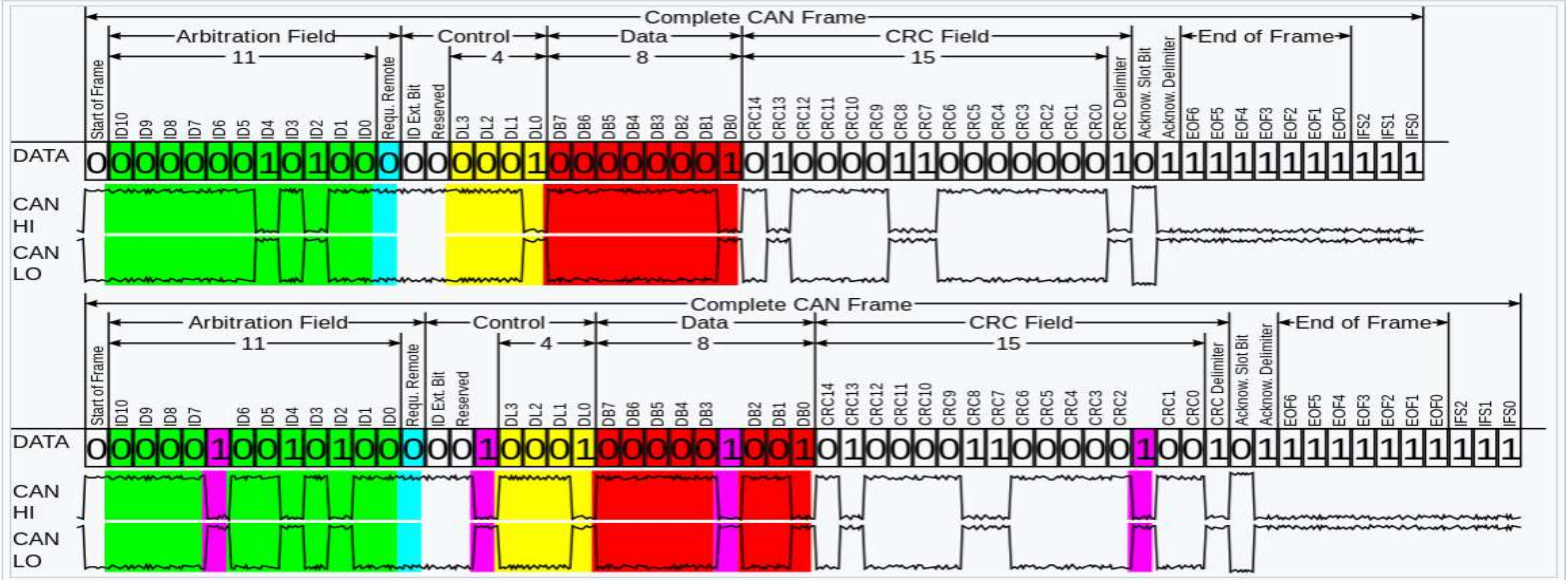

Node A
10010101111
Node B
10111111010
Node B loses arbitration!
no longer transmitting
Transmits recessive, but bus is dominant.
Node C
10010110001
Node C loses arbitration!
no longer transmitting
CAN Bus

The **lower** the arbitration value, the **higher** the priority.

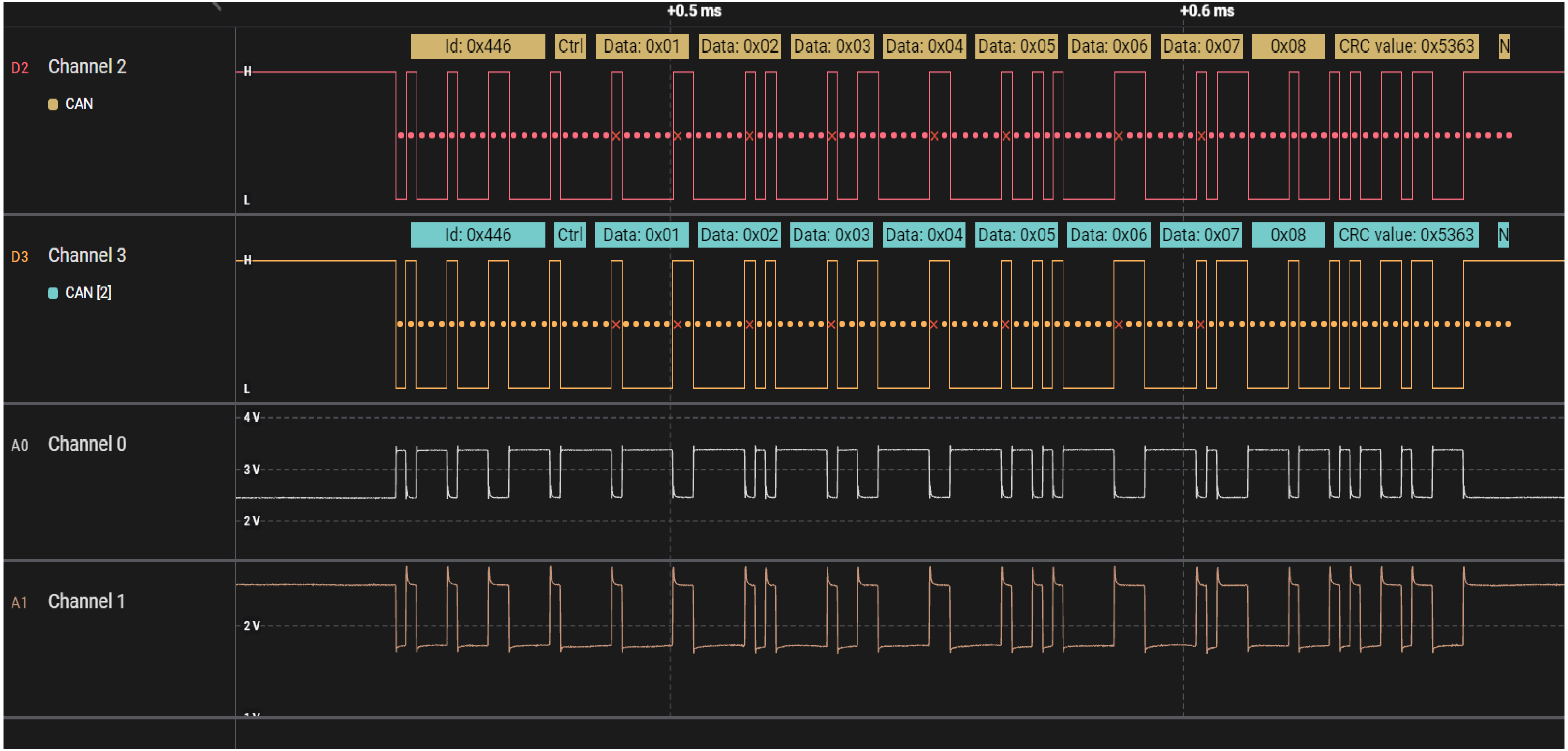

Node A
Identifier: 1 0 0 1 0 1 0 1 1 1 1
Value: $1199_{10}$
Node B
Identifier: 1 0 1 1 1 1 1 1 0 1 0
Value: $1530_{10}$

**Node A** has higher priority.

# Stuff bits

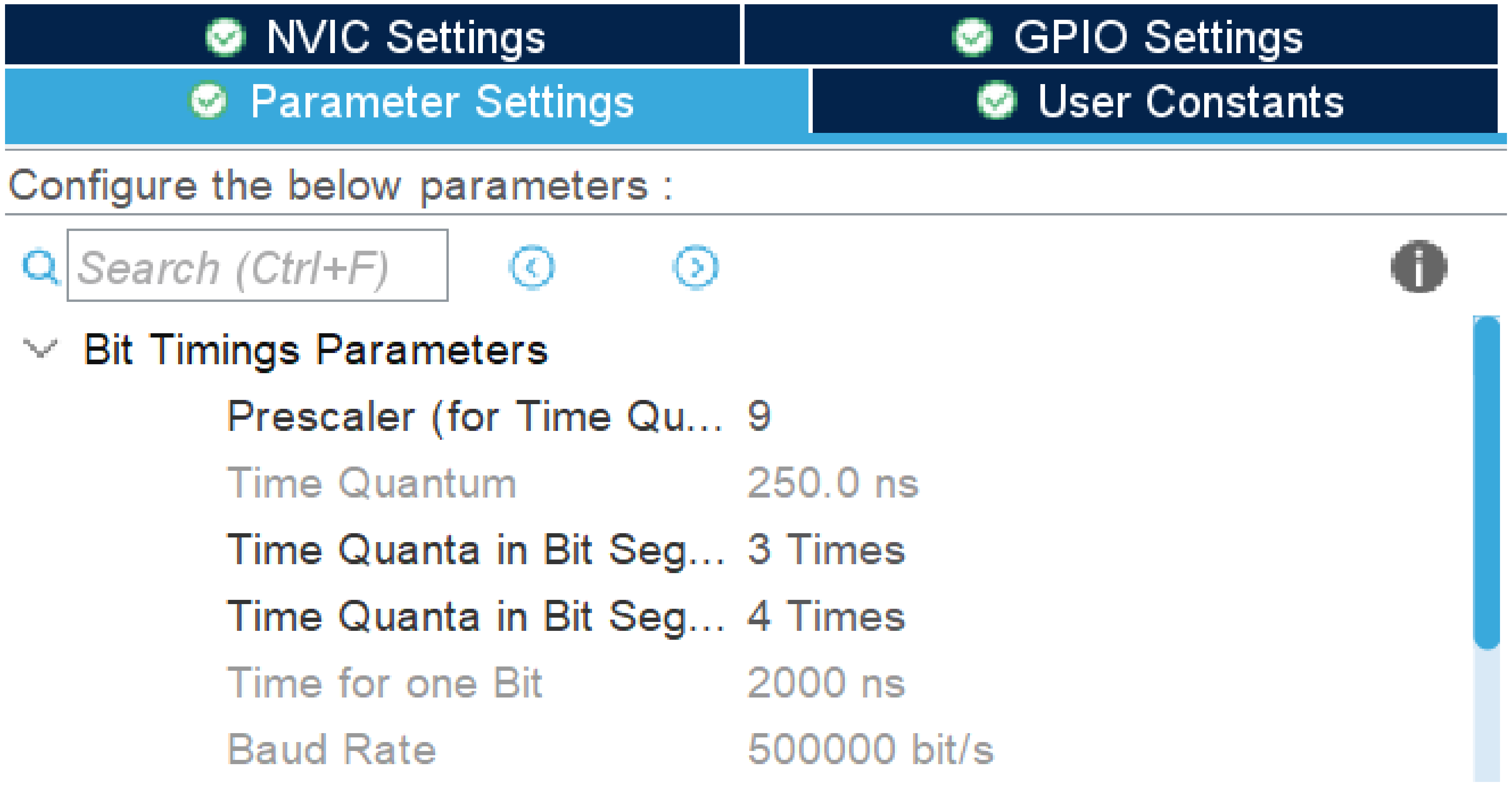


CAN frame before and after the addition of stuff bits (in purple). An incorrect CRC is used for bit stuffing illustration purposes.


+0.5 ms
+0.6 ms
D2
Channel 2
CAN
Id: 0x446
Ctrl
Data: 0x01
Data: 0x02
Data: 0x03
Data: 0x04
Data: 0x05
Data: 0x06
Data: 0x07
0x08
CRC value: 0x5363
N
H
L
D3
Channel 3
CAN [2]
Id: 0x446
Ctrl
Data: 0x01
Data: 0x02
Data: 0x03
Data: 0x04
Data: 0x05
Data: 0x06
Data: 0x07
0x08
CRC value: 0x5363
N
H
L
A0
Channel 0
4 V
3 V
2 V
A1
Channel 1
2 V

## BAUT RATE: Example

- The CAN bus to move at 500kbps, or 500,000 bps. One bit's time is 1/500000 seconds, or 2000 nanoseconds. To obtain the necessary 2000ns in the Time for one Bit section, we must therefore modify Prescaler, Time Quanta in Bit Segment 1 and Time Quanta in Bit Segment 2.
- Since Prescaler provides a rounded value of 250.0 ns in The clock quantum, I use it with a 9.
- Prescaler = 9
- Time Quantum = 250.0
- Time Quanta in Bit Segment 1 = 3 Times
- Time Quanta in Bit Segment 2 = 4 Times
- **Time for 1 Bit = TQ + (TQ *TQ Bit 1) + (TQ* TQ Bit 2)**
- Time For 1 Bit = 250 + (250* 3) + (250*4)
- Time For 1 Bit = 250 + 750 +1000 = 2000ns



Configure the below parameters :

Bit Timings Parameters

| | |
|---|---|
| Prescaler (for Time Qu... | 9 |
| Time Quantum | 250.0 ns |
| Time Quanta in Bit Seg... | 3 Times |
| Time Quanta in Bit Seg... | 4 Times |
| Time for one Bit | 2000 ns |
| Baud Rate | 500000 bit/s |

INTELLIGENT SOLUTIONS

1. **Time Quantum = 250.0:** This value signifies the fundamental time unit or quantum of time within the system, and it is set at 250.0 nanoseconds. The Time Quantum serves as a foundational unit for measuring and adjusting various temporal parameters within the CAN protocol.
2. **Time Quanta in Bit Segment 1 = 3 Times:** The Bit Segment 1 represents a portion of the overall time frame for a bit in the CAN protocol. The designation "3 Times" implies that this specific phase of the bit is subdivided into three time segments, each quantified by the value of the Time Quantum. Each of these segments contributes to defining the total time of a bit, influencing the synchronization and communication timing.
3. **Time Quanta in Bit Segment 2 = 4 Times:** Similar to Bit Segment 1, Bit Segment 2 is another component of the time frame for a bit. In this case, "4 Times" indicates that this phase is subdivided into four time segments, all quantified by the Time Quantum value. The segmentation in both Bit Segment 1 and Bit Segment 2 is crucial for shaping the temporal characteristics of a bit and, consequently, the overall communication on the CAN bus.

## Conclusion on CAN Bus Protocol:

- The CAN Bus protocol has proven to be a robust and widely adopted communication standard, offering a range of benefits for various applications. Its ability to facilitate reliable and real-time data exchange among multiple devices has made it a cornerstone in automotive, industrial, and other embedded systems.
- The decentralized nature of CAN, allowing multiple nodes to communicate without a central master, enhances system flexibility. The protocol's prioritization mechanism, based on message identifiers, ensures efficient data transmission in scenarios with varying levels of criticality.

- However, while CAN Bus provides a solid foundation for communication, its effectiveness depends on careful network design, considering factors such as the number of devices, data rates, and application requirements. Testing and performance analysis are crucial to identifying potential bottlenecks and optimizing network efficiency.
- As technology evolves, newer protocols may offer enhanced features, but CAN remains a reliable and widely supported choice, especially in environments where real-time communication and fault tolerance are paramount. Continuous monitoring and adaptation to changing conditions are key practices for maintaining the protocol's efficiency in dynamic operational scenarios."