CAN Protocol

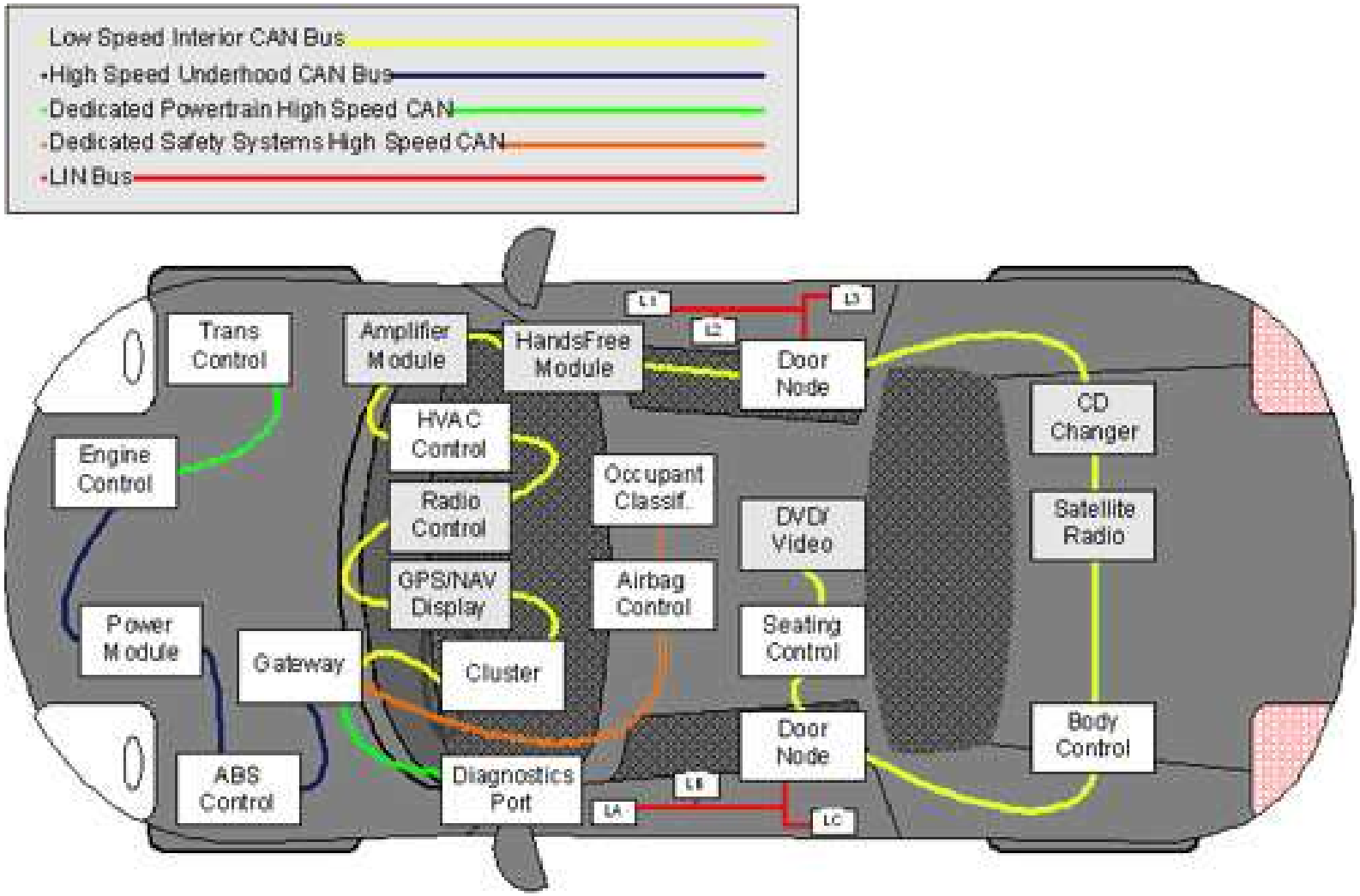
What is CAN ?

- The Controller Area Network protocol (CAN or CAN Bus) is a two-wire (twisted-pair), bidirectional serial bus communication method that allows electronic subsystems to be linked together and interact in a network.

History of CAN

- Pre CAN: Car ECUs relied on complex point-to-point wiring
- 1986: Bosch developed the CAN protocol as a solution
- 1991: Bosch published CAN 2.0
- 1993: CAN is adopted as international standard (ISO 11898)
- 2003: ISO 11898 becomes a standard series
- 2012: Bosch released the CAN FD 1.0 (flexible data rate)
- 2015: The CAN FD protocol is standardized (ISO 11898-1)
- 2016: The physical CAN layer for data-rates up to 5 Mbit/s standardized in ISO 11898-2

Controller
Area
Network



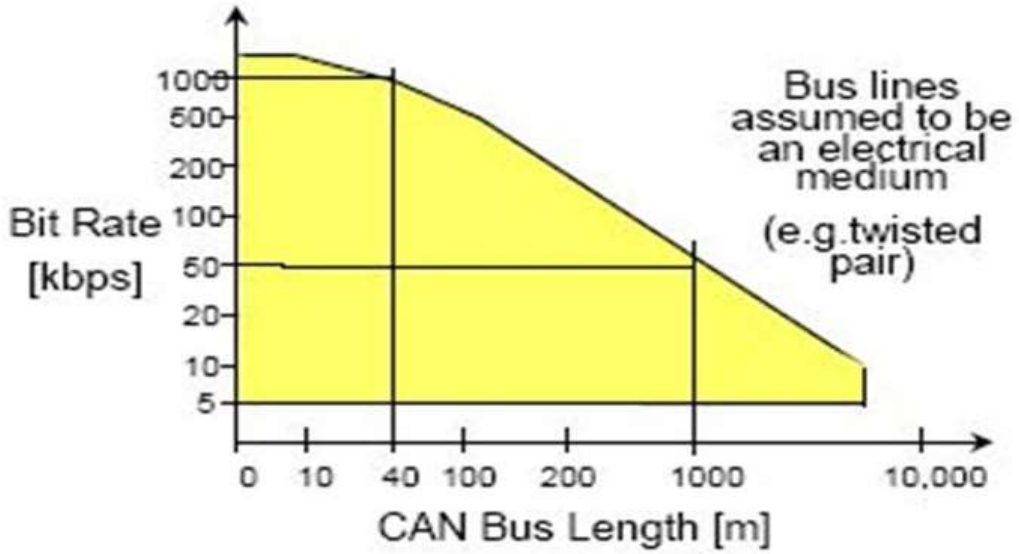
Comparing CAN with other protocols

	CAN	Flexray	Ethernet	Most
Bus	Dual twisted pair	Dual twisted pair	Wire or optical fibre	optical fibre
Speed	1MBPS	10MBPS	100MBPS and above	28.4MBPS
Data	8 bytes	264 bytes	1500Bytes	NA
Type	Asynchronous	Synchronous	point to point	point to point
Fault tolerance	High	Medium	Low	Low

- *CAN Speed*
- *CAN Data*
- *CAN Bus*
- *CAN Logic*
- *CAN Controllers and Transceivers*
- *CAN Network Topology*

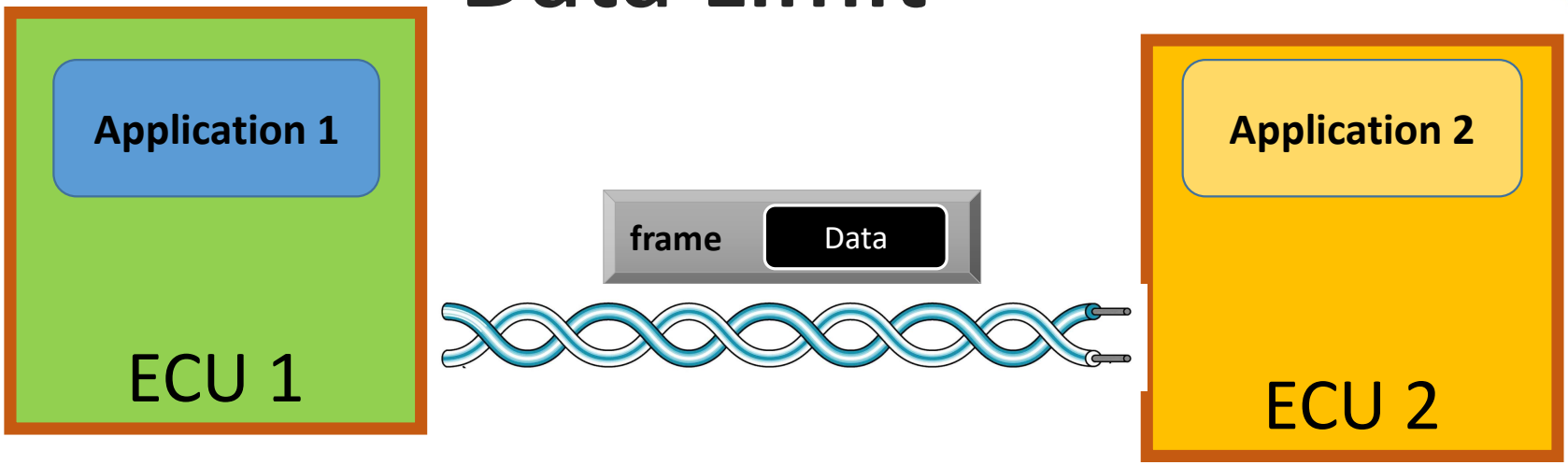
CAN Speed

- CAN frames can go up to 1mbps speed with the bus length at max of 40 mts.
- As the bus length increases above the maximum speed reduces.
- In industry usually the CAN is configured at 500kbps speed



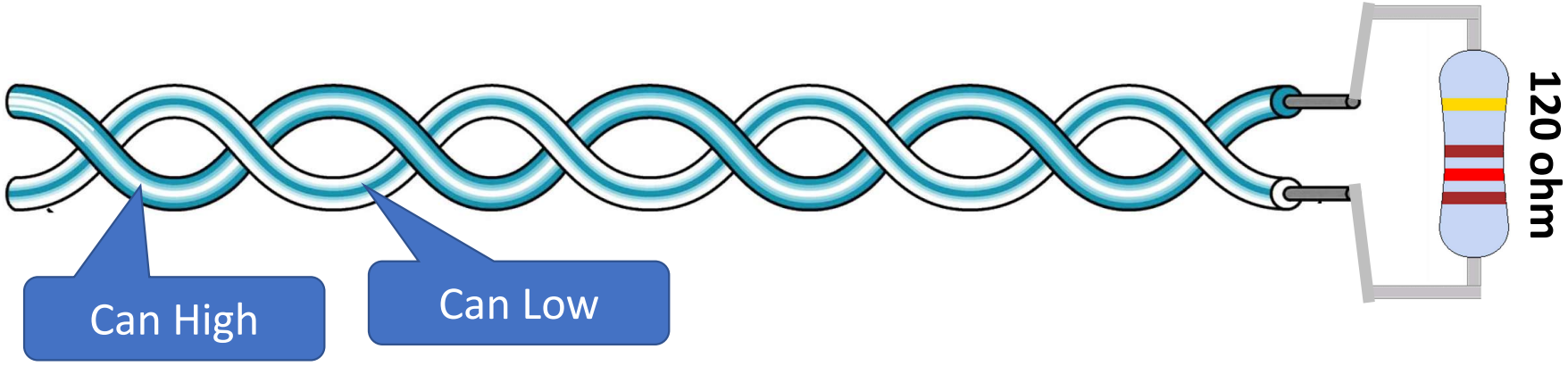
- Lets say the baud rate is 500kbps → In 1 second, 500 *1000 bits can be transmitted

Data Limit



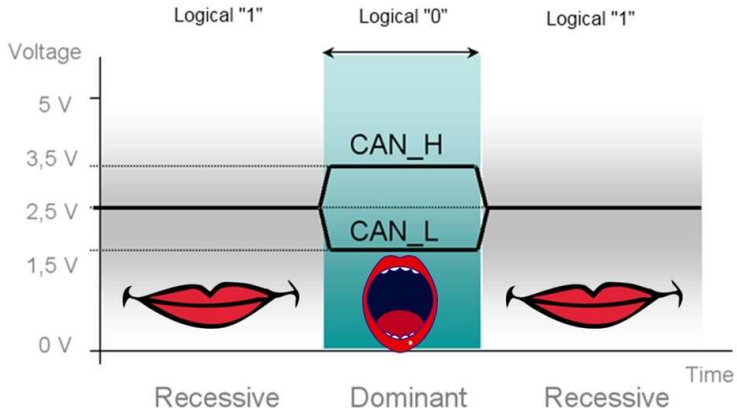
- Frame has 2 parts: PCI(Protocol Control Information) and Data
- As per CAN protocol How much data can be sent in each frame? → 8 Bytes
- Is the data size variable? → Yes we can have 0 bytes or 1 byte or 2 bytes etc. Max is 8 bytes.
- Is it possible to have data in bits? → No. Its in bytes. We can't have 2 bit or 27 bits etc its always in bytes.

CAN Bus



- CAN bus is a Dual-Twisted pair wires called CanHigh and CanLow
- Differential Voltage Logic
  - **Logic1 = Recessive bit**
  - **Logic0 = Dominant bit**

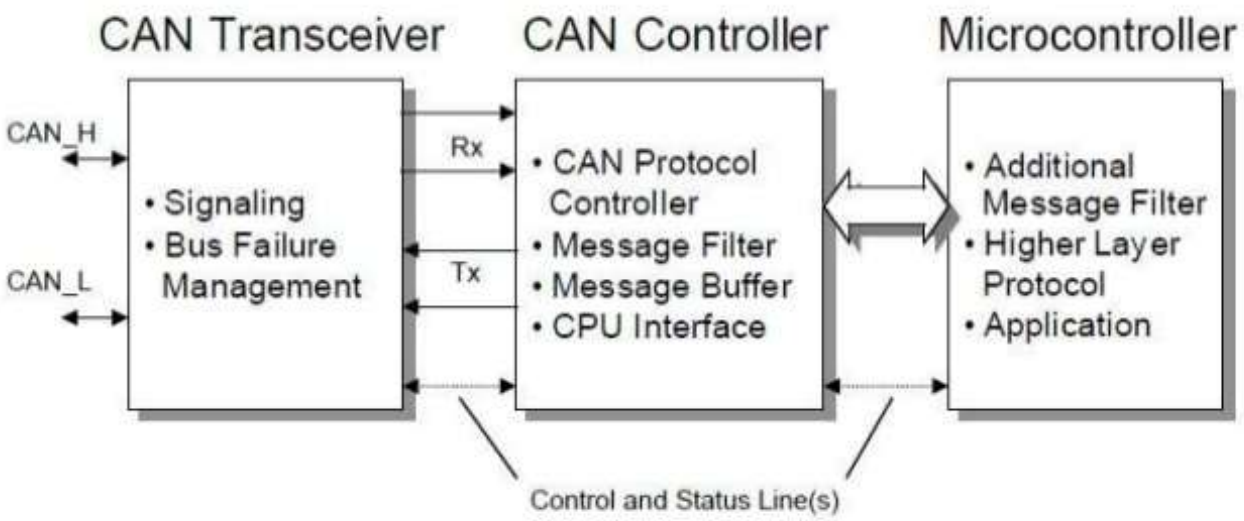
	CAN_H	CAN_L	Delta V
Logic 1	2.5v	2.5v	0
Logic 0	3.5v	1.5v	2



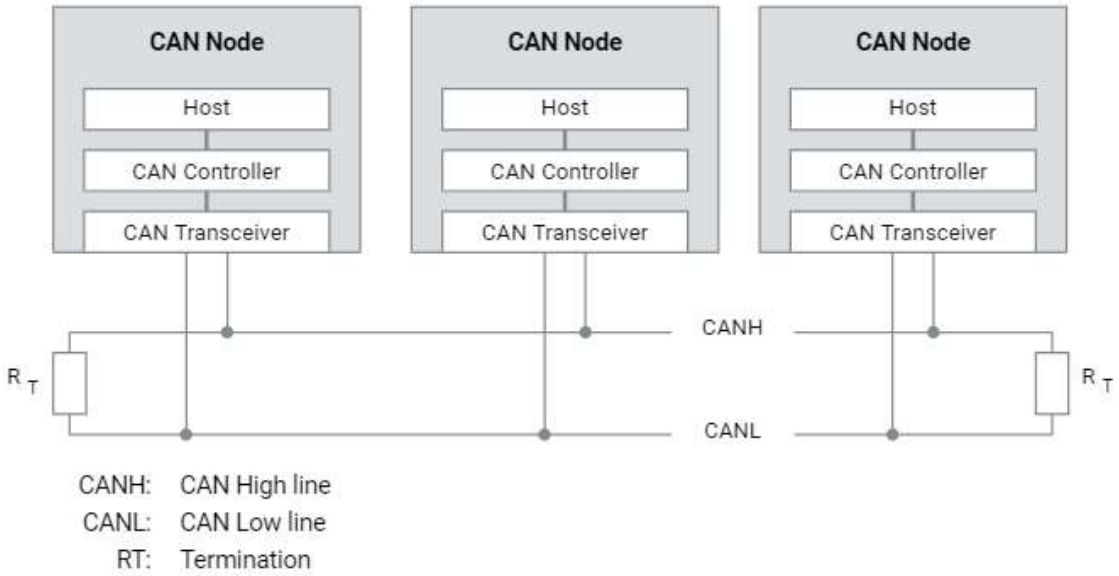
- Why twisted pair? → Noise Immunity
- Terminated with 120 ohm resistor → Impedance matching and minimize reflective waves.

Micro controllers work with TTL and not CAN logic

Stand alone CAN Controller



CAN Network Topology



CAN Properties

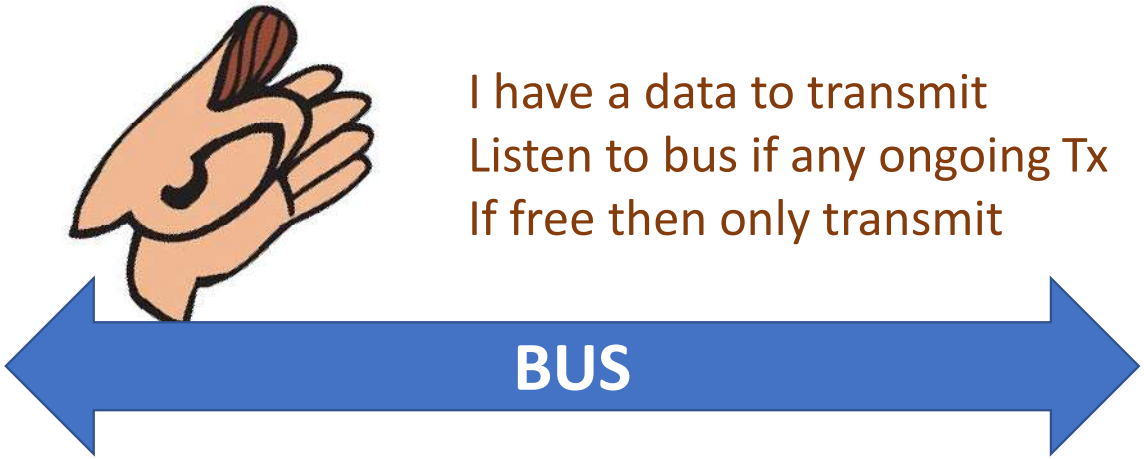
- Wired – Wireless
- Serial – Parallel
- Synchronous – Asynchronous
- Peer2Peer – Broadcast
- Message Based Protocol
- Message Filtering
- Simplex – Half Duplex – Full Duplex
- Acknowledgement method
- CSMA-CA Protocol
- Temporary and Permanent Node failures

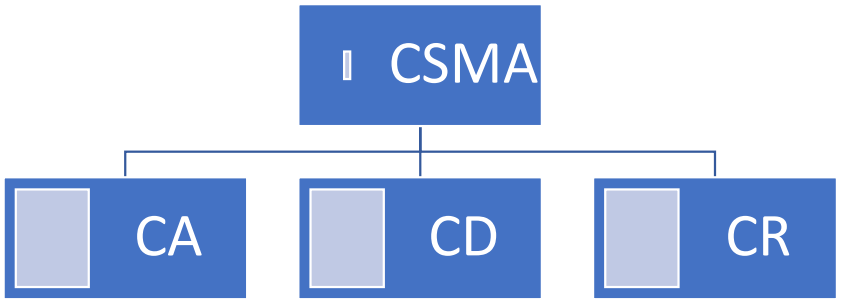
Message Based Protocol

- CAN protocol is message based protocol and not Node based protocol.
- Each CAN frame is identified with Message ID and no info about the Tx or Rx node.
- Node based Can be understood with mobile phones. Usually Peer-Peer are node based.
- Message ID has 3 purpose:
  - Identify the Data carried by the frame
  - Decides the message Priority and hence decides which frame gets the bus if clash happens.
  - Used in Message Filtering

CSMA-CA Protocol

CSMA-CA stands for Carrier Sense Multiple Access Collision Avoidance Protocol

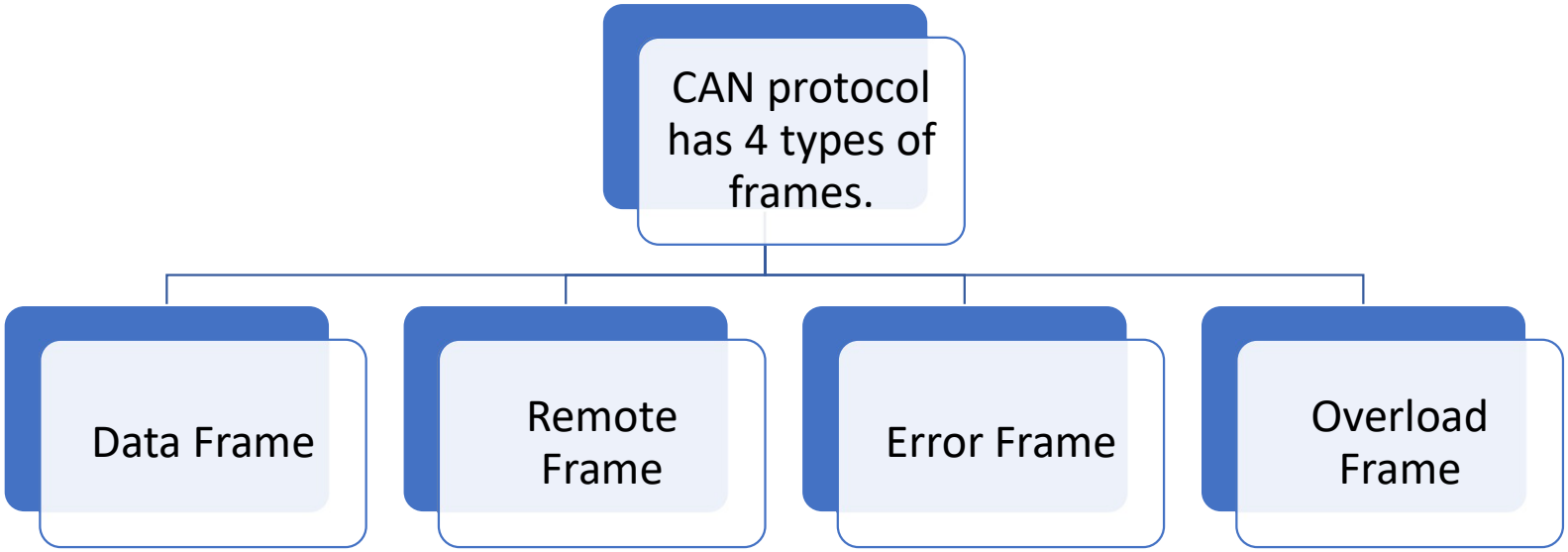




In CSMA : A transmitter attempts to determine whether another transmission is in progress before initiating a transmission using a carrier-sense mechanism

CA: Collision Avoidance → Doesn't let collision happen between its frames
CD: Collision Detection → If collision happens between the frames, it is detected and a proper action taken.
CR: Collision Resolution → If collision happens between frames, its resolved in that instance only successfully.

Types of CAN Frames

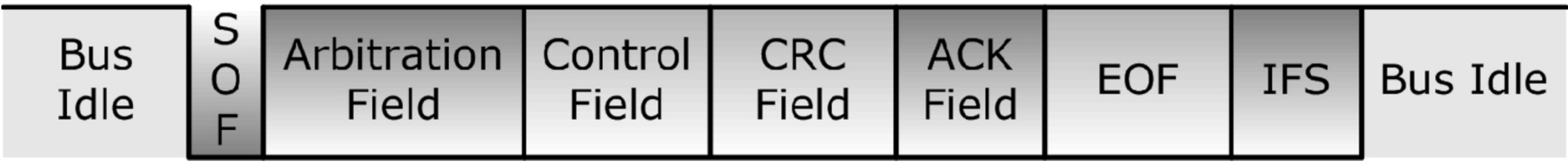


Data Frame

- Frame carrying the data from transmitter node to all receiver nodes.
- Serves the main purpose of CAN protocol
- Two types of Data Frames
  - Standard Frame : 11 bit message ID
  - Extended Frame : 29 bit message ID {split as 11 bit Base ID + 18 bit ID Extension}
- In CAN when we are normally talking of frames, we are referring Data frames only.

Remote Frame

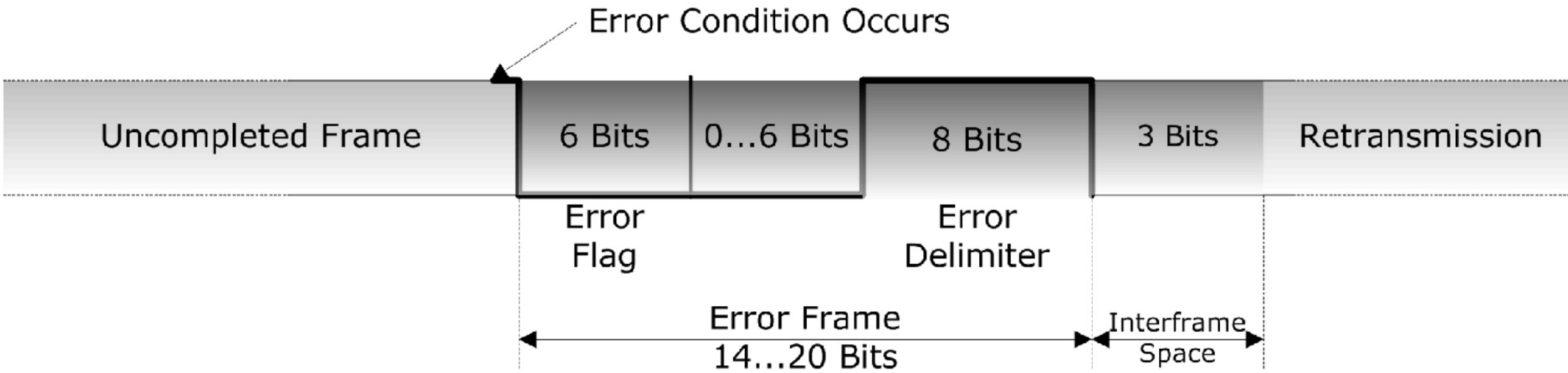
- Frame Requesting for a Data Frame
- A node wants a particular Data frame, so It sends the Remote frame with same Message ID on the Bus.
- This remote frame is followed by the Data frame on the bus transmitted by the node who owns that data frame.
- Now a days Remote frame is obsolete and not used in CAN protocol.



Error Frame

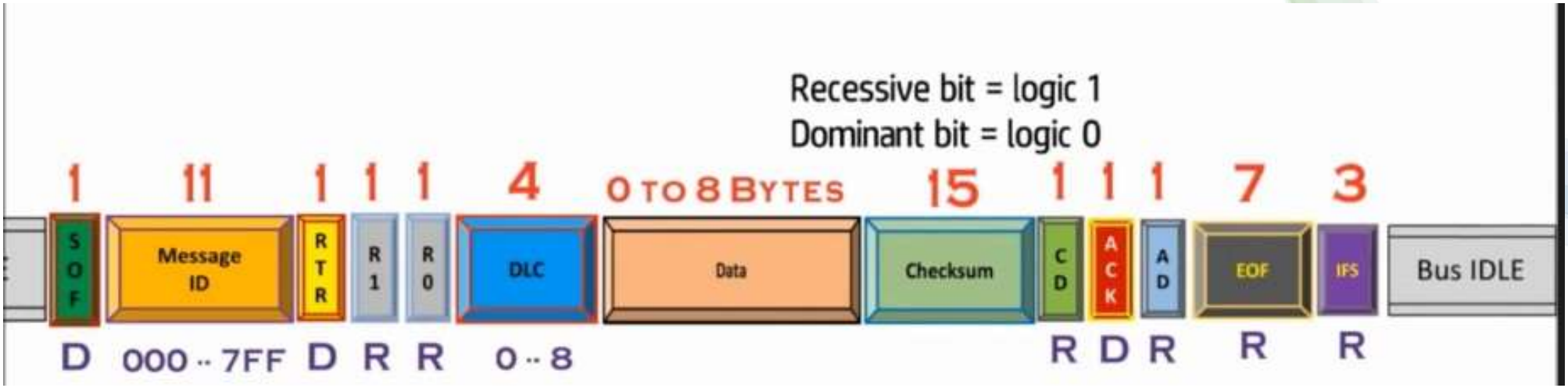
- Error Frame signals an error condition in the data frame being transmitted currently.

- If a node detects an error in the data frame being transmitted on the bus currently, then it destroys that data frame and signals all other nodes by transmitting a error frame.
- The error can be detected by any of the receiver nodes or the transmitter node.

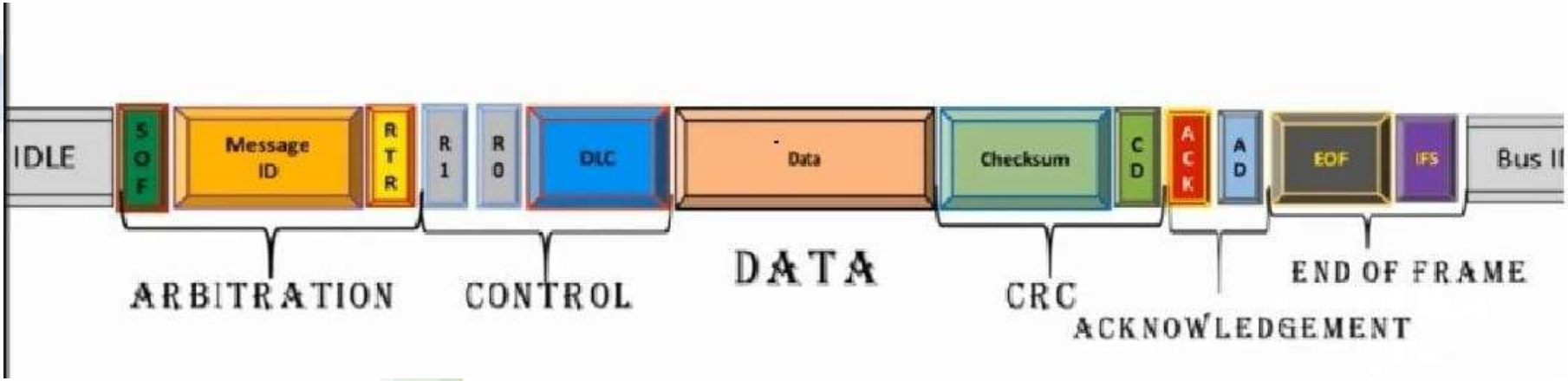


Overload Frame

- Overload frame is transmitted by a node when it is overloaded and needs some time to process the data frame received previously.
- Strategy is that since the overloaded node needs some time so it buys that time by keeping the bus busy with its overload frame preventing other nodes to start transmitting data frames.
- Maximum of 3 consecutive overload frames can be transmitted by each node after a data frame.



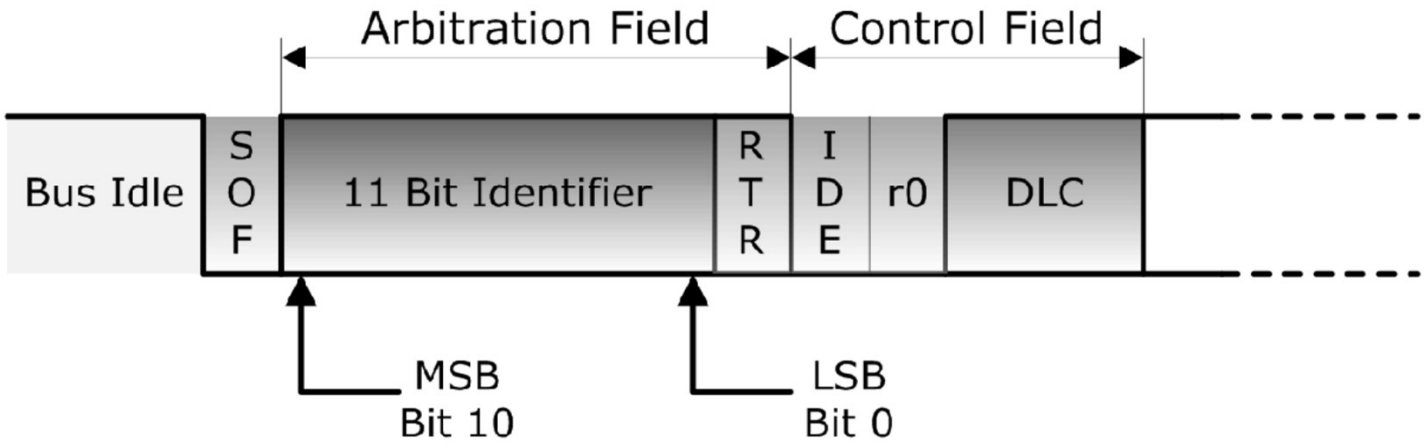
Frame Format



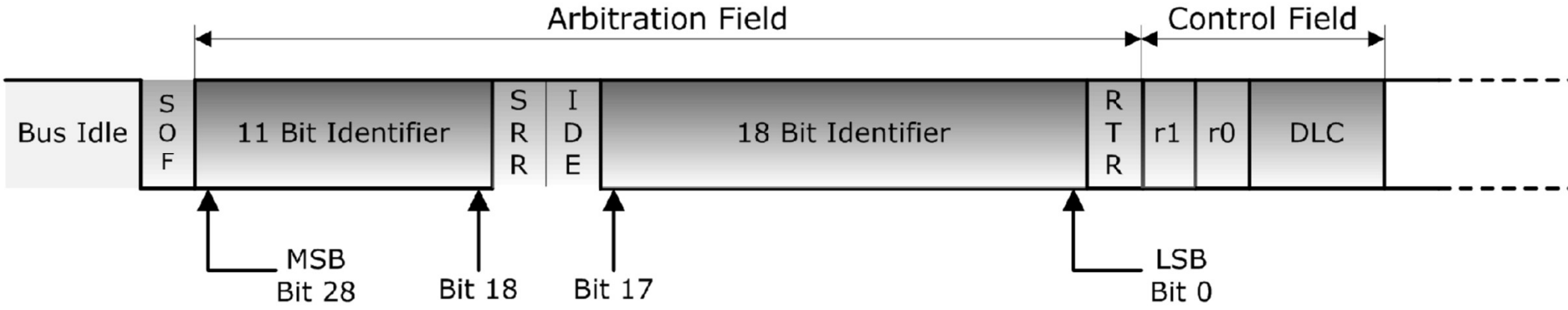
Extended Frame Format

Why Extended Frame

- CAN is a message based protocol
- We know with standard frame format the message ID being 11 bits we can have maximum of 512 frames per network.
- In real time scenario, this number is not always sufficient.
- So a extended frame format was proposed where the message ID is 29 bit long.
- In a network both Standard CAN and extended CAN frames can co-exist.
- These frames are differentiated by a new bit called as IDE(Identifier Extension Indication bit).



Standard Format: 11 Bit Message Identifier



Extended Format: 29 Bit Message Identifier

Additional Bits

SRR: Substitute Remote Request → In place of RTR bit in Standard frame format.

IDE: Identifier Extension Indication Bit → Indicates the presence of Extended Identifier or not and hence differentiates Standard frame and extended frame.
Dominant (0): Standard frame
Recessive (1): Extended frame

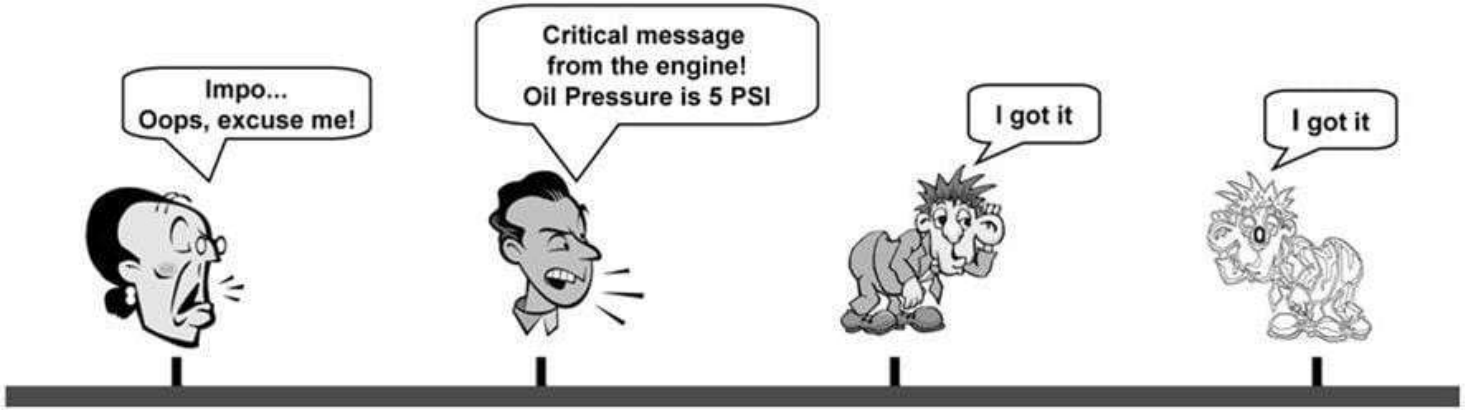

CAN Bus Arbitration

What is Bus Arbitration ?

- If multiple nodes wants to transmit frame simultaneously on the bus, the conflict is resolved by a process called Bus Arbitration
- SOF, Message ID and RTR bits participate in bus arbitration and are together called Arbitration field.
- The node which wins the bus arbitration becomes the Transmitter node and gets to send its complete frame and other nodes become Rx node.
- The nodes which lost arbitration will try to transmit their frames again once the bus becomes IDLE
- If Two nodes tries to put different bits on the bus simultaneously the bus will take the dominant bit value and ignore the Recessive bit

Bus Arbitration

- Arbitration – needed when multiple nodes try to transmit at the same time
- Only one transmitter is allowed to transmit at a time.
- A node waits for bus to become idle
- Nodes with more important messages continue transmitting



CAN Bus

Main Rules of Bus Arbitration

- Bus is considered idle after transmitted message plus Intermission Field

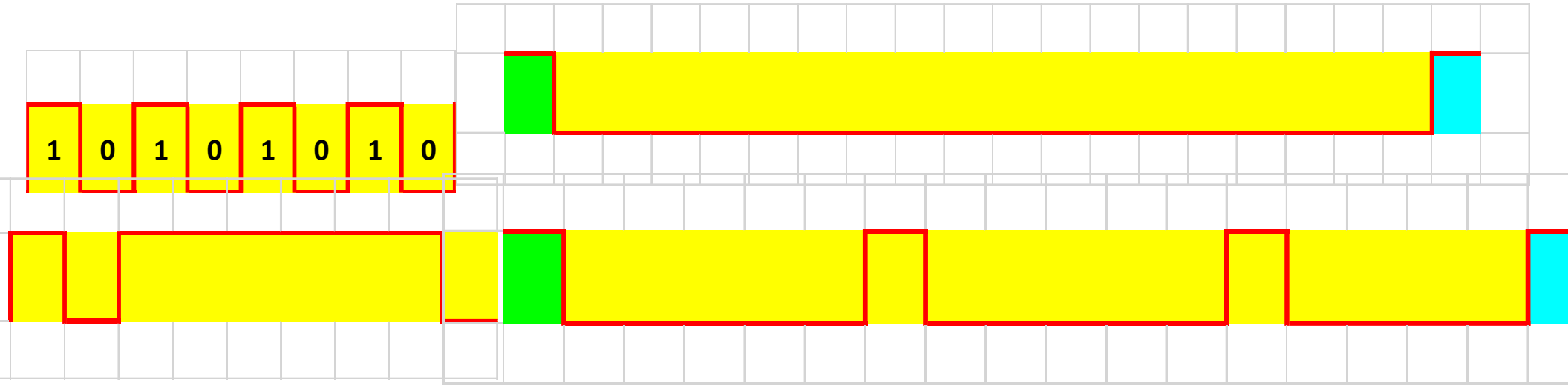
- Node that transmits message with lowest ID (highest priority) wins arbitration, continues to transmit. Competing nodes switch to receiving mode.

- Nodes that lost arbitration will start new arbitration as soon as bus is free for access again => No message gets lost

Bit Stuffing

Need for Bit Stuffing

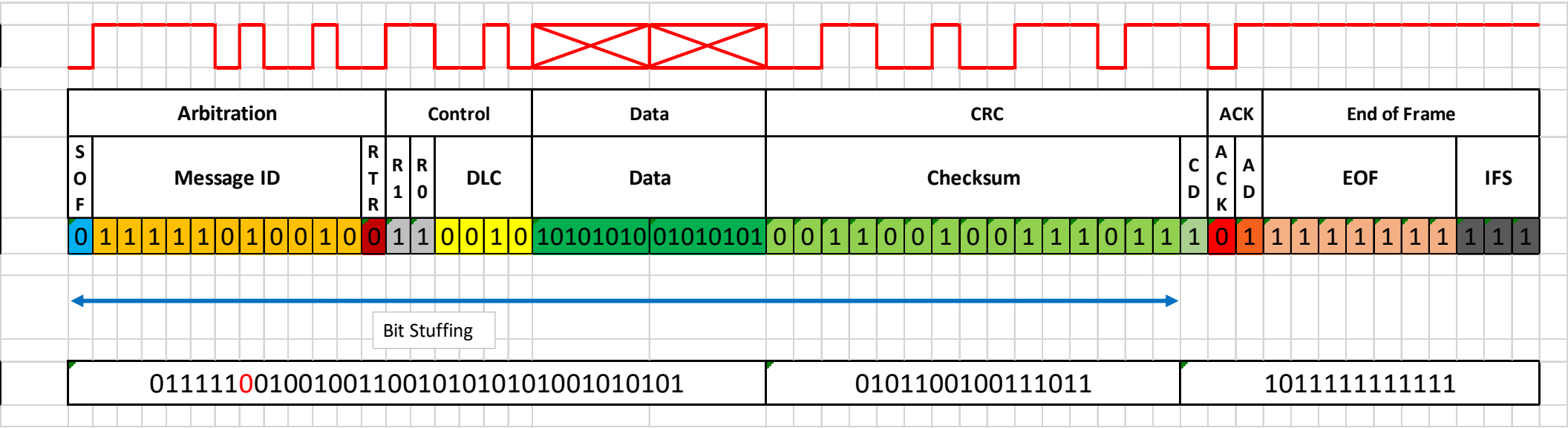
- CAN Protocol is a Asynchronous Protocol.
- There is no Clock signal to tell which bit is currently on bus or to synchronize between Rx nodes with Tx nodes.
- In CAN network all Rx nodes synchronize initially with Tx node with falling edge of SOF bit.



- So having bus state constant or unchanged for long time is risky in terms of Rx nodes being in synch with Tx node.

- If CAN bus has constant value for 5 bit lengths atleast then a complimentary bit is inserted next to it. This is called as bit stuffing

Which part of frame is bit stuffed



- The Bit stuffing is applicable from beginning of SOF till end of checksum field.
- Bit stuffing is not applicable for Fixed form part (i.e CD to IFS)
- Stuffed Bits are considered for CRC computation in CAN frame
- Tx side Frame is bit stuffed and sent on bus → Bit Stuffing
- Rx side the stuffed bits are removed and then processed further → Bit DeStuffing

CAN Errors

What is Can Error ?

- All the Rx nodes must receive the same information which the Tx application is intended. If some problem happens in this then we call as CAN error.

- The problem may happen in Tx node, or CAN bus or Rx node. CAN protocol has capability to detect any such problem and raise an error flag.

- So Depending on which node detects an error, we have Tx errors and Rx errors

Reaction to CAN Error

- When a node (Tx or Rx) detects a CAN error it starts sending the error flag on the bus immediately.
- This Error flag is designed in a way that it destroys the data frame currently present on the bus.
- Depending on this error flag other nodes also may put its own error flags on CAN bus.
- All Error flags and delimiters together forms the error frame on the bus.

CAN Errors

There are 5 types of CAN errors

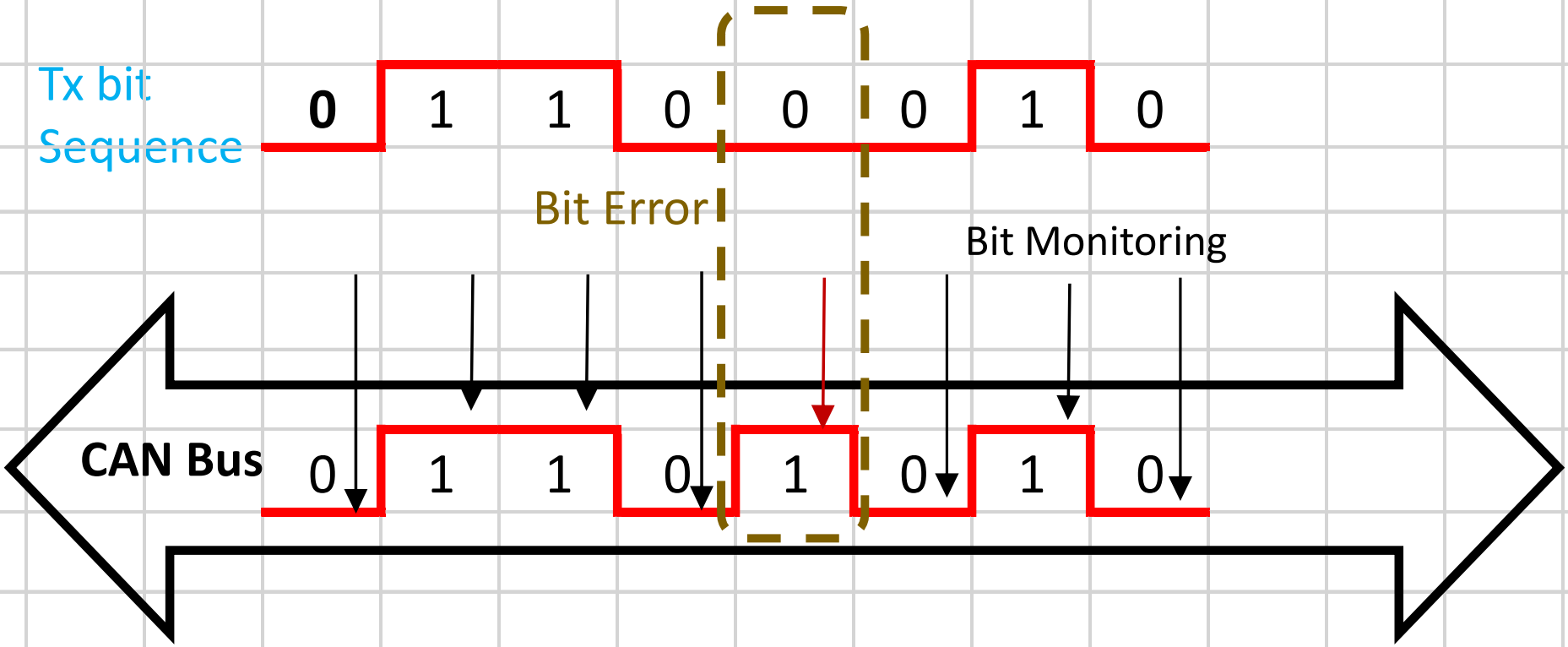
- Bit Error
- Acknowledgement Error
- CRC Error
- Stuff Error
- Form Error

✓ Bit Error and Ack Error are called Transmitter errors

✓ CRC, Stuff and form errors are called Receiver errors

Bit Error

When the transmitter node monitors a different bit value than the value it transmitted in that bit, then its called as a bit error



Bit Error Exceptions

Use cases:

1. Tx puts Dominant (0) bit but finds a Recessive(1) bit on the bus
2. Tx puts Recessive (1) bit but finds a Dominant (0) bit on the bus

Exceptions:

1) During Arbitration, Tx puts a Recessive bit but finds a Dominant Bit then this means ***Lost in Arbitration*** and not ***Bit Error***.

Note: During Arbitration if Tx node puts a Dominant bit but finds a Recessive bit on the bus, then this is a Bit Error only.

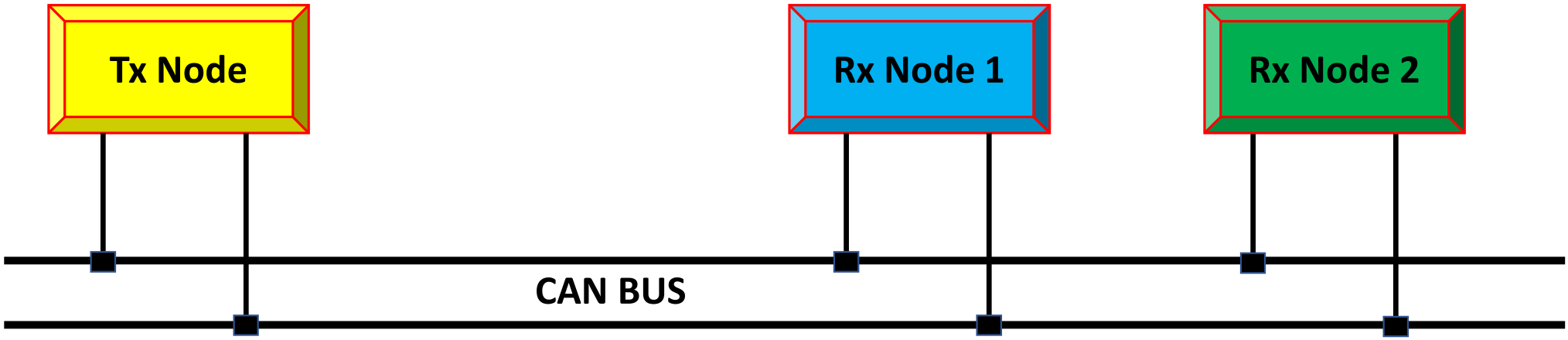
2) During Acknowledgement Slot, Tx node puts a Recessive bit but finds a Dominant Bit then this means ***Acknowledgement Received*** and not ***Bit Error***.

Acknowledgement Error

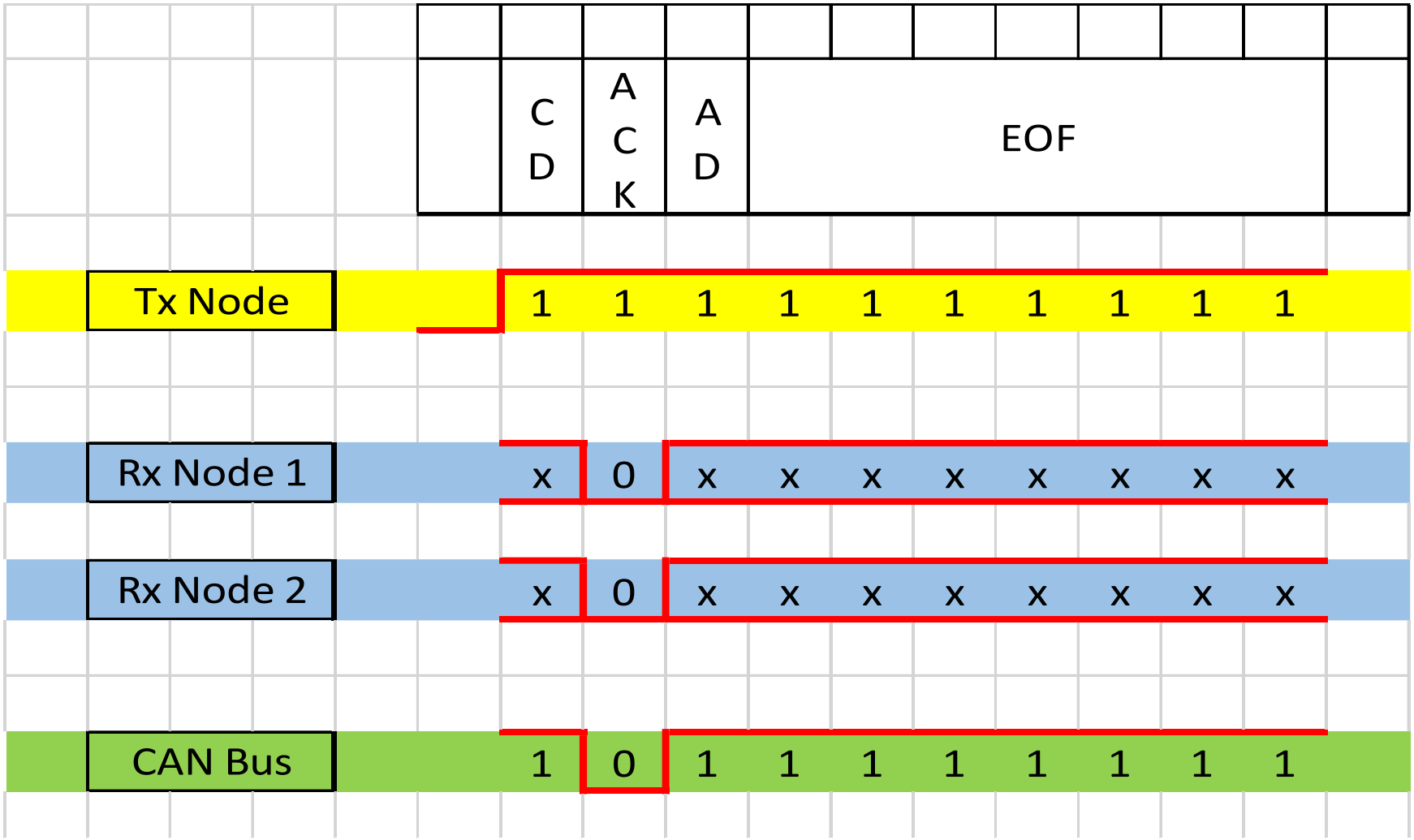
When the transmitter node monitors a Recessive bit value in Ack Slot, then it means it did not get a Acknowledgement. This is Acknowledgement Error

CRC Error

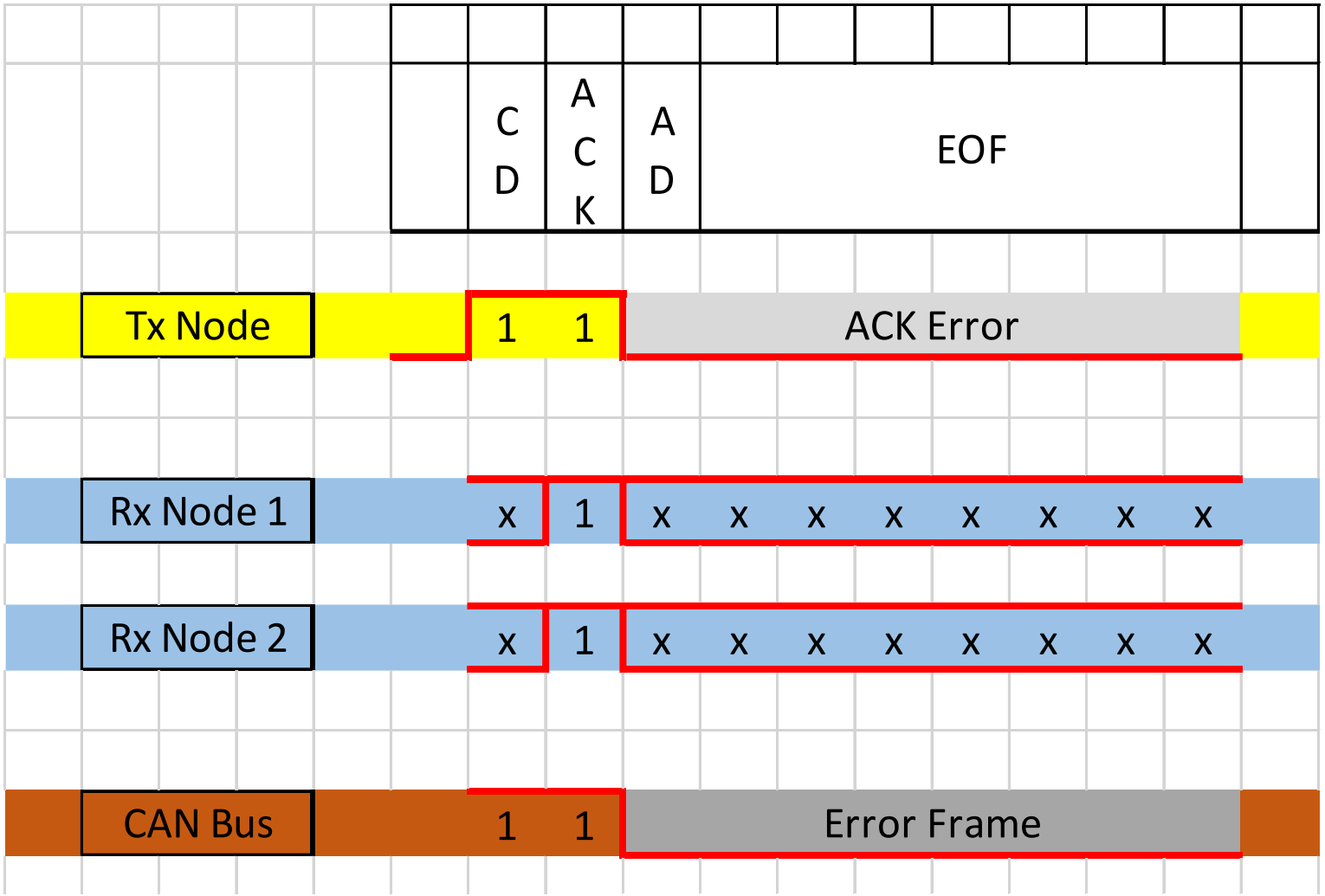
When the Reciever Node has not given acknowledgement but still monitors a dominant bt in ACK slot, then it’s a CRC Error.



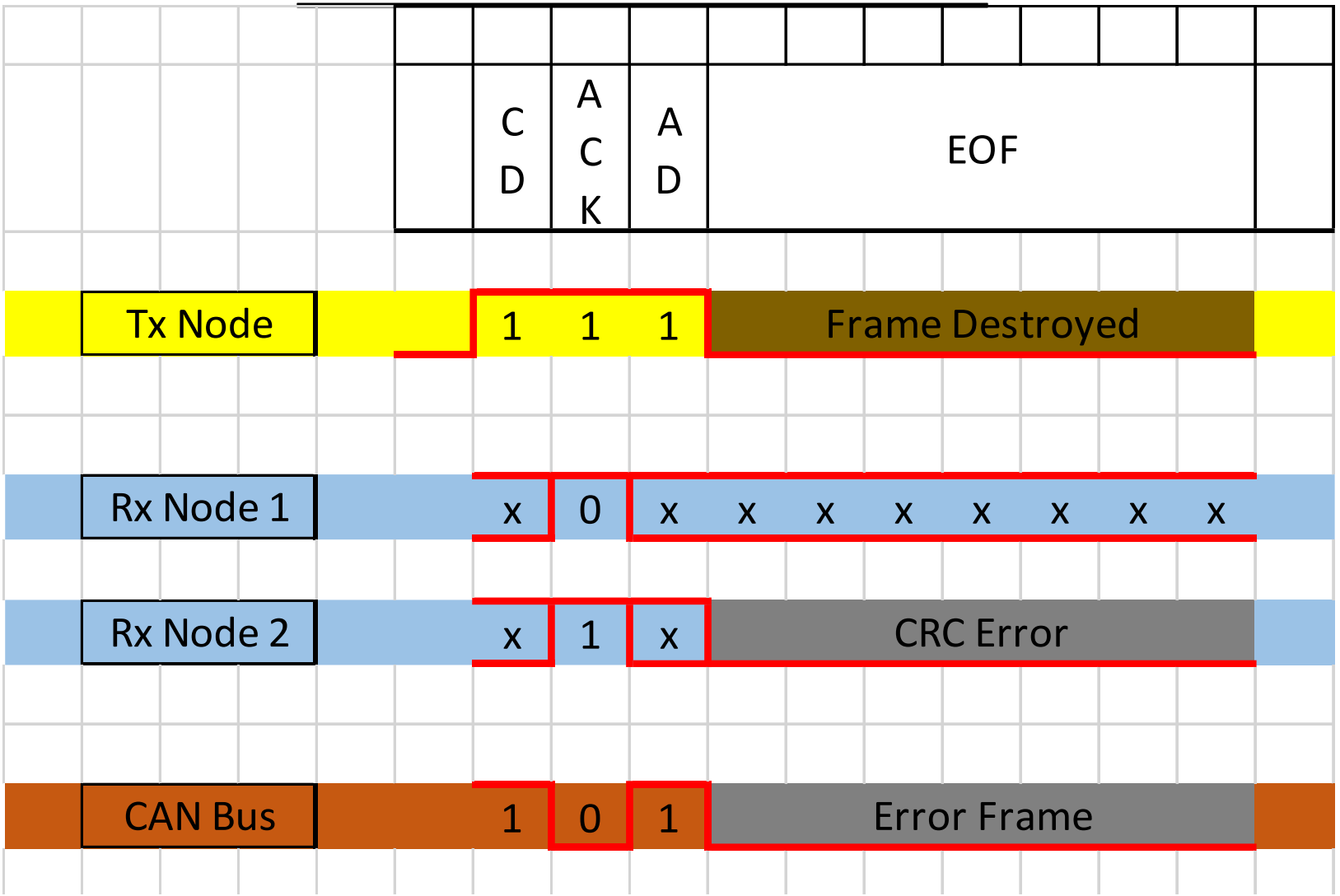
Case 1: No Error



Case 2: Acknowledgement Error



Case 3: CRC Error



Stuff Error

When a receiver receives at-least 6 consecutive bits of same polarity, then it means the bit stuffing rule is violated. This is Stuff Error.

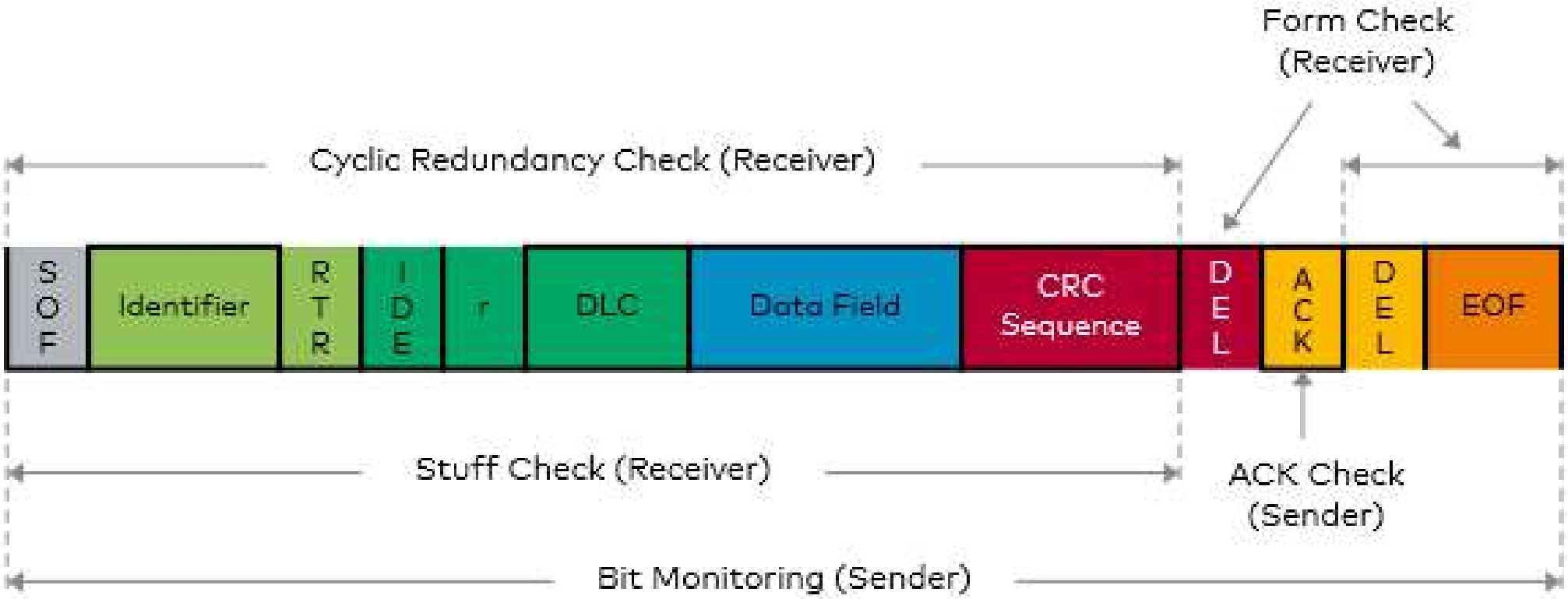
- In CAN protocol, if frame has consecutive 5 bits of same polarity then a complimentary bit is inserted after that called Stuff bit. This helps in better synchronization between Rx and Tx nodes.

- Bit Stuffing is applied from SOF to Checksum field. So the scope of stuff error also is the same.

- The moment the Stuff error is detected, the Rx node raises an error flag the very next bit, hence destroying the current data frame on the CAN Bus.

Form Error

When the Fixed form of the CAN data frame (CD + AD + EOF) is altered then Form error occurs.



CAN FD Intro

What is CAN-FD ?

- CAN-FD is a communication protocol based on CAN.

- Can-FD is an improvement over CAN.

- Modern CAN controllers support both CAN and CAN-FD frames
- Note: CAN-FD is not an substitute for CAN protocol. A single Automotive network can have both CAN frames and CAN-FD frames.
- So CAN-FD protocol must be compatible with CAN protocol.

Advantages of CAN-FD over CAN

- **DATA:** CAN-FD frame can carry upto 64 bytes of data where as CAN carries 8 bytes

- **SPEED:** CAN-FD frame maximum speed upto 5Mbps whereas CAN frame upto 1Mbps.

- Note: Data and CRC fields can be transmitted with higher speed. Remaining fields of the frame will be transmitted with lower speed same as CAN speed.

Thank You