In-Vehicle Security

CPS and IoT Security

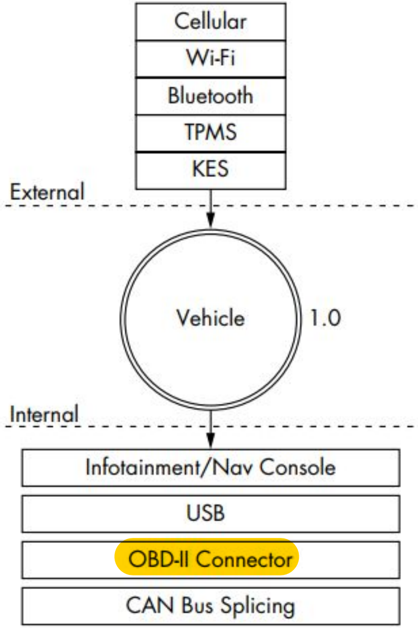
Alessandro Brighente

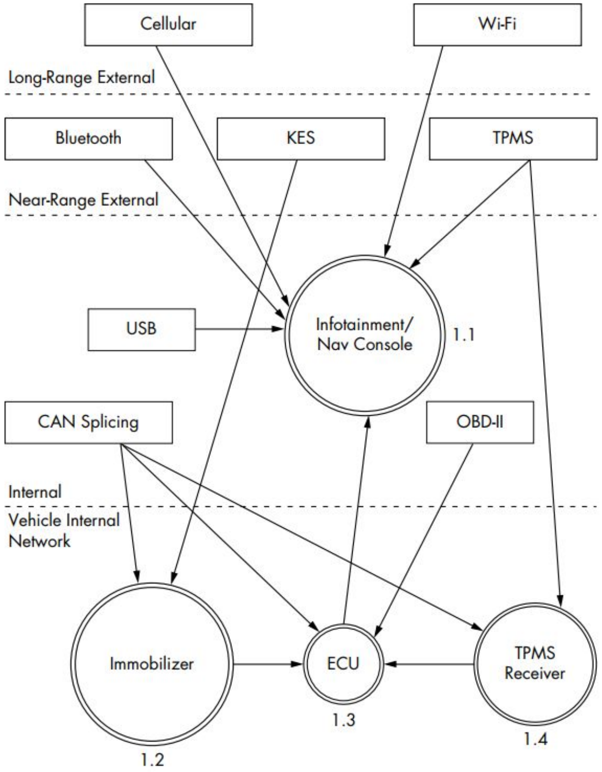
Master Degree in Cybersecurity

Università degli Studi di Padova

SPRITZ Security & Privacy Research Group

What is there in a Car?





Vehicle Components and Network

- Modern vehicles contain a large number of Electronic Control Units (ECUs)
- ECUs are embedded systems that control one or more (sub)system in a car or other vehicles
- Examples: engine control module, powertrain control module, transmission control module, suspension control module,...
- ECUs are nodes in a rather complex in-vehicle network, where they should communicate one another to report many different types of information

What is CAN Bus

- Controller Area Network (CAN) is an in-vehicle network bus-based standard developed in 1986 by Bosch
- It allows in-vehicle components (ECUs) to communicate one another
- It has broad application in automotive systems, including power train, suspension, and braking
- In 2003, it became a standard series with ISO 11898
- The Society of Automotive Engineers (SAE) standardized CAN bus communications to have asynchronous data rate up to 1 Mbps at a 40 m distance

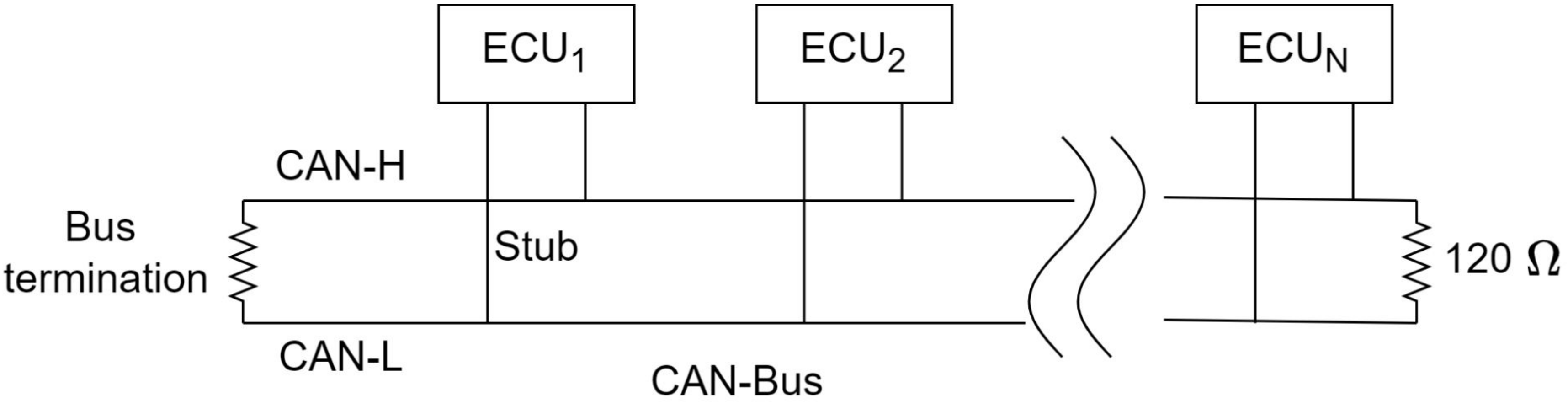
Why CAN Bus

- **Robustness:** ideal for safety applications thanks to its durability and reliability (bit stuffing, bit monitoring, frame check, ack and CRC checks)
- **Low cost:** objective is to reduce errors, weight, wiring and costs
- **Flexibility:** it is a message-based protocol, so nodes can be added or removed without updating the system
- **Efficiency:** messages with high priority are clearly marked and have prioritized access to the bus

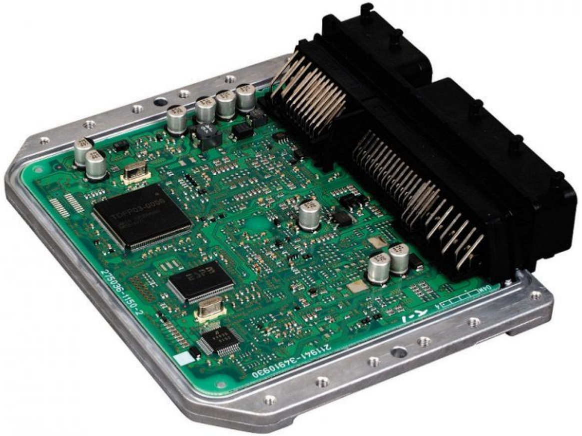
CAN Bus Wiring

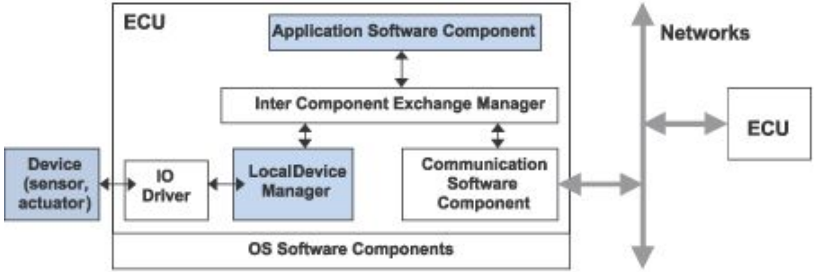
- To achieve reliability, it is important to prevent message collision such that no data gets lost
- Thanks to the shared bus and the message handling system, CAN allows for a reduced number of wires to achieve reliability
- Redundancy is the best way to achieve reliability, however it comes with a higher communication implementation cost (huge number of wires)
- Point-to-point communications were used before CAN

CAN Bus Architecture



Electronic Control Unit

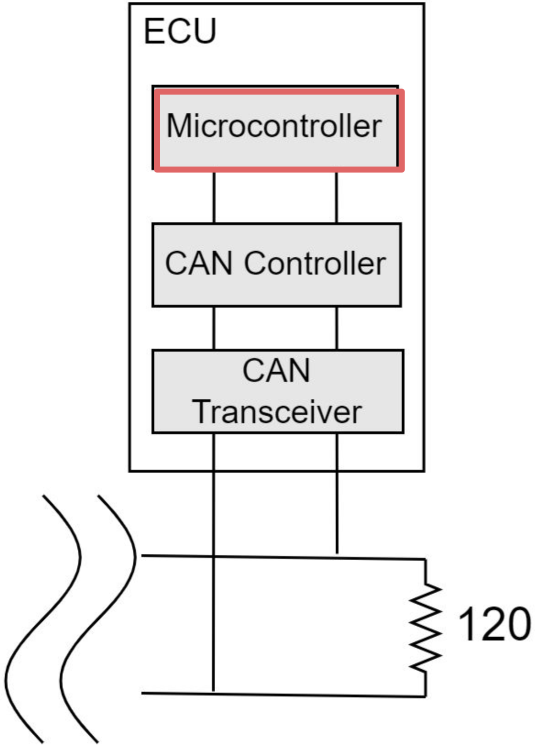




ECU Architecture

Microcontroller:

- central processing unit of the ECU to decide what the received signal means and what messages it wants to transmit
- Allows for the connection of other devices, such as sensors and actuators

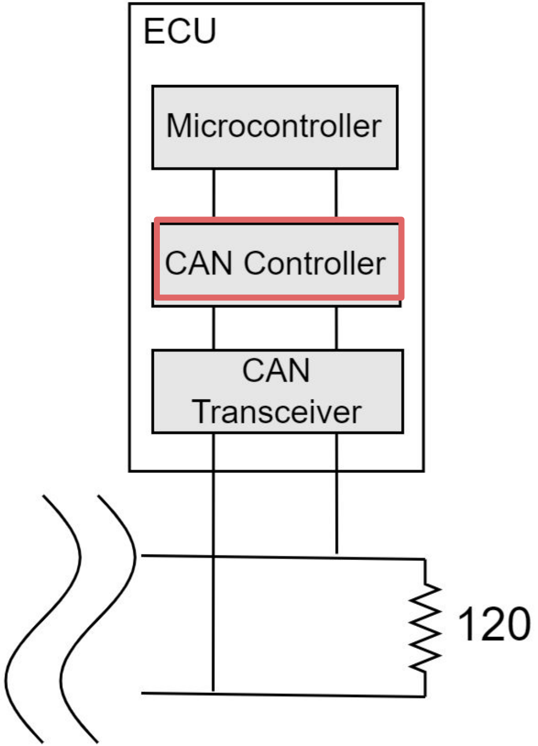


SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

CAN controller:

- Stores the received serial bits and passes messages to the processor
- Transmits bits serially on the bus upon reception from the processor

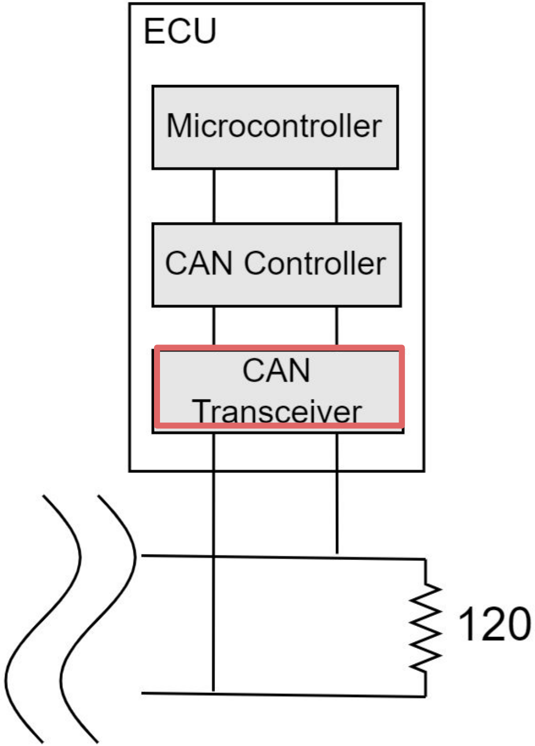


SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

CAN transceiver:

- Protects the CAN controller from overvoltage
- Converts CAN bus levels to levels that the CAN controller can use
- Converts the data stream from the CAN controller to CAN bus levels



CAN Bus Communication

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

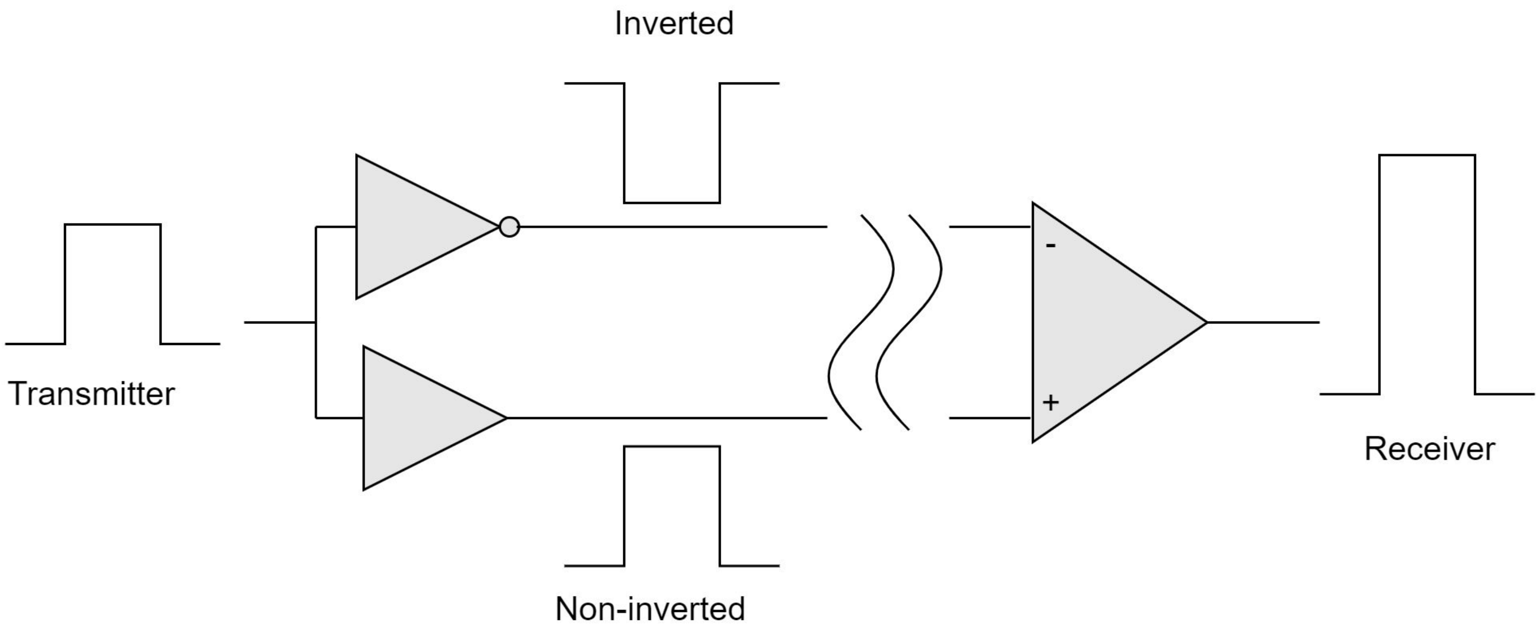
- CAN bus uses **differential** **wired-AND signals**
- It uses two signals, CAN high (CAN-H) and CAN low (CAN-L)
- They are either driven to a dominant state with CAN-H > CAN-L
- Or driven to a recessive state, with CAN-H $\leq$ CAN-L
- Dominant state = 0 bit
- Recessive state = 1 bit

CAN Bus Communication

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- CAN bus uses **differential** wired-AND signals

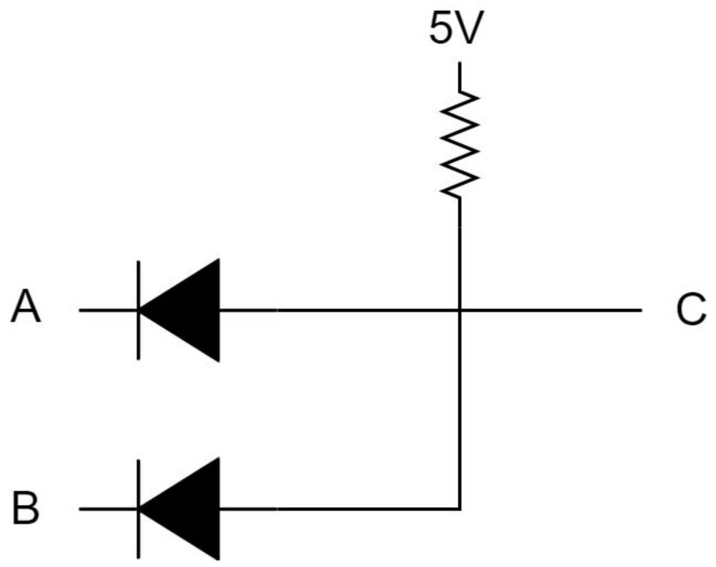


CAN Bus Communication

SPRITZ Security & Privacy Research Group
Università degli Studi di Padova

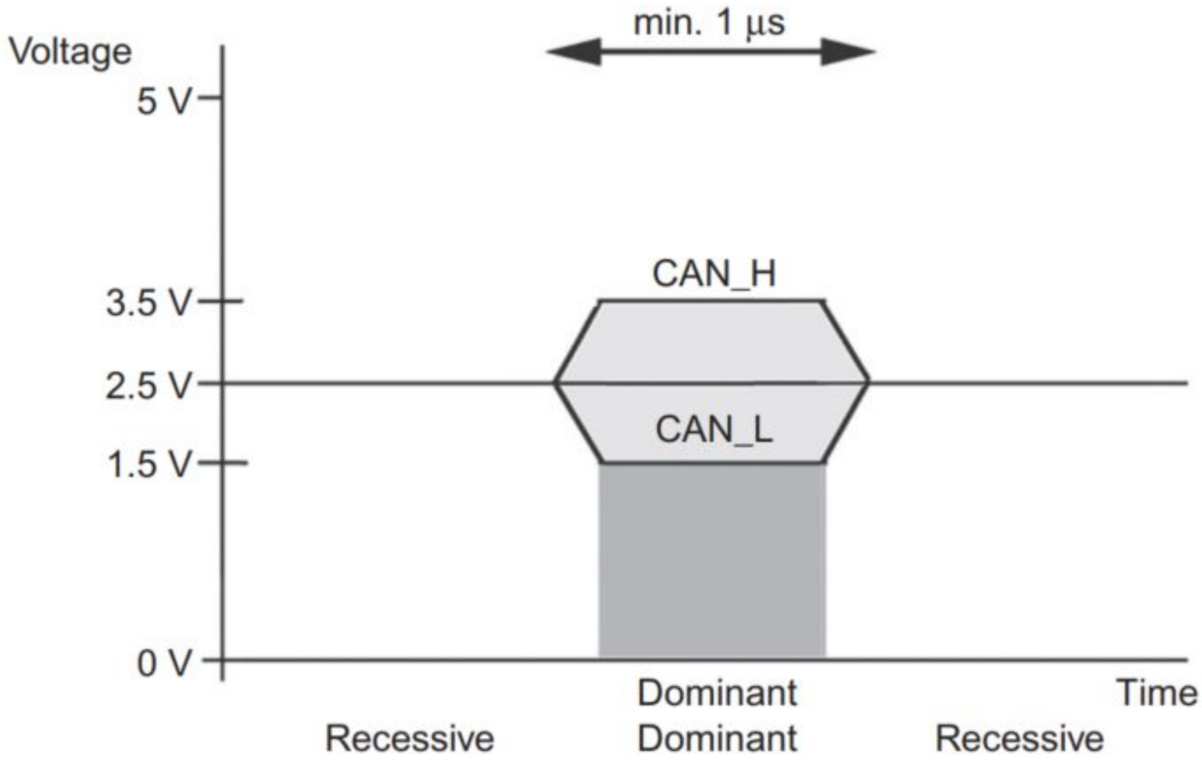
- CAN bus uses differential **wired-AND signals**

Input		Output
A	**B**	**A AND B**
0	0	0
0	1	0
1	0	0
1	1	1



CAN Bus Communication

- Either dominant or recessive

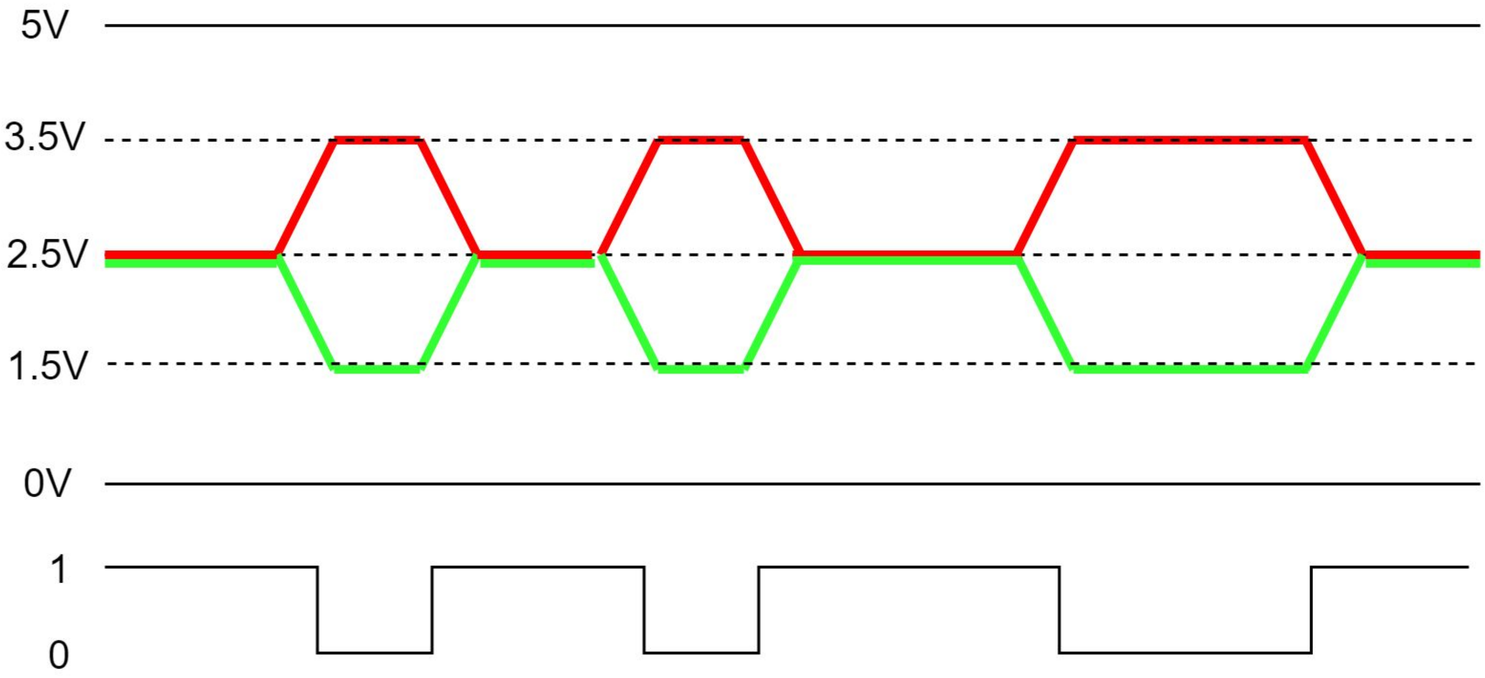


CAN Bus Communication

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- Dominant state = 0 bit
- Recessive state = 1 bit

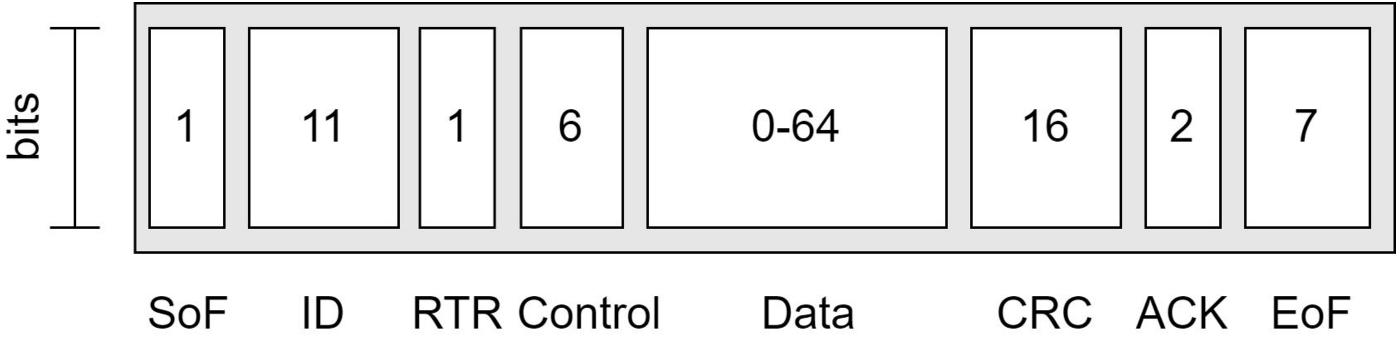


Physical and Data Link Layer

- The CAN bus is described by a data link and physical layer
- ISO 11898-1 describes the data link layer, while ISO 11898-2 describes the physical layer
- From ISO 11898-2:
  - CAN nodes must be connected via a two-wire bus with baud rates up to 1 Mbit/s or 5 Mbit/s (CAN FD)
  - Maximal cable lengths should be between 500 m (125 kbit/s) and 40 m (1 Mbit/s)
  - The CAN bus termination must be a 120 Ohms resistor

CAN Frames

- The standard CAN frame is composed by 8 message fields
- CAN 2.0A and 2.0B differ in the length of the ID field



Message fields of the CAN frame

CAN Frame Message Fields

- **Start of Frame (SoF):** dominant 0 to tell the other a node wants to talk
- **ID:** frame identifier. The lower the ID, the higher the priority. It is not an identifier of the sender
- **Remote Transmission Request (RTR):** indicates whether a node sends data or requests data from another node
- **Control:** contains the Identifier Extension Bit (IEB) which is dominant 0 for 11 bits. It also contains the 4 bit Data Length Code (DLC) that specifies the length of the data bytes to be transmitted (0 to 8 bytes)

CAN Frame Message Fields

SPRITZ SECURITY & PRIVACY RESEARCH GROUP

UNIVERSITÀ DEGLI STUDI DI PADOVA

- **Data:** payload in common network terminology. CAN signals that can be extracted and decoded for information
- **CRC:** check for data integrity
- **ACK:** indicates whether a node has acknowledged and received data correctly
- **End of Frame (EoF):** end of the CAN frame

Validity of the CAN Frame

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- CAN frames need to satisfy certain conditions to be valid
- If a node detects its transmission of an erroneous message, it takes actions accordingly
- This is the CAN bus error handling, and each node has its own error counter and state
- The state is in the set (active, passive, bus off) depending on the counter value

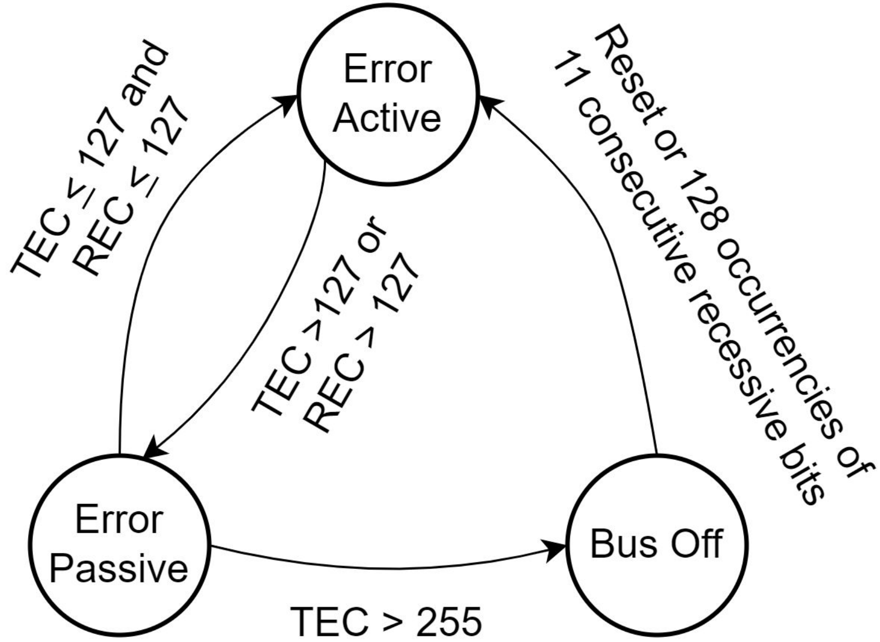
Validity of the CAN Frame

- Counters are Transmit Error Counter (TEC) and Receive Error Counter (REC)
- TEC = TEC+ 8 if an error occurs during transmission
- REC = REC + 1 if an error in the reception
- Success decreases the counter by one in both cases
- Nodes start at the error active frame
- A node transitions to the error passive state if the value of the counter exceeds 127

Validity of the CAN Frame

- In the error passive state the node can only write recessive error flags, thus not affecting the bus traffic
- The node transitions to the Bus off state is the counter exceeds 255
- In this case, the node no longer takes part to the bus traffic
- To get back to error active, the node needs to be either reset or should count 128 occurrences of 11 consecutive recessive bits

Validity of the CAN Frame



Error detection and fault confinement states

Bus Off Mode

- When an ECU transitions to the bus off mode it means that something serious is happening
- To prevent accidents but still allow the driver to reach a safe place, the ECU in bus off usually runs with predefined parameters and reduced functionality (e.g., limited RPM)
- We call an ECU in this state as *limp home*
- In this state, warning limps are turned on on the driver's dashboard
- Depending on the severity, the limp mode is eventually disabled

Errors in CAN Bus Frames

- In CAN bus transmissions we can have no less than five types of errors

1. *Bit error*: occurs when an ECU, after comparing its transmitted bits with those on the CAN bus detects a mismatch (does not hold in arbitration mode)
2. *Stuff error*: after transmitting five consecutive same polarity-bits, the ECU sends an opposite bit to maintain soft synchronization. An error occurs if stuffing does not happen
3. *CRC error*: computed one different from the received one

Errors in CAN Bus Frames

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- In CAN bus transmissions we can have no less than five types of errors

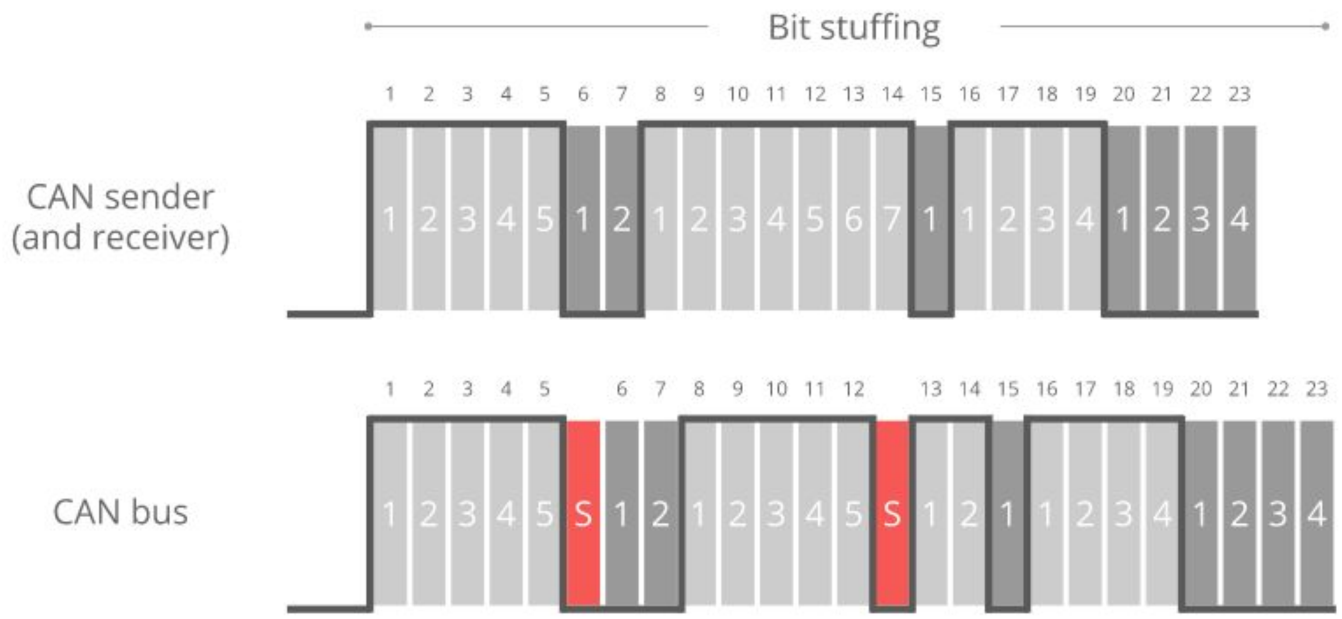
4. *Form error*: if a fixed-form bit field (e.g. EoF) has a bit error the ECU raises a form error
5. *ACK error*: when a node receives a frame it responds with a dominant bit in the ACK frame. If none respond, then ACK error

Error Flags

- When a node detects an error it needs to warn the others on the bus
- The node raises an *error flag* which may come in two forms: active and passive
- Nodes that are in **error-active mode** issue an *active error flag* which consists of 6 dominant bits
- The transmitted frame causes other nodes to violate the bit-stuffing rule, transmit their own error frame caused by the stuff error, and terminate any on-going transmissions or reception

Something More on Errors

SPRITZ Security & Privacy Research Group
Università degli Studi di Padova

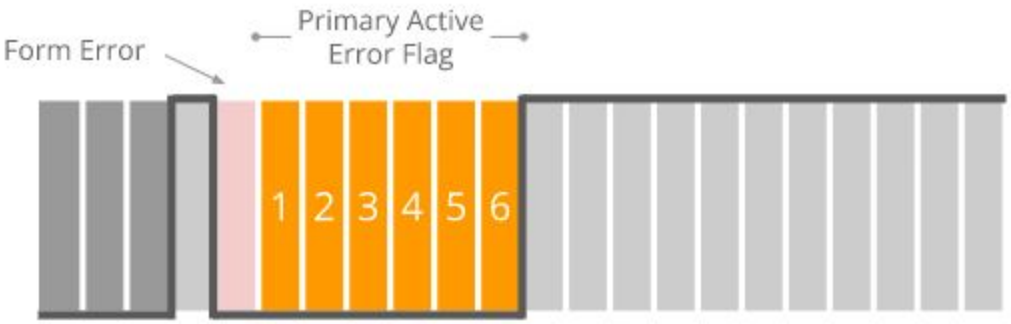


Every 5 consecutive bits of same polarity, add one for synch

See source on: https://www.csselectronics.com/pages/can-bus-errors-intro-tutorial

Error Flags

- Bit stuffing error is a violation of this rule and is visible to all CAN nodes
- Hence, we can have a bit error that makes the transmitter raise this flag, and all other nodes raise this error to notify the overall CAN
- We call them as *primary Active Error Flag* and *secondary Active Error flags*



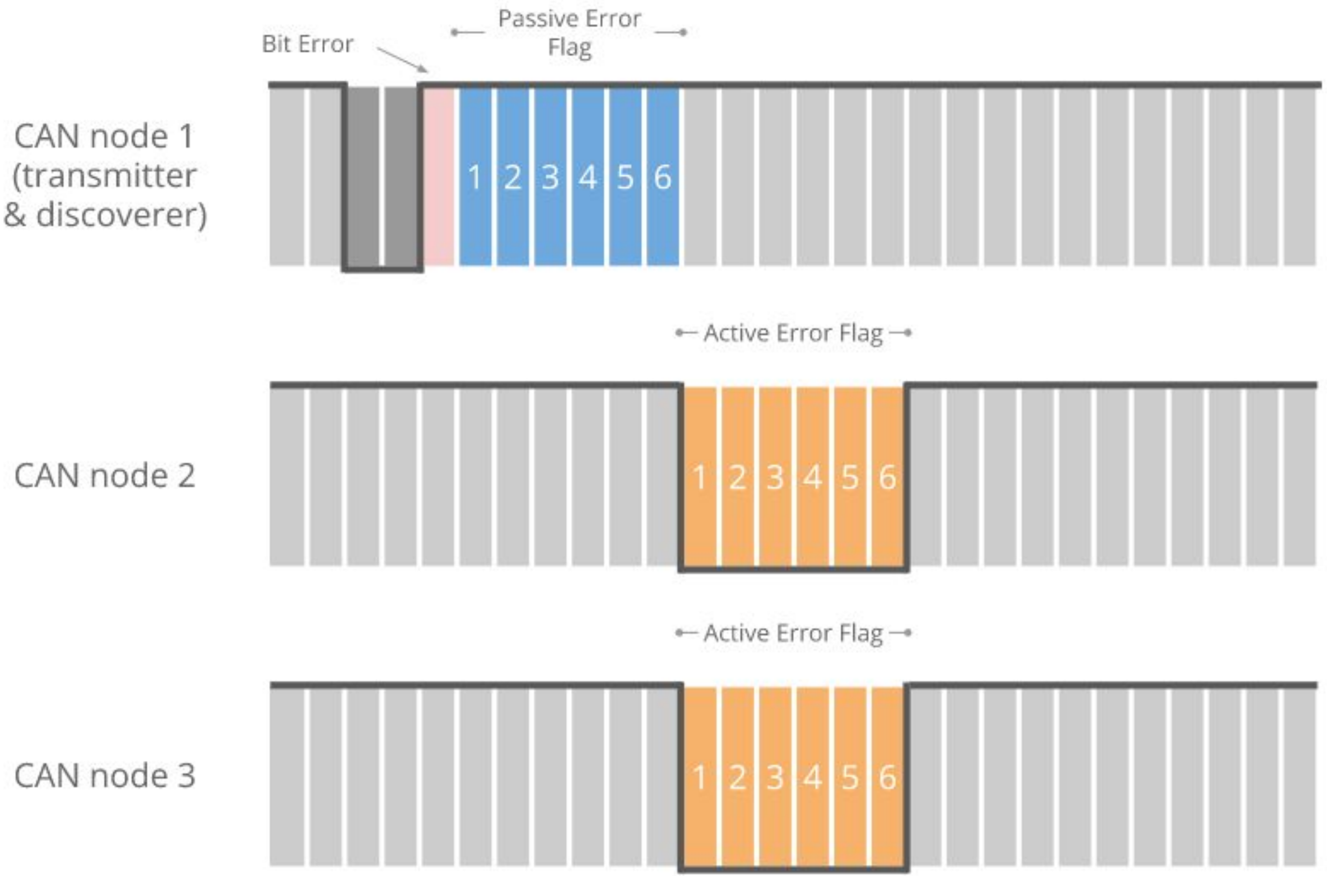
Passive Error Flags

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- When the node is in passive error mode, and is transmitting, it can only transmit recessive bits as error flag
- This is in turn detected as a bit stuffing error by all other nodes
- If all other nodes are in error active, they will create an active error flag
- If instead the error passive node raises an error passive flag while receiving, this will not be detectable by other nodes in th network

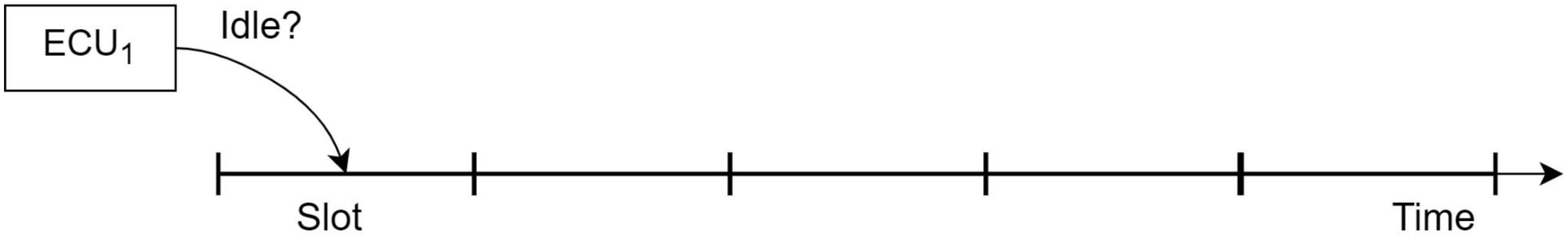
Passive Error Flags



Bus Arbitration

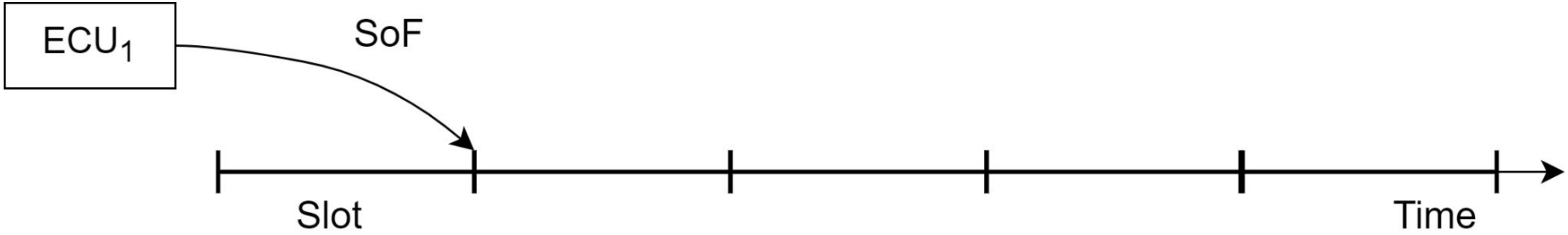
- It refers to the process by which nodes get a share of the bus resources to transmit
- CAN arbitration protocol is both priority-based and non-preemptive
- Non-preemptive: a message cannot be preempted by a higher priority one if this was queued after the transmission began
- The media access protocol alternates contention and transmission phases

Bus Arbitration



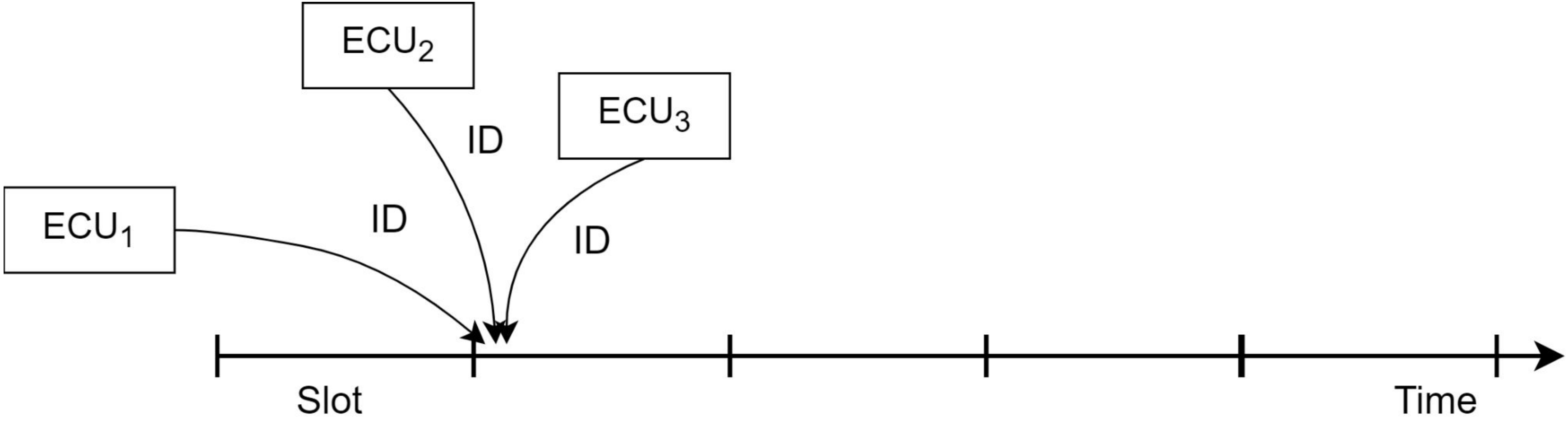
- Time is divided into slots, each with a length sufficient for a message to go back and forth in the whole bus
- A node wishing to transmit checks whether the channel is idle

Bus Arbitration



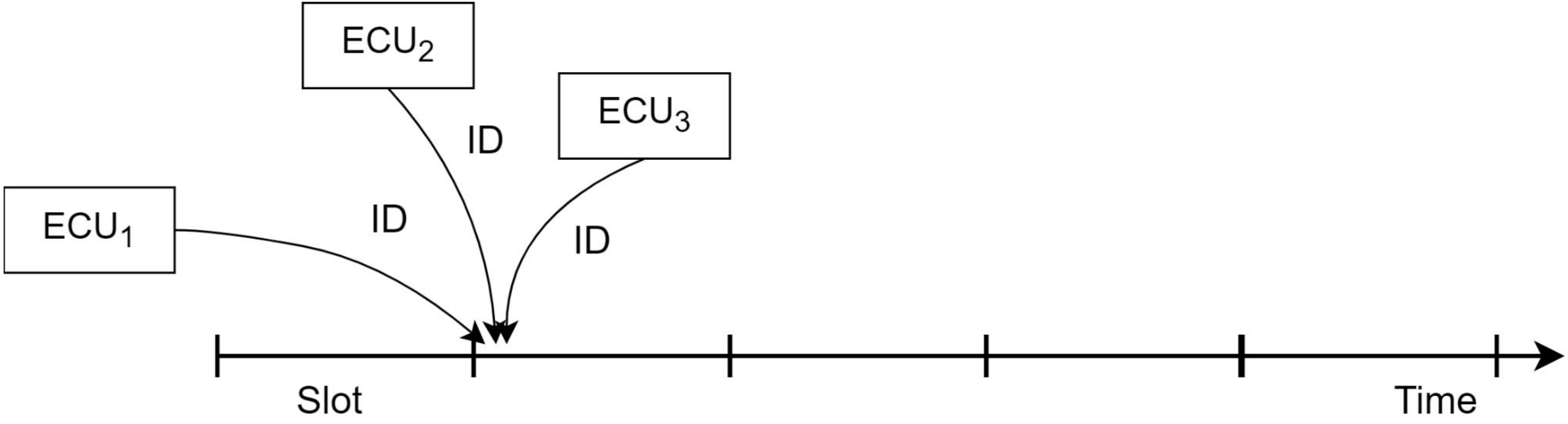
- If idle, it waits the next slot and start a contention phase by transmitting a SoF bit

Bus Arbitration



- All other nodes wishing to transmit send their ID
- The CAN controller compares its output with the actual bus level at the end of each bit cycle

Bus Arbitration



- The node loses contention in case it submitted a recessive level (high) and detects a dominant level (low)
- Error is dominant transmitted and recessive detected

Bus Arbitration Example

S1	1	0	0	0	1	0	0	0
S2	0	0	1	1	1	0	0	1
S3	0	0	1	1	1	0	1	0
Bus	0	0	1	1	1	0	0	1

S3 wins the arbitration at the 7th bit

Bus Arbitration

- After the node that wins the competition terminates its transmission with an EoF, there should be new transmissions
- The CAN bus uses a 3 bit inter-frame symbols to separate packets
- After this, the bus becomes idle again and arbitration shall commence again

Accessing CAN Bus

- To access CAN bus you need to connect to the On Board Diagnostic (OBD) port
- Basically all cars nowadays use the OBD-II standard port
- It is usually located near the driver's or passenger's seat
- To talk with the CAN bus you need a converter, such as that in the figure
- This converts CAN bus data to something readable via a USB port

The Bus-Off Attack

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- The bus-off attacks is a denial of service attack that exploits how ECUs access the bus
- Proposed by Cho et al. in 2016 (pretty recent, nah?)
- The goal of the attacker is to compromise the network and in particular multiple healthy (i.e., non compromised ECUs) with a minimal number of messages
- Notice that this is very different from flooding

Bus-Off Attack: Adversary Model

- The attacker can compromise an in-vehicle ECU either physically or remotely to gain its control
- We do not require the adversary to reverse engineer messages or checksums to be able to deliver the attack
- Once the ECU is compromised, the attacker can inject messages with any ID, DLC and data on the CAN bus

The Attack: Hint

- The error handling mechanism of CAN bus forces ECUs that measure a certain number of errors to go bus-off
- The attacker exploits this feature of CAN bus to iteratively isolate ECUs
- Injecting attack messages, the adversary coerces the TEC of an uncompromised/healthy victim ECU to continuously increase
- Eventually, the attacker forces the error confinement to force the victim or even the entire network to shutdown

The Attack

- Suppose that a victim V periodically sends a message M
- The attacker can hence deliver a successful bus-off attack by (all conditions must be fulfilled)
  - Using the same ID
  - Transmitting at the same time as M
  - Having at least a bit that is dominant in the attack message while being recessive in M (all preceding bits are equal)

The Attack

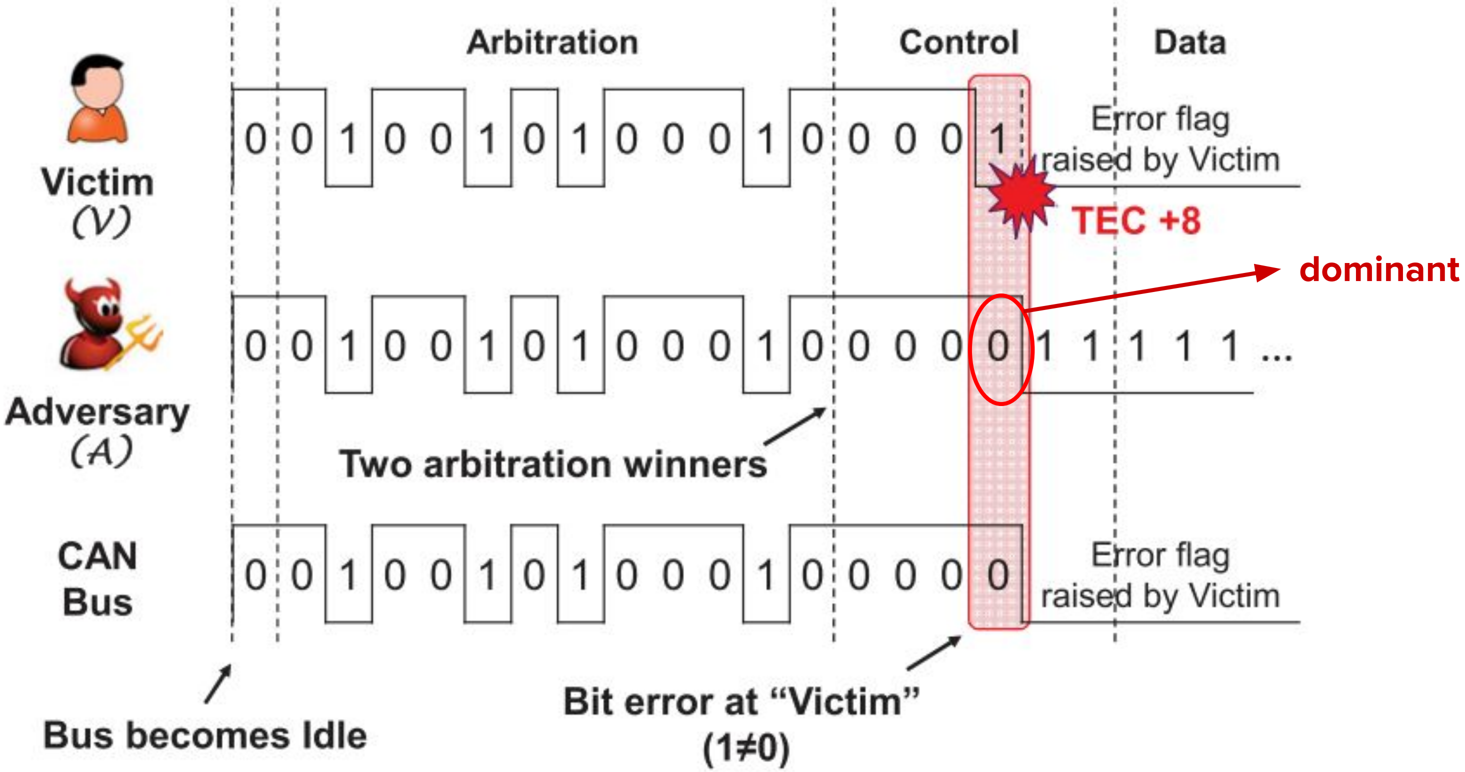


Figure from the original paper: "Error handling of in-vehicle networks makes them vulnerable"

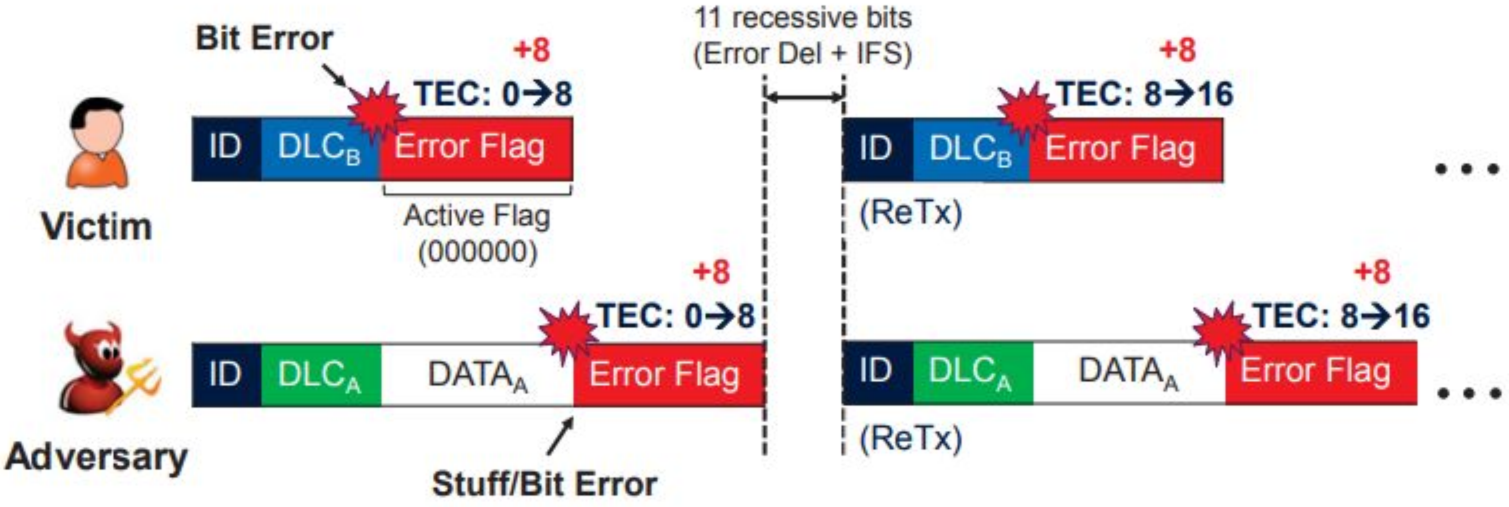
Attack Phases

- **Phase 1: Victim in Error-Active**
- Both adversary and victim start in error-active mode, and the attacker targets one of the victim's periodical messages
- The attacker sends the malicious message and the victim's TEC increases
- The attacker's TEC also increases, due to the stuff or bit errors triggered at the adversary node
- Both nodes automatically retransmit the failed message

Attack Phases

- **Phase 1: Victim in Error-Active**
- Both adversary and victim start in error-active mode, and the attacker targets one of the victim's periodical messages
- The attacker sends the malicious message and the victim's TEC increases
- The attacker's TEC also increases, due to the stuff or bit errors triggered at the adversary node
- Both nodes automatically retransmit the failed message

Phase 1



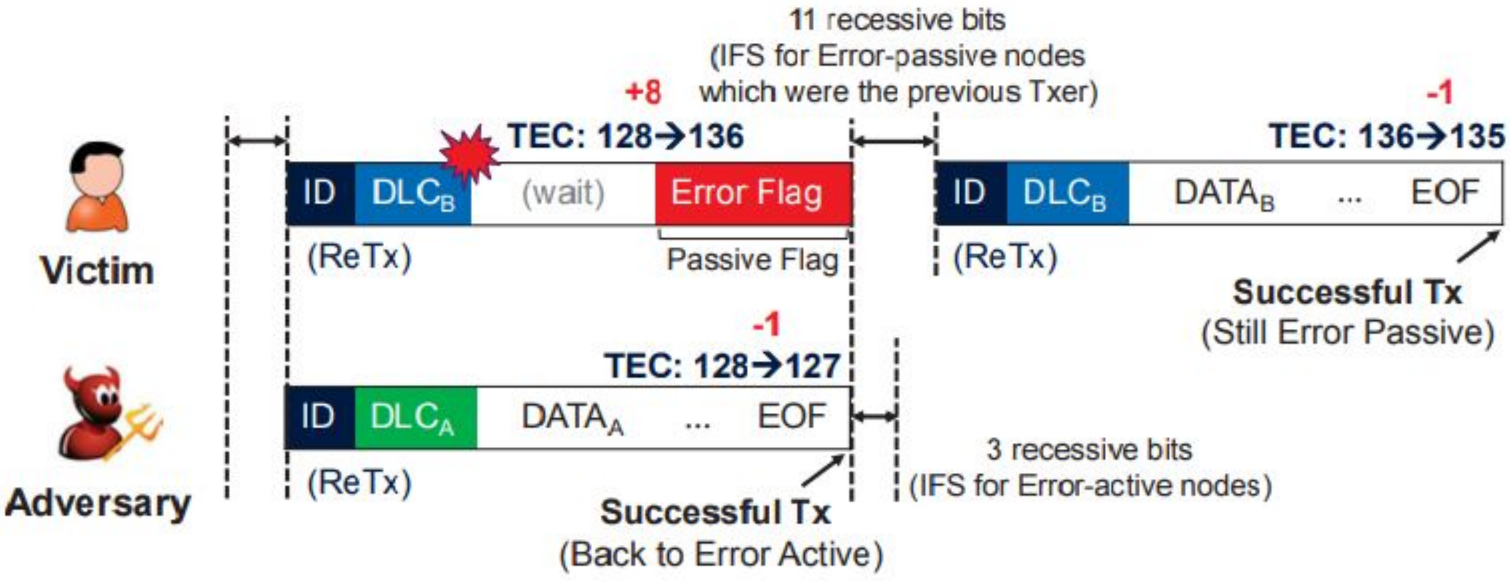
(a) Phase 1 – Victim in error-active mode.

- Thanks to the automatic retransmission, the attacker bring the victim to error-passive by sending a single message

Attack Phases

- **Phase 1 to 2**
- After 16 retransmissions, both victim's and attacker's TEC=128 ➔ error-passive
- As a consequence, the victim attempts the delivery of a passive error flag with 6 recessive bits
- The attempt to transmit this flag persists until the end of the attacker's frame, after which it is successful (TEC = TEC -1)
- Due to the successful transmission, the adversary does not go to error-passive

Phase 1 to 2



(b) Transition from Phase 1 to 2.

- TEC of victim and attacker differ

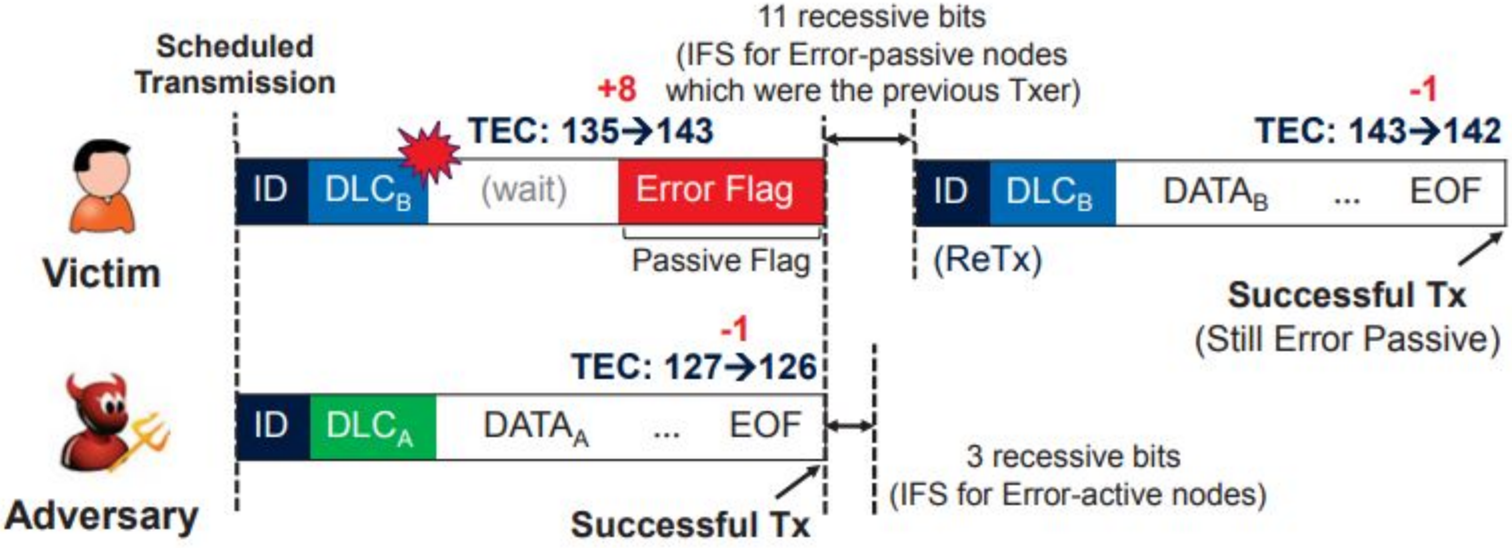
Attack Phases

- **Phase 2: Victim Error-Passive**
- Once the scheduled interval of the target message is elapsed, V retransmits M again
- At the same time, the adversary reinjects malicious M
- Since the victim is in error-passive, the attacker’s TEC decreases by 1, whereas the victim’s increases by 7 (+8 -1) thus maintaining the error-passive
- The attacker iterates until the victim’s TEC>255, i.e., bus-off

Phase 2

SPRITZ
Security & Privacy
Research Group

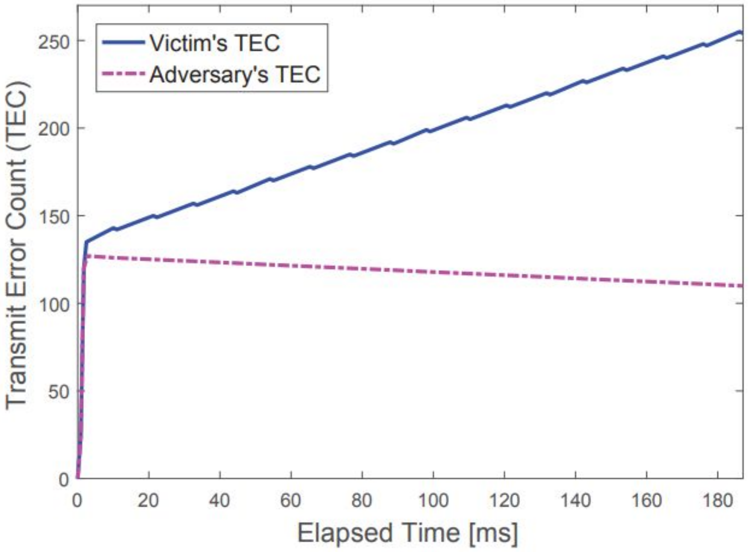
Università
degli Studi
di Padova

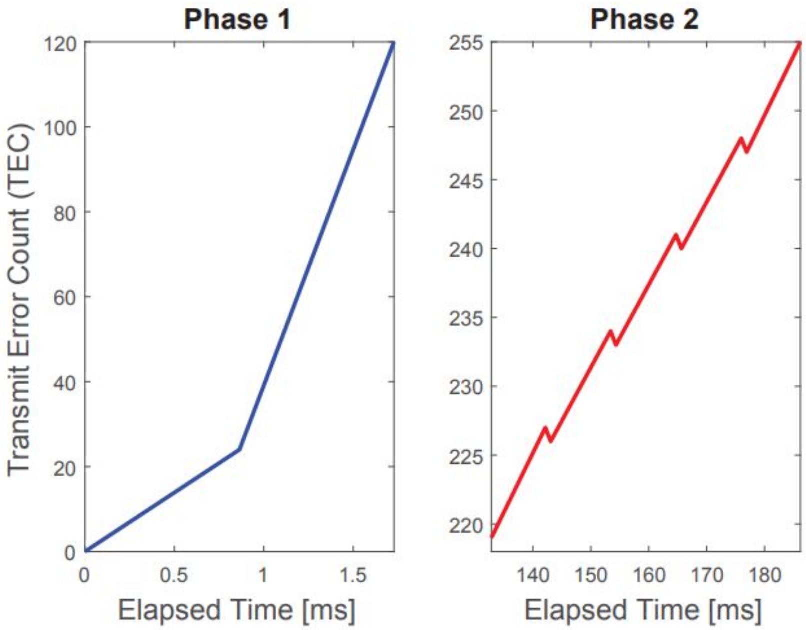


(c) Phase 2 – Victim in error-passive mode.

- TEC of victim and attacker differ

TECs Behavior





- TEC of victim and attacker
- TEC of victim

Important Note

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- Remember that CAN ID do not contain actual information on the transmitter
- The ID contains information that define the time interval and priority (i.e., safety-critical) of messages
- The attacker should hence target low-value IDs (dominant) to disconnect possibly safety-critical ECUs
- Examples: acceleration, braking

Difficulties in Setting the ID

- Each ECU cannot acquire the IDs of all received messages
- Only of those that pass through its message filter
- We can distinguish hence between *received* message and *accepted* message
- The attacker can only read the ID from accepted messages, thus meeting the requirements on the same ID depends on the filter of the victim ECU
- If there is no filter then done, but filters can be remotely manipulated

Preceded ID

- The constraint on synchronization must be precis at a bit level
- Although CAN messages are periodic, jitter may impair the effectiveness of the attack
- However, all messages and priorities are periodic and we can exploit this to infer when a certain message is going to be transmitted
- The exact time is hence 3 bit-time after the completion of the preceded ID