Outline

↓ LEC. 8

- Bus systems: basics
- Protocols
  - K-Line
  - CAN: Controller Area Network
  - LIN
  - FlexRay
  - MOST
  - In-car Ethernet
- ECUs
- Safety

The CAN Bus

- "Controller Area Network"
- 1986 → BOSCH
- Network topology: Bus
- Many (many) physical layers

- Common:
  - Up to 110 nodes
  - At 125 kBit/s: max. 500m

- Always: Two signal levels
  - low (dominant)
  - high (recessive)

→ Wired AND

The CAN Bus

- In the following: ISO 11898
  - Low Speed CAN (up to 125 kBit/s)
  - High Speed CAN (up to 1 MBit/s)
- Specifies OSI layers 1 and 2
  - Higher layers not standardized by CAN, covered by additional standards and conventions
  - e.g., CANopen

Collision Resolution

- Random access, collision free
  - CSMA/CR with Bus arbitration
  - (sometimes called CSMA/BA – bitwise arbitration)
- Message oriented
- Does not use destination addresses
  - Implicit Broadcast/Multicast

Physical layer (typical)

125 kbs → $l \le 400\,m$

$2\,l \cdot 10^{-8} \le 2 \cdot 10^{-6}$

$l \le 100\,m$

In generale

$\frac{1}{R} \ge 2\,l \cdot 10^{-8}$

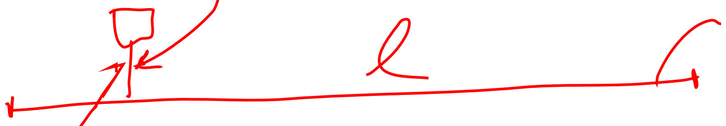
$l \le \frac{1}{2R \cdot 10^{-8}}$

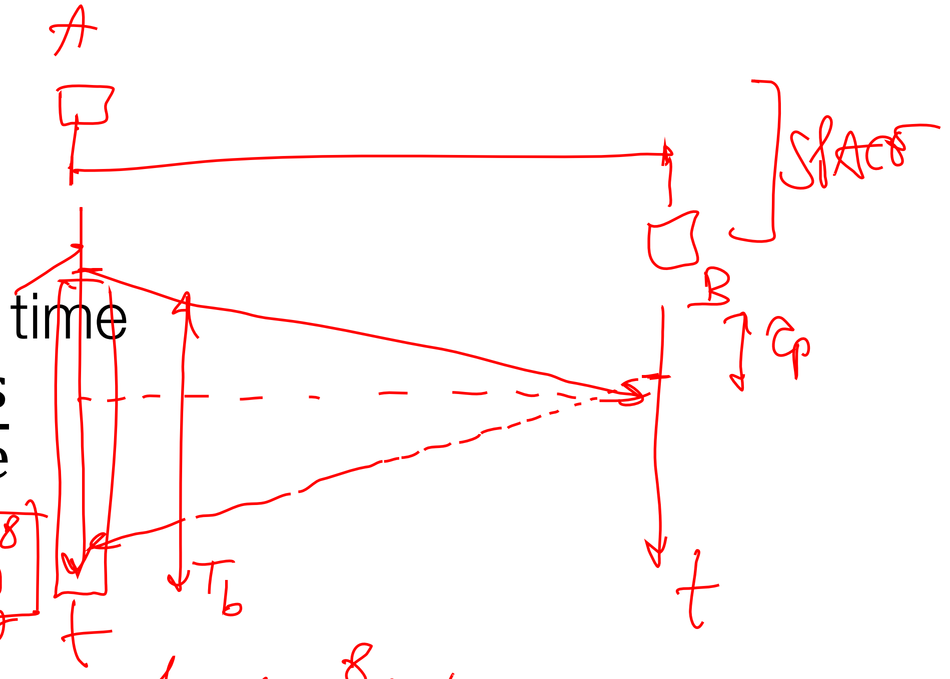
- High Speed CAN
  - $R =$ 500 kBit/s
  - Twisted pair wiring
  - Branch lines max. 30 cm
  - Terminating resistor mandated (120 Ω)
  - Signal swing 2 V
  - Error detection must happen within one Bit's time

⇨ bus length is limited to $l \le 50\text{m} \times \frac{1\ \text{MBit/s}}{\text{data rate}}$

Branch line ≤ 30 cm

Bus length ≤ 500 m





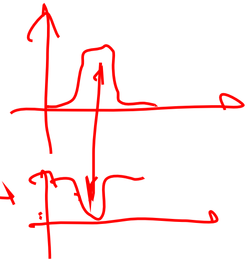
$2 \cdot 10^{-6}\ s$

$= \frac{2}{10^6\ b/s} = \frac{1}{5 \cdot 10^5\ b/s} = \frac{1}{R} = T_b \ge 2\,\tau_p = 2\,l/c = \boxed{2 \cdot l\, 10^{-8}}$

$c \simeq \frac{1}{3} c_0 = \frac{1}{3}\, 3 \cdot 10^8\ m/s$

Physical layer (typical)

High Speed
Low Speed



- Low Speed CAN
  - Up to 125 kBit/s
  - Standard two wire line suffices
  - No restriction on branch lines
  - Terminating resistors optional
  - Signal swing 5 V

uses Twisted

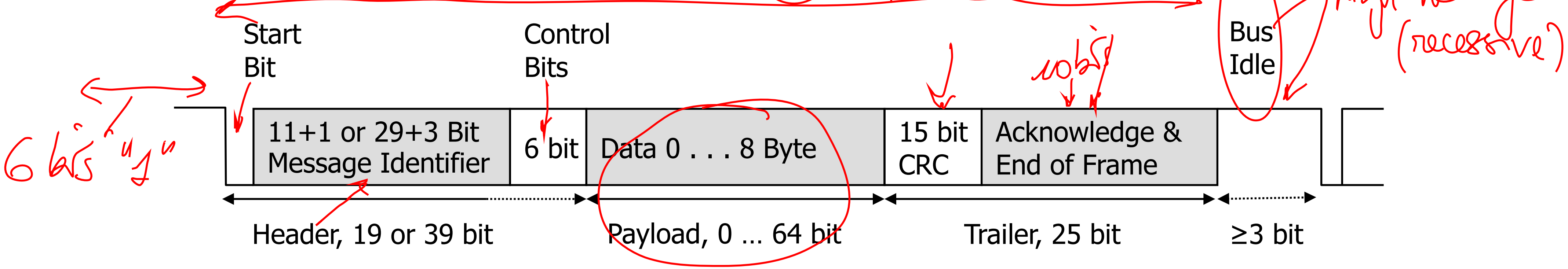
Swing up in terms of data-rate allows to support more easily large voltage swings.

- Single Wire CAN
  - 83 kBit/s
  - One line vs. ground
  - Signal swing 5 V

CAN in Vehicular Networks

- Address-less communication
  - Messages carry 11 Bit (CAN 2.0A) or 29 Bit (CAN 2.0B) message identifier
  - Stations do not have an address, frames do not contain one
  - Stations use message identifier to decide whether a message is meant for them
  - Medium access using CSMA/CR with bitwise arbitration
  - Link layer uses 4 frame formats
    Data, Remote (request), Error, Overload (flow control)
  - Data frame format:

FRAMES



CAN in Vehicular Networks

- CSMA/CR with bitwise arbitration
  - Avoids collisions by priority-controlled bus access
  - Each message contains identifier corresponding to its priority
  - Identifier encodes “0” dominant and “1” recessive: concurrent transmission of “0” and “1” results in a “0”
  - Bit stuffing: after 5 identical Bits one inverted Stuff-Bit is inserted (ignored by receiver)
  - When no station is sending the bus reads “1” (recessive state)
  - Synchronization happens on bit level, by detecting start bit of sending station

CAN in Vehicular Networks

Priority embedded inside

- CSMA/C~~A~~ R with bitwise arbitration
  - Wait for end of current transmission
    - wait for 6 consecutive recessive Bits "1"
  - Send identifier (while listening to bus)
  - Watch for mismatch between transmitted/detected signal level
    - Means that a collision with a higher priority message has occurred
    - Back off from bus access, retry later

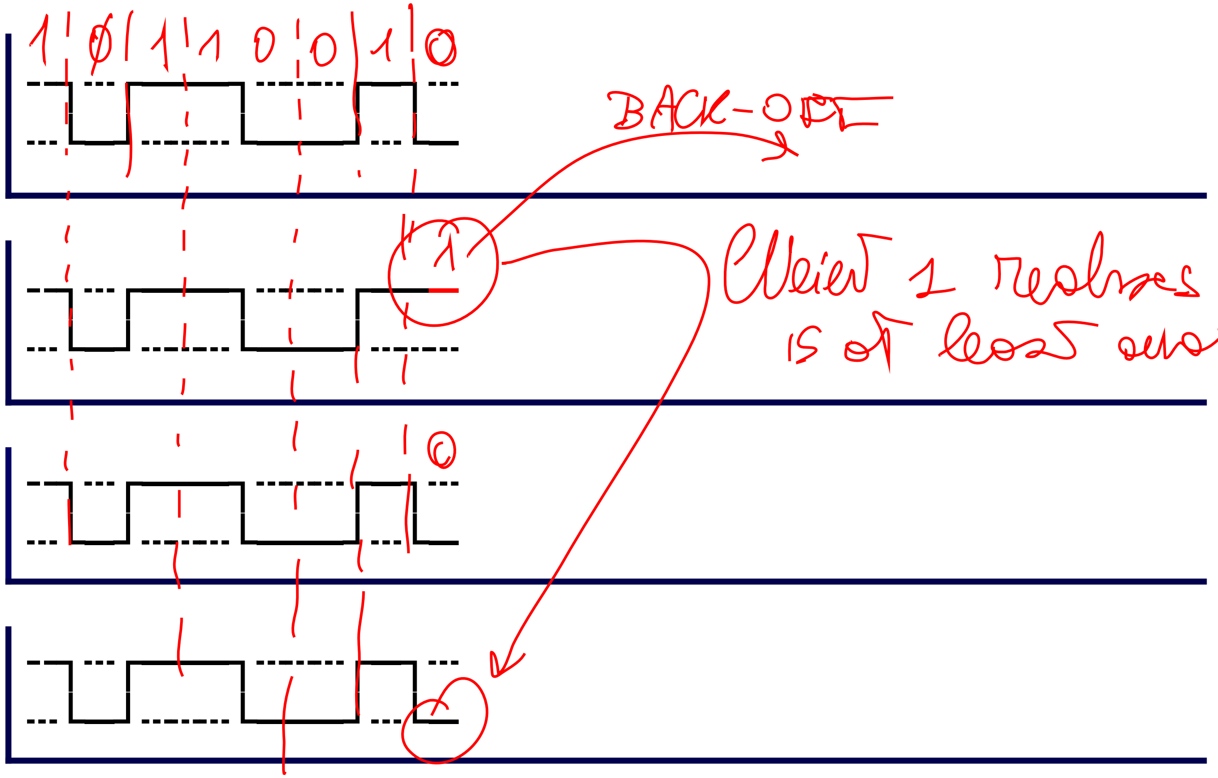
If the message that a node wants to transmit has highest one

- Realization of non-preemptive priority scheme
- Real time guarantees for message with highest priority
  - i.e., message with longest "0"-prefix

Priority is associated to # of "0"s in the prefix

The CAN Bus

- CSMA/CA with bitwise arbitration (CSMA/CR)
  - Client 2 recognizes bus level mismatch, backs off from access



BACK-OFF

Client 1 realises that there is at least another node transmitting ∅
⇓
At least another node has higher prio

6Tb

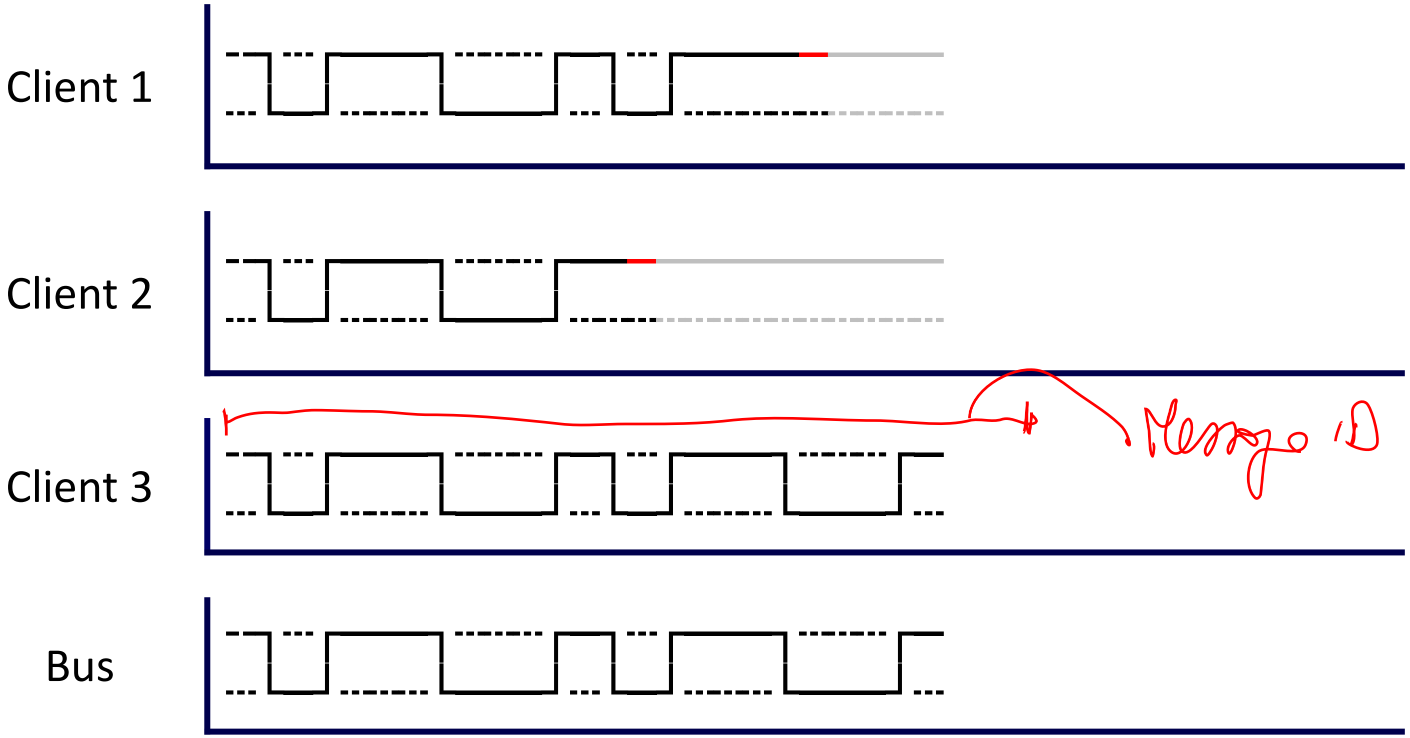
The CAN Bus

- CSMA/CA with bitwise arbitration (CSMA/CR)
  - Client 1 recognizes bus level mismatch, backs off from access

Client 1, i.e. corresponding to this bit, realizes that at least one node has higher priority.

The CAN Bus

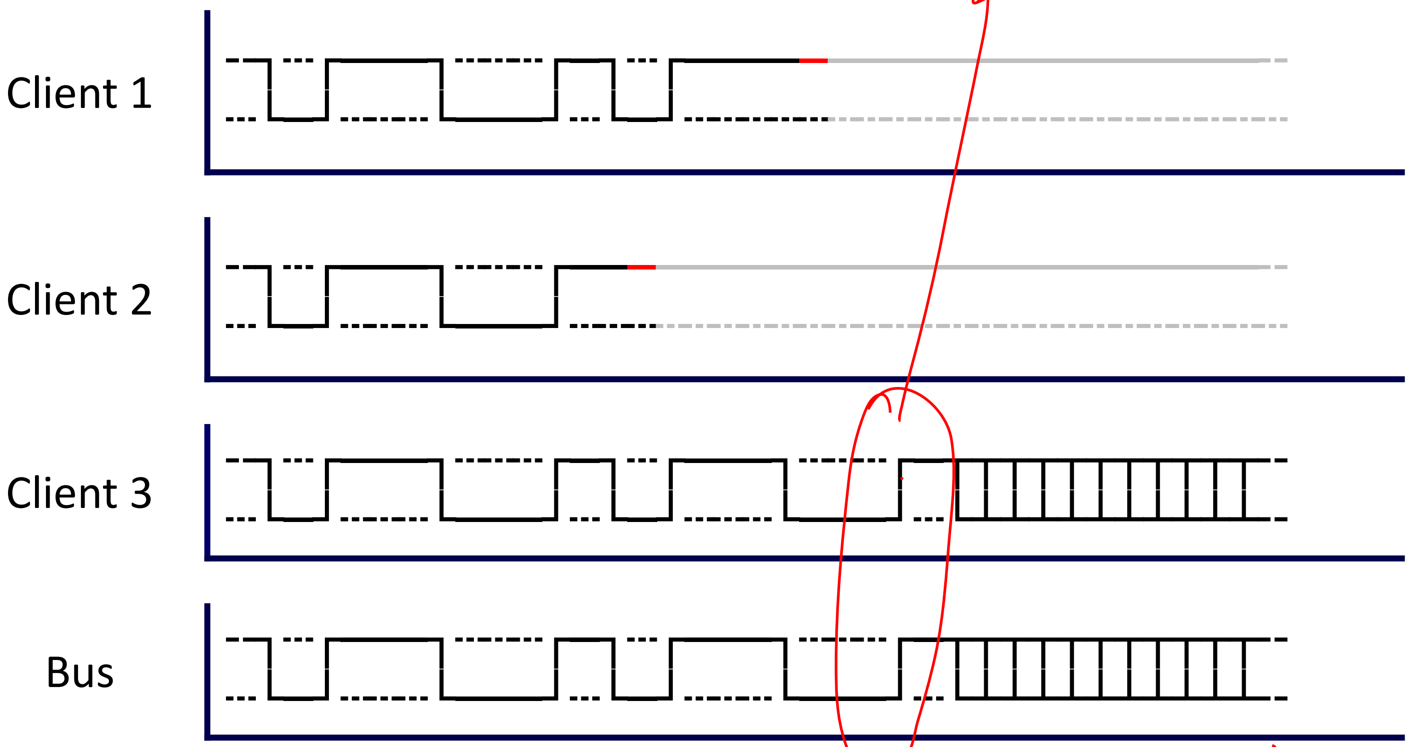
- CSMA/CA with bitwise arbitration (CSMA/CR)
  - Client 3 wins arbitration



The CAN Bus

- CSMA/CA with bitwise arbitration (CSMA/CR)
  - Client 3 starts transmitting data

6 control bits



PAYLOAD

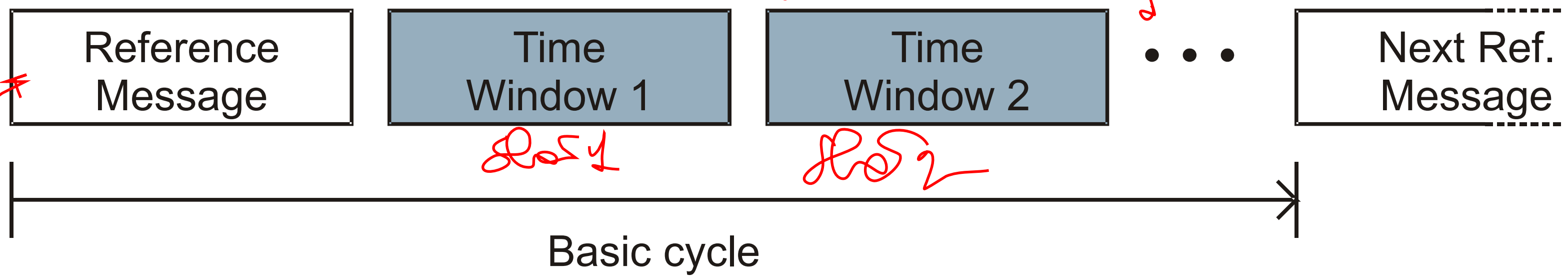
random access

The CAN Bus: Time-Triggered CAN (TTCAN)

- ISO 11898-4 extends CAN by TDMA functionality
- Solves non-determinism of regular CAN
  - Improves on mere "smart" way of choosing message priorities
- One node is dedicated "time master" node
- Periodically sends reference messages starting "basic cycles"
- Even if time master fails, TTCAN keeps working
  - Up to 7 fallback nodes
  - Nodes compete for transmission of reference messages
  - Chosen by arbitration

Arbitration Time slot

select next time slots depending on priority



The CAN Bus: TTCAN Basic Cycle

- Basic cycle consists of time slots
  - Exclusive time slot
    - Reserved for dedicated client
  - Arbitration time slot
    - Regular CAN CSMA/CR with bus arbitration

(Very similar to the idea used in IEEE 802.15.4 with beacons)

- Structure of a basic cycle arbitrary, but static
- CAN protocol used unmodified

  ➔ Throughput unchanged

Impossible for TTCAN because of reference message.

- TTCAN cannot be seen replacing CAN for real time applications
  - Instead, new protocols are being used altogether (e.g., FlexRay)

The CAN Bus: Message Filtering

- Message filtering
  - Acceptance of messages determined by message identifier
  - Uses two registers
    - Acceptance Code (bit pattern to filter on)
    - Acceptance Mask ("1" marks relevant bits in acceptance code)

Regardless of the values of the last 4 bits, I am interested

11 bits (CAN 2.0A)

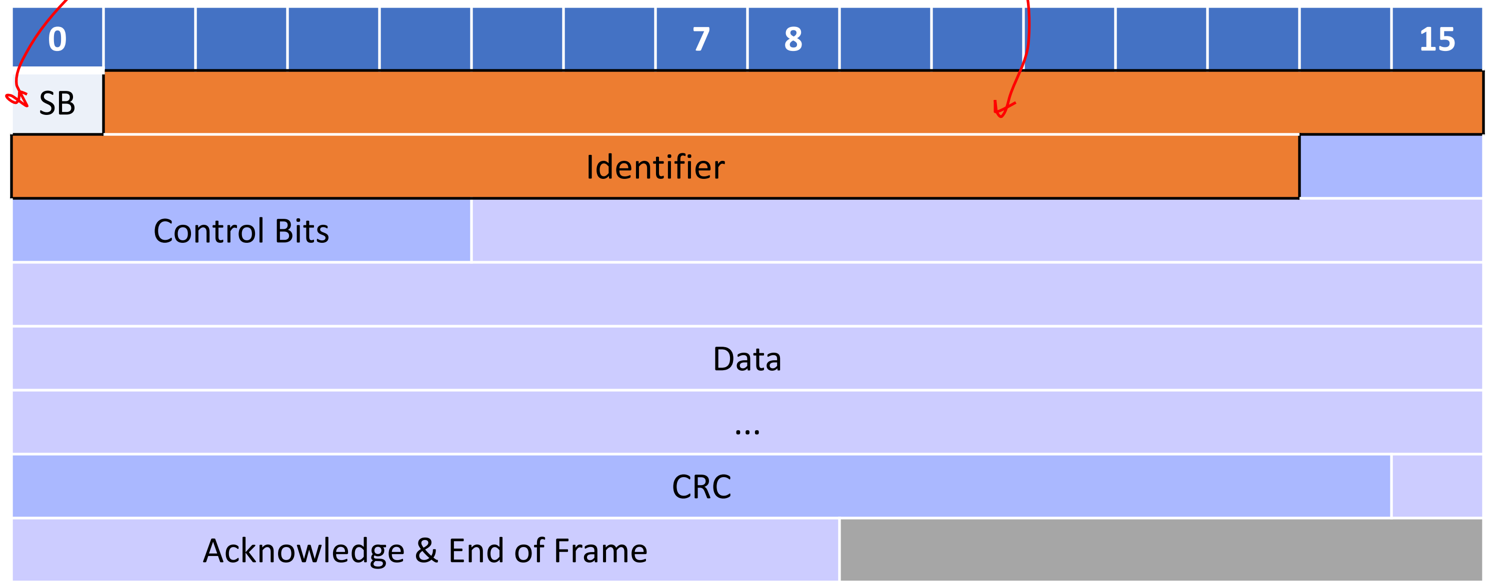
M.ID (filtering)

Bit	10	9	8	7	6	5	4	3	2	1	0
Acceptance Code Reg.	0	1	1	0	1	1	1	0	0	0	0
Acceptance Mask Reg.	1	1	1	1	1	1	1	0	0	0	0
Resulting Filter Pattern	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**X**	**X**	**X**	**X**

If coherent with the code, it's accepted

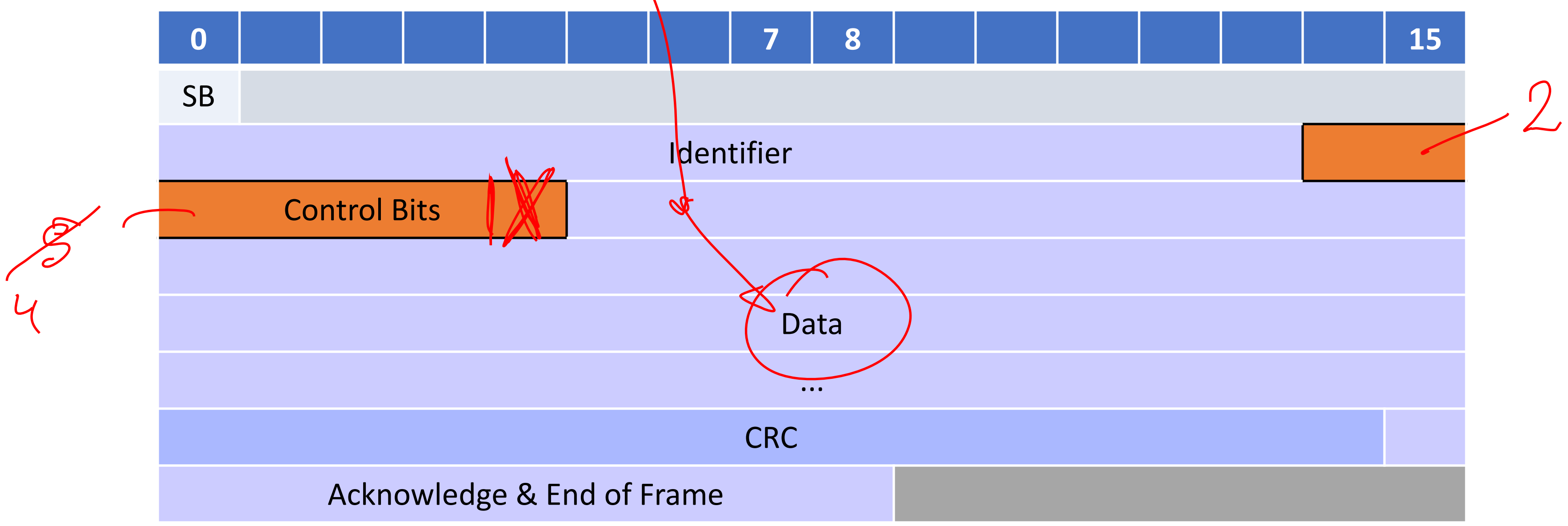
The CAN Bus: Data Format

- NRZ
- Time synchronization using start bit and stuff bits (stuff width 5)
- Frame begins with start bit
- Message identifier 11 Bit (CAN 2.0A), now 29 Bit (CAN 2.0B)



The CAN Bus: Data Format

- Control Bits
  - Message type (Request, Data, Error, Overload)
  - Message length
  - ...



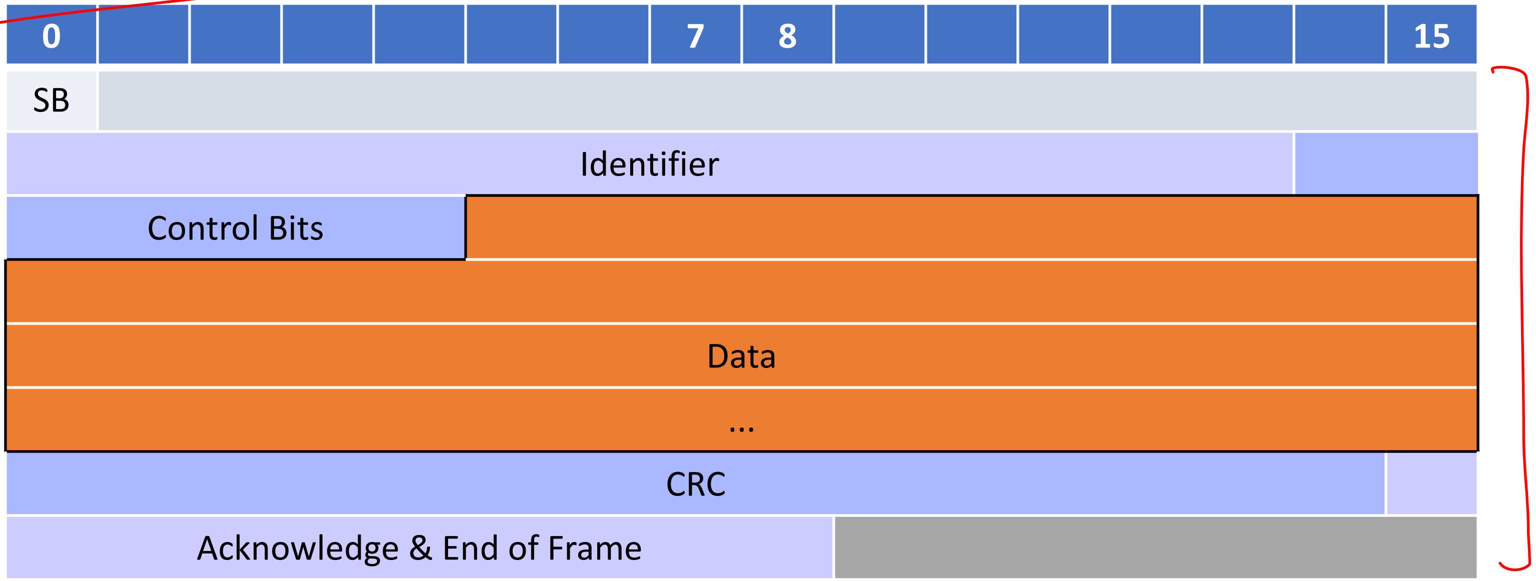
The CAN Bus: Data Format

- Payload
  - Restriction to max. 8 Byte per message (64 bits)
  - Transmission time at 500 kBit/s: 260 μs (using 29 Bit ID)
  - i.e., usable data rate 30 kBit/s

$$\frac{16 \cdot 8 \text{ bits}}{5 \cdot 10^5 \text{ b/s}} =$$

$T_{frame} = 260 \mu s$

At home!
frame rate of payload
↳ very inefficient if payload is 1 B.



After a successful Transmission, every node sends 6 recessive bits "1"

The CAN Bus

- Error detection (low level)
  - Sender checks for unexpected signal levels on bus
  - All nodes monitor messages on the bus
    - All nodes check protocol conformance of messages
    - All nodes check bit stuffing → Synchronization
  - Receiver checks CRC

→ Check if I am the intended destination

- If any(!) node detects error it transmits error signal → special frame
  - 6 dominant Bits with no stuffing "0"
- All nodes detect error signal, discard message

The CAN Bus

- Error detection (high level)
  - Sender checks for acknowledgement
    - Receiver transmits dominant “0” during ACK field of received message
  - Automatic repeat of failed transmissions
  - If controller finds itself causing too many errors
    - Temporarily stop any bus access

    skip
  - Remaining failure probability ca. 10^{-11}

layer 1 & layer 2

The CAN Bus: Transport Layer

- Not covered by ISO 11898 (CAN) standards
  - Fragmentation
  - Flow control
  - Routing to other networks

There may be one node acting as a router

- Add transport layer protocol
  - (1) ISO-TP
    - ISO 15765-2
  - (2) TP 2.0
    - Industry standard
  - …

(1)

The CAN Bus: ISO-TP

- ISO-TP: Header
  - Optional: 1 additional address Byte
    - Regular addressing → additional byte
      - Transport protocol address completely in CAN message ID
    - Extended addressing
      - Uniqueness of addresses despite non-unique CAN message ID
      - Part of transport protocol address in CAN message ID, additional address information in first Byte of TP-Header

4 bits

  - 1 to 3 PCI Bytes (Protocol Control Information)
    - First high nibble identifies one of 4 types of message of CAN
    - First low nibble and addl. Bytes are message-specific

0	1		2	3	4	5	6	7
(opt) Addl. Address	PCI high	PCI low	(opt) Addl. PCI Bytes		Payload			

→ 8 bytes

with respect to CAN protocol without ISO-TP

The CAN Bus: ISO-TP

- ISO-TP: Message type "Single Frame"
  - 1 Byte PCI, high nibble is 0
  - low nibble gives number of Bytes in payload
  - PCI reduces frame size from 8 Bytes to 7 (or 6) Bytes, throughput falls to 87.5% (or 75%, respectively)
  - No flow control

~~(Addr)~~

0		1	2	3	4	5	6	7
0	Len	Payload						

0	1		2	3	4	5	6	7
(Address)	0	Len	Payload					

The CAN Bus: ISO-TP

↓ Fragmentation of all frames!

- ISO-TP: Message type "First Frame"
  - 2 Bytes PCI, high nibble is 1
  - low nibble + 1 Byte give number of Bytes in payload
  - After First Frame, sender waits for Flow Control Frame

To be sent from destination to transmitter

0	1		2	3	4	5	6	7
(Address)	1	Len		Payload				

- ISO-TP: Message type "Consecutive Frame"
  - 1 Byte PCI, high nibble is 2
  - low nibble is sequence number SN (counts upwards from 1)
    - Application layer can detect packet loss
  - No additional error detection at transport layer

You can start from 1 as the CAN system is simple!

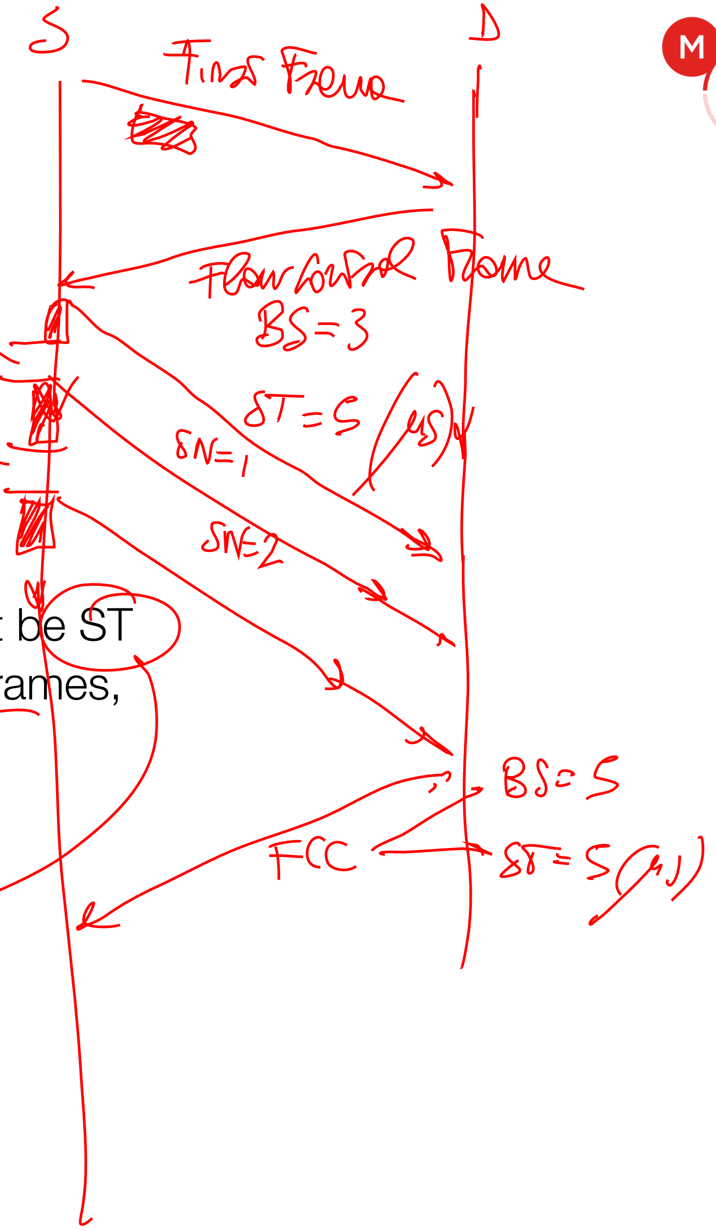
0	1		2	3	4	5	6	7
(Address)	2	SN	Payload					

The CAN Bus: ISO-TP

- ISO-TP: Message type “Flow Control Frame”
  - 3 Bytes PCI, high nibble is 3
  - low nibble specifies Flow State FS
  - FS=1: Clear to Send
    - Minimum time between two Consecutive Frames must be ST
    - Sender may continue sending up to BS Consecutive Frames, then wait for new Flow Control Frame
  - FS=2: Wait
    - Overload
    - Sender must wait for next Flow Control Frame
  - Byte 2 specifies Block Size BS
  - Byte 3 specifies Separation Time ST

≤ 4B

≤ 2B



0	1		2	3
(Address)	3	FS	BS	ST

2

The CAN Bus: TP 2.0

- Connection-oriented
- Communication based on channels
- Specifies Setup, Configuration, Transmission, Teardown

- Addressing
  - Every ECU has unique logical address; additional logical addresses specify groups of ECUs → Multicast
  - for broadcast und channel setup: logical address + offset = CAN message identifier
  - Channels use dynamic CAN message identifier

The CAN Bus: TP 2.0 Broadcast

- Repeated 5 times (motivated by potential packet loss)
- Fixed length: 7 Byte
- Byte 0: 23 = 0010|0011
  - logical address of destination ECU → Broadcast address
- Byte 1: Opcode 24 = 0010|0100
  - 0x23: Broadcast Request
  - 0x24: Broadcast Response
- Byte 2, 3, 4:
  - Service ID (SID) and parameters
- Byte 5, 6:
  - Response: 0x0000
  - No response expected: alternates between 0x5555 / 0xAAAA

"Special" address →

0	1	2	3	4	5	6
Dest	Opcode	SID, Parameter			0x55	0x55
					0xAA	0xAA

} 1st TX
} 2nd TX

The CAN Bus: TP 2.0 Channel Setup

- Byte 0:
  - logical address destination ECU — Specific one!
- Byte 1: Opcode
  - 0xC0: Channel Request
  - 0xD0: Positive Response
  - 0xD6 .. 0xD8: Negative Response
- Byte 2, 3: RX ID
  - Validity nibble of Byte 3 is 0 (1 if RX ID not set)
- Byte 4, 5: TX ID
  - Validity nibble of Byte 5 is 0 (1 if TX ID not set)
- Byte 6: Application Type
  - cf. TCP-Ports

towards higher layer

0	1	2	3	4	5	6
Dest	Opcode	RX ID	V	TX ID	V	App

The CAN Bus: TP 2.0 Channel Setup

- Opcode 0xC0: Channel Request
  - TX ID: CAN msg ID requested by self
  - RX ID: marked invalid
- Opcode 0xD0: Positive Response
  - TX ID: CAN msg ID requested by self
  - RX ID: CAN msg ID of original sender
- Opcode 0xD6 .. 0xD8: Negative Response
  - Reports errors assigning channel (temporary or permanent)
  - Sender may repeat Channel Request

At home

- After successful (positive response) exchange of Channel Request/Response: dynamic CAN msg IDs now assigned to sender and receiver next message sets channel parameters

0	1	2	3	4	5	6
Dest	0xC0		1	TX ID	0	App

channel request

The CAN Bus: TP 2.0

- TP 2.0: set channel parameters
  - Byte 0: Opcode
    - 0xA0: Channel Setup Request (Parameters for channel to initiator)
    - 0xA1: Channel Setup Response (Parameter for reverse channel)
  - Byte 1: Block size
    - Number of CAN messages until sender has to wait for ACK
  - Byte 2, 3, 4, 5: Timing parameters
    - E.g., minimal time between two CAN messages

At home!

Setup

- TP 2.0: misc. channel management and teardown
  - Byte 0: Opcode
    - 0xA3: Test – will be answered by Connection Setup Response
    - 0xA4: Break – Receiver discards data since last ACK
    - 0xA5: Disconnect – Receiver responds with disconnect, too

Control & Teardown

0	1	2	3	4	5
0xA0	BS	Timing			

Channel setup Request

The CAN Bus: TP 2.0

- TP 2.0: Data transmission via channels
  - Byte 0, high nibble: Opcode
    - MSB=0 – Payload
      - /AR=0 – Sender now waiting for ACK
      - EOM=1 – Last message of a block
    - MSB=1 – ACK message only (no payload)
      - RS=1 – ready for next message (➔ flow control)
  - Byte 0, low nibble
    - Sequence number
  - Bytes 1 .. 7: Payload

Opcode Nibble			
0	0	/AR	EOM

Opcode Nibble			
1	0	RS	1

0		1	2	3	4	5	6	7
Op	SN	Payload						

Main Takeaways

- CAN
  - Still standard bus in vehicles
  - Message oriented
  - CSMA with bitwise arbitration (CSMA/CR with BA or CSMA/BA)
    - Impact on determinism
    - TTCAN (TDMA)
  - Error detection
  - Transport layer: ISO-TP vs. TP 2.0
    - Flow control, channel concept

address

lecture 8