Fieldbuses

Jean-Luc Scharbarg - ENSEEIHT - Dpt. SN

October 2020

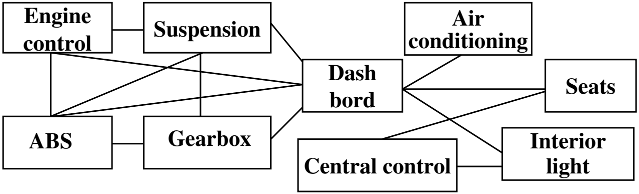
Overview of the module

- Fieldbuses for different applicative contexts
  - Automotive context
    - CAN, TTCAN, FTTCAN, LIN, Flexray
  - Avionics context
    - ARINC429, ARINC629
    - Integrated Modular Avionics paradigm
  - Satellite context
    - MilStd1553, Spacewire
  - Factory automation context
    - Profibus, FIP, PNet, Foundation Fieldbus, Interbus

The automotive context

From point-to-point to multiplexed communications

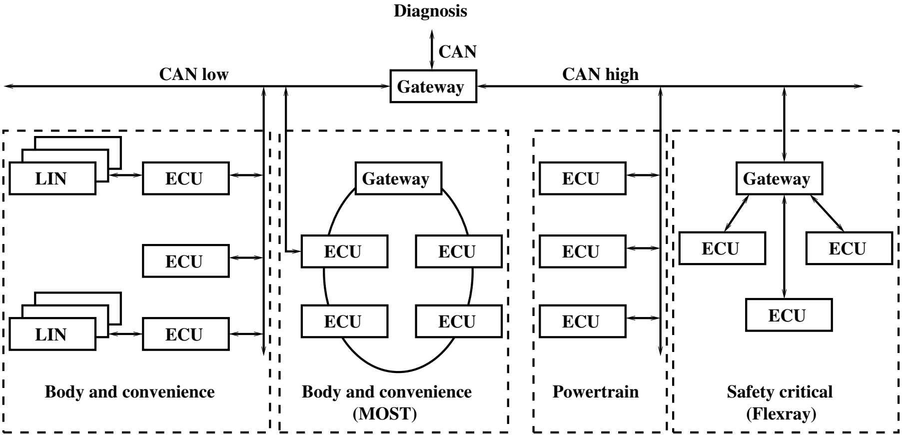
- Before 1970, mechanical and hydraulic systems for engine control, brake system, gear system, ...
- After 1970, huge increase in the number of electronic systems
- Performance and reliability of hardware components + software technologies $\Rightarrow$ implementation of complex functions that improve the comfort and safety (Antilock Braking System, Electronic Stability Program, active suspension, GPS, doors, entertainment, ...)
- Early days of automotive electronics: 1 function = 1 Electronic Control Unit (ECU: a microcontroller and sensors and actuators)
- Exchanges of data between functions via dedicated links (e.g. speed estimated by the engine control and used by the suspension control)



From point-to-point to multiplexed communications

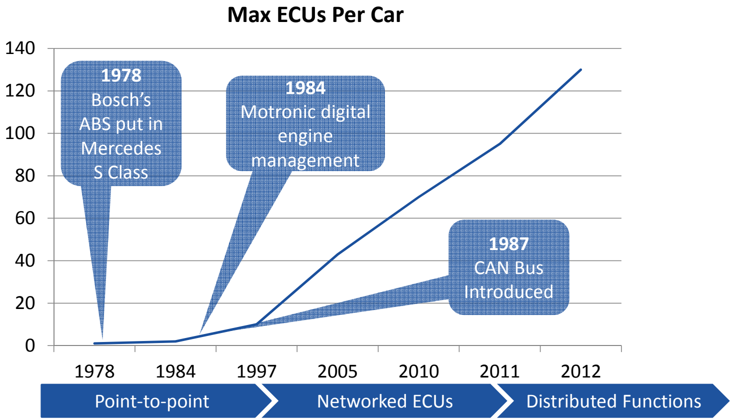
- A function is either too complex or too small for a single ECU $\Rightarrow$ functions distributed over several ECUs or several functions on the same ECU
- Too many information exchanges $\Rightarrow$ need for a shared communication medium.
  - Reduces the weight of the electronic systems: reduction of 15 Kg with the replacement of the dedicated links by a shared bus for the control of the doors of a BMW
  - Necessary to guarantee that the communication delays can be bounded
- First automotive bus: Controller Area Network (Bosch, 1980s)
- First embedded CAN bus: Mercedes class S (1991)
- Today: a complex architecture with several communication technologies

Automotive electronic architecture - 2006



The Modern Car
How much electronics?

ETAS



DRIVING | EMBEDDED EXCELLENCE

Vehicle Domains: Powertrain
(Or what does all that stuff do?)

- Engine Management
  - Injection/Spark timing
  - Emissions control
- Transmission Control
  - Gear selection
  - Terrain Adjustment
- Real-time issues
  - Pressure wave control on diesel engines
  - Deadlines a function of angular rotation
  - Lots of data communication

DRIVING | EMBEDDED EXCELLENCE

Vehicle Domains: Chassis
(Or what does all that stuff do?)

ETAS

- Braking
  - Anti-Lock Braking (ABS) since 1978
- Traction Control
  - Electronic Stability (ESP) since 1995
- Steering assist
- Adaptive cruise control

- Real-time issues
  - Wheel rotation driving
  - Slip detect
  - Brake force distribution

DRIVING | EMBEDDED EXCELLENCE

Vehicle Domains: Body
(Or what does all that stuff do?)

- Wiper control / rain sensing
- Wing mirrors
- Vehicle access
- Window lift/anti-trap/pinch
- Electronic seats
- Heating/ventilation
- Park pilot
- Lane departure warning
- Airbags
- Blind spot warning

- Real-time issues
  - End-to-end latency guarantees
    - e.g for brake lights
  - Distributed functionality

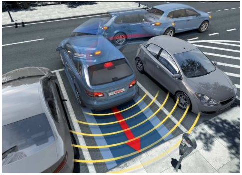


Image: Robert Bosch GmbH

Vehicle Domains: In-Vehicle Infotainment (IVI)
(Or what does all that stuff do?)

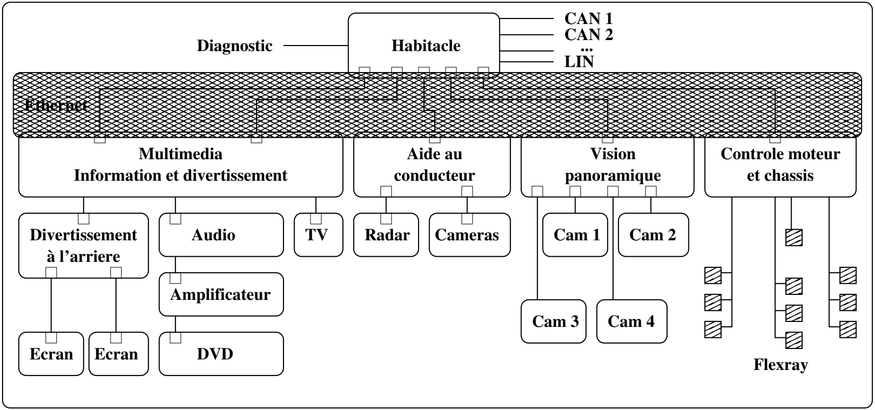
- Radio/CD/MP3 integration
- Navigation/Mapping
- TV
- Internet
- Telephony

- This area accounts for an increasing part of the "user experience"

- Real-time issues
  - Quality of service similar to those for similar "PC" applications

DRIVING | EMBEDDED EXCELLENCE

A more up-to-date architecture



Different networks for different requirements

- Classification for automotive communication protocols defined by the Society for Automotive Engineers (SAE) in 1994
  - Class A networks
    - Low-cost, low rate (< 10 Kb/s) technology
    - Transmission of simple control data
    - Mainly integrated in the body domain (doors, ...)
    - LIN (Local Interconnect Network) and TTP/A
  - Class B networks
    - From 10 to 125 Kb/s
    - Data exchanges between ECU (reduce the number of sensors by sharing information)
    - J1850 and low-speed CAN
  - Class C networks
    - From 125 Kb/s to 1 Mb/s
    - Powertrain and chassis domain
    - High-speed CAN
  - Class D networks
    - Over 1 Mb/s
    - Multimedia data and x-by-wire applications
    - MOST (Media-Oriented System Transport), TTP/C, Flexray

Event Triggered versus Time Triggered

- The Event Triggered paradigm
  - The system takes into account, as quickly as possible, any asynchronous event (e.g. an alarm)
  - Necessity of a mechanism in order to avoid collisions
  - Efficient bandwidth usage (no unnecessary transmissions)
  - Evolution of the system without redesigning existing nodes
  - Tricky to verify that timing constraints are met
  - Problematic detection of node failure
- The Time Triggered paradigm
  - Time Division Multiple Access (TDMA) $\Rightarrow$ static frame scheduling (well suited for periodic messages)
  - Timing behavior fully predictable
  - Immediate identification of missing messages $\Rightarrow$ missing nodes
  - Inefficient in terms of bandwidth usage
  - Most of the time very bad for aperiodic messages
  - Problematic evolution of the system
  - Need for a synchronization of the whole system
- Several automotive networks with both event-triggered and time-triggered capabilities

Communication technologies in a car

- CAN: event-triggered with priorities
- TTP/C: time-triggered (one regular slot per station)
- Flexray: a time-triggered part (regular slots per station) and an event-triggered part with priorities
- TTCAN: time-triggered parts (slots for messages) and event triggered parts with priorities
- LIN: a low-cost master-slave (time-triggered)
- TTP/A: a low-cost master-slave (time-triggered)
- MOST: a multimedia network

The CAN bus: history

1983: Start of the development of CAN (Controller Area Network) at Robert Bosch Gmbh

1985: first specification (**V1.0**) of the CAN bus

1986: Start of the standardization activity of CAN at ISO

1989: Start of the first industrial applications of CAN

1991: Specification of the extended version of the protocol (**CAN 2.0**): part **2.0A** (identifiers on 11 bits), part **2.0B** (identifiers on 29 bits)

1991: first car (Mercedes class S) integrating 5 ECU interconnected by a CAN bus at 500 Kb/s

2001: Specification of Time-Triggered CAN

2012: Specification of CAN-FD

Main technical characteristics

- CAN specification concerns level 1 and 2 of the OSI model
- Several definitions of the application level (CANopen, OSEK, ...)
- broadcast bus with CSMA technique
- MAC with priorities and a non-destructive arbitration mechanism
- A unique identifier for each message:
  - defines the priority for the bus access
  - allows the filtering of messages in reception
- A powerful set of mechanisms for error management:
  - automatic retransmission of corrupted frames
  - error counters for each CAN controller
  - ...
- At most eight bytes of data per frame

A brief overview of the physical layer

- Bit coding is Non-Return-to-Zero (NRZ) $\Rightarrow$ a constant level for the whole duration of a bit
- Logical level 0 = dominant bit
- Logical level 1 = recessive bit
- In case of simultaneous transmissions:
  one dominant bit + one recessive bit = one dominant bit on the bus
- Minimum bit duration = two times the propagation delay
- Limitation of the bandwidth as a function of the length of the bus

maximal bandwidth	**Length of the bus**	
1 Mbit/s	40 meters	CAN High Speed ISO 11898
500 Kbit/s	100 meters	
250 Kbit/s	250 meters	
125 Kbit/s	500 meters	CAN Low Speed ISO 11519-2
10 Kbit/s	5 Kilometers	

Bit-stuffing

- Goal: create edges on the signal in order to limit the synchronization problems
- Solution: after 5 identical bits, insertion of a stuff bit of opposite value

To be transmitted	1 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 1
On the medium	1 0 1 1 1 1 1 S 1 0 0 1 1 0 0 0 0 0 S 1 1 1 1 S 0 1 1
Received	1 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 1

- Worst-case: insertion of $\left\lfloor \frac{(n-1)}{4} \right\rfloor$ bits

To be transmitted	1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
On the medium	1 0 0 0 0 0 S 1 1 1 1 S 0 0 0 0 S 1 1 1 1 S 0 0 0 0 S

Frame types

- Data frame: broadcasts the data associated to an identifier on the bus
  - Standard CAN (2.0A): identifier on 11 bits
  - Extended CAN (2.0B): identifier on 29 bits
- Remote Transmission Request frame (RTR): requests the broadcasting of the data associated to an identifier on the bus:
  - no data in the frame
  - no guarantee on the delay before the answer
- Error frame
  - Emitted by each station which detect a transmission error
  - Generates the transmission of error frames by all the other stations
- overload frame
  - Used to delay the transmission of a frame on the bus
  - Format quite similar to the format of an error frame
  - Quite rarely used

Format of the standard data frame

SoF	Arbitration	Control	Data	CRC	ACK	EoF	Inter
1	12	6	0 a 64	16	2	7	3

SoF: one dominant bit which indicates the start of a transmission

Arbitration: **ID** (identifier) associated with the data, on 11 bits, followed by one dominant bit **RTR** (Remote Transmission Request) which indicates that it is a data frame

Control: two dominant bits, followed by four bits **DLC** (Data Length Code) which indicate the number of bytes in the data field

Data: transmitted payload, from zero to eight bytes

CRC: error detection code on 15 bits, followed by one recessive bit which delimits the end of the CRC

ACK: one bit which is transmitted recessive by the emitter of the frame and dominant by every station which received correctly the frame, followed by one recessive bit

Eof: sequence of seven recessive bits

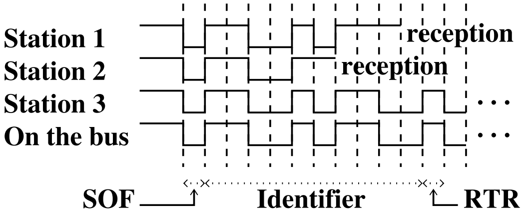
Format of the extended data frame

SoF	Arbitration	Control	Data	CRC	ACK	EoF	Inter
1	32	6	0 a 64	16	2	7	3

- The only difference with the standard data frame format: the arbitration field
- Format of the arbitration field:
  - The most significant part of the identifier (**Base ID**) on 11 bits,
  - One recessive bit **SRR** (Substitute Remote Request),
  - One recessive bit **IDE** (IDentifier Extension),
  - The least significant part of the identifier (**ID Extension**) on 18 bits,
  - The **RTR** bit (Remote Transmission Request)
- The same bus can support both formats, thanks to the **IDE** bit

Control of the bus access

- The start of frame transmissions are synchronized for all the stations
- Bit by bit resolution collision on the arbitration field $\Rightarrow$ the transmitted frame with the higher priority wins
- Principle: each station transmits one bit and then listen; received value $\neq$ emitted value $\Rightarrow$ the station has lost the arbitration
- Consequence: one round trip for the signal before the transmission of a new bit $\Rightarrow$ limit the maximum bandwidth



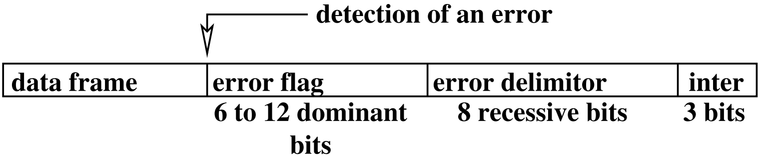
Exercise

- At a given time t we have the following pending frames on a CAN bus
  - standard frame f1 with identifier 10,
  - extended frame f2 with identifier 10 (basic part) + 15 (extension)
  - standard frame f3 with identifier 7,
  - standard frame f4 with identifier 14,
  - extended frame f5 with identifier 13 (basic part) + 1 (extension)

Question: Give the sequence of frame transmissions on the bus

Mechanisms for transmission errors

- No automatic correction of errors
- Principle: each station which detects an error immediatly sends an error frame (six dominant bits)



- Between 17 and 23 bits for an error frame
- The corrupted frame will participate to a future arbitration process, with the same priority

Different types of errors

- Bit-stuffing error: 6 consecutive bits with the same level during the SoF, arbitration, control, data and CRC fields $\Rightarrow$ the error frame propagates the error to all the stations
- error on a bit level: emission of a dominant bit and reception of a recessive one
- CRC error: the received CRC is different from the calculated one (on the SoF, arbitration, control and data fields)
- Acknowledgement error: the first bit of the **ACK** field is received recessive $\Rightarrow$ no station received correctly the frame
- Format error: wrong value for a fixed bit (e.g. last bit of the **CRC** field is received dominant)

Fault confinement

- The problem: a faulty station can disturb the whole system (e.g. permanent transmission of error frames)
- The solution: automatic disconnection of faulty stations or limitation of their error indication capabilities
- Implementation
  - An error counter on frames transmitted by each station (**TEC** : Transmit Error Counter) :
    - transmission of a corrupted frame: $+8$ (up to 256)
    - transmission of a correct frame: -1 (if $TEC > 0$)
  - An error counter on frames received by each station (**REC** : Recieve Error Counter) :
    - reception of a corrupted frame: $+1$ (up to 128)
    - reception of a correct frame: -1 (if $REC > 0$)
  - The state of a station depends on the value of its counters
    - **error-active**: normal behavior ($REC < 127$ and $TEC < 127$)
    - **error-passive**: no more error signalling ($REC \geq 127$ and $127 \leq TEC \leq 255$)
    - **bus-off**: disconnection from the bus (no more emission nor reception) ($TEC > 255$)

Example of an engine configuration

- CAN bus at 250 kb/s $\Rightarrow$ load $\simeq$ 20 %

Frame	Emitter	DLC	Period (ms)
1	Engine control	8	10
2	Steering angle sensor	3	14
3	Engine control	3	20
4	Automatic gearbox	2	15
5	ABS	5	20
6	ABS	5	40
7	ABS	4	15
8	Body control	5	50
9	Suspension	4	20
10	Engine control	7	100
11	Automatic gearbox	5	50
12	ABS	1	100

CAN schedulability analysis

- Mandatory to guarantee that deadlines are not missed
- First presented in 1994 by Tindell
- Considers sporadic flows with known frame lengths
- Implemented in a tool (Volcano) and used by all the car makers
- Flaw in the analysis, shown in 2006
- Corrected schedulability analysis (Davis et al, Real-Time System 2007)
- Illustration of the analysis on the following example

Message	Priority	Period	Deadline	Data bytes
A	1	2.5 ms	2.5 ms	7 bytes
B	2	3.5 ms	3.25 ms	7 bytes
C	3	3.5 ms	3.5 ms	7 bytes

Bandwidth of the CAN bus: 125 Kb/s

Identifiers on 11 bits

Summary of CAN schedulability analysis

- The considered model
  - A set of periodic CAN messages with, for each message m_i
    - a period T_i
    - a delay D_i
    - a number of data bytes S_i
  - Maximum transmission time for a frame of a message m_i

$$
\begin{array}{lll}
C_i & = & MaxL_i \times \tau_{bit} \\
 & \text{with} & \\
MaxL_i & = & 55 + 10 \times S_i \quad \text{for identifiers on 11 bits} \\
MaxL_i & = & 80 + 10 \times S_i \quad \text{for identifiers on 29 bits} \\
\tau_{bit} & = & \text{Transmission time of one bit}
\end{array}
$$

  - On the example, $C_i = (55 + 10 \times 7) \times 0.008 = 1\ ms$
- The expected result: the worst-case response time R_i for each message m_i
  - m_i is schedulable iff $R_i \leq D_i$
  - The considered set of CAN messages is schedulable iff all the messages in the set are schedulable

Summary of CAN schedulability analysis

- The worst-case response time R_i can be divided in three parts

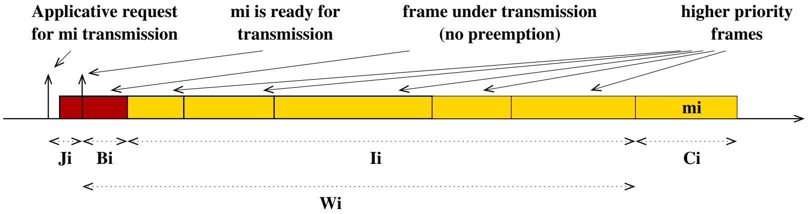
$$R_i = J_i + W_i + C_i$$

- J_i: maximum jitter of the CAN controller
- W_i: the maximum waiting time in controller queue, until the frame wins the arbitration
- C_i: the transmission time of the frame
- W_i can be divided in two parts

$$W_i = B_i + I_i$$

- B_i: the blocking time, due to the non-preemptive characteristics of CAN
- I_i: the interference time, due to the transmission of frames with higher priority

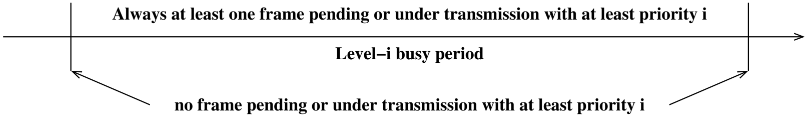
Summary of CAN schedulability analysis



- Worst-case analysis $\Rightarrow$ find for each message m_i the scenario of frame transmissions which leads to the longer W_i

Summary of CAN schedulability analysis

- Based on the busy period concept
- Priority level-i busy period
  - begins when a frame with priority i or higher becomes ready for transmission and there is no pending frame with priority i or higher
  - Ends as soon as there are no more pending frames with priority i or higher

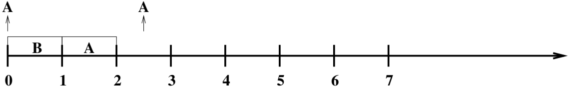


- The longer W_i occurs for a frame of message m_i transmitted during the longest priority level-i busy period
- Longest priority level-i busy period: begins with a critical instant for message m_i, immediately after the start of transmission of a frame with a lower priority and the maximum possible length among these frames

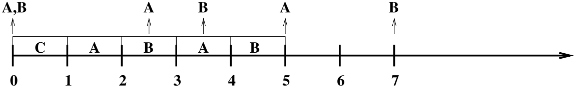
Illustration of CAN schedulability analysis

Message	Priority	Period	Deadline	C_i
A	1	2.5 ms	2.5 ms	1 ms
B	2	3.5 ms	3.25 ms	1 ms
C	3	3.5 ms	3.5 ms	1 ms

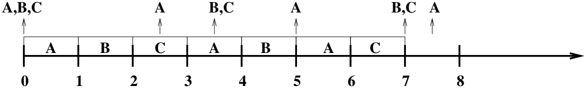
- Upper possible W_i for message A: 1 $ms \Rightarrow R_A = 2$ ms



- Upper possible W_i for message B: 2 $ms \Rightarrow R_B = 3$ ms



- Upper possible W_i for message C: 2.5 $ms \Rightarrow R_A = 3.5$ ms



Exercice on CAN schedulability analysis

- 4 stations, bandwidth = 500 Kb/s, dentifiers on 11 bits

Message	Station	Identifier	Period	Deadline	Data bytes
A	1	5	0.8 ms	0.5 ms	2 bytes
B	2	12	1.2 ms	0.7 ms	4 bytes
D	3	23	1.2 ms	0.9 ms	6 bytes
E	4	36	1.2 ms	0.9 ms	6 bytes

- 4 stations, bandwidth = 500 Kb/s, dentifiers on 11 bits

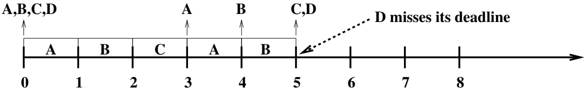
Message	Station	Identifier	Period	Deadline	Data bytes
A	1	5	0.8 ms	0.5 ms	2 bytes
B	2	12	1.2 ms	0.7 ms	4 bytes
C	2	14	1.2 ms	0.7 ms	4 bytes
D	3	23	1.2 ms	0.9 ms	6 bytes
E	4	36	1.2 ms	0.9 ms	6 bytes

- Definition of an offset between B and C

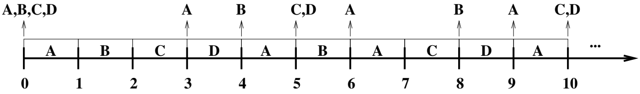
Limits of native CAN MAC : static identifiers

- Static identifier for each message (all the frames of a given message have the same priority) ⇒ fixed priority scheduling policy (e.g. Rate monotonic)

Message	Station	Identifier	Period	Deadline	Data bytes
A	1	5	3 ms	3 ms	7 bytes
B	2	12	4 ms	4 ms	7 bytes
C	3	23	5 ms	5 ms	7 bytes
D	4	36	5 ms	5 ms	7 bytes



- Dynamic priorities (Earliest Deadline First) can give better results



Limits of native CAN MAC : static identifiers

- One proposed solution allowing EDF scheduling

Identifier on n bits

deadline of the frame	identifier of the message
p bits	$n-p$ bits

- The priority of a frame depends, first on its deadline, second on the identifier of the associated message
- The previous example, with $p = 3$ and the deadline in *ms*

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$
A	$011\ldots$	-	-	$011\ldots$	$010\ldots$
B	$100\ldots$	$011\ldots$	-	-	$100\ldots$
C	$101\ldots$	$100\ldots$	$011\ldots$	-	-
D	$101\ldots$	$100\ldots$	$011\ldots$	$010\ldots$	-
Winner	A	B	C	D	A

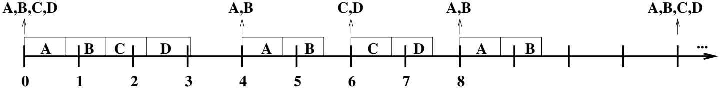
- Update the identifiers of pending frames
- Limit the number of available messages

Limits of native CAN MAC : purely event-triggered

- Event-Triggered paradigm $\Rightarrow$ the jitter can be high
- The jitter: the difference between the shortest response time and the longest one

Message	Identifier	Period	Deadline	Data bytes	Tx time
A	5	4 ms	4 ms	4 bytes	0.760 ms
B	12	4 ms	4 ms	4 bytes	0.760 ms
C	23	6 ms	6 ms	4 bytes	0.760 ms
D	36	6 ms	6 ms	4 bytes	0.760 ms

- A possible scheduling of frames



- Jitter for messages C and D: 1.520 ms

Limits of native CAN MAC : data field and bandwidth

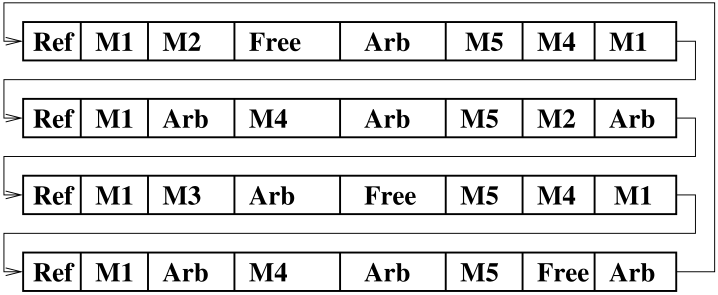
- At most 8 bytes in the data field
  - Well-suited for the transmission of control data
  - Large overhead
  - Slow software download
  - Splitting of long messages
- Bandwidth limited due to delay propagation (bit by by arbitration)
  - Very limited bandwidth in case of long bus lines (trucks, autobuses)

Main proposed evolutions of CAN MAC

- Integration of the time triggered paradigm in the CAN MAC
  - Control of the jitter for the periodic traffic
  - Possibility to integrate some dynamic priority scheduling
  - The synchronization between all the stations becomes mandatory
  - Keep some event triggered paradigm for non periodic traffic
  - The two main proposed solutions
    - The Time-Triggered CAN (TTCAN) standard (2001)
    - Flexible Time-Triggered CAN (FTTCAN), proposed by the university of Aveiro
- CAN with Flexible Data rate (CAN FD)
  - Up to 64 data bytes per frame
  - Higher bandwidth for the data phase

Time-Triggered CAN: general overview

- Master/slave architecture
- Static scheduling of CAN messages, stored in the system matrix
- Each station has a partial knowledge of the matrix
- Synchronization between stations via a periodic message transmitted by the TTCAN master
- The system matrix: exclusive, arbitrary and free windows



Time-Triggered CAN: general overview

- Characteristics of the system matrix
  - Sequence of different elementary cycles
  - All the windows of the same column have the same length
  - Windows of different columns can have different lengths
  - The number of lines is a power of 2
- Different types of windows
  - The exclusive window
    - Allocated to a given message (dedicated to periodic transmissions, time-triggered)
    - No collision, no retransmission
  - The arbitrary window
    - Any message can be transmitted (dedicated to aperiodic transmission, event-triggered)
    - native CAN MAC for collision resolution, possibly retransmission
  - The free window
    - for the evolution of the application
- Every station knows its part of the system matrix

Time-Triggered CAN: a first example

- A set of three periodic messages with harmonic periods

Message	Priority	Period	Deadline	Data bytes
A	1	4 ms	4 ms	7 bytes
B	2	8 ms	8 ms	7 bytes
C	3	16 ms	16 ms	7 bytes

Bandwidth of the CAN bus: 125 Kb/s

Identifiers on 11 bits

- One possible TTCAN matrix: One line is 4 ms

Ref	A	Arb	B
Ref	A	Arb	C
Ref	A	Arb	B
Ref	A	Arb	Free
0.520 ms	1.030 ms	1.360 ms	1.090 ms

- No jitter for any message

Time-Triggered CAN: a second example

- A set of four periodic messages

Message	Priority	Period	Deadline	Data bytes
A	1	4 ms	4 ms	4 bytes
B	2	4 ms	4 ms	4 bytes
C	3	6 ms	6 ms	4 bytes
D	4	6 ms	6 ms	4 bytes

Bandwidth of the CAN bus: 125 Kb/s
Identifiers on 11 bits

- One possible TTCAN matrix: One line is 3 ms

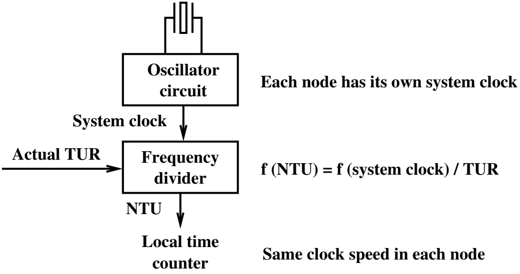
Ref	A	C	B
Ref	Arb	A	D
Ref	B	C	A
Ref	Arb	B	D
0.520 ms	0.790 ms	0.830 ms	0.860 ms

- Jitter for A (0.670 ms) and B (0.670 ms), no jitter for C and D

Time-Triggered CAN: timing

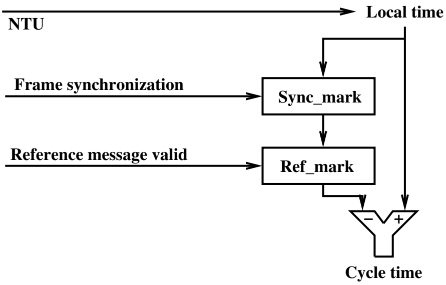
- Two possible levels for Time-Triggered CAN
  - Level 1: the time-triggered behaviour is insured by a periodic reference message broadcasted by the master $\Rightarrow$ one data byte
  - Level 2: the clock drift is corrected with each reference message $\Rightarrow$ 4 data bytes
- Three time bases of the TTCAN protocol
  - The local time
    - The time base of each node
    - Obtained by dividing the native clock of the node by a Time Unit Ratio
    - Network Time Unit: common unit for the local times of all the stations
  - The cycle time: time ellapsed since the SOF of the last received reference message
  - The global time (only for TTCAN level 2): a kind of global clock for the whole network
    - Each node Compensates the phase drift between its local time and the global time received in each reference message by updating its local offset
    - Each node adapts its Time Unit Ratio at each reference message

Time-Triggered CAN: local time



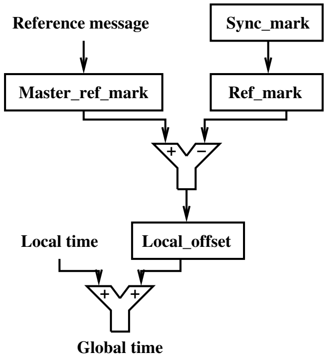
- TTCAN level 1
  - TUR (Time Unit Ratio): constant value
  - Local time counter: 16 bit integer value, incremented once each NTU (Network Time Unit)
- TTCAN level 2
  - TUR: non integer value, may be adapted to compensate clock drift or to synchronize to an external time base
  - Local time counter: 16 bit integer value, extended by a fractional part of N (at least 3) bits, incremented 2^N times each NTU $\Rightarrow$ higher time resolution

Time-Triggered CAN: cycle time



- Hardware capture in *Sync_mark* of the value of the local time counter at each SOF bit
- A frame is recognized as a reference message ⇒ the current *Sync_mark* becomes the new *Ref_mark*
- Cycle time: the difference between the local time counter and *Ref_mark*

Time-Triggered CAN: global time



- The master transmits its view of *Ref_mark* in each reference message
- Local offset: difference between the *Ref_mark* of the master and the local *Ref_mark*

Time-Triggered CAN: compensate clock drift

- Principle: adapt the TUR that generates the local NTU from the local system clock
- df: factor by which the local NTU has to be adjusted

$$df = \frac{Ref\_mark - Ref\_Mark_{previous}}{Master\_ref\_mark - Master\_ref\_Mark_{previous}}$$

- Adaptation of TUR

$$TUR = df \times TUR_{previous}$$

- Small example

	Master_ref_mark	Ref_mark
Actual	2000	4300
Previous	1000	3200

$$df = \frac{1100}{1000} = 1.1 \Rightarrow TUR = 1.1 \times TUR_{previous}$$

TTCAN exercice

- TTCAN bus with 250 kbs bandwidth
- The reference message consumes 400 μs at the beginning of each cycle
- Set of messages

	Period	Deadline	Data bytes
$f1$	3 ms	4 ms	2
$f2$	3 ms	4 ms	2
$f3$	6 ms	7 ms	4
$f4$	6 ms	7 ms	4
$f5$	9 ms	10 ms	6
$f6$	9 ms	10 ms	6
$f7$	12 ms	13 ms	8
$f8$	12 ms	13 ms	8

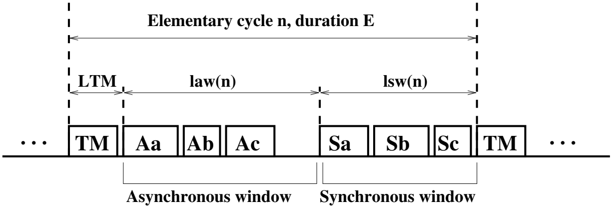
Build a matrix such that all the messages respect their deadlines

Flexible Time-Triggered CAN

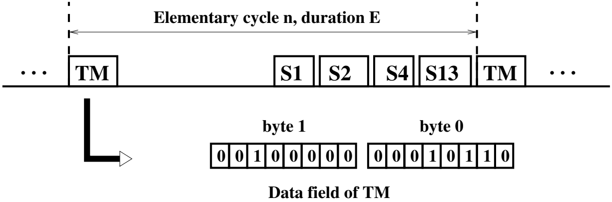
- Main goals of FTTCAN
  - Sharing of time between periodic (time-triggered) and aperiodic (event-triggered) traffic
  - temporal isolation between the two types of traffic
  - Flexible management of the periodic traffic
  - Efficient management of the aperiodic traffic
  - Possibility to change dynamically the scheduling of the transmissions
- Basic principles of FTTCAN
  - Can be seen as an evolution of TTCAN
  - Master/slave architecture
  - Time is divided in elementary cycles, each composed of
    - ★ A trigger message from the master at the beginning of each elementary cycle (synchronization, content of the cycle)
    - ★ An asynchronous window (for aperiodic messages)
    - ★ A synchronous window (for periodic messages)
  - The trigger message includes the identifiers which have to be transmitted in the synchronous window
  - Native CAN MAC in both synchronous and asynchronous windows

Flexible Time-Triggered CAN

- The elementary cycle of FTT-CAN



- Management of the synchronous window



CAN with Flexible Data rate (CAN FD)

- Native CAN: bandwidth limited due to bit by bit arbitration
- Mandatory only during the arbitration phase
- Increase the bandwidth after the arbitration phase and increase the number of bytes in the data field
- New frame format