Communication systems for vehicle electronics

Presentation overview

- ***Background***

  automotive electronics as an application area for real-time communication

- ***Real time protocols***

  LIN – Local Interconnection Network

  CAN – Controller Area Network

  TTCAN, - Time Triggered CAN (based on “Controller Area Network” (CAN)

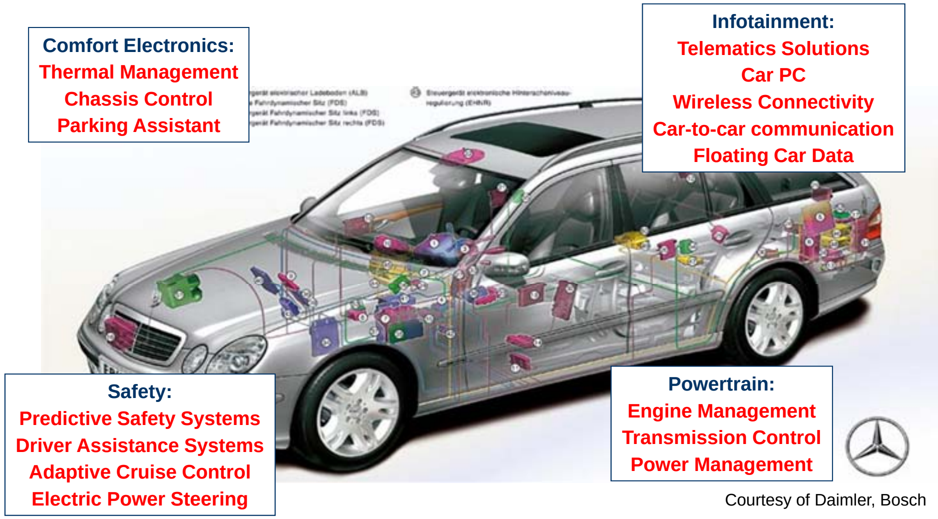
  CAN FD – CAN with Flexible Data-rate

  FlexRay, based on BMW’s “ByteFlight”

- ***Hybrid scheduling***

  combining static scheduling with fixed priority scheduling analysis

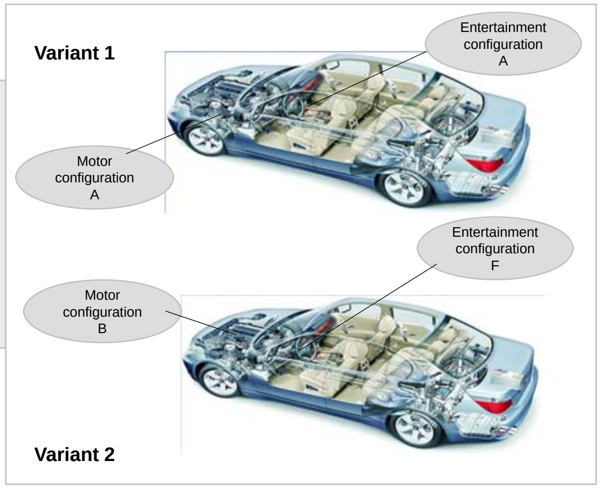
A premium passenger car is controlled and managed by 80+ Embedded Systems



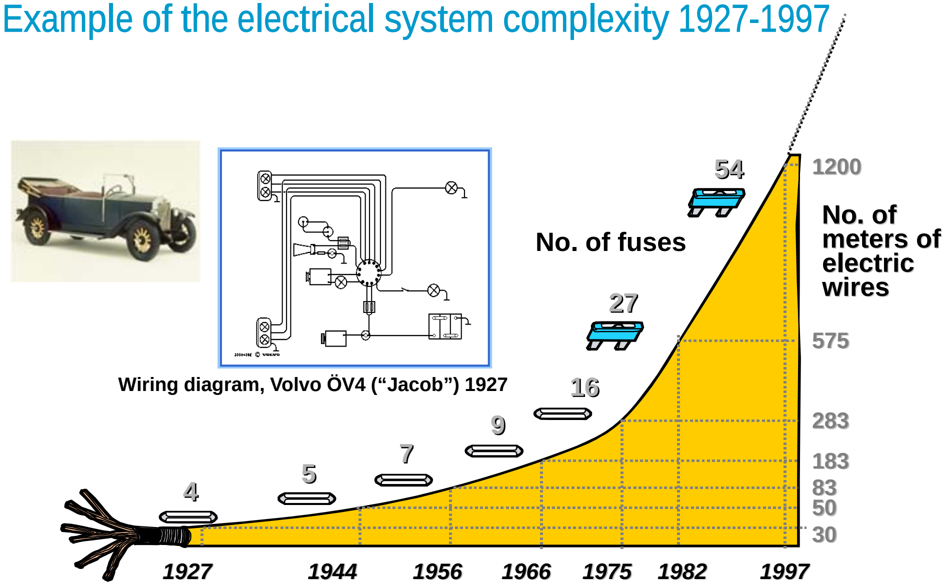
Courtesy of Daimler, Bosch

Virtual differentiation between variants

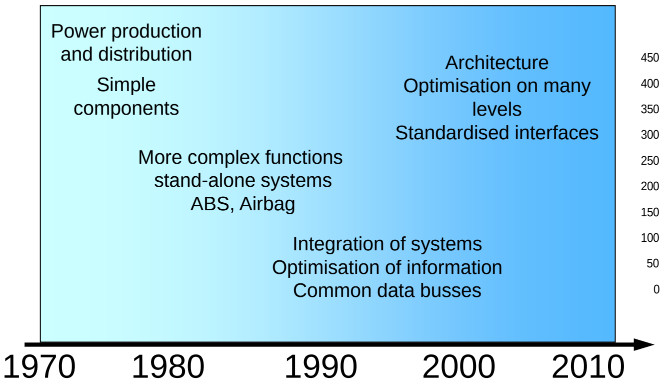
- **All variants of a specific model are physically identical and differ only in their individual software configuration**
- **The various included physical components can be activated or deactivated by the software**

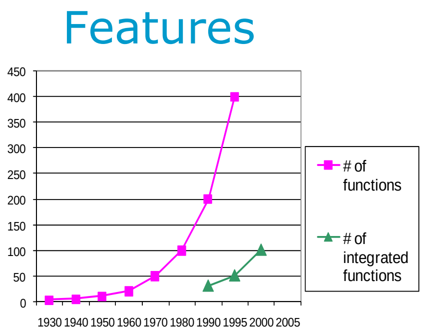


Example of the electrical system complexity 1927-1997

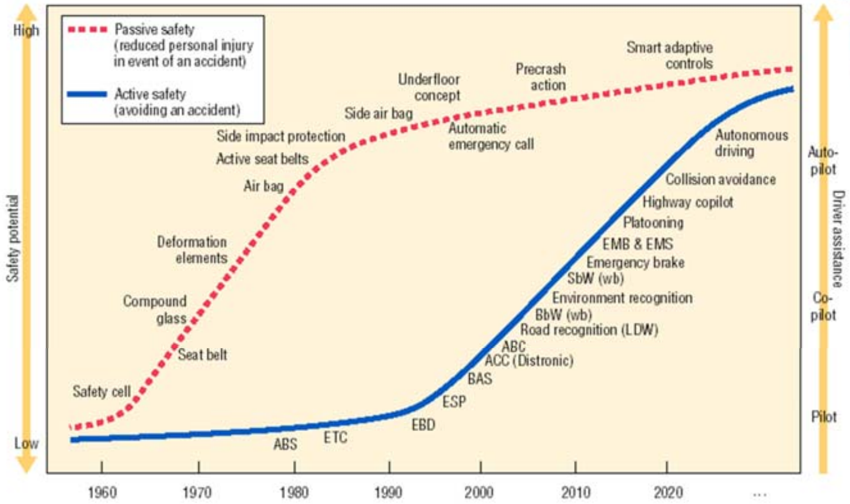


The evolution of functional requirements on the electrical system

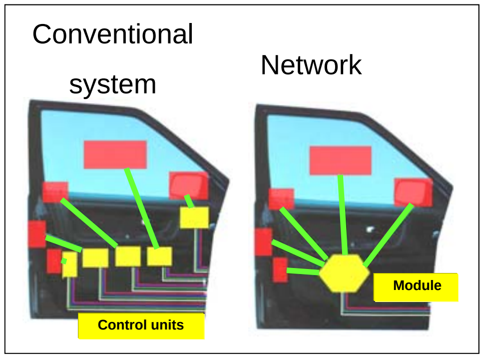


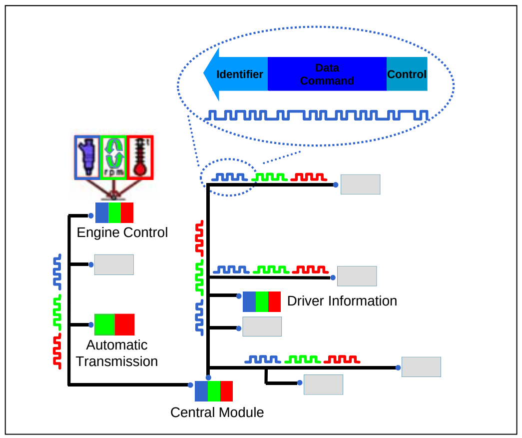


Automotive electronics roadmap

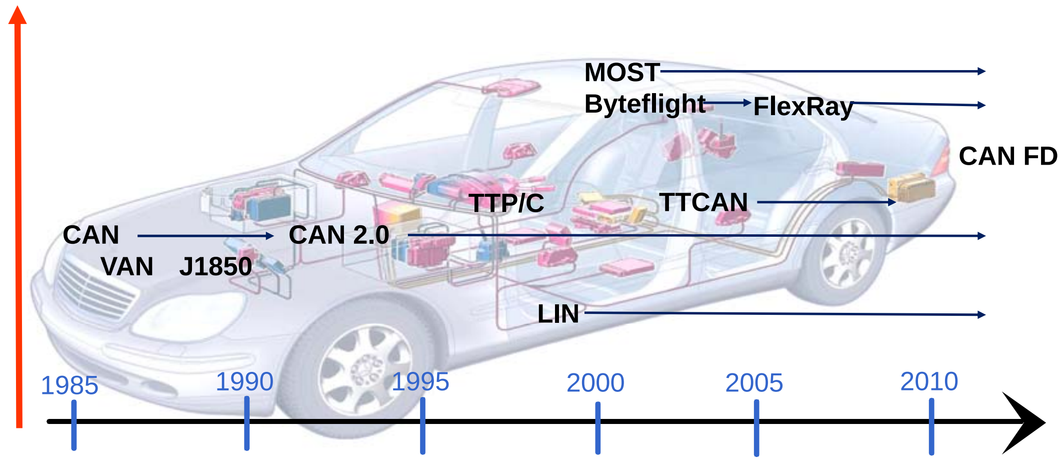


Multiplex Networks

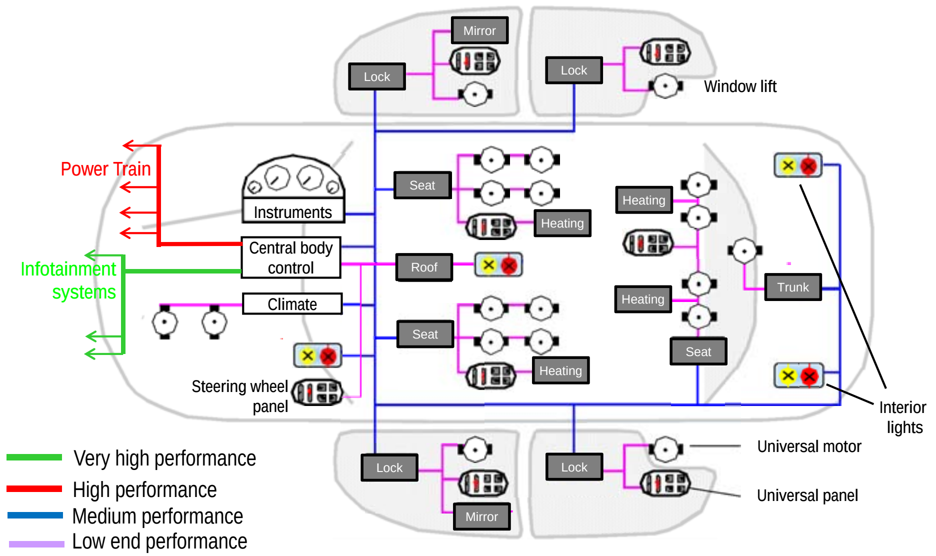




Evolution of protocols



Example of the electrical system…



The LIN protocol, started in 1998

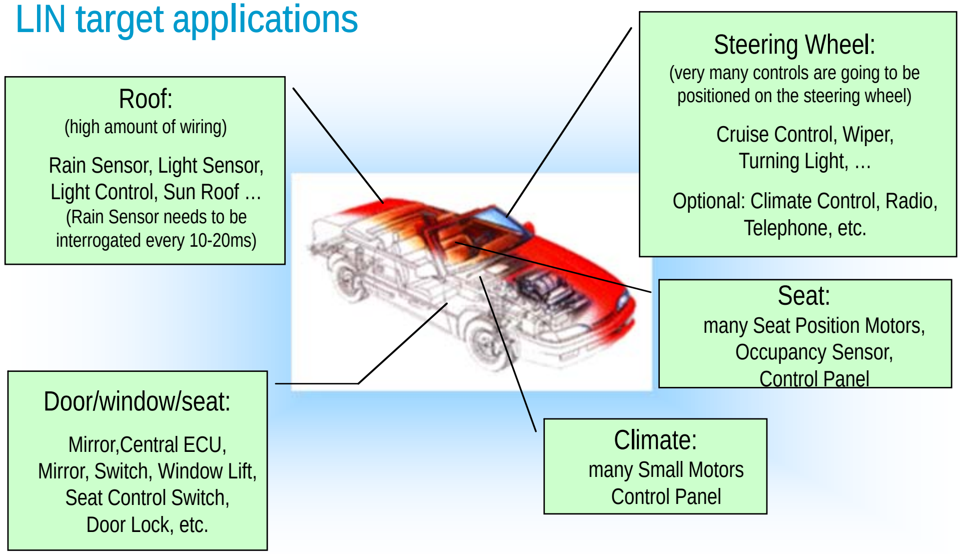
LIN Local Interconnection network
predecessor: VOLCANO Lite

LOCAL INTERCONNECT NETWORK

Cooperation between partners:
Freescale, VOLVO CAR, BMW, AUDI, Volkswagen, Daimler-Chrysler
Mentor Graphics (former: Volcano Communication Technology)

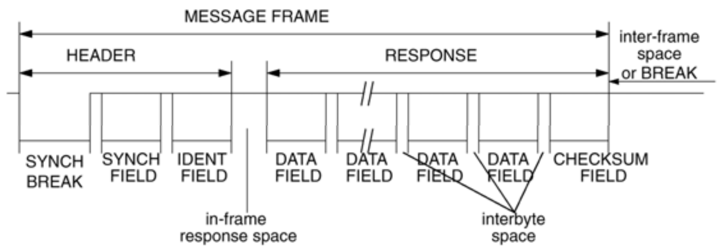
Objectives:
Low cost, modest performance and safety requirements, flexible system architecture

LIN target applications

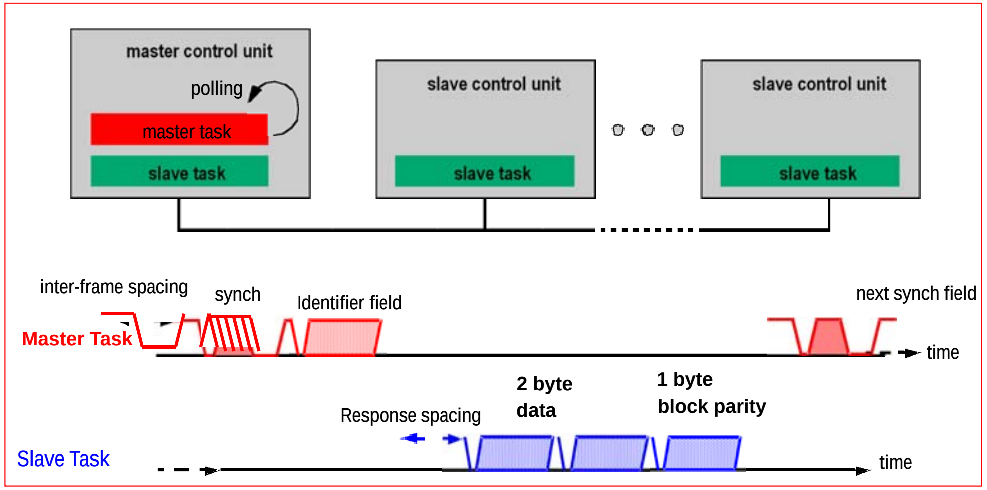


LIN protocol features

- Bus topology
- Master-slave protocol, no arbitration required
- UART protocol, 10 bits (uses "sync break" facility)
- 8 bits of data in a block
- 2-8 blocks of data per frame
- Single wire
- Maximum 20 kbits/s



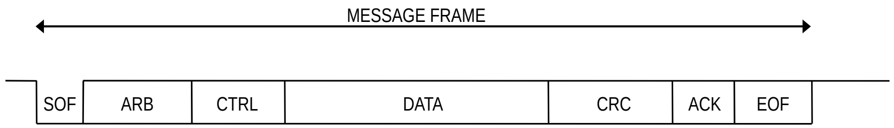
LIN bus communication



CAN – Controller Area Network

- Bus topology
- CSMA/CR (Carrier sense, Multiple Access/ Collision Resolution)
- Error detection capabilities
- Supports "atomic broadcast"
- 0-64 bytes of data per frame
- Twisted pair
- Maximum 1 Mbit/s

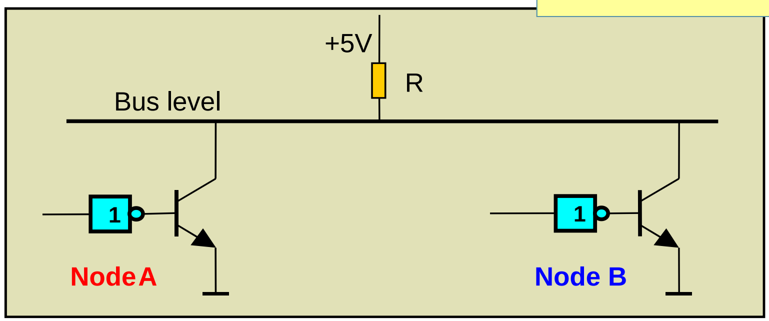
ARB	*Arbitration (identifier)*
CTRL	*Control information*
DATA	*0-8 bytes*
CRC	*Checksum*
ACK	*Acknowledge*
EOF	*End of frame*



Bus collission detection

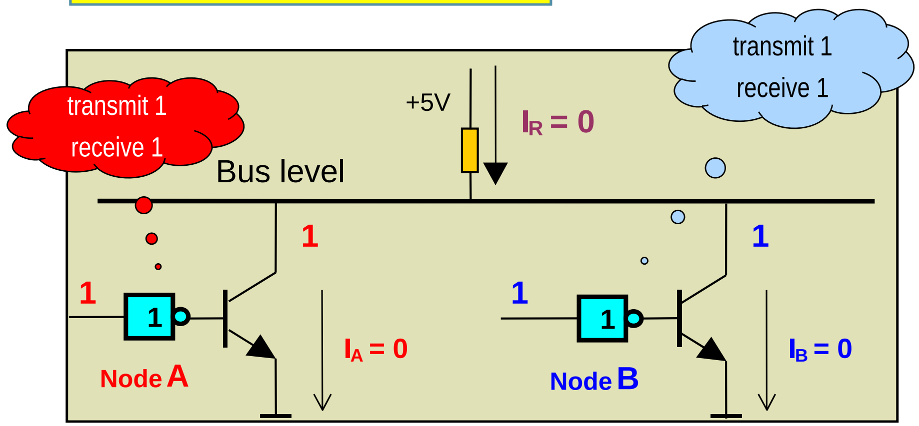
Bus transceivers
"Open collector"
Bus level:
Recessive (bit) "1"
Dominant (bit) "0"

Idle bus (recessive level)

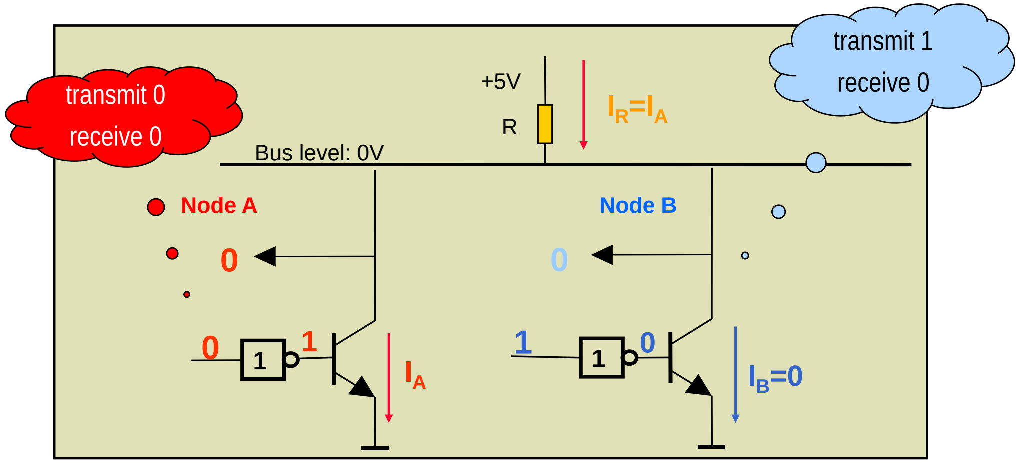


Bus arbitration

Two nodes transmitting same level (1)



Collission Resolution



Node B aborts transmission since the received bit differs from the transmitted bit

Three messages collide...

Arbitration field (identifier with priority)
Nodes "own" specific message identifiers.

EXAMPLE: Three nodes start simultaneously

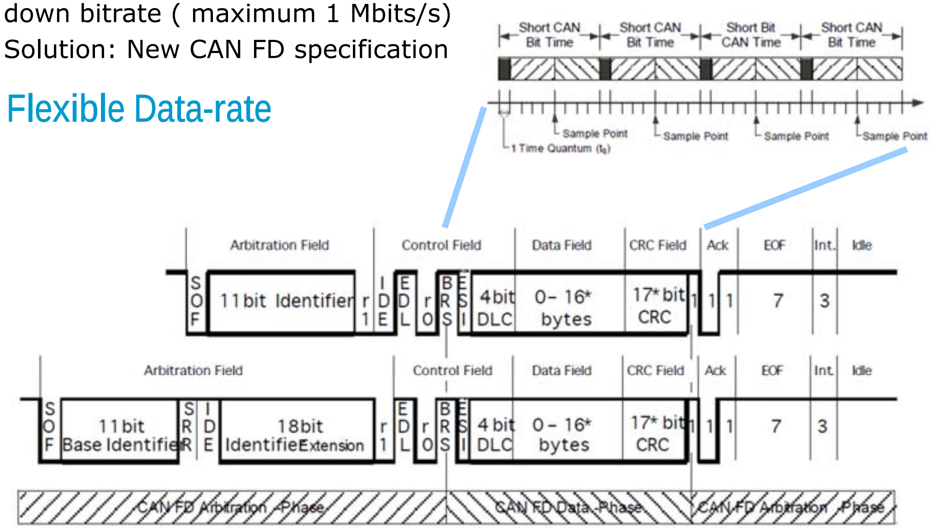
Node A transmits: $257 (0010 0101 0111)
Node B transmits: $360 (0011 0110 0000)
Node C transmits: $25F (0010 0101 1111)

Bit number	SOF	1	2	3	4	5	6	7	8	9	10	11	12	13
Bus level	D	D	D	R	D	D	R	D	R	D	R	R	R	R
Node A	0	0	0	1	0	0	1	0	1	0	1	1	1	1
Node B	0	0	0	1	1	Aborts								
Node C	0	0	0	1	0	0	1	0	1	1	Aborts			

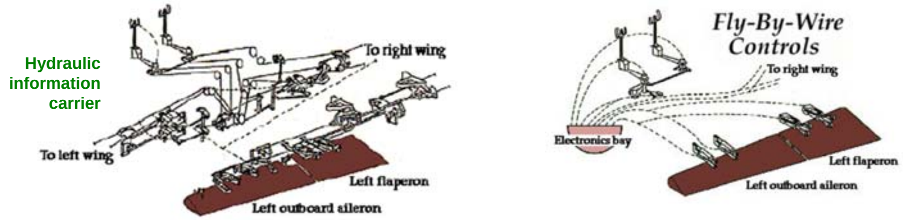
Standard/Extended CAN drawback....

- Protocol bus arbitration, acknowledge and error handling slow down bitrate (maximum 1 Mbits/s)
- Solution: New CAN FD specification

CAN Flexible Data-rate



By-wire control



The F-8 Digital Fly-By-Wire (DFBW) flight research project validated the principal concepts of all-electric flight control systems now used on nearly all modern high-performance aircraft and on military and civilian transports. The first flight of the 13-year project was on May 25, 1972.

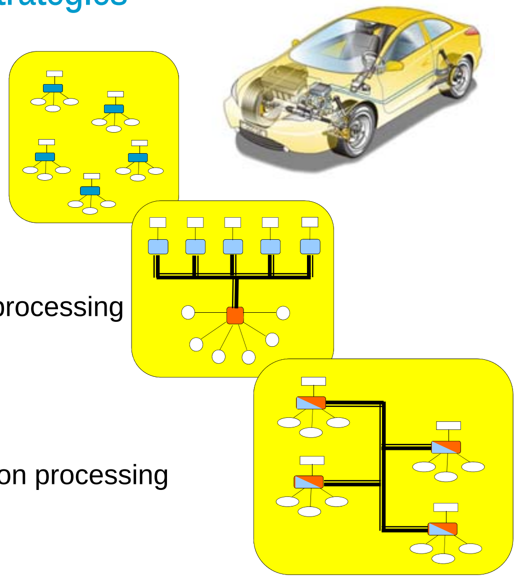
Courtesy of Dryden Flight Research Center

Control system implementation strategies

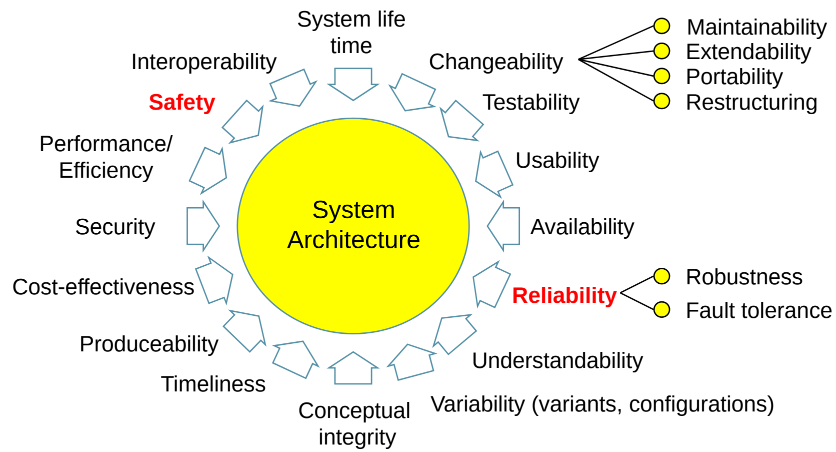
Local control
- Local information processing
- Independent control objects

Centralized global control
- Local and central information processing
- Interconnected control objects

Distributed global control
- Local and distributed information processing
- Interconnected control objects



Non-functional requirements



Tradeoffs from Safety/Reliability requirements

The extremes from reliability requirements leads to safety requirements.

Safety requirements implies redundancy, (Fail-Operational, Fail-Safe, etc).

Safety requirements also demands predictability, we has to show, a priori, that the system will fulfill it's mission in every surrounding at every time.

- ***In a distributed environment, only time triggered protocols with redundant buses can provide this safety.*** *Contemporary TTP's are:*

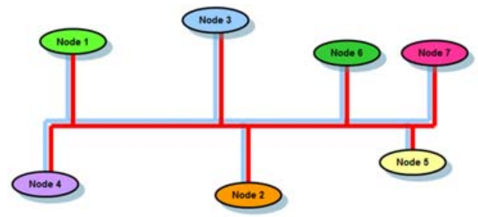
TTCAN, based on *Controller Area Network* (CAN) which is widely used in today's vehicular electronic systems.

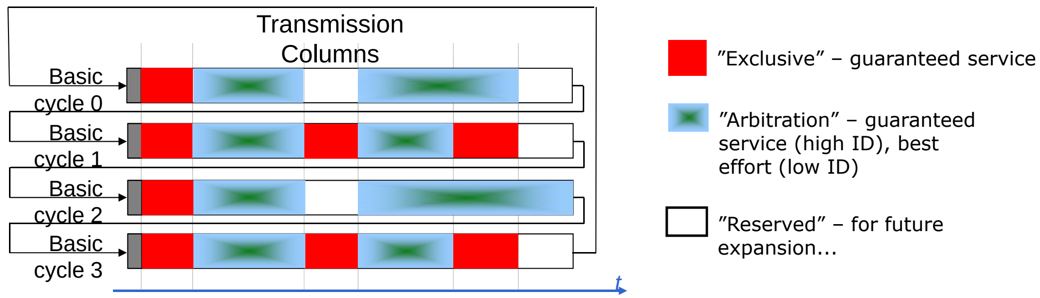
FlexRay, based on BMW's “ByteFlight”. Operational in contemporary automotive electronic systems.

TimeTriggered Ethernet. TTEthernet expands classical Ethernet with services to meet time-critical, deterministic or safety-relevant conditions.

Time Triggered CAN

- Based on the CAN protocol
- Bus topology
- Media: twisted pair
- 1Mbit/s

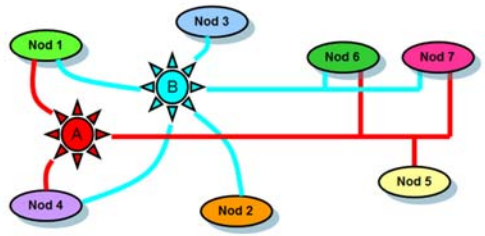




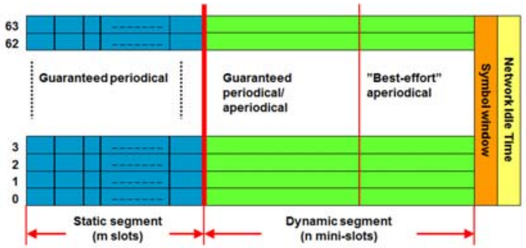
Time is global and measured in *network time units* (NTU's)

Flexray

- Double channels, bus or star (even mixed).
- Media: twisted pair, fibre
- 10 Mbit/s for each channel



Redundant channel can be used for an alternative schedule



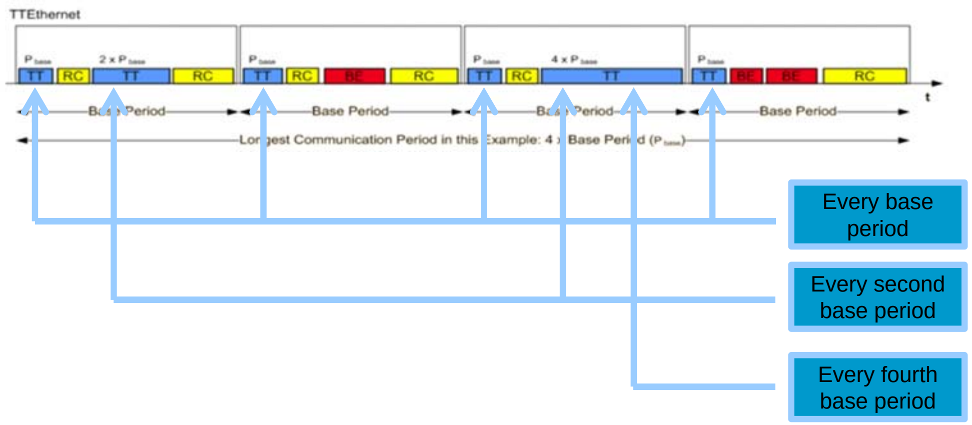
"Static segment" (TTCAN "Exclusive") – guaranteed service

"Dynamic segment" (TTCAN "Arbitration") – guaranteed service (high ID), "best effort" (low ID)

Max 64 nodes on a Flexray network.

Time Triggered Ethernet

- Classic Ethernet bus topology
- 1 Gbit for each channel



Compare with TTCAN "basic cycles"

Comparisons

All protocols targets real time applications.
Provides for *time* AND *event* triggered paradigms.

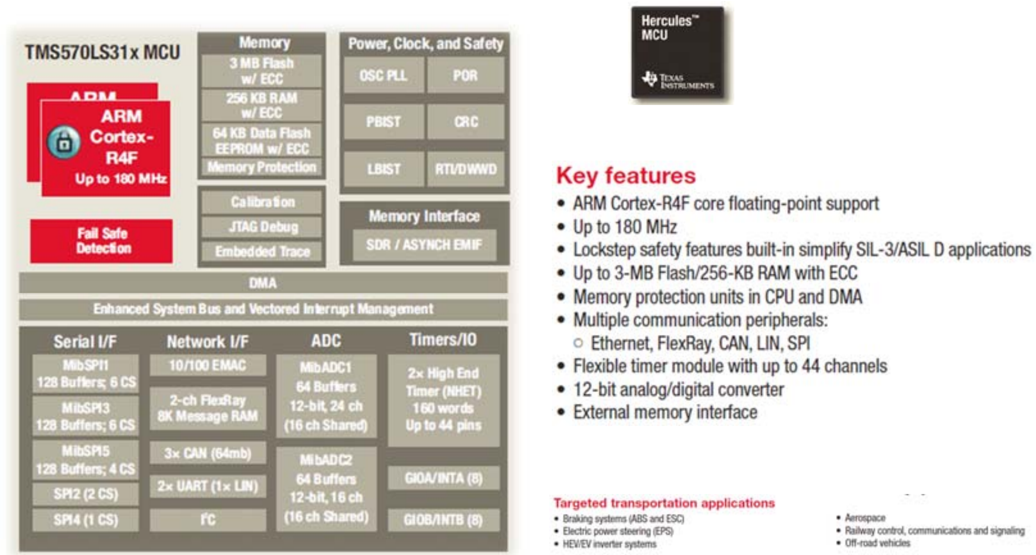
All protocols are suitable for scheduling tools.
Commercial production tools are available.

CAN, many years experiences, a lot of existing applications.
Implies migration of existing CAN applications into TTCAN and CAN FD.

Flexray is the automotive industries initiative.
New hardware, promoted in for example "AUTOSAR".

TTEthernet.
Proven technology with lots of existing hardware,

What to choose?

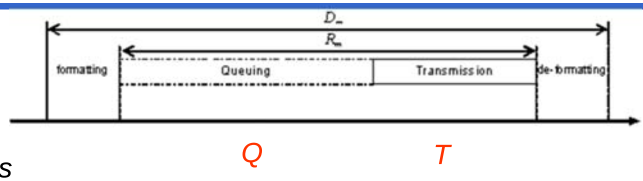


Combining time triggering with events: Example of Hybrid scheduling for *TTCAN*

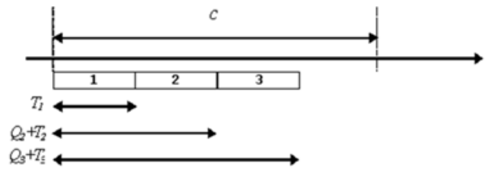
Messages are sorted into three different categories:

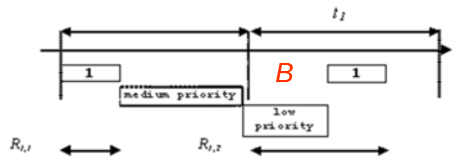
- Hard real-time, for minimal jitter with guaranteed response time.
- Firm real-time, for guaranteed response time, but can tolerate jitter.
- Soft real-time, for "best effort" messages.

TTCAN detailed study



Response time analysis

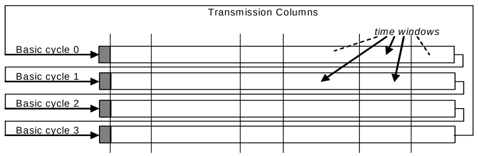




$$Q_i = \sum_{\forall j: P_j < P_i} \left\lceil \frac{Q_i}{t_j} \right\rceil T_j$$

$$R_i = B_i + T_i + Q_i$$

Time triggered messages M^h



After structuring:
M : {M^h, M^f, M^s}, assume that at least M^h is defined. We now construct a matrix cycle. Due to protocol constraints, the schedule has to fulfil:

$$\mathbf{LCM}(M^h{}_p) = x\,2^n$$

where:
- LCM is *least common multiple* period for the M^h message set;
- x is the preferred length of a basic cycle within LCM;
- n is the number of *basic cycles*.

Hardware constraints:
Hwc1: $1 \le x \le 2^y$, has to be consistent with a hardware register, y bits
Hwc2: $0 \le n \le k$, always a power of 2, constraint in hardware.
Hwc3: # of triggers ≤ Tr, columns in the matrix cycle. Limited by the number of available trigger registers.

Multiple solutions satisfies the equation...

Choose a strategy:

Strategy 1:

Minimize number of *basic cycles*, requires a longer *basic cycle*, and more *triggers*.

Strategy 2:

Minimize length of *basic cycles*, increase probability of finding a feasible schedule for large message

Persuing the strategies...

Construct a schedule for the following set:

M^h = (*M1*, *M2* , *M3*) with the following attributes (NTU):
$M1_p$ = 1000, $M1_e$ = 168
$M2_p$ = 2000, $M2_e$ = 184
$M3_p$ = 3000, $M3_e$ = 216

It's obvious that:

LCM(*M1*, *M2* , *M3*) = 6000.

and:

$6000 = x\,2^n$

Strategy 1

Minimizing number of basic cycles yields: $2^n = 1$, so $n = 0$ and $x = 6000$.
Hwc1 and Hwc2 are fulfilled.
Total numbers of *triggers* for *N* messages in one *basic cycle* is:

$$\sum_{i=1}^{N} \frac{LCM(\,\mathbf{M}\,)}{M^i}$$

in this case:

$$\#\text{ of } triggers = \frac{6000}{1000} + \frac{6000}{2000} + \frac{6000}{3000} = 11$$

So, strategy 1, leads to a solution with:

- 1 *basic cycle* and 11 triggers.
- MAtrix cycle length is 6000 NTU.

Basic Cycle Triggers																	
0	168	352		1000		2000	2168		3000		3352		4000	4168		5000	
M_1	M_2	M_3		M_1		M_1	M_2		M_1		M_3		M_1	M_2		M_1	

Strategy 2

n = 0:
6000 = x 2^0 ⇒ *x = 6000*
(same as strategy 1)

n = 1:
6000 = x 2^1 ⇒ *x = 3000*

n = 2:
6000 = x 2^2 ⇒ *x = 1500*

n = 3:
6000 = x 2^3 ⇒ *x = 750*

n = 4:
6000 = x $\mathbf{2^4}$ ***⇒ x = 375***

n = 5:
6000 = x 2^5 ⇒ *x = 187.5*

Basic cycle								*Trigger* Information / Minimum *Triggers*
1 (at 0)	0	168	352	-	-	-	-	
2 (at 375)	-	-	-	-	-	-	-	
3 (at 750)	-	-	-	-	-1000	-	-	
4 (at 1125)	-	-	-	-	-	-	-	
5 (at 1500)	-	-	-	-	-	-	-	
6 (at 1875)	-	-	2000	2168	-	-	-	
7 (at 2250)	-	-	-	-	-	-	-	
8 (at 2625)	-	-	-	-	-	-	-	
9 (at 3000)	3000	-	-	-	-	3352	-	
10 (at 3375)	-	-	-	-	-	-	-	
11 (at 3750)	-	-	-	-	-	-	4000	
12 (at 4125)	4125	4168	-	-	-	-	-	
13 (at 4500)	-	-	-	-	-	-	-	
14 (at 4875)	-	-	-	-	-	-	5000	
15 (at 5250)	-	-	-	-	-	-	-	
16 (at 5625)	-	-	-	-	-	-	-	
1	M_1	M_2	M_3					3
2								0
3					M_1			1
4								0
5								0
6			M_1	M_2				2
7								0
8								0
9	M_1					M_3		2
10								0
11							M_1 ?	1
12	? M_1	M_2						2
13								0
14							M_1	1
15								0
16								0

Strategy 2

Avoid this conflict with the requirement that:
a *basic cycle* shall be *at least as long as* the shortest period in the message set.

Applying this restriction we get:
n = 2, (*x* = 1500)
which yields a feasible schedule:

Basic cycle									*Trigger* Information / Minimum *Triggers*
	1	0	168	352	-	-	-	1000		
	2	-	-	-	2000	2168	-	-		
	3	3000	-	3352	-	-	4000	4168		
	4	-	-	-	5000	-	-	-		
1		M_1	M_2	M_3				M_1		4
2					M_1	M_2				2
3		M_1		M_3			M_1	M_2		4
4					M_1					1

Verifying the events... (M^f)

Basic Cycle	Grey slots are supposed to be allocated for M^h NTU-slots (Columns)								
1		q_0							
2		q_1					q_2		
3		q_3		q_4			q_5		
.....		...			...		...		...
2^n		q_{N-3}					q_{N-2}		q_{N-1}

```
for each message m in M^f :
  for message m = 1 up to last_m
    for virtual message VM_i = 1 up to last_VM
      if( Q_m + T_m ) falls within ( VM_i,start , VM_i,completion )
        Q_m = VM_i,completion
      else
```

$$Q_m = \sum_{\forall j: P_m < P_j}^{1} \left\lceil \frac{Q_m}{t_j} \right\rceil T_j$$

```
      endif
    end
  end
end
```


Thank you for your attention.