Operating Systems, Concurrency and Time

real-time communication and CAN

Johan Lukkien

(Courtesy: Damir Isovic, Reinder Bril)

Question

- Which requirements to communication arise from real-time systems?
- How does CAN – as an example – work?

Networking in real-time context

Heterogeneous

Distributed applications

- components even on different networks
- guarantees and admission tests
- online adaptation

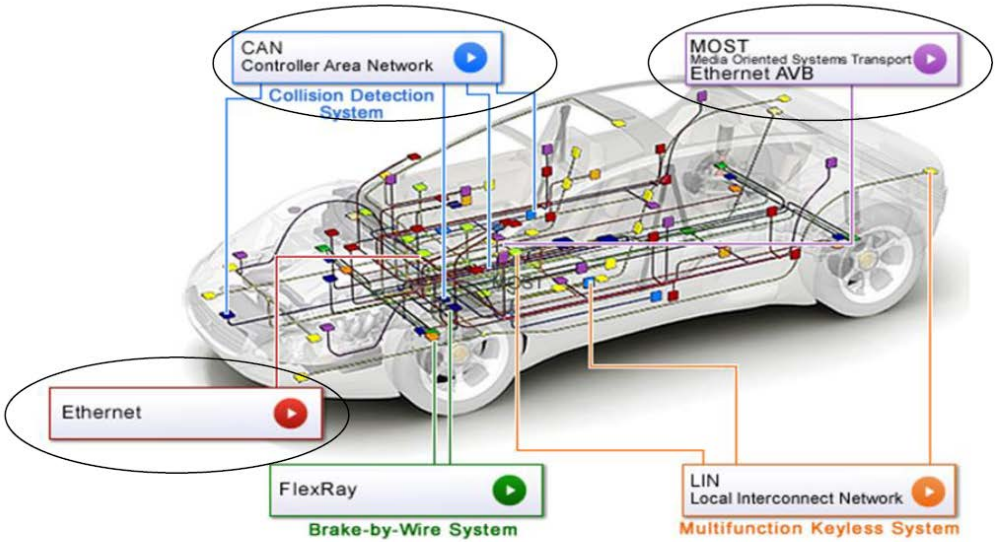


Figure courtesy of www.renesas.eu

Open (more than before)

- embedded data available for (new) data oriented applications
- security for privacy and safety

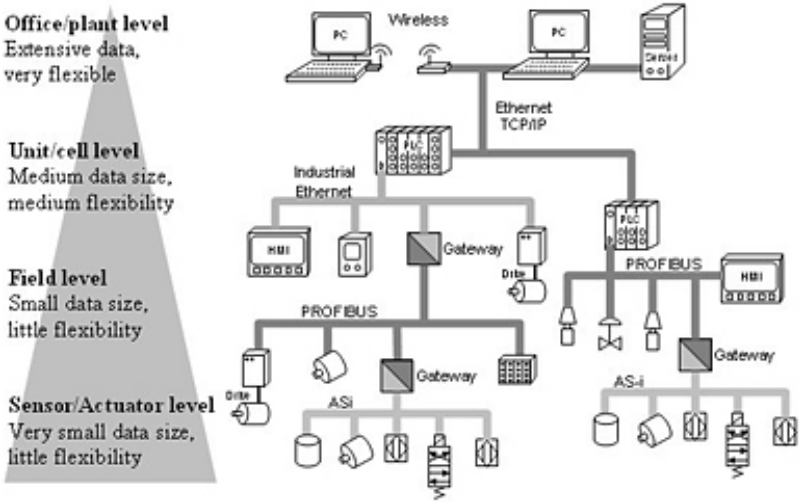
Data and Control: connecting the 'physical' to the 'Cyber'

- information about the physical world to higher layers ICT
  - local sensors and actuators
- requirements of power use, throughput, latency, jitter

Fieldbus technology

Fieldbus for distributed control

- intended for lowest layer in control system: connecting PLCs, sensors, actuators
- reduction of wires and complexity (replaces 4-20mA standard)
- high reliability, predictability; relatively low data rates and small messages



from http://www.profibusaustralia.com.au/

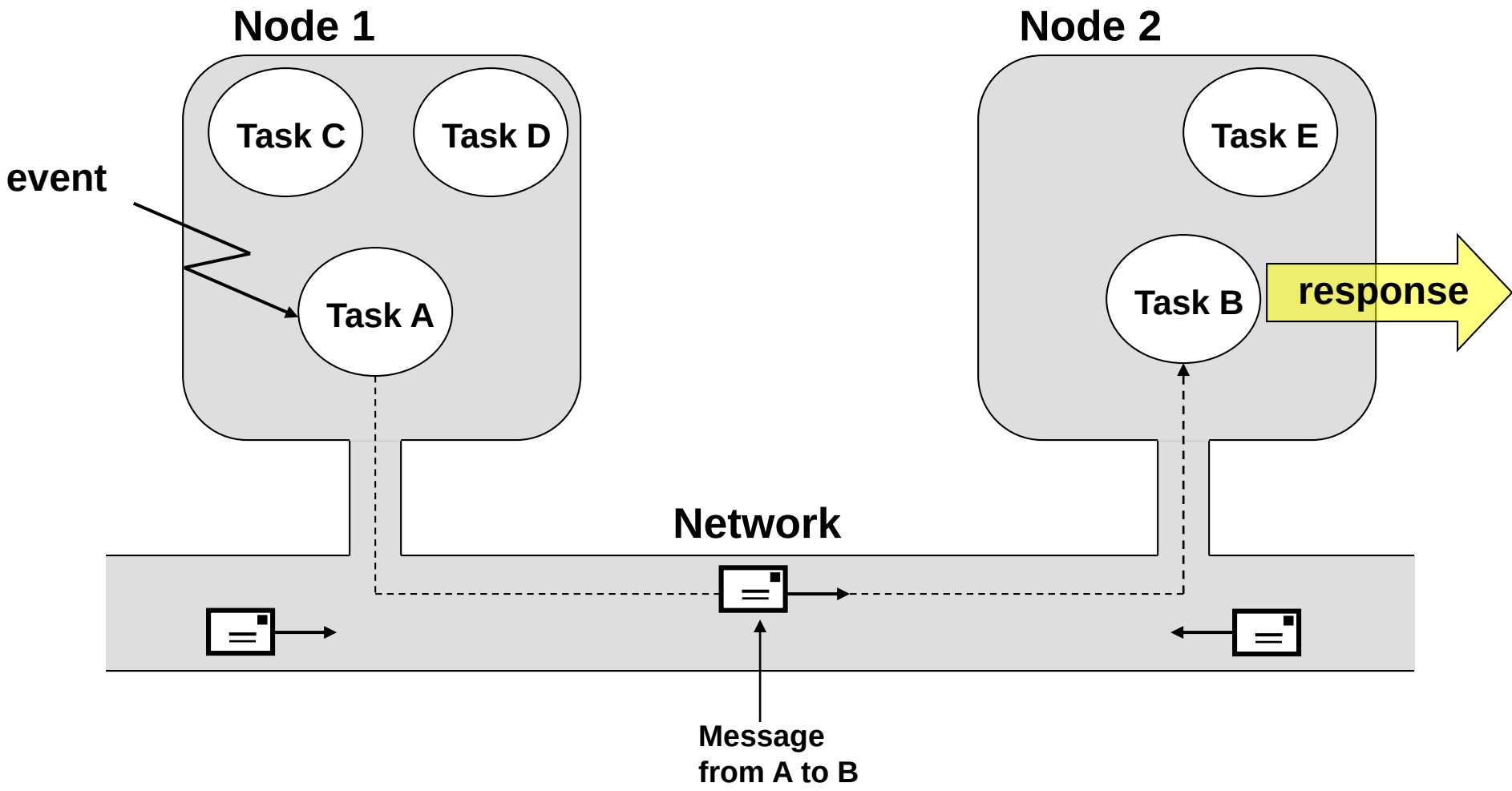
Networking

- physically: bus, ring
- logically: bus (shared medium), mostly
- OSI model: focus on PHY, MAC (LLC) and application layer
  - messages have application meaning ('commands')
  - not straightforward to integrate with other networks (or IP networks)
  - PLCs might perform gateway functionality, or be integrated in IP

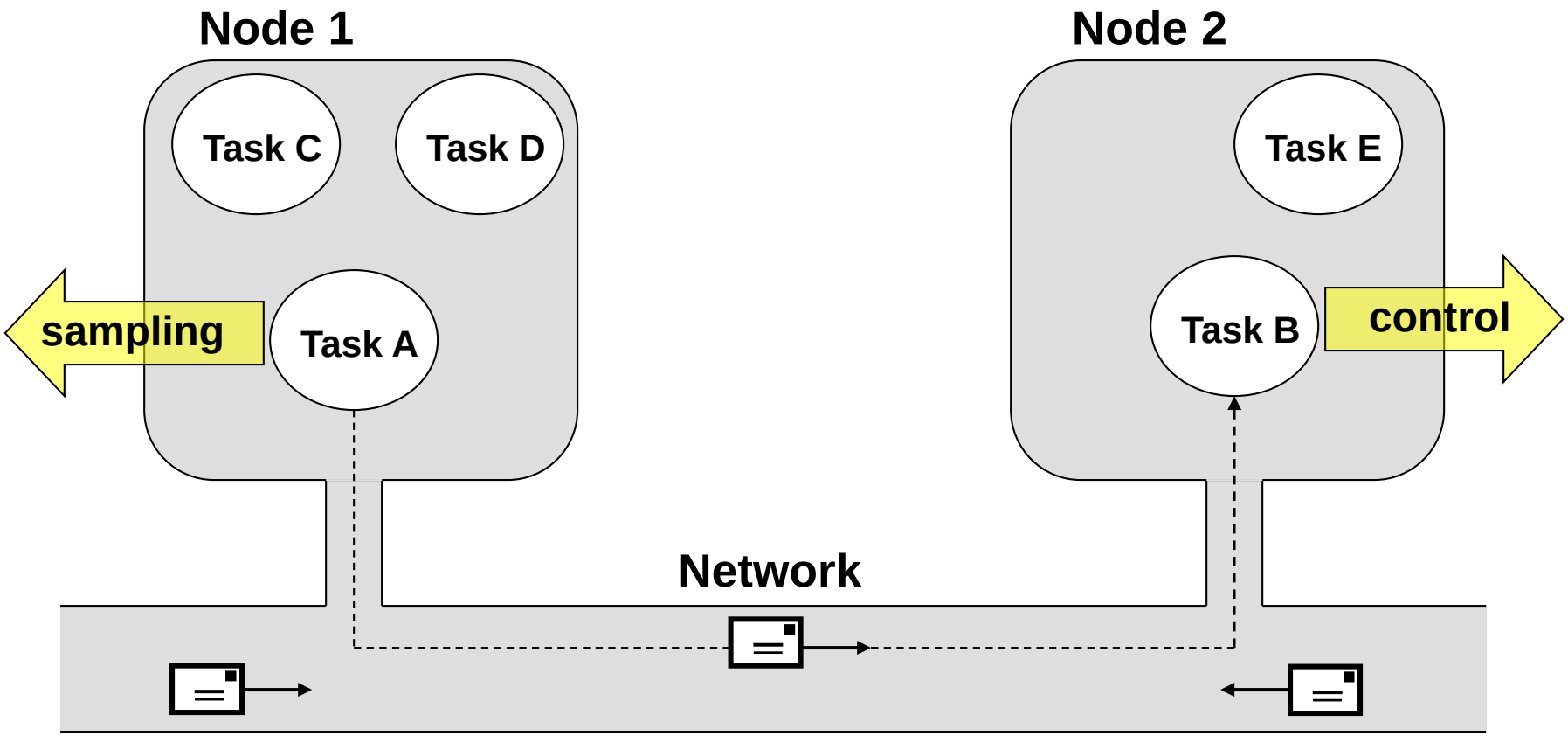
Standards

- IEC 61158, general fieldbus description
- AS-i, **CAN**, ETHERCAT, LON, PROFIBUS

Event-trigged communication



Time-trigged communication



Distributed real-time systems require robust real-time communication

Non real-time systems

- Throughput
- Average response time
- Average latency

Real-time systems

- Predictability
- Timing requirements on individual response times and latencies
- Require predictable communication network
- Analysis *before* the system is operating

Challenge

- to construct computer systems that have at least **reliability** and **safety** that are at least as good as the (mechanical) systems they are replacing

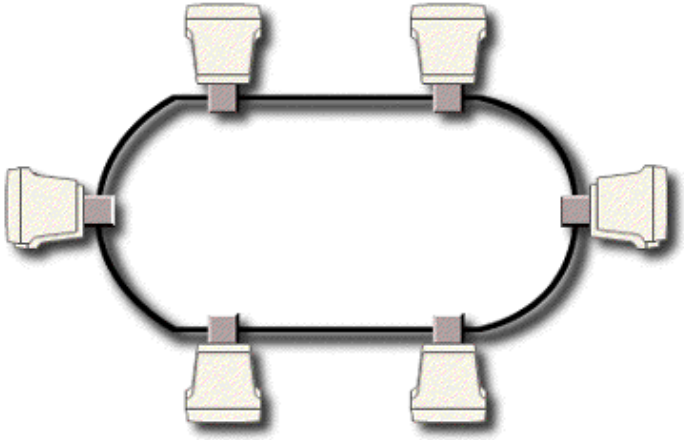
Communication protocols

(Old) Ethernet

- Addressed broadcast messages
- Collision → Nodes resend after a random time
- Impossible to determine transmission times → Not suitable for hard real-time systems

Token ring

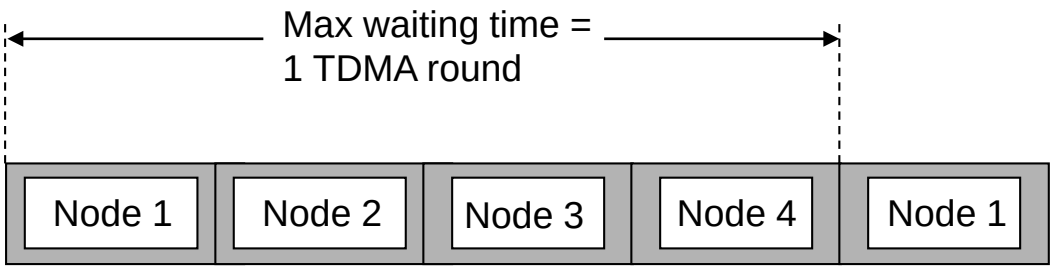
- Circulating token
- No collisions
- RT guarantees possible



Suitable for RT-communication

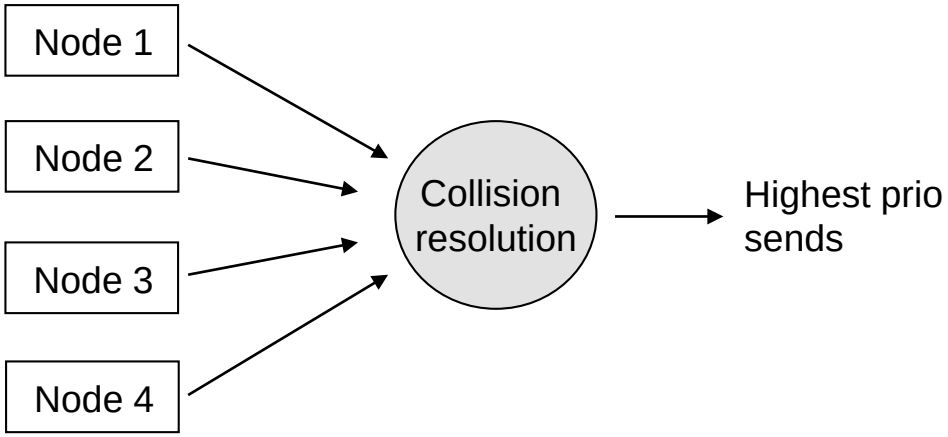
TDMA

- Time-trigged (periodic)
- Predictable
- High testability
- *<u>Example</u>: TTP-protocol*



CSMA/CR (collision resolution)

- Priority based
- Online scheduled
- Flexible
- *<u>Example</u>: CAN-protocol*



CAN – Control Area Network

Originally developed for automotive industry needs

- 1983: BOSCH starts CAN development (Intel joins 1985)
- 1987: First CAN chip
- 1990: First car with CAN (Mercedes S-class)
- 1993: ISO standard

Now used in industry applications as well

- Very common in machinery
- CAN-controllers developed by Philips, Intel, NEC, Siemens …

An implementation of CSMA/CR

- Priority based
- CR is the central mechanism
- *Bitwise arbitration* to resolve collisions

CAN Variants: ISO 11898

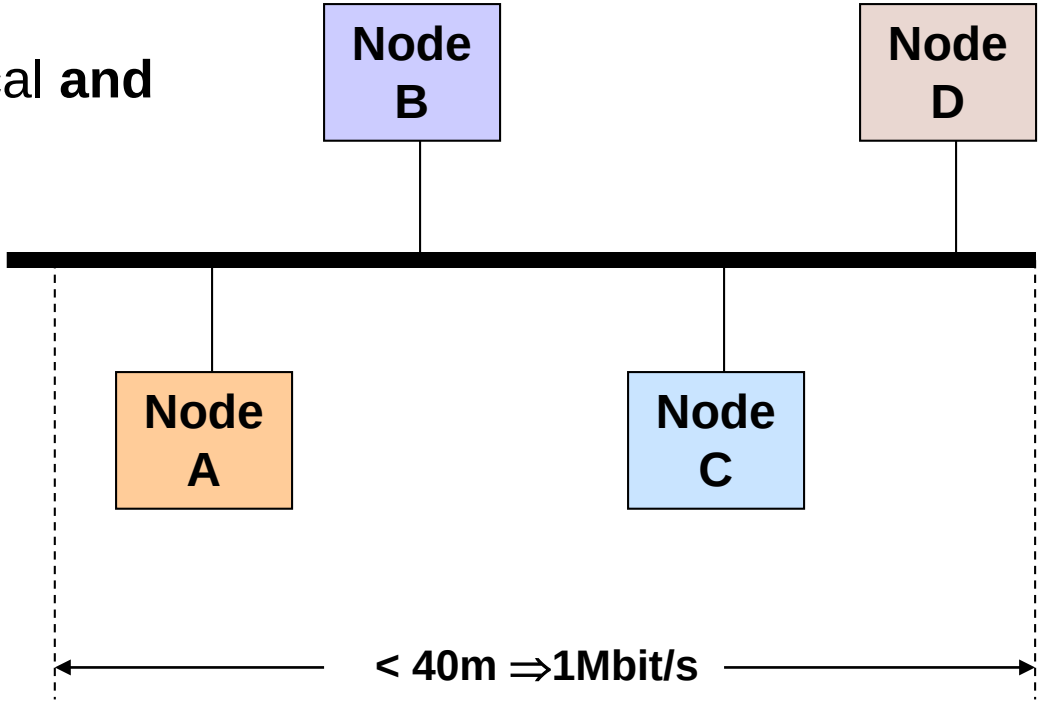
- ISO 11898: physical and datalink layers
- 11898-1 (2003): general architecture, PHY, DLL, LLC, MAC
- 11898-2 (2006): High Speed (1 Mps)
- 11898-3 (2006): Fault tolerant, low speed (40K-125Kbps) PHY
- 11898-4 (2004): Time Triggered CAN, TTCAN
- 11898-5 (2007): extend 11898-2 with low power
- 11898-6 (2013): extend -2 and -5 with configurable frames and wakeup

- 15765-2 (2011): diagnostic messaging on CAN

Operation

- Synchronous serial communication
- A shared medium (cable) with connected nodes
- Broadcast – data transmitted as frames can be picked up by all other attached nodes
- 1 Mbit/s at 40m bus length

- Value of the bus is the logical **and** of values put on by stations
  - 0 highest priority
- priority is derived from the **identifier** (= type or class or function) of a message
  - not from source or destination (these are not part of the message)



Message format

Data frames

- Used for data transmission e.g., sampling values from a sensor
- Standard CAN frame (CAN 2.0 A), 11 bits identifier
- Extended CAN frame (CAN 2.0 B), 29 bits identifier

Remote Frames

- Used for information requests.
- The transmitting node is asking for information of the type given by the identifier.

Error frames

- Used for error signaling

Overload Frames

- Used to delay the transmission of the next message frame
- The node sending the Overload Frame is not ready to receive additional messages at this time

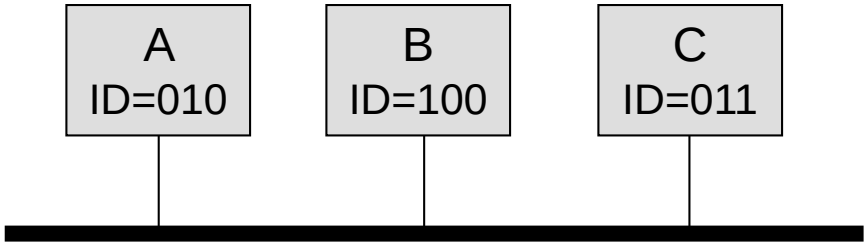
CAN-frame (version 2.0 A, standard format)

SOF	ID	RTR	Control	Data	CRC	CRC DEL	ACK	ACK DEL	EOF	IFS
1 bit	11 bits	1 bit	6 bits	0-8 bytes	15 bits	1 bit	1 bit	1 bit	7 bits	min 3 bits

SOF - *Start of Frame*, start bit (always 0), used for signaling that a frame will be sent (the bus must be free)
ID - *Identifier*, identity for the frame and its priority
RTR - *Remote Transmission Request*
Control - indicates the length of the data field
Data - message data
CRC - *Cyclic Redundancy Check*,
CRC DEL - *CRC delimiter*
ACK - *Acknowledgement*
ACK DEL - *ACK delimiter*
EOF - *End of Frame*
IFS - *Inter Frame Space*, resending wait time

Arbitration mechanism

<u>Example:</u>
Assume a simplified CAN-system with only three ID-bits and three functions, (identifiers A, B, C) mapped onto distinct nodes:



000 – highest priority

111 – lowest priority

which gives:

A-high prio, C-middle, B-low

How does the arbitration look like if the nodes are sending simultaneously?

Node	ID	Bit 0	Bit 1	Bit 2	
A	010	0	1	0	→ Send the rest of the frame
B	100	1	→ abort! (bit 0 ≠ bus value)		
C	011	0	1	1	→ abort! (bit 2 ≠ bus value)
Bus value:		0	1	0	

Acknowledgement and error signaling

Error detection with check sum (CRC)

- If the frame is *received* correctly, the ACK-bit is set to **0** by a receiver

Error signaling

- The node that detects an error puts instantly **000000** on the bus
- Because **0** is the dominant value, all nodes will detect the error rapidly
- Some CAN-systems have **1** as the dominant bit → bit-pattern for error signaling is **111111**

Bit stuffing

- Patterns 000000 and 111111 must be avoided: insert extra bits at sender, reconstruct at receiver

Original frame:	…00101000000101…
Sender puts extra bits:	…00101000001010 1…

CAN and time

CAN is time deterministic

- Possible to calculate how long time it takes to deliver a frame
- CAN acts as a non-preemptable resource, accessed under fixed priority: latency and response time can be predicted/computed

How many bits are sent in a CAN-frame?

SOF	ID	RTR	Control	Data	CRC	CRC DEL	ACK	ACK DEL	EOF	IFS
1 bit	11 bits	1 bit	6 bits	0-8 bytes	15 bits	1 bit	1 bit	1 bit	7 bits	min 3 bits

Sum = $47 + 8n$

(n = nr of data bytes)

Bit stuffing leads to slightly larger messages (ignored here)

Literature

R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien, *Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised*, Real-Time Systems, ISSN 1573-1383 (online), http://www.springerlink.com/content/8n32720737877071/, January 30th, 2007, ISSN 0922-6443 (print), 35(3): 239-272, April 2007.

Exercises

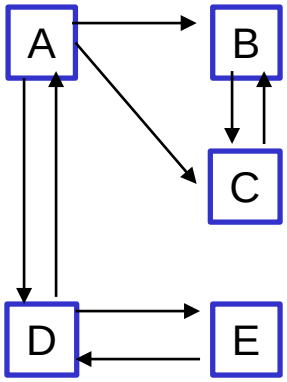
- **N.1** Consider a 1Mbps CAN with 11 bit identifiers. How long does it take to transmit a 4 byte and a 10 byte payload respectively?

- **N.2** Three periodic tasks T1, T2 and T3 share a 250kbps CAN with 11 bit identifiers. The tasks have periods of 1ms, 2ms and 3ms respectively, and transmit 4, 8 and 12 bytes respectively. T1 has the highest priority and T3 the lowest.
  1. assign CAN identifiers to implement these priorities
  2. what is the total transmission time per job of each task?
  3. draw the timing diagram considering a phasing of 0
  4. what is the utilization of this CAN?

Exercises

- **N.3** Consider the diagram below: an arrow from X to Y means that function Y (i.e., the node(s) that execute Y) needs a message from X. Two nodes (ECU/CPU) execute the functions and there are two mappings.

	Mapping 1	Mapping 2
ECU1:	{ A, D }	{ A, B, C }
ECU2:	{ B, C, E }	{ D, E}



All functions send messages with the same period and same length.

Determine, for the two alternatives, how many CAN messages are sent from one ECU to the other.