# Fieldbuses in real-time distributed embedded systems

José Alberto Fonseca, Paulo Pedreiras

DET – IEETA
University of Aveiro
Aveiro-Portugal


**2nd International Workshop on Embedded Systems, Internet Programming and Industrial IT**
**Kiel, Germany, 20-22 August, 2003**

# Industrial communications and fieldbuses – a little of history

- Pre-history of *fieldbuses* started 20 years ago [Thomesse 1999]
- Main motivation at that time (factory automation, in particular process industries): to reduce and simplify cabling in the interconnection of field instruments and the so-called control rooms.
- This led to the name: *fieldbus* coming from *field*, (the industrial plant designation) + *bus* topology used almost always.
- Subsequent history:
  - Commercial interests of companies and countries
  - Promotion of solutions originated at the academia
  - Complex standardization efforts with odd ends.

# Industrial communications and fieldbuses
## – a little of history

- Development of BitBUS by Intel in the eighties.
  - HDLC (High-Level Data Link Control) coming from networks.
  - 8051 based microcontrollers at the fieldbus nodes.
- National standardization efforts
  - Starting almost in parallel in Europe (198x)
  - PROFIBUS in Germany
  - FIP (Factory Instrumentation Protocol) in France
  - P-NET in Denmark
- A milestone: Producer-Distributor-Consumer Model
  - In FIP by J.P. Thomesse.
- 1984: Start of the dissemination of CAN – *Controller Area Network*

# Industrial communications and fieldbuses – a little of history

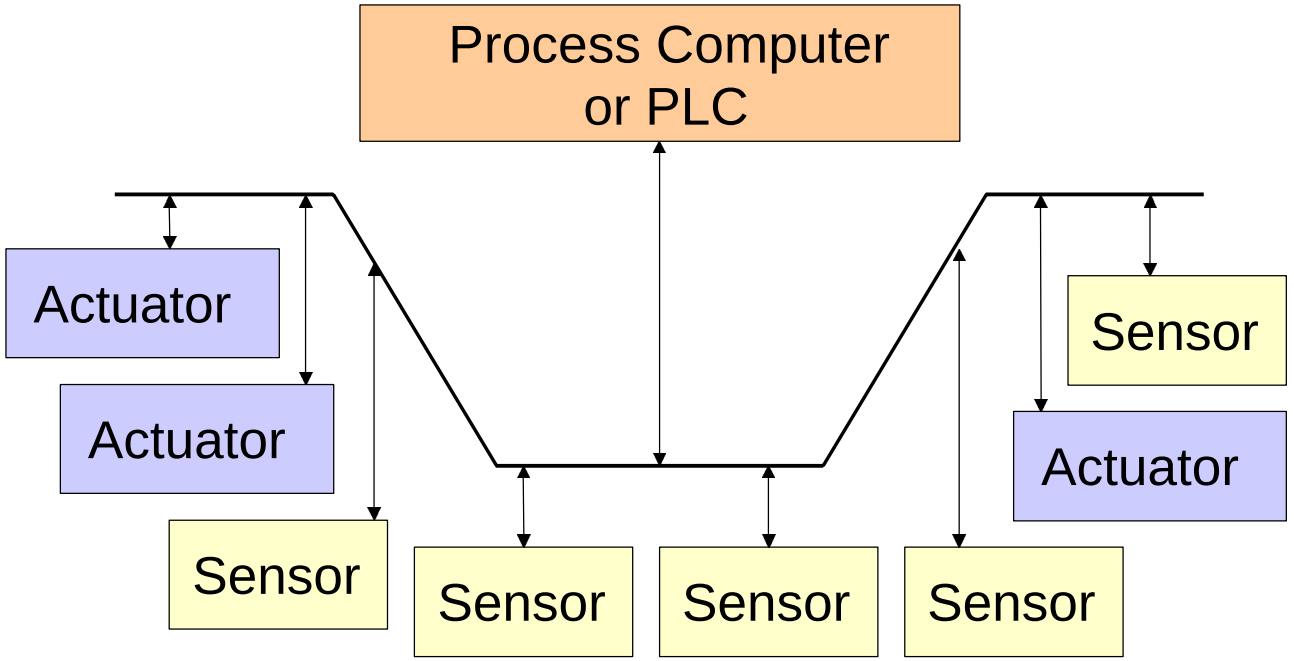
- European Standardization: EN50170
  - Three incompatible profiles (EN50170-1, -2 and –3)
  - PROFIBUS, WorldFIP, P-NET
- *Time-Triggered* paradigm [Kopetz 1994].
- *Fieldbuses* without a bus topology
  - *Interbus* (“active ring”)
  - *LonWorks (power-line, twisted pair, fiber optic, …)*
  - *…*
- IEC 61158 standardization with 8 profiles…
- Properties of the MAC – Medium Access Control of the LAS (Link Active Scheduler) - profile 1 in IEC 61158 (Fieldbus Foundation).

# Industrial communications and fieldbuses – a little of history

- Fieldbuses: currently an important research topic.
- In related conferences:
  - SICICA - *Symposium on Intelligent Components for Control Applications*
  - ETFA - *Emerging Technologies and Factory Automation*
  - RTSS - *Real-Time Systems Symposium*
  - EUROMICRO RTS (*Real-Time Systems*)
- In Journals:
  - *Journal of Real-Time Systems, Control Engineering Practice*
  - *IEEE Transactions on Industrial Electronics, on Communications*
- Object of specific conferences:
  - FeT - *Fieldbus Technology and Applications Conference* (IFAC)
  - WFCS - *Workshop on Factory Communication Systems* (IEEE)

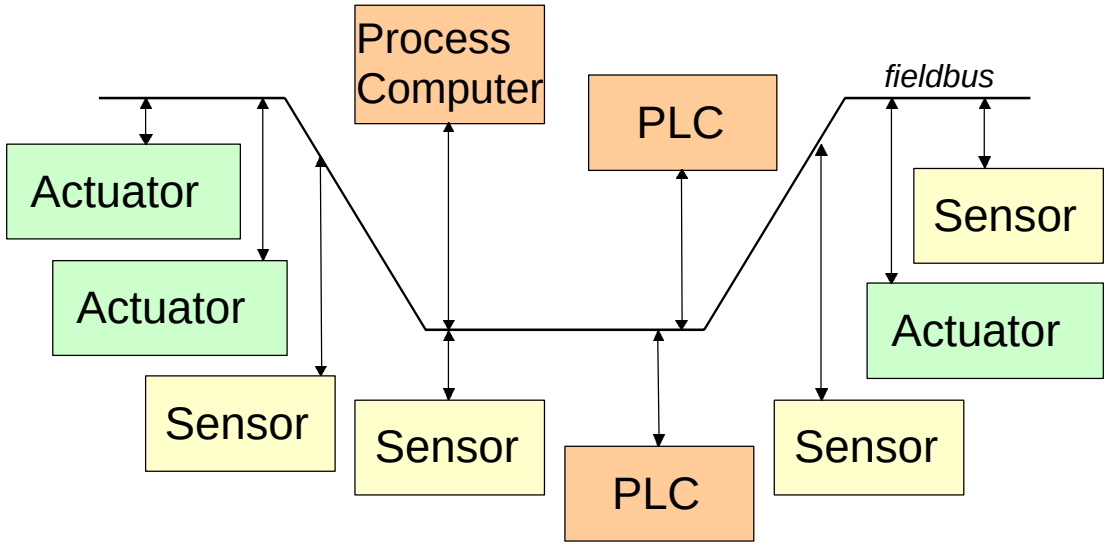
# From *Fieldbuses* to Distributed Embedded Systems

- First *fieldbus*-based systems:
  - Infrastructure to simplify and reduce the number of cables
  - Centralized architecture



# From *Fieldbuses* to Distributed Embedded Systems

- Evolution of *fieldbus*-based architectures:
  - Control and command distributed by different devices
  - Makes coordination more complex
  - But offers new possibilities: task distribution, parallelization, redundancy at lower costs, …

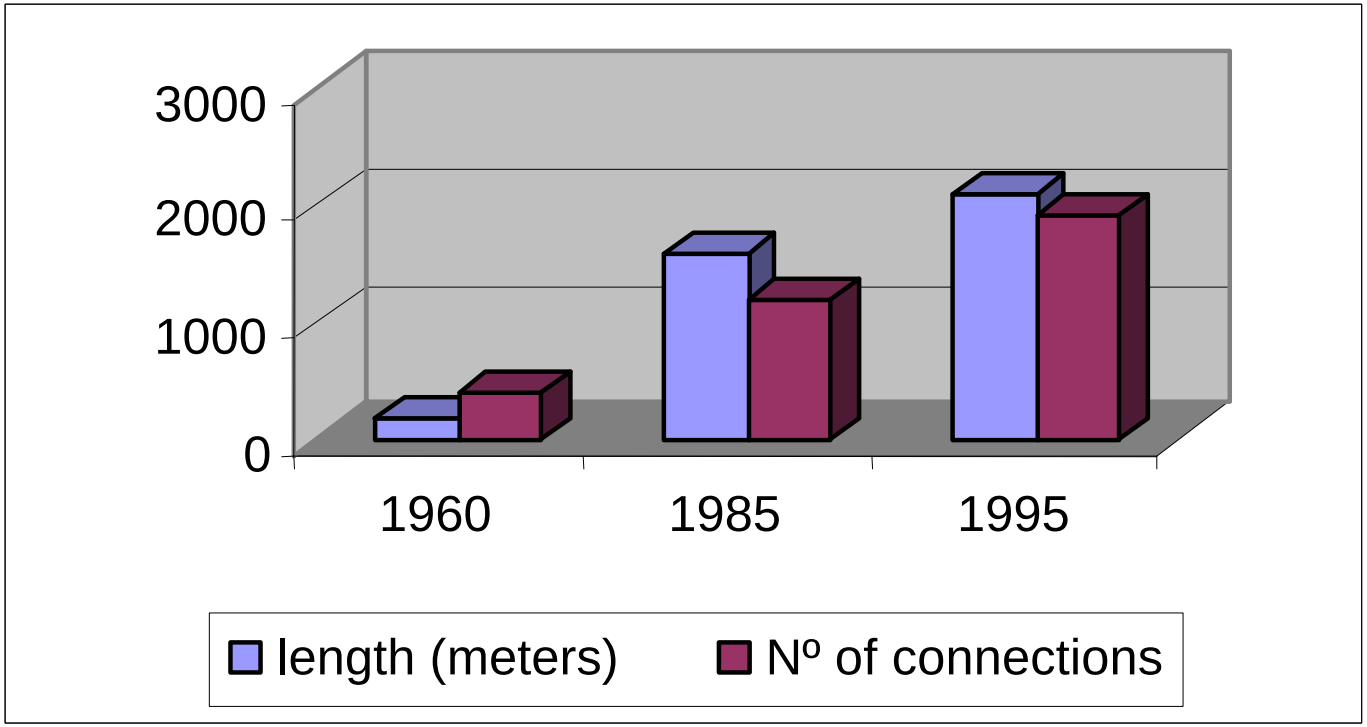


# From *Fieldbuses* to Distributed Embedded Systems

- From the “macroscopic” level to the integration into systems:
  - Control, command, monitoring places moved from a central point to the proximity of the subsystem.
  - Shared data, enabling redundancy, cross-relations, ...
- Paradigmatic example of using DESs: the car (automotive appl.)
  - The motivation to use electronics in cars came from
    - The improvement of passenger comfort.
    - The navigation control and motor control.
    - The inclusion of entertainment systems.
  - Consequences:
    - Complex installation of cabling on board.
    - Significant weight due to cabling.

# From *Fieldbuses* to Distributed Embedded Systems

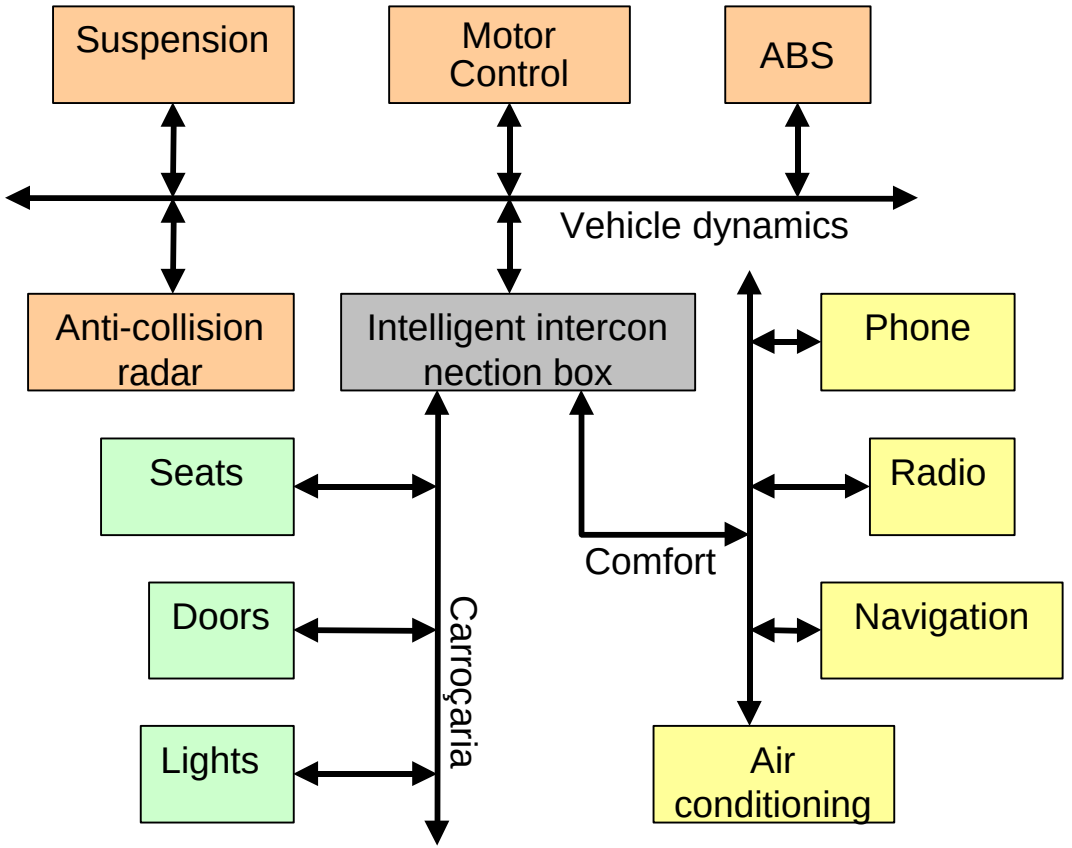
- Evolution of cabling in a car:




Bichet, C. "*Les Standards de Communications dans l'Automobile*" Actes INNOCAP'99, Grenoble, France, april 1999.

# From *Fieldbuses* to Distributed Embedded Systems

- Typical architecture of the communication network in a car:



# From *Fieldbuses* to Distributed Embedded Systems

- Field communication networks in cars:
  - CAN (*Controller Area Network*) bus [Bosch 1991]
  - Protocols built on top of CAN or similar to CAN:
    - TTCAN – *Time Triggered CAN* [Führer et al. 2000]
    - FTT-CAN – *Flexible Time-Triggered CAN* [Almeida et al. 2002]
    - VAN – *Vehicle Area Network*
  - Other important protocols:
    - TTP – *Time Triggered Protocol* [Kopetz 1991]
    - FlexRay [Belshner 2001]

# From *Fieldbuses* to Distributed Embedded Systems

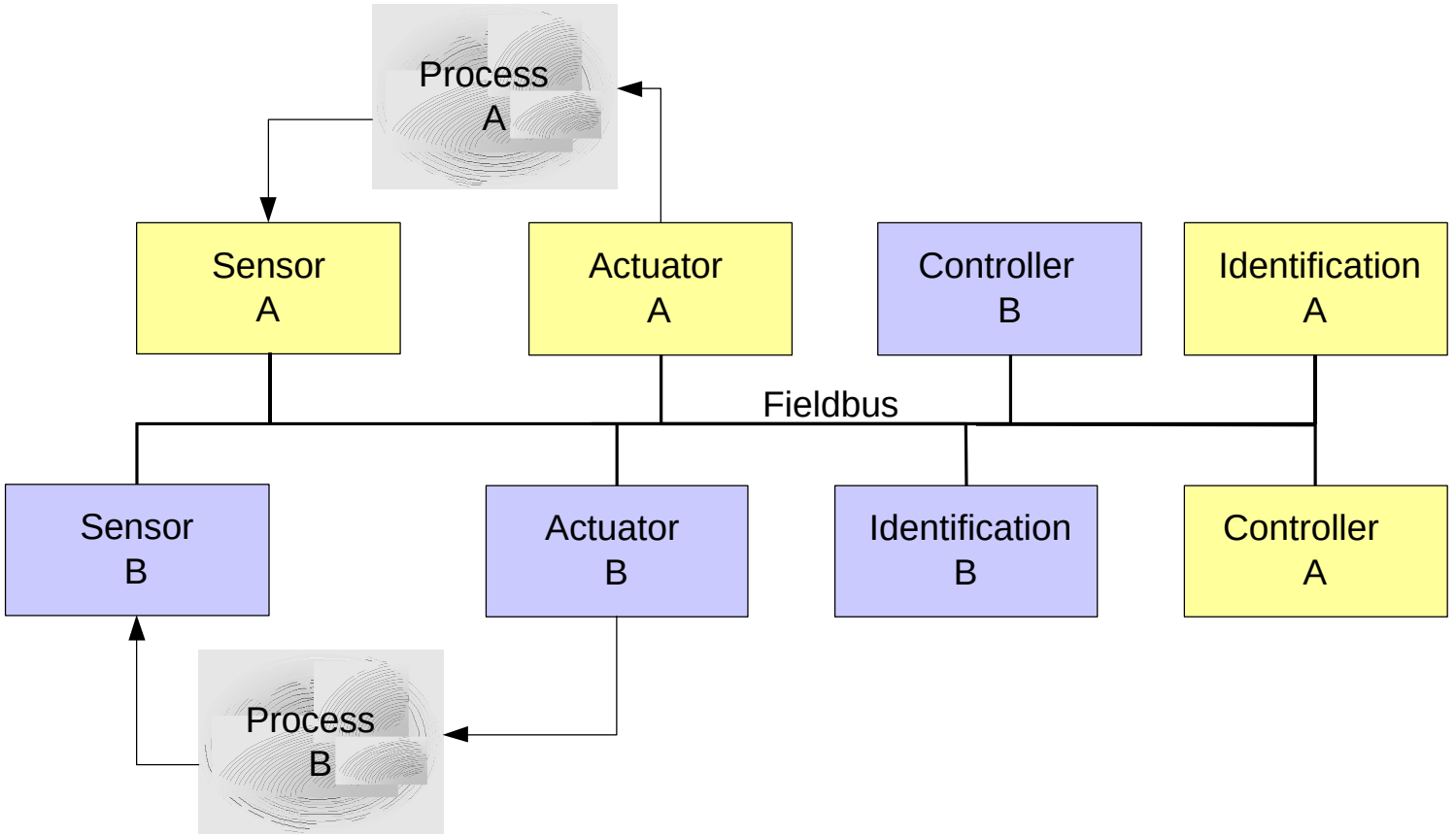
- Why CAN has become so widely used?
  - High number of CAN nodes in automotive
  - Intrinsic properties (some explored years after the initial standard)
  - Low cost of controllers and drivers, enabling small production in other applications beside automotive
  - Integration of CAN interfaces in several popular microcontrollers:
    - 8051
    - PIC (Microchip)
- Interesting side effects: CAN had some contribution to revert the tendency to abandon 8-16 bit microcontrollers.

# From *Fieldbuses* to Distributed Embedded Systems

- The use of fieldbuses led to a change in the embedded systems architecture:
  - From centralized to distributed systems.
  - Small processors close to the subsystems under control.
  - Each processor with bus controller and driver becomes a node.
  - Reduction in the complexity of tasks executed at the nodes.
- Examples:
  - In control systems:
  - Sampling and actuation are displaced to a close position from the sensor and the actuator.
  - Other functions (control, identification, ...) can be executed in different nodes.

# From *Fieldbuses* to Distributed Embedded Systems

- Example: Distributed control system with 2 control loops:



## Real-time and control requirements

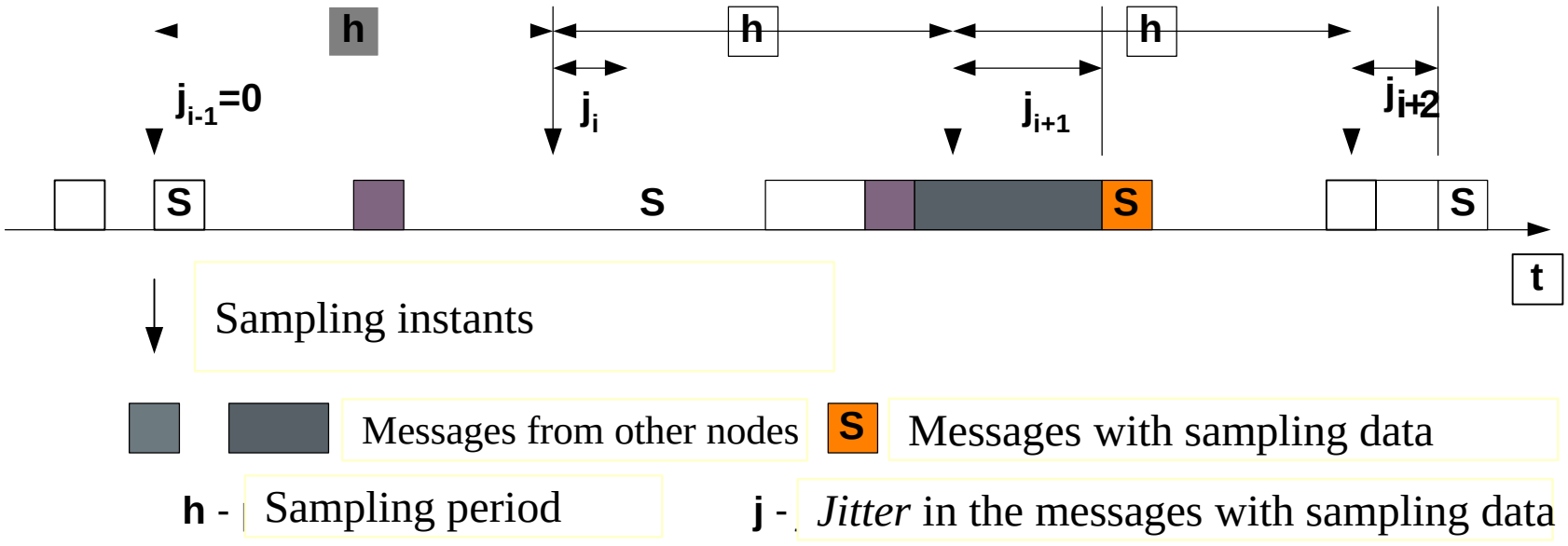
- New feature in DESs - Distributed Embedded Systems:
  - The existence of a shared communication network, connecting different subsystems that can operate independently.
- The data streams produced by the subsystems can be completely independent
  - Example: A and B control loops in the previous figure.
  - Some streams can be periodic (e.g. controls).
  - There may be a need to transmit sporadic data (e.g. detection of alarm situations).

## Real-time and control requirements

- The transmission of independent data flows:
  - Originates conflicts in the use of the bus.
  - The MAC- Medium Access Control must solve them.
  - It is clear that data can not always be transmitted when the node wants.
- In real-time systems some problems can occur:
  - The deadline to transmit the data must be respected.
  - Periodic data streams can show fluctuations in the periodicity (*jitter*).
  - Determining the worst-case situations related to transmission latency is fundamental.

# Real-time and control requirements

- Network induced *Jitter* during the transmission of periodic data:



# Real-time and control requirements

- Determination of the worst-case values:
  - Knowledge of the message set is required.
  - Safety critical messages must have higher priority of transmission
  - The MAC must be able to handle higher priorities:
  - Maximum blocking time due to lower priority messages must be known and bounded.
  - Normally it is not possible to interrupt the transmission of messages – non-preemptive systems (reduced length of data, packing not used, ...)

## Real-time and control requirements

- The MAC plays a fundamental role, determining the ability of the fieldbus to support a real-time distributed system.
- Some examples:
  - The indeterminism shown by the CSMA-CD medium access control makes it unusable for hard real-time.
  - The CAN medium access control, CSMA-CA (*collision avoidance – or other*), shows adequate characteristics.
- Strangely, the theory leading to the determination of the worst-case transmission time of periodic messages sent over CAN was only published in 1994 [Tindell and Burns].


# Real-time and control requirements

- Examples of problems from the control perspective:
  - Vacant sampling [Hong 1995].
  - Read-in / Read-out network induced jitter [Stothert and MacLeod 1998].
- Some of the proposed solutions:
  - Taking the jitter into consideration at the system model level.
  - Minimizing the interference among periodic data streams.
  - Synchronizing the sampling and the actuation, imposing a fixed offset in between.

# Current paradigms concerning distributed embedded systems

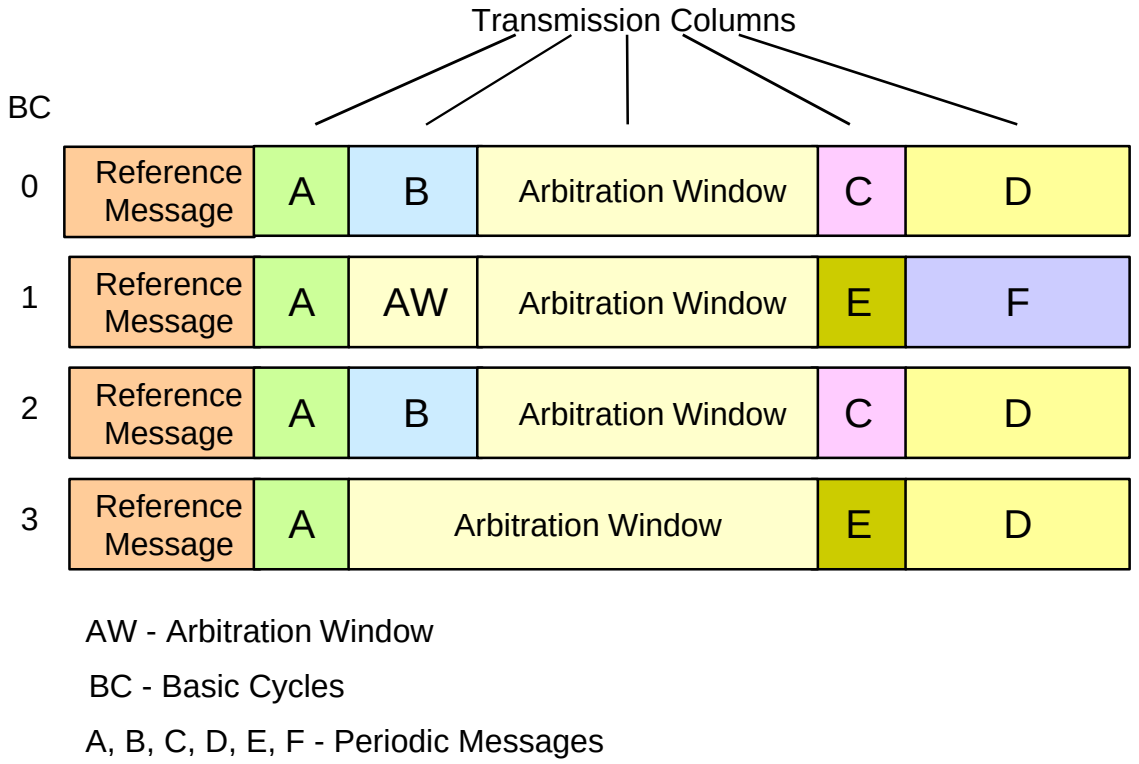
- *Time-triggered* systems paradigm [Kopetz 1991]
  - Assumes a coherent notion of time among the distributed system elements.
  - Imposes the transmission of data streams in dedicated, pre-allocated time slots.
  - Collisions are avoided and response time becomes deterministic.
  - Some interesting properties such as scalability and composability, this one important in what concerns timeliness guarantees.
- NOTE: the discussion about the vantages and advantages of time-triggered or event-triggered transmission paradigms is not finished yet.

# Examples of similar time-triggered protocols

- TTCAN (*Time-Triggered* CAN)
  - Standardization effort initiated by ISO in 1999.
  - Target applications: automotive.
  - It consists in a session layer enabling time-triggered communication on CAN.
  - The communication is scheduled in a time-slots schedule matrix pre-defined prior to the system operation.
  - Periodic data streams use permanently specific fixed time-slots.
  - Event-triggered communication is also possible within the so-called arbitration windows (time-slots using normal CAN MAC).

# TTCAN – Time-triggered CAN

- TTCAN schedule matrix




AW - Arbitration Window

BC - Basic Cycles

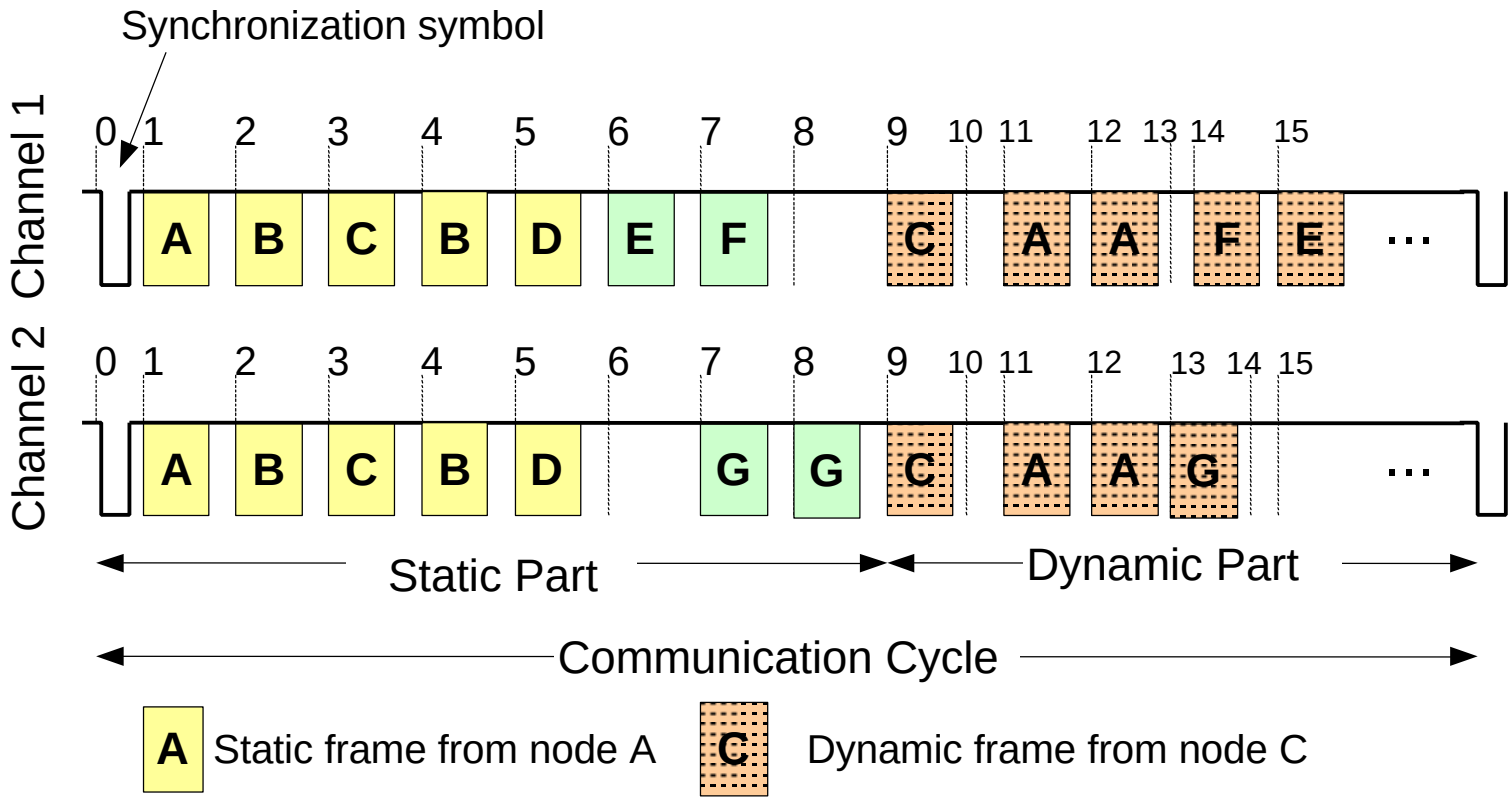
A, B, C, D, E, F - Periodic Messages

# Examples of time-triggered protocols

- FlexRay:
  - Specification presented first in September 2001 (v.1.9.7)
  - Initial promoters: BMW, DaimlerChrysler, Bosch
  - Transmission rate targeted to 5Mbit/s net
  - Uses electrical and optical transmission media
  - The bus time is divided in slots called cycles in which part of the traffic is defined statically and part can be transmitted dynamically.
  - It uses redundancy at the physical transmission media, some of the nodes transmitting simultaneously in both channels.

# Examples of time-triggered protocols: FlexRay

- Definition of a communication cycle in FlexRay:

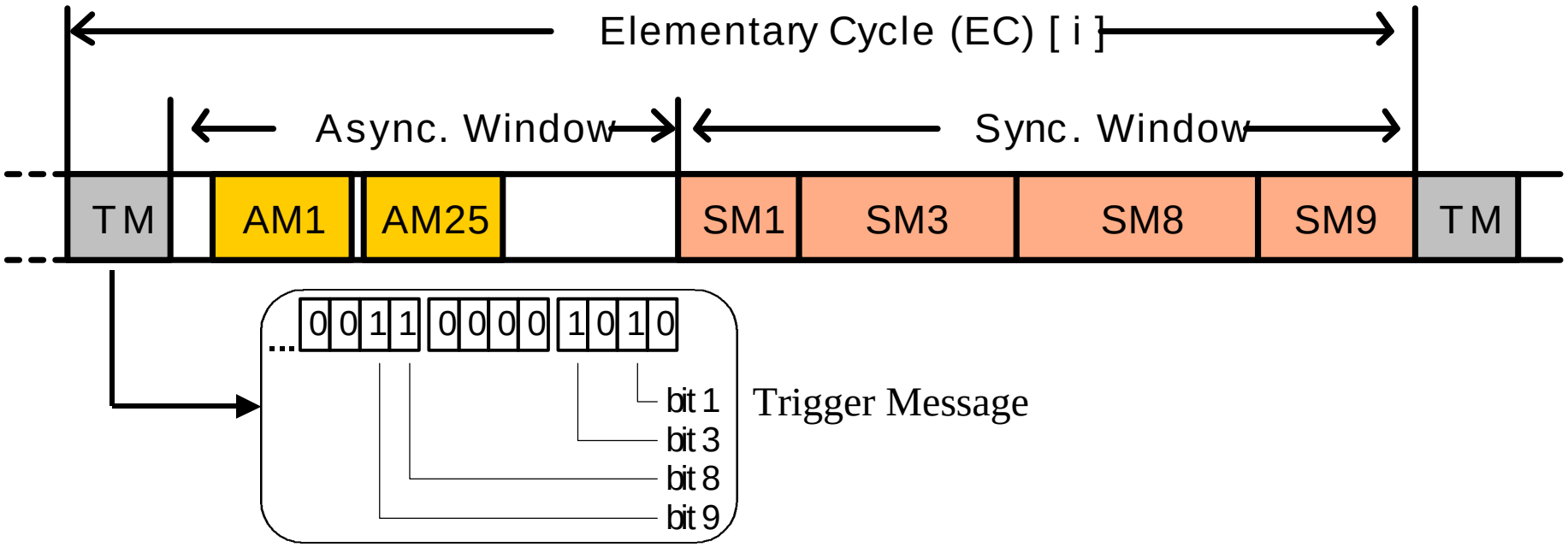


# Example of time / event triggered protocols

- FTT CAN: *Flexible Time Triggered Protocol on CAN*
- Developed in Aveiro in 1999
- Centralized scheduling of periodic data streams.
- Uses an admission control mechanism to allow changes in the scheduling during the system operation, keeping the timeliness guarantees.
- Takes profit of the native MAC of CAN.
- Supports *time-triggered* and *event-triggered* communication with temporal isolation.

# FTT CAN: *Flexible Time Triggered Protocol on CAN*

- FTT CAN: *Flexible Time Triggered Protocol on CAN*



# Conclusions

- Industrial communications, in particular fieldbuses, have evolved from a “macroscopic” level in factory automation to a “microscopic” level as a key component of distributed embedded systems.
- The use of fieldbuses in real-time systems (often Hard-RT) poses some challenges:
  - Response times must be bounded and correctly determined
  - Perturbations in the transmission timeliness of control parameters (samples, set-points) must be taken into consideration.

# Conclusions

- Automotive industry is currently the main driving force in the development of distributed embedded systems. The standard used in automotive will have, after some time, a significant impact in other applications (e.g. automation).
- The current trend is to combine time-triggered and event-triggered communications to benefit from the advantages of both.
- There is still some uncertainty about which protocol will win the automotive competition:
  - TTP ?
  - A CAN based protocol: TTCAN, FTT-CAN, ... ?
  - FlexRay ?