# Do *SAFETY-CRITICAL SYSTEMS* really need to be *STATIC* ?

**Luís Almeida**

DET – IEETA
Universidade de Aveiro
Aveiro-Portugal

DATE 2005, Munich, Germany
11 March 2005

# Background

Nowadays, current complex embedded systems are **distributed** (DES)

- ✓ Cars, planes, industrial machinery ...

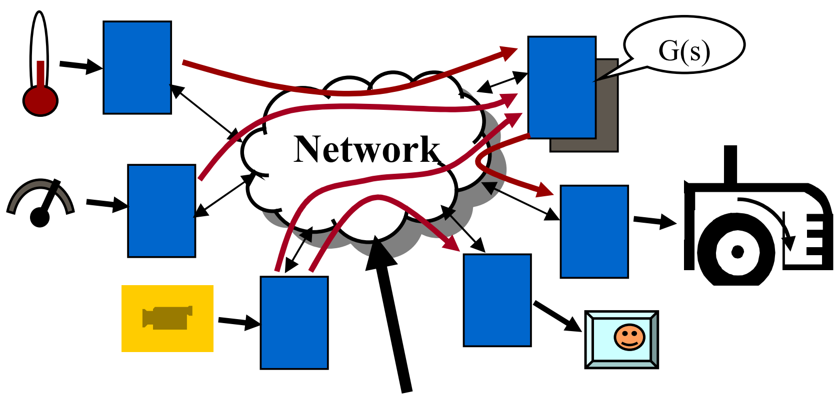
There is also a trend to **increase integration** among subsystems as a way to

- ✓ **Improve efficiency** in using systems resources
- ✓ **Reduce** number of active **components** and **costs**
- ✓ **Manage complexity**

# Background

Higher integration and distribution lead to a **stronger impact of the network** on the global system properties:

- ✓ Composability, timeliness, flexibility, dependability...




**We will focus on the network services**

# Current approach

**Safety** concerns have typically led to **static** approaches in the design of DES

- ✓ Static implies we **always know** what we **should be observing** at each instant
  (conflict **flexibility** versus **safety)**
- ✓ **Fault-tolerance** mechanisms become **simpler**
- ✓ Proliferation of **static Time-Triggered** architectures using **TDMA** with pre-allocated slots
  **(TTP, TT-CAN, FlexRay, SAFEbus, SwiftNet)**

# However

Static approaches:

- ✓ Tend to be **inefficient** in the use of system resources → potential for higher costs
- ✓ Do not easily accomodate **changes** in the **operational environment** or **system configuration**

# Moreover

There is a growing interest in using DES in **dynamic operational scenarios**:

- ✓ Systems with variable number of users, either humans or not (traffic control, radar...)
- ✓ Systems that operate in changing physical environments (robots, cars...)
- ✓ Systems that can self-reconfigure dynamically to cope with hazardous events or evolving functionality (cars, planes, ...)

***QoS adaptation, graceful degradation, survivability***

# Network requirement

**Dynamic (flexible) management of bandwidth** while guaranteeing both real-time and safety constraints.

- ✓ Act upon periodic communication, e.g. related to control information (potentially bandwidth consuming)
- ✓ Adapt transmission rates according to effective needs
- ✓ Explore subsystems that operate ocasionally
- ✓ Explore variable sampling/tx rates according to the current system control stability state

# Problem

How to implement such level of **flexibility** without jeopardizing **timeliness** and **safety**?

Hints

- ✓ Combining **flexibility** with **timeliness** requires the use of adequate **communication paradigms and protocols**
- ✓ Combining **flexibility** with **safety** requires **constraining flexibility** and guaranteeing sufficient resources

# Flexibility and timeliness

The communication protocol must exhibit/support:

- ✓ Bounded communication delays
- ✓ On-line changes to the communication requirements → dynamic traffic scheduling
- ✓ On-line admission control (based on appropriate schedulability analysis)

**Dynamic planning-based scheduling paradigm**

# Flexibility and safety

A form of constraining flexibility must be supported:

- ✓ Possible solution – **Mode change protocols**
  - ✓ set of predefined modes
  - ✓ on-line mode switching
  - ✓ requires a priori definition of all possible modes

10 subsystems with 2 states each → $2^{10}$ possible modes !

Each being independently verified

# Flexibility and safety

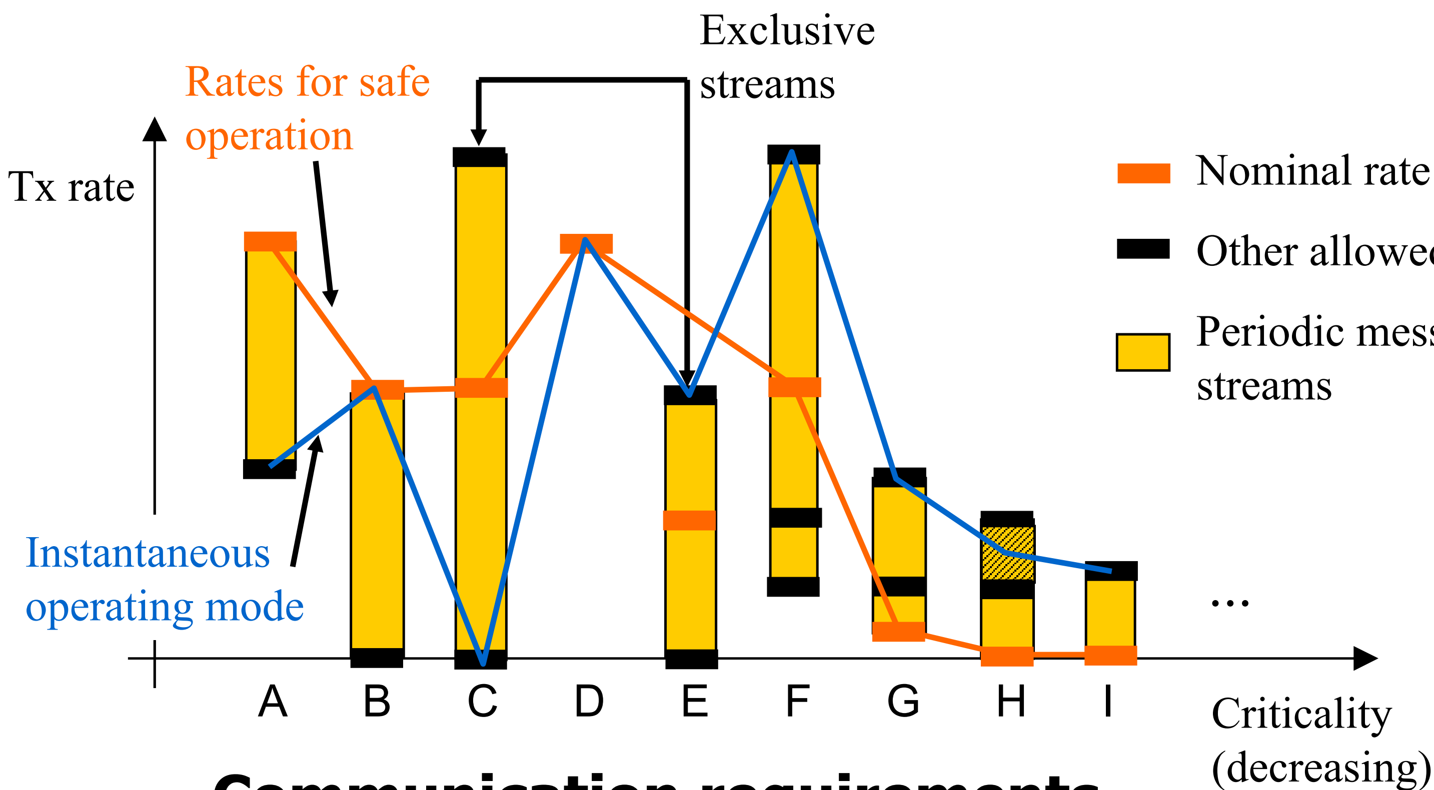
Alternatively, flexibility can also be constrained by **extending the characterization** of message streams with:

- ✓ **safety constraints**
  Nominal rate, level of criticality
- ✓ **change attributes**
  Permitted changes

➞ **Resources are reserved** according to safety constraints (one mode to verify off-line)

Online, subsystems can **use more or less resources** if they are **available** and that **change is permitted**

# Constraining Flexibility



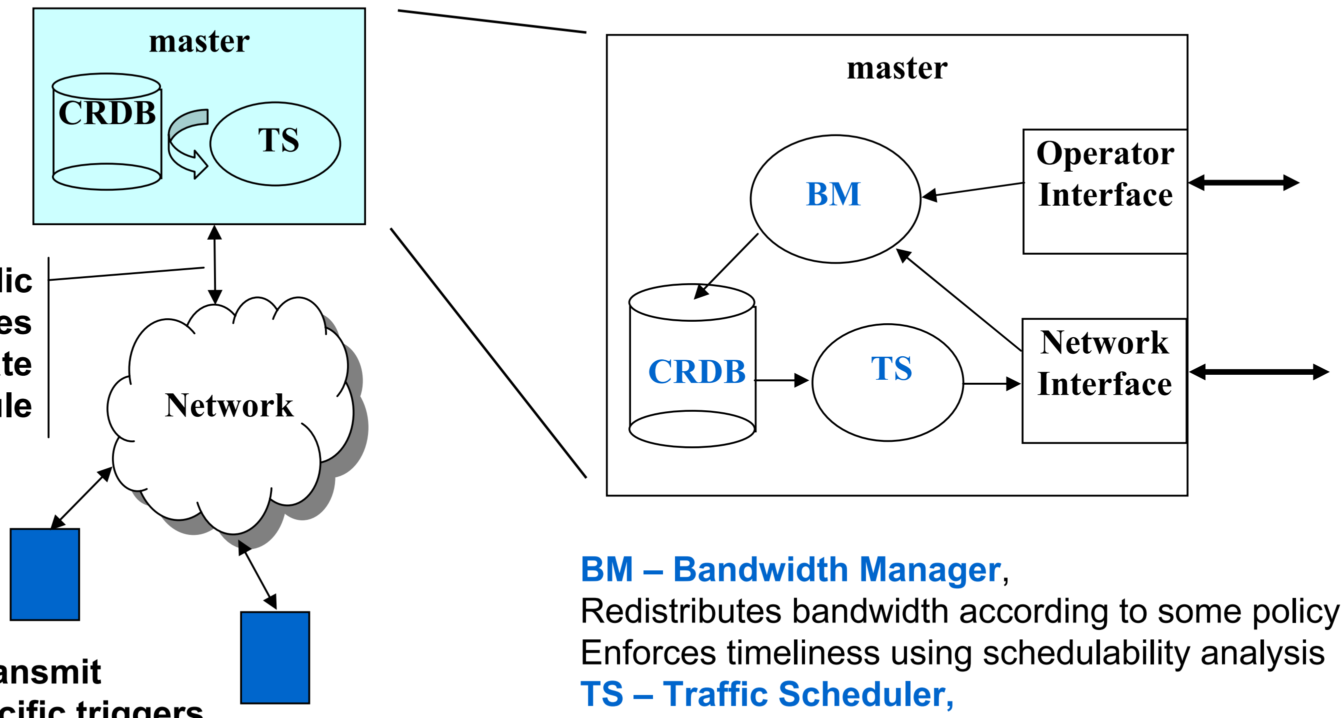

**Communication requirements**

# Architectural requirements

- Maintain a Communication Requirements Database (**CRDB**)
- Support for:
  - on-line changes to either message set as well as scheduling policy with **low latency**
  - on-line admission control and bandwidth management with **low latency**
  - **Replication**

# Possible architecture

Master-slave paradigm, for flexibility control




**Transmits periodic trigger messages with adequate schedule**

**System nodes transmit according to specific triggers**

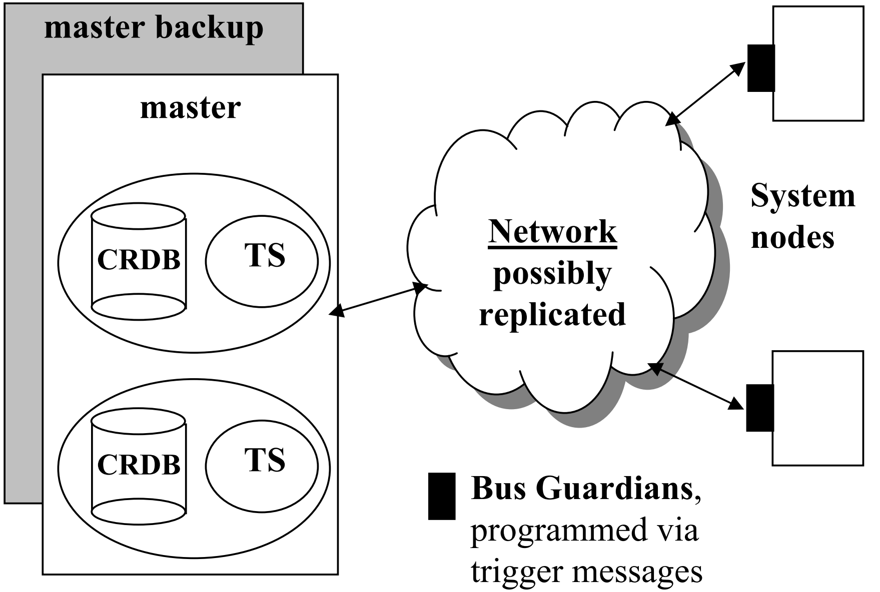
**BM – Bandwidth Manager**,
Redistributes bandwidth according to some policy
Enforces timeliness using schedulability analysis
**TS – Traffic Scheduler,**
Constantly scans CRDB, building traffic schedules

# Possible architecture

## Fault-tolerance features

Coherency between databases:
- consistency in change requests
- CRDB / scheduler_state transfer
- verification of trigger schedules

- ✓ Detection of omissions
- ✓ Master/network replication
- ✓ Fail-silent nodes
  - ✓ System nodes: time domain (BGs)
  - ✓ Masters: time and value domains (internal replication)



# Implementation

This architecture is the basis of the **FTT** (Flexible Time-Triggered) architecture

Two protocols have already been developed according to this architecture

- ✓ **FTT-CAN** and **FTT-Ethernet**
  - ✓ **Efficient master-slave implementation**
  - ✓ **Efficient combination of sync(TT)/async(ET) traffic**

# Conclusion

Concerning DES we have observed:

- Growing interest in dynamic operational scenarios (QoS adaptation, graceful degradation, survivability)
- This requires flexible (dynamic) bandwidth management (particularly wrt the periodic traffic)
  - **Increased bandwidth efficiency**
    → more functionality or better service with same bandwith

We have shown a possible architecture that

- Supports such flexible management of the periodic traffic with
  - **Guaranteed timeliness**
  - **High safety level**