Networks for Reconfigurable Embedded Systems

NeRES 2007

An **ARTIST2** workshop

http://www.artist-embedded.org/artist/-NERES-2007-.html

Luis Almeida, Paulo Pedreiras
Electronic Systems Lab-IEETA / DET
Universidade de Aveiro
Aveiro, Portugal

Motivation

Dynamic reconfiguration
(or reconfigurability)

within **distributed embedded systems**

What ?

Why ?

How ?

Examples ?

Can we answer this by the end of the day?

Dynamic Reconfiguration
What?

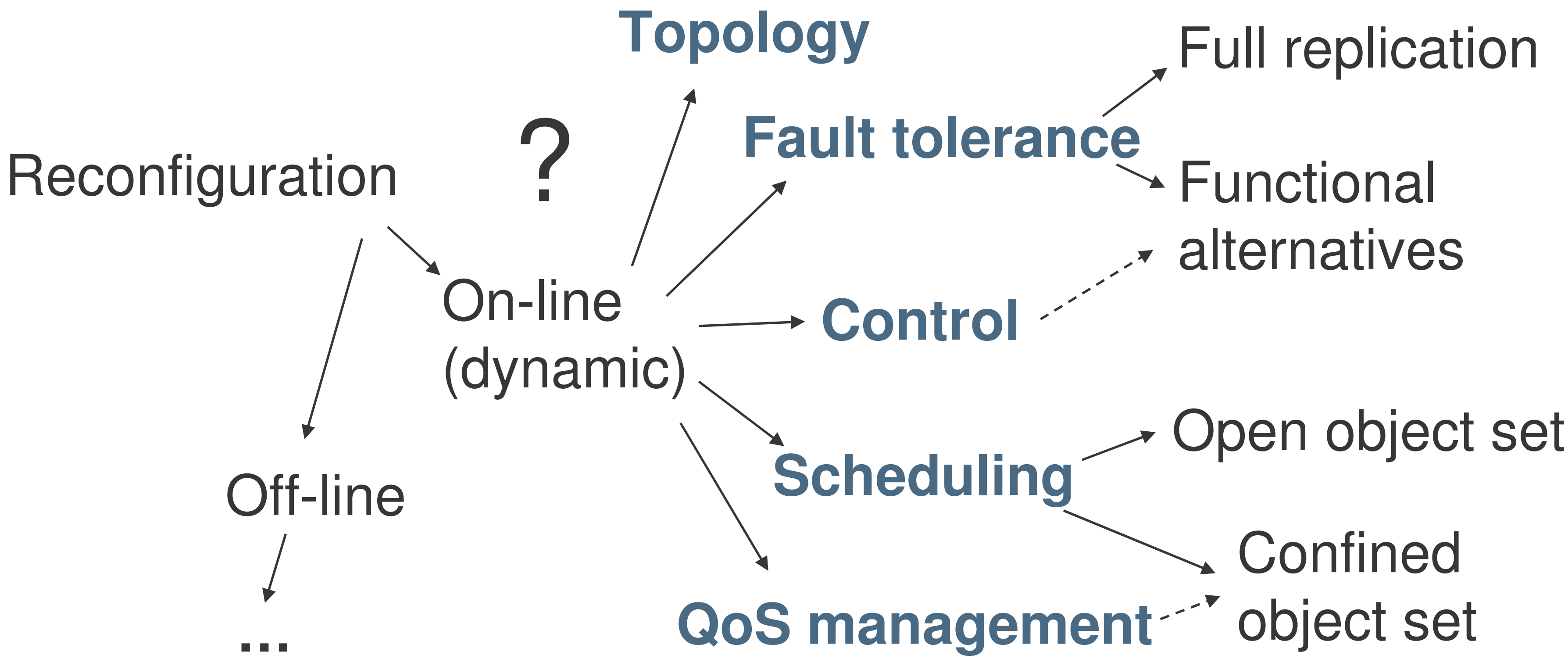
- **Reconfigurability** denotes the capability of a system that can **dynamically change its behavior**, usually in response to dynamic changes in its environment.

- In the context of **wireless communication** reconfigurability tackles the changeable behavior of wireless networks and associated equipment…
- In the context of **Control reconfiguration**, a field of fault-tolerant control within control engineering, reconfigurability is a property of faulty systems meaning that the original control goals specified for the fault-free system can be reached after suitable control reconfiguration

in Wikipedia

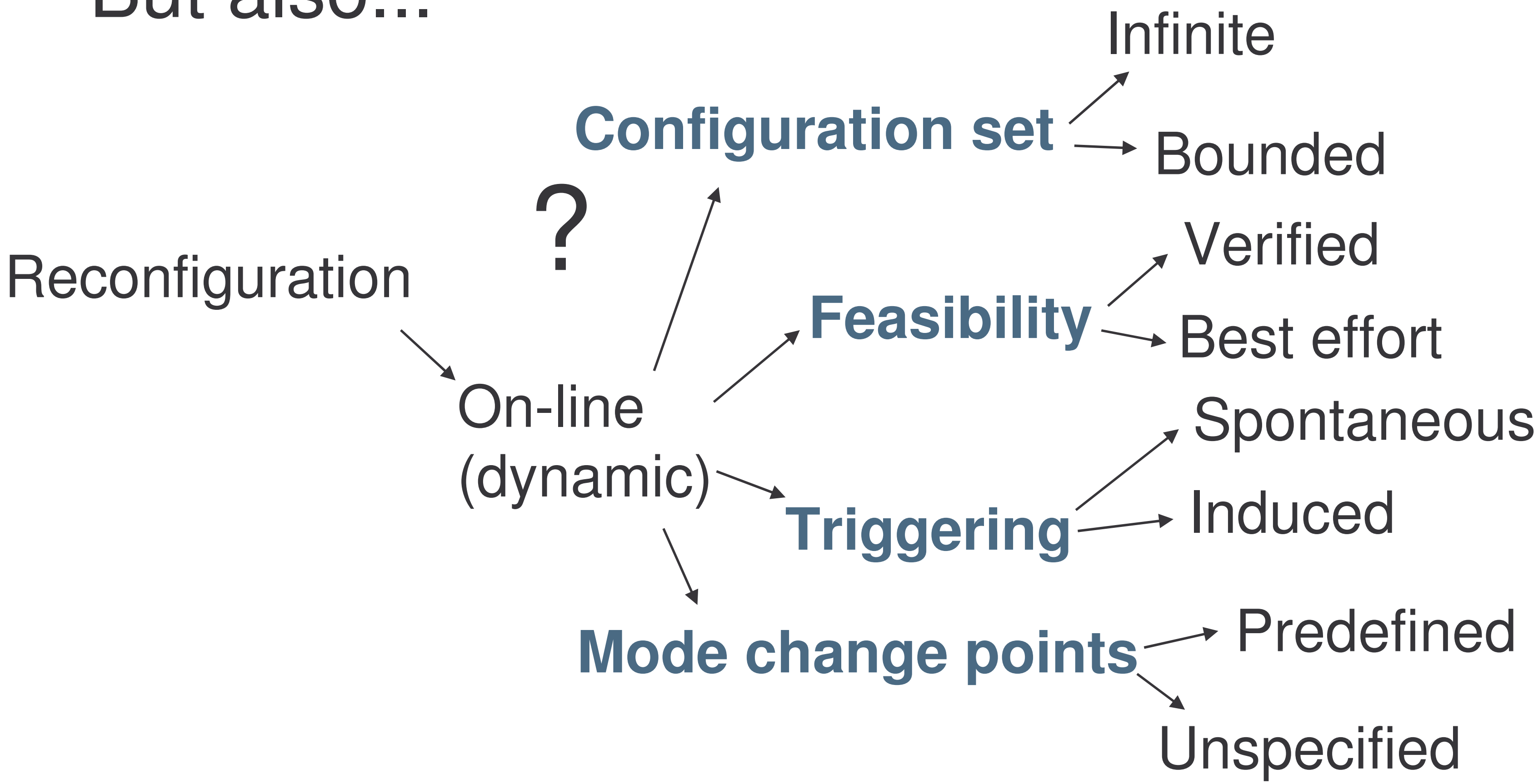
Dynamic Reconfiguration *What?*

Broad concept! Is there a taxonomy?



Dynamic Reconfiguration
What?

But also...



Dynamic Reconfiguration
Why? How? Examples?

In the workshop

12 presentations covering DR in

- ✓ ***Middleware*** *for distributed real-time systems*
- ✓ *Communication for* ***industrial automation***
- ✓ ***Mobile*** *real-time* ***wireless*** *ad-hoc networks*
- ✓ ***Dependable systems***
- ✓ ***Reconfigurable control***
- ✓ ***Industrial case studies***

Workshop goals

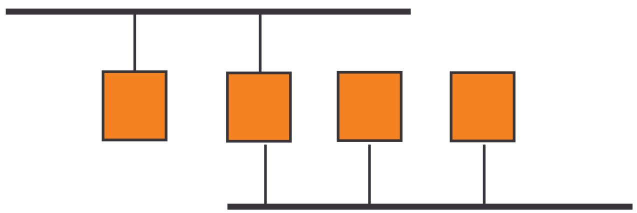
- ✓ discuss the **motivations, interest and challenges** of reconfigurability in distributed real-time embedded systems;
- ✓ deduce the **network requirements** to support **flexible reconfigurability** under real-time and safe operation;
- ✓ discuss the **adequacy of existing protocols and middlewares**;
- ✓ discuss how to provide **real-time communication in highly flexible networks** and identify the potential of current protocols;
- ✓ deduce further **network requirements** to support real-time communication in **highly flexible networks**.

Dynamic Reconfiguration
A few considerations

The challenges of
real-time distributed reconfiguration

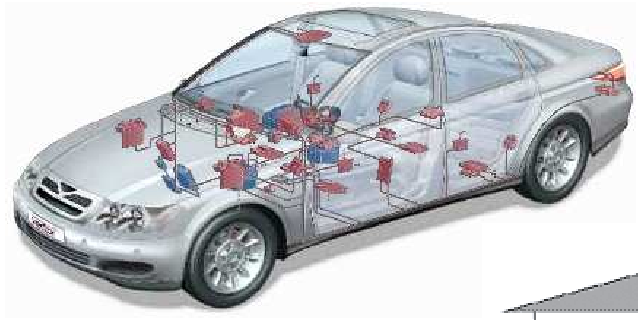
Luis Almeida
lda@det.ua.pt

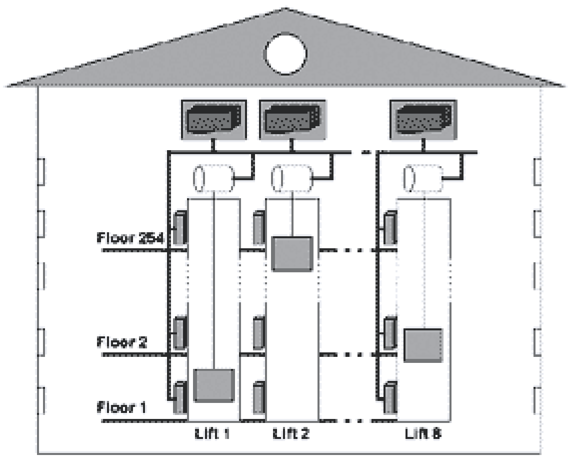
Background



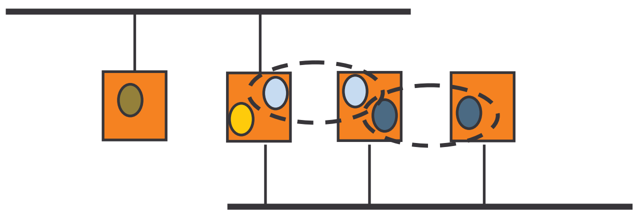
Nowadays, current complex embedded systems are **distributed** (DES)

- ✓ Cars, planes, industrial machinery ...



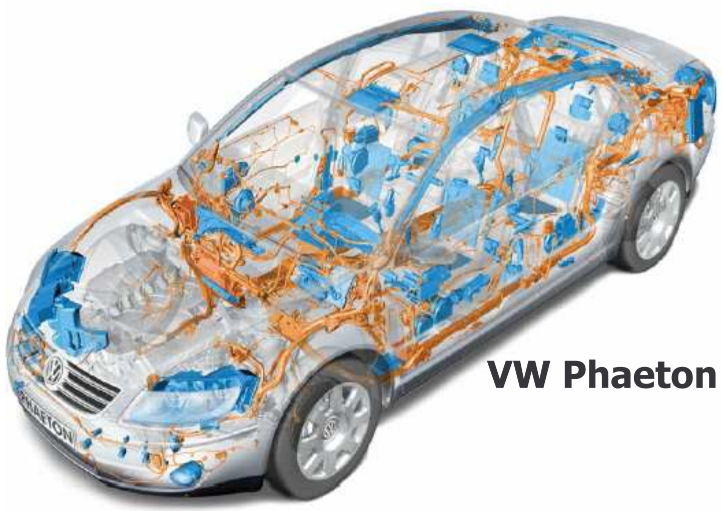


Background



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**



VW Phaeton:

→ **11.136 electrical** parts in total

→ **61 ECUs** in total

→ external diagnosis for 31 ECUs via serial communication

→ **optical bus** for high bandwidth Infotainment-data

→ sub-networks based on proprietary serial bus

→ **35 ECUs** connected by **3 CAN-busses** sharing:

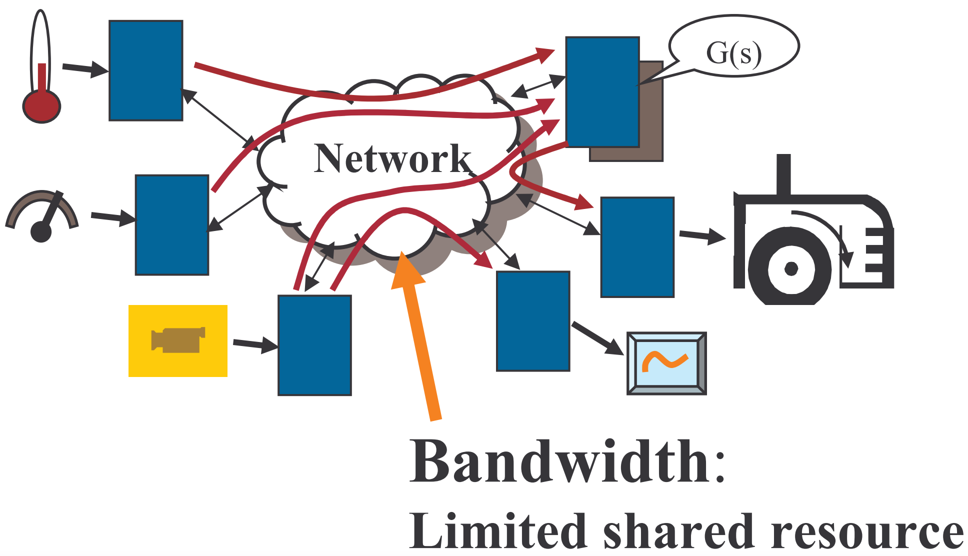
→ appr. **2500 signals** in **250 CAN messages**

(Loehold, WFCS2004)

Background

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

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



Current approach

Safety concerns have typically led to **static** approaches in the design of complex 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 or variable load (traffic control, radar, telecom…)
- ✓ 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, trains, production cells…)

QoS adaptation, graceful degradation, survivability

Alternatively

Common protocols that do **not constrain the load** generated by each node could be used (Ethernet, CAN, …)

✓ **High level of flexibility**
(any node can change its submitted load at any time)

But if any change can happen at run time what **guarantees** can we get with respect to **timeliness** and **safety ?**

What we want

To be able to **connect any component** to the system, **on-line**, being sure that:

- ✓ **Nothing bad** will happen
- ✓ The system will **do its best to integrate** the added component:
  - ✓ It can **accept** the new component **without any adjustment** on the system
  - ✓ It can **accept** it upon system **adjustment**
  - ✓ It can **reject** the new component

What we want

Allow the system to **adjust on-line** according to **effective instantaneous needs**:

- ✓ **Free and reuse the resources** of subsystems that **operate occasionally/fail** when off
- ✓ **Adapt the resources** used by each subsystem on-line to:
  - ✓ **Minimize the resources** used (e.g. to minimize BW usage, energy, …)
  - ✓ **Maximize the service** delivered with a fixed level of resource usage

What we want

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

- ✓ Explore subsystems that **operate ocasionally**
- ✓ Act upon **periodic communication**, e.g. adapting transmission rates according to effective needs
  - ✓ Explore **variable sampling/tx rates** according to the current system control stability state
  - ✓ Explore **variable number of users/services** and provide the best QoS to each one at every instant considering system resources

Problem

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

Hints

- ✓ Basically, we need to **constrain flexibility**
- ✓ Concerning **timeliness** we need adequate **communication paradigms and protocols** (particularly with admission control)
- ✓ Concerning **safety** we must assure that the **resources** needed for **safe operation** are always available

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)
- ✓ (Traffic policing)

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

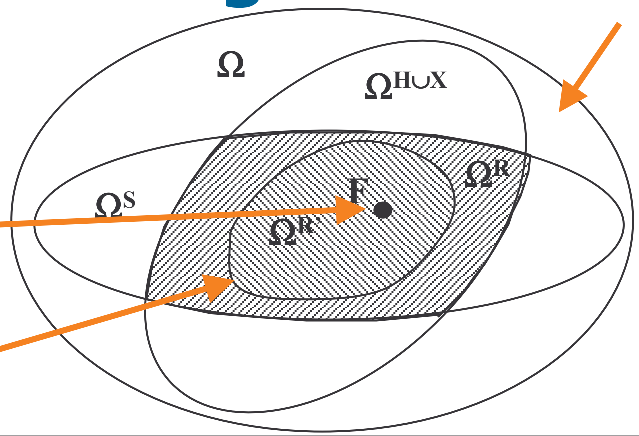
10 subsystems with 2 states each → 2^{10} possible modes !

Each being independently verified

Flexibility and safety

Alternatively, flexibility can also be constrained by defining a **boundary** for the **configurations space** considering:

- ✓ **safety constraints**
  Nominal rates
- ✓ **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**

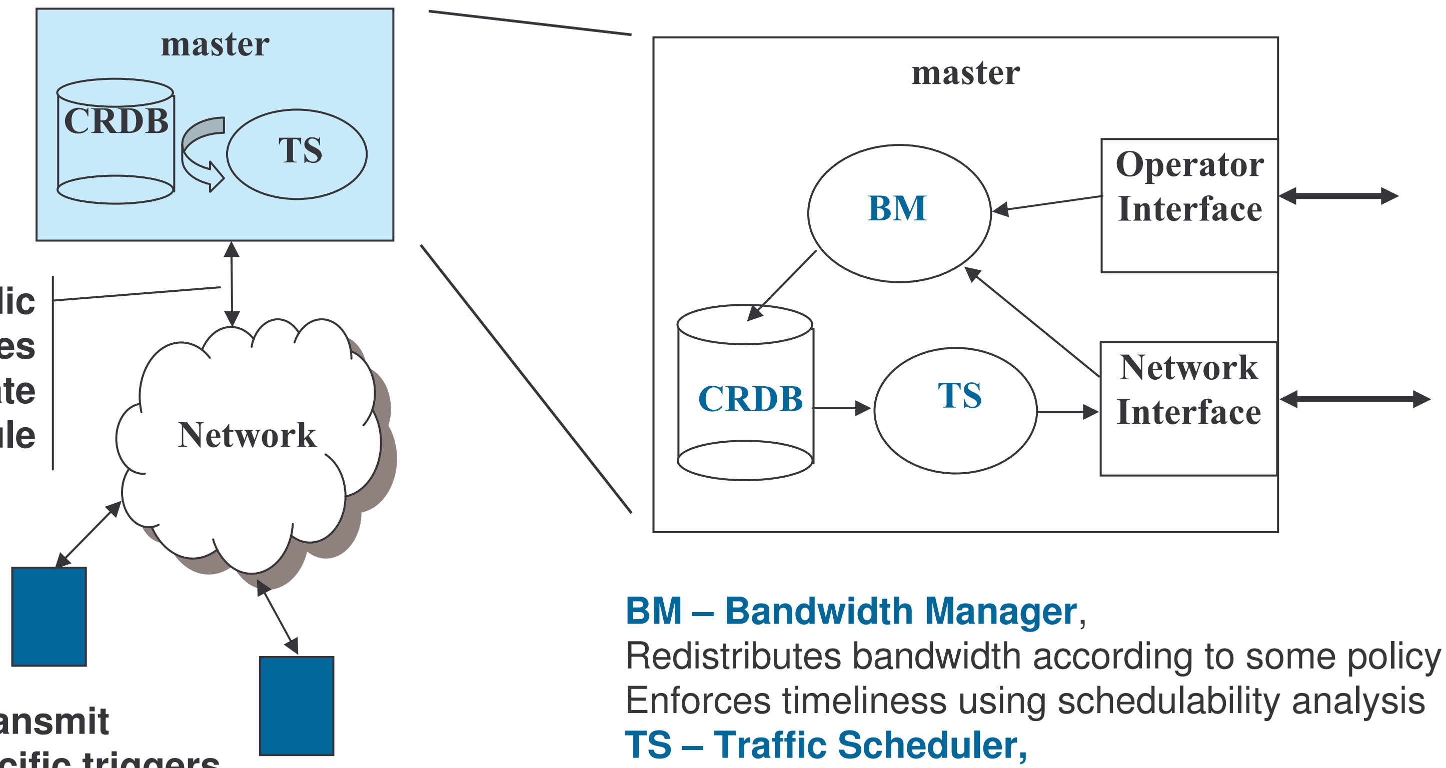
Architectural requirements

- ✓ 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**

(low latency = few ms)

Possible architecture

Master-slave paradigm, for flexibility control



Transmits periodic trigger messages with adequate schedule

System nodes transmit according to specific triggers

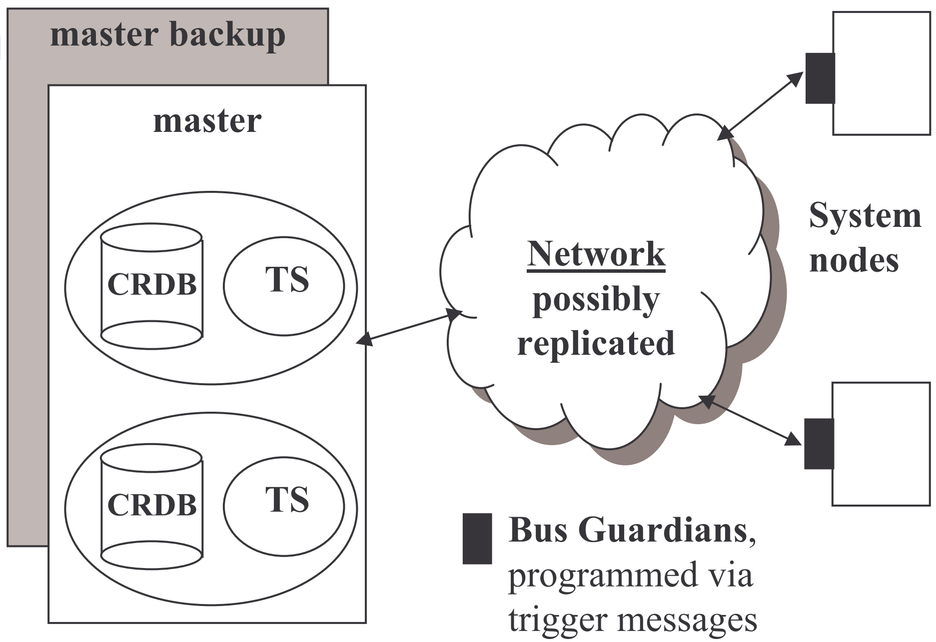
BM – Bandwidth Manager,
Redistributes bandwidth according to some policy
Enforces timeliness using schedulability analysis
TS – Traffic Scheduler,
Constantly scans CRDB, building traffic schedules

Possible architecture

Fault-tolerance features

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

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



Our implementation

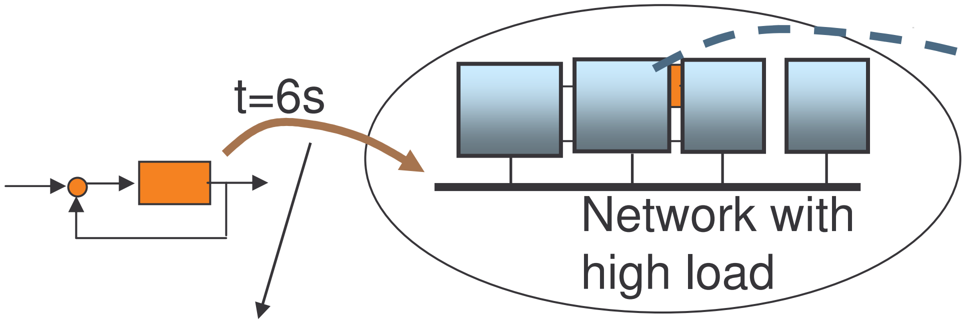
This architecture is the basis of the
FTT (Flexible Time-Triggered) architecture

Three protocols have already been developed according to this architecture

- ✓ **FTT-CAN**, **FTT-Ethernet** and **FTT-SE**
  - ✓ **Efficient master-slave implementation**
  - ✓ **Efficient combination of sync(TT)/async(ET) traffic**
  - ✓ **Guaranteed on-line changes to the sync traffic**
  - ✓ **Support for dynamic QoS management**
  - ✓ **Support for Holistic TT system design (network-centric)**

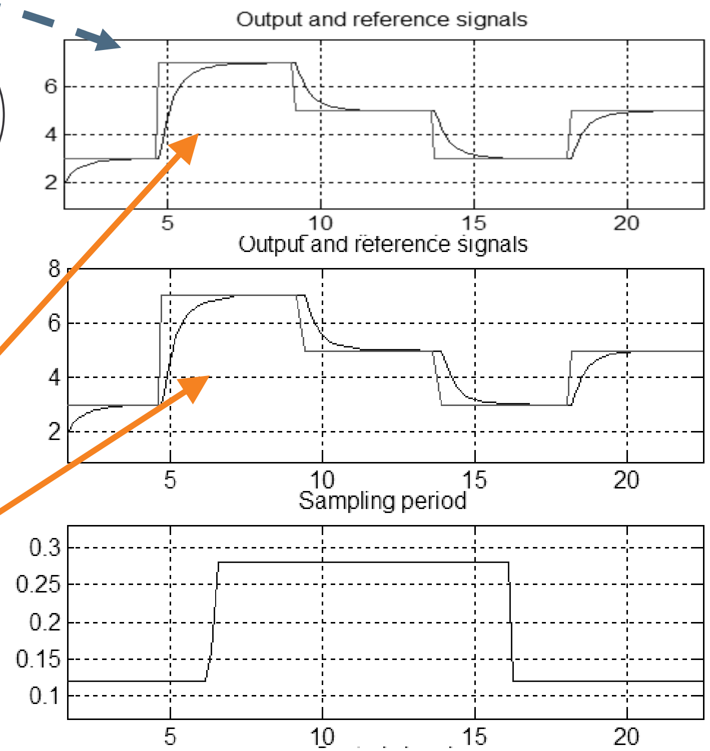
Our implementation

Example in control applications (*truetime* simulation)



Rejected with fixed h
Accepted with variable h

h	ISE	Degrad.
0.12s	48.2	0
0.12s 0.28s	50,3	4,4%



Conclusion

We have seen that:

- ✓**Dynamic Reconfiguration** (DR) at the network level does help in getting
  - ✓**Increased bandwidth efficiency** → more functionality or better service with same bandwith
- ✓With an adequate **architecture** it is possible to support a flexible management (DR) of the periodic traffic with
  - ✓**Guaranteed timeliness**
  - ✓**High safety level**