From a Federated to an Integrated Architecture for Dependable Embedded Systems

H. Kopetz

TU Wien

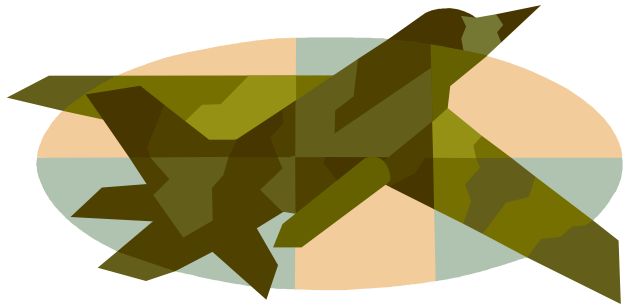
September 2004

Outline

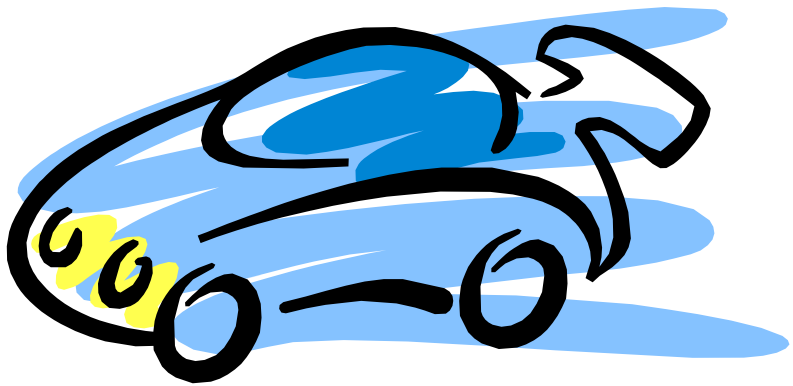
- Introduction
- Federated versus Integrated Architecture
- The Challenge
- The Time-Triggered Architecture
- Fault Tolerance in the TTA
- Conclusion

Examples of Safety Critical Systems--No Backup

Fly-by-wire Airplane: There is no mechanical or hydraulic connection between the pilot controls and the control surfaces.



Drive-by-wire Car: There is no mechanical or hydraulic connection between the steering wheel and the wheels.



The 10^{-9} Challenge

- ***Critical system services*** **must be more reliable than any one of the components**: *e.g., System Dependability 1 FIT--Component dependability 1000 FIT (1 FIT: 1 failure in 10^9 hours)*
- Architecture **must be distributed and support fault-tolerance** to mask component failures.
- System as a whole is **not testable** to the required level of dependability.
- The safety argument is based on a **combination** of **experimental evidence** about the expected failure modes and failures rates of **fault-containment regions (FCR)** and a **formal dependability model** that depicts the system structure from the point of view of dependability.
- **Independence** of the FCRs is a critical issue.

Independence of FCRs

The independence of failures of different FCRs is the most critical issue in the design of an ultra-dependable system.

There are two basic mechanisms that compromise the independence of FCRs

- Missing fault isolation among the FCRs
- Error propagation--the consequences of a fault, the *ensuing error*, propagates to a healthy FCR by an erroneous message.

Federated Architecture

In a *federated architecture* each *Distributed Application Subsystem* (DAS) is implemented on its own stand-alone *distributed hardware base*, consisting of *nodes* dedicated to jobs and physical communication channels (a network) among the nodes.

This has the following consequences:

- Each DAS is physically separated from other DASes
- Clear boundaries of responsibility and error propagation
- Limited sharing of hardware and communication resources--many nodes and networks.
- Integration of functions difficult--multiple sensors necessary
- In a large system there are many nodes and communication links (physical contact points).

Examples of DASes Onboard a Car

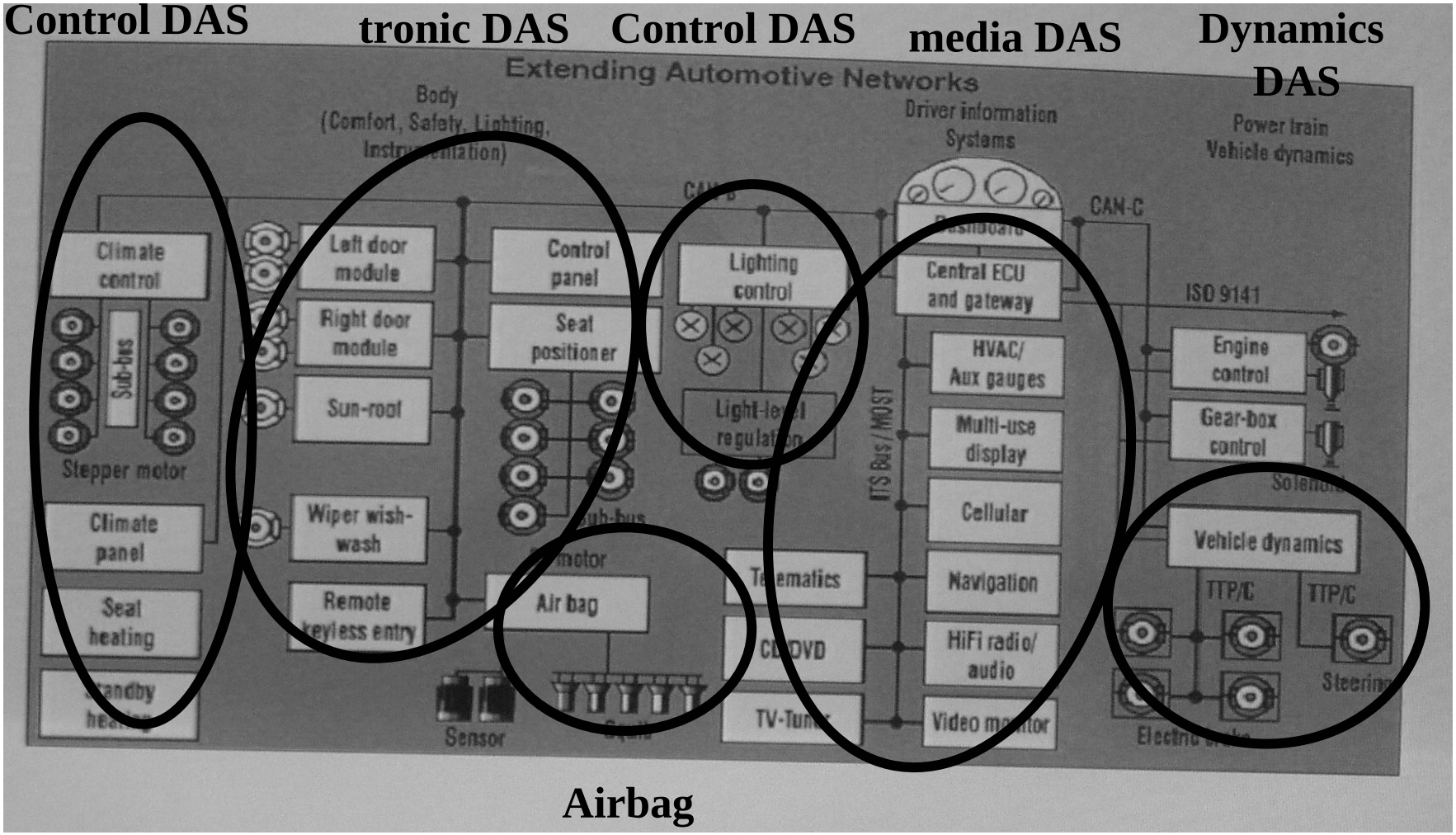
Climate Control DAS

Body Elec-tronic DAS

Ligthning Control DAS

Multi-media DAS

Vehicle Dynamics DAS



Airbag DAS

Integrated Architecture

A number of technical and economic advantages could be realized if the different DASes were integrated into a single architecture

- Cost savings by the reduction of nodes, sensors and wiring points (results also in an increase in hardware reliability).
- Better integration of functions--more flexibility
- Implementation of fault tolerance simplified

But

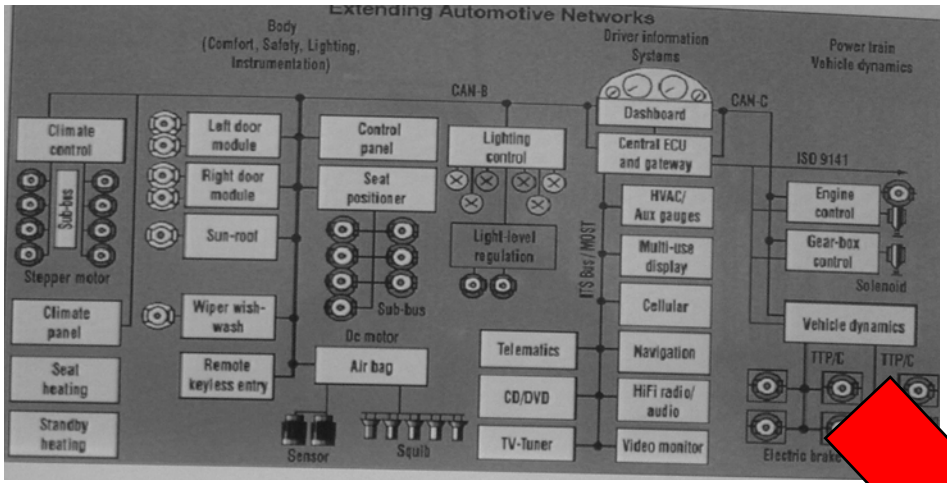
- Independence of individual DAS compromised--increased potential of error propagation from one DAS to another DAS
- Integration increases complexity and diagnostics
- Allocation of responsibility more difficult

The Challenge

The ideal future avionics systems would combine the complexity management advantages of the federated approach, but would also realize the functional integration and hardware efficiency benefits of an integrated system.

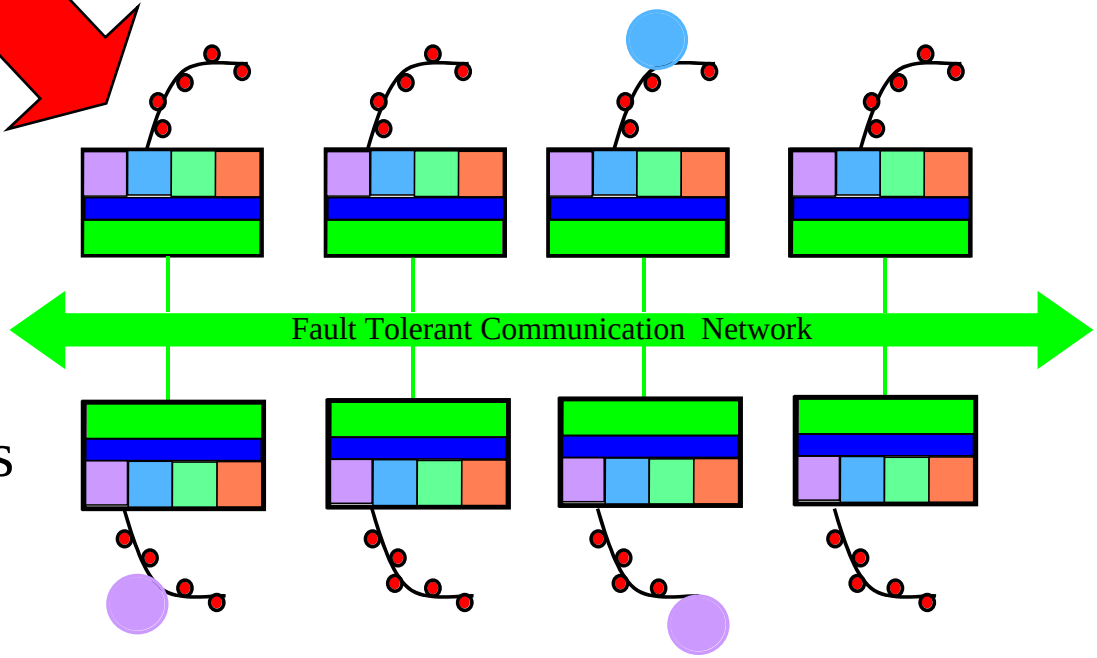
Hammett Robert. *Flight Critical Electronics System Design,* IEEE AESS Systems Magazine, June 2003, p.32

From a Federated to an Integrated Architecture



Federated Architecture:
"Every functions has its own ECU"

Integrated Architecture:
Backbone Network with integrated fault-tolerance
Intelligent Sensors and Actuators connected by field-buses



The Time-Triggered Architecture (TTA)

provides an execution environment for real-time applications. It is

- a *distributed architecture that support fault tolerance*, where a node can be a single chip computer (SoC).
- It provides a *fault-tolerant global time-base* of high precision at every node.
- an *integrated architecture*, where different application subsystems (DAS) up to the *highest criticality class* can be integrated into a single framework.
- a *platform architecture* that provides technology invariant interfaces to the application software.
- a *generic architecture*, which can be deployed in different application domains (e.g., automotive, aerospace, train signaling, process control, mutimedia) where real-time performance is an issue.

Kopetz, H, Bauer, G. , The Time-Triggered Architecture, Proc. of the IEEE, Jan 2003, Vol 91 p. 112-126

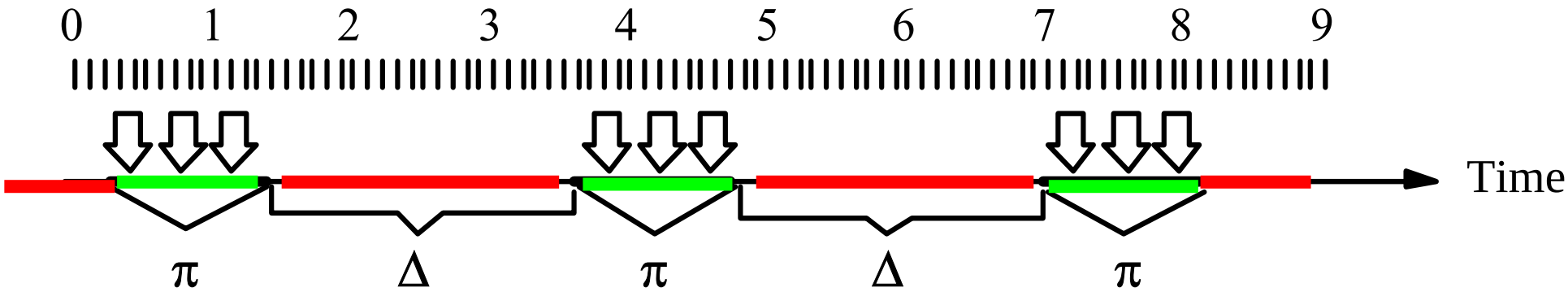
History of the TTA

The TTA has been developed over a time-frame of more than 20 years:

- Started in 1979 at the Technical University of Berlin and continued at the Technical University of Vienna since 1982.
- More than 50 Mio US $ have been invested in the TTA.
- The TTA is presently deployed in industrial applications at
  - Aerospace Applications (Honeywell)
  - Railway Signalling (Alcatel)
  - Automotive *Drive-by-Wire* Prototypes (many companies)
- On July 1, 2004 the three-year European Integrated Project DECOS (about 15 Mio US $) was started to further develop the TTA.

Fault Tolerant *Sparse* Time Base in the TTA

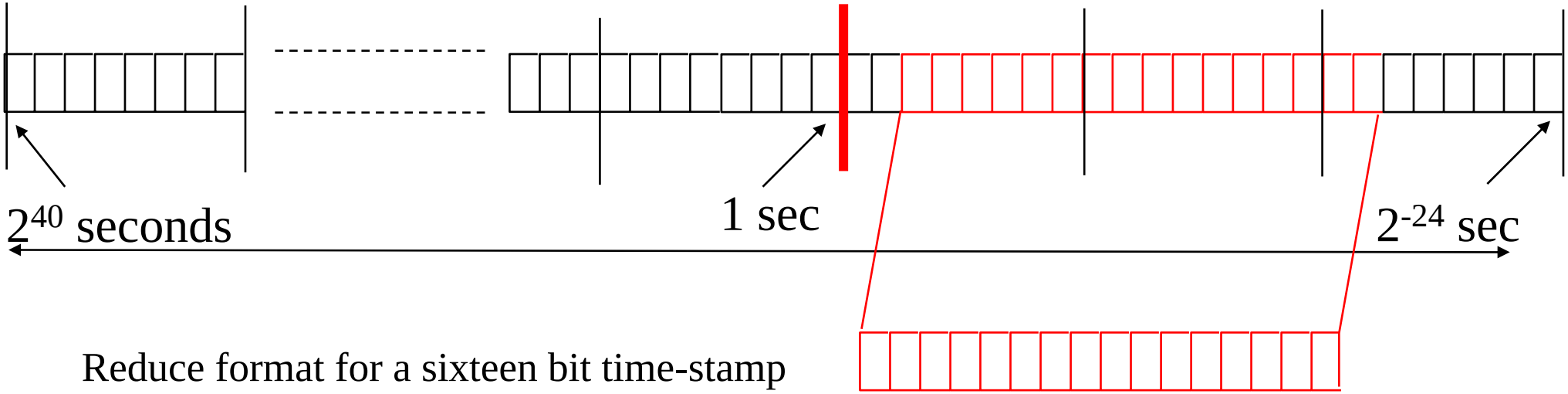
If the occurrence of events is restricted to some active intervals with duration π with an interval of silence of duration Δ between any two active intervals, then we call the timebase π/Δ-sparse, or sparse for short.



Events ⇩ are only allowed to occur at subintervals of the timeline

In a sparse time base, instants can be represented by integers.

Standardized Time Format in the TTA

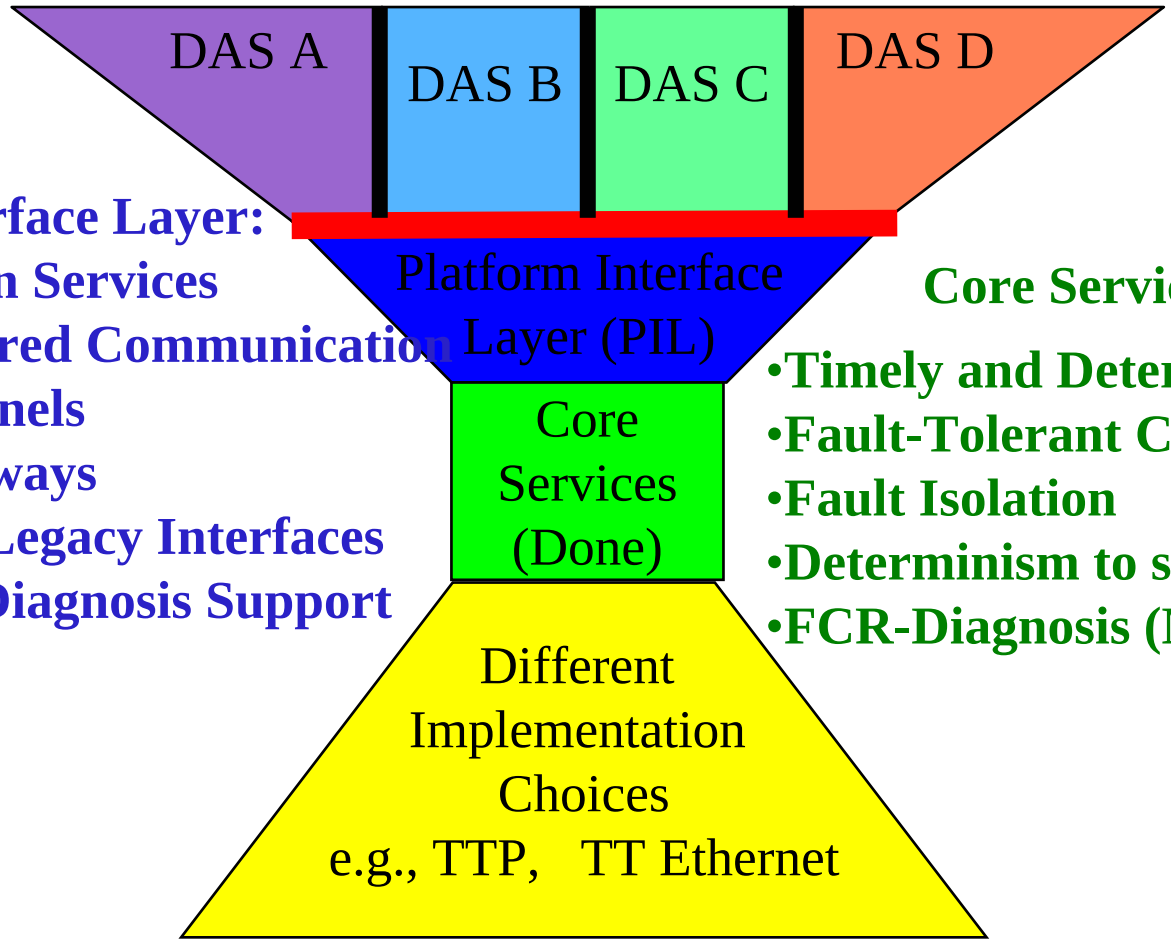


Reduce format for a sixteen bit time-stamp horizon 2^{-2} seconds, granularity 2^{-18} seconds

If desired for for efficiency reasons, a shorter time-stamp can be extracted from this global time representation as shown above.

If periods are *power-of-two compatible* to the full second, a single bit (*the periodicity bit*) can be used to denote the periodicity of an activity. The phase of the *start-of-period* can be denoted by the bit pattern to the right of the periodicity bit.

The TTA is a *Platform* Architecture



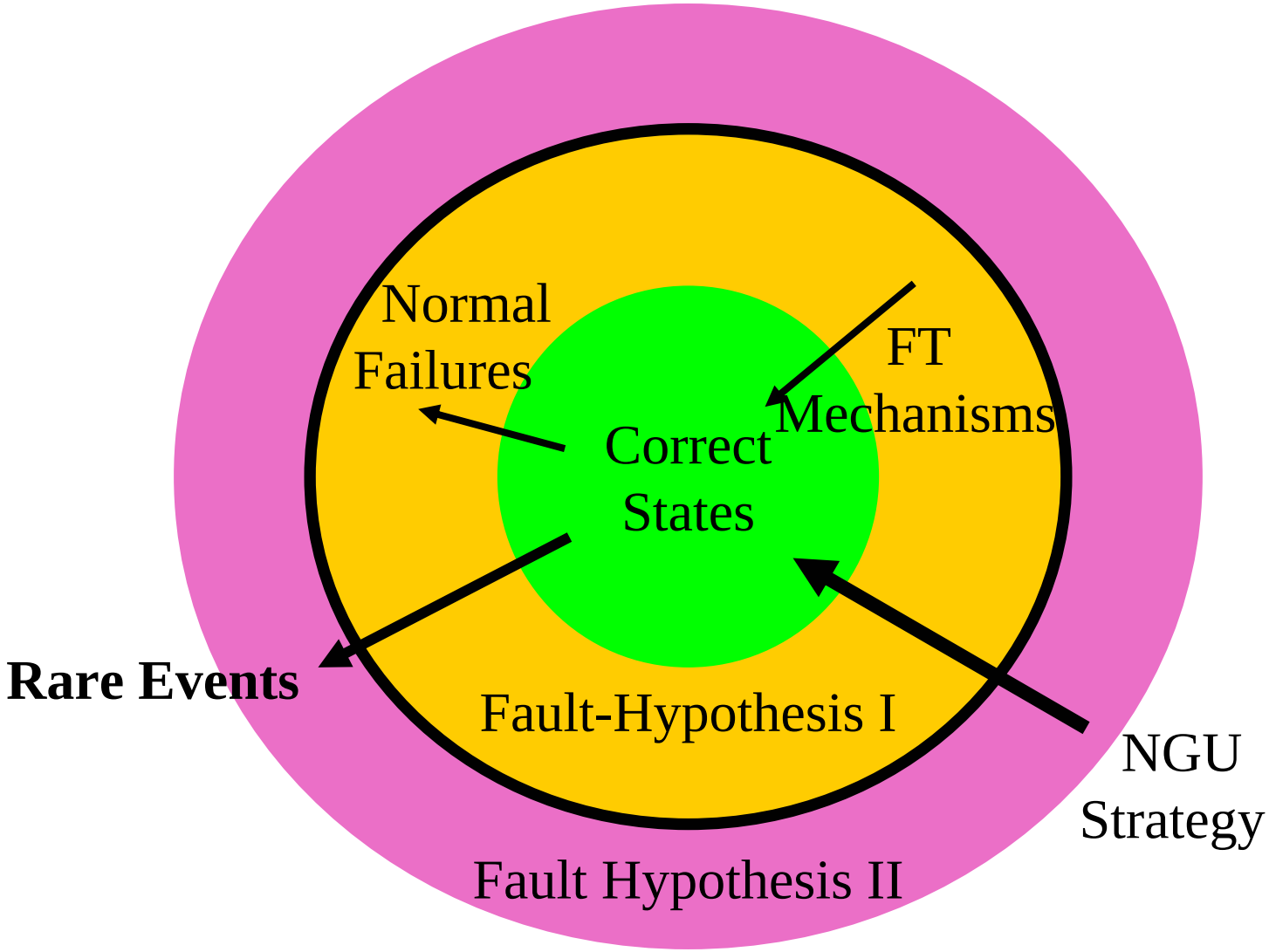
Distributed Application Systems (DAS)

Platform Interface Layer:
- **Encapsulation Services**
- **Event-Triggered Communication**
- **Virtual Channels**
- **Hidden Gateways**
- **Provision of Legacy Interfaces**
- **Application Diagnosis Support**

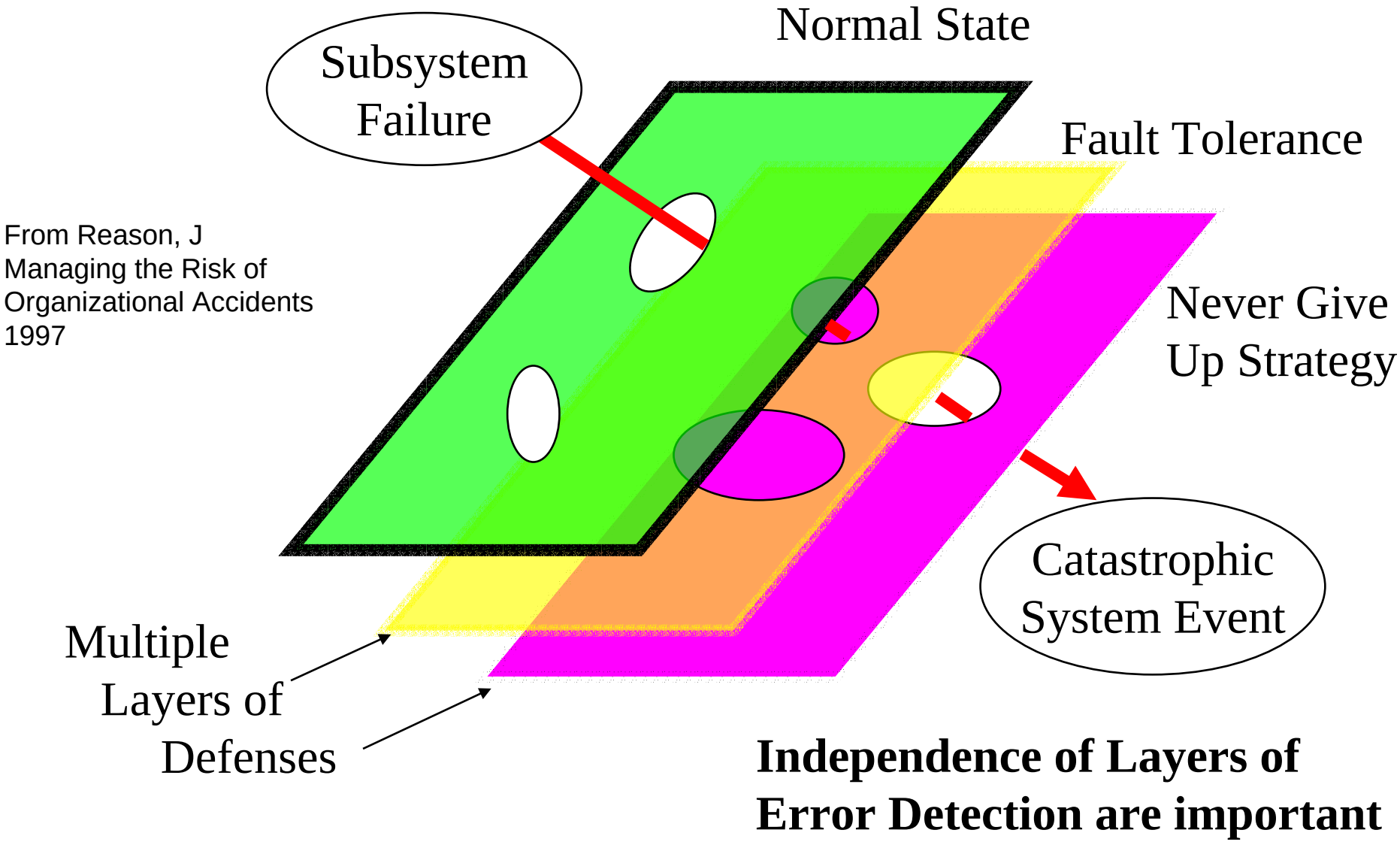
Core Services (done for TTP)
- **Timely and Deterministic Transmisson**
- **Fault-Tolerant Clock Synchronization**
- **Fault Isolation**
- **Determinism to support TMR**
- **FCR-Diagnosis (Membership)**

Technology invariant interface

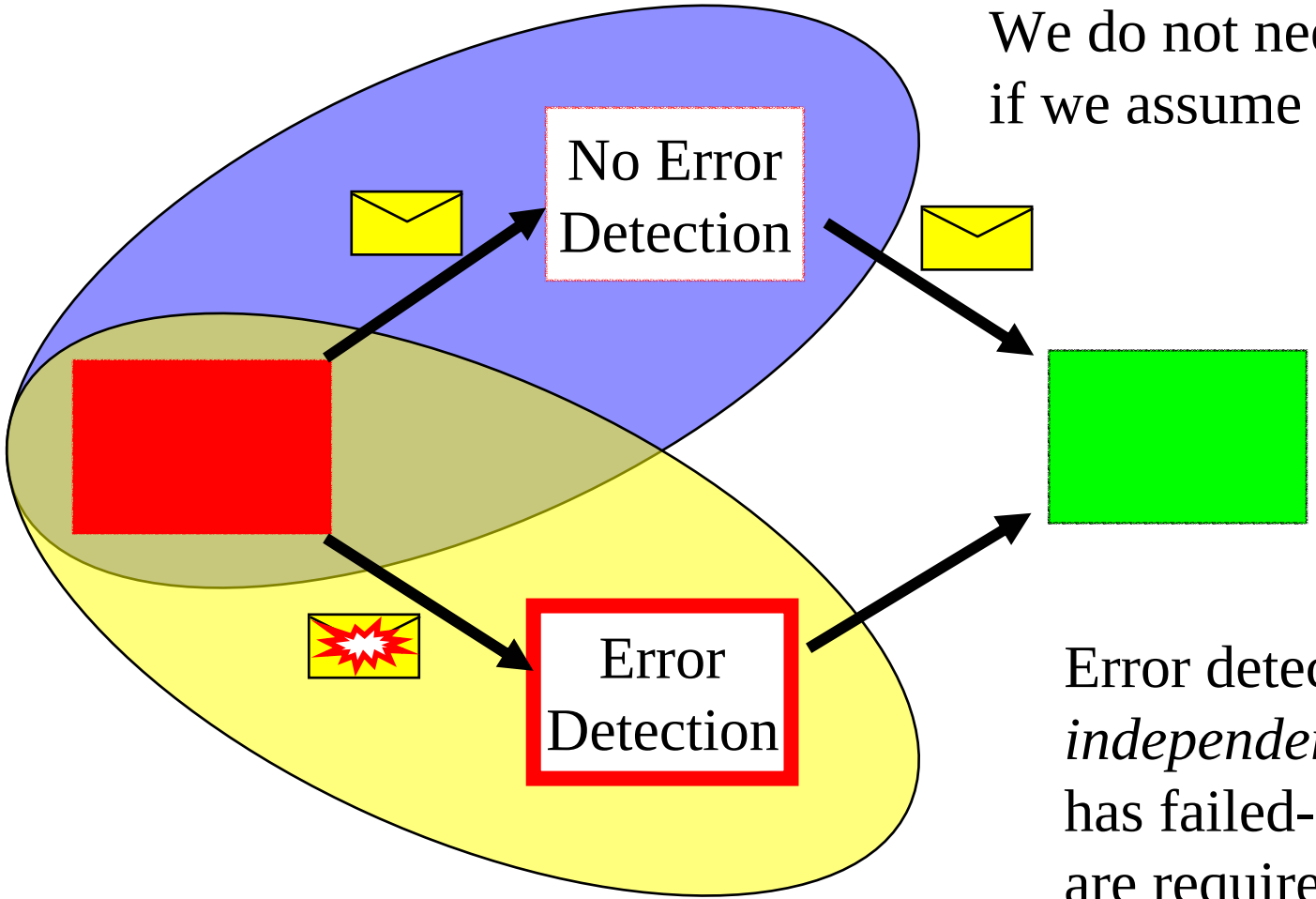
Fault Hypothesis in the TTA



Approach to Safety: The *Swiss-Cheese* Model



Fault Containment vs. Error Containment



We do not need an error detector if we assume *fail-silence*.

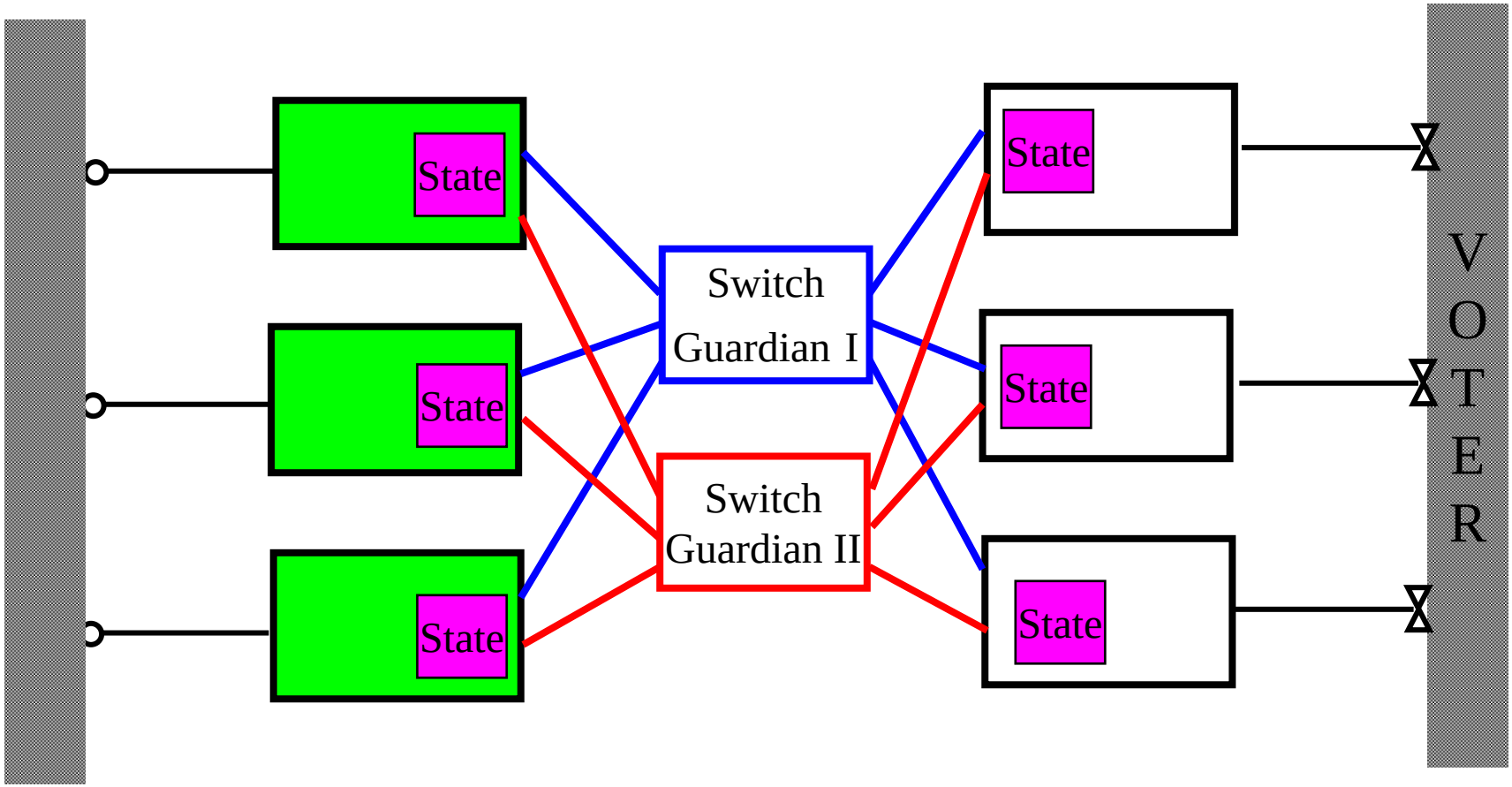
Error detecting FCR must be *independent* of the FCR that has failed--***at least two*** FCRs are required if a *restricted* failure mode is assumed.

Error Containment Region (ECR) in the TTA

In a distributed computer system the consequences of a fault, the ensuing error, can propagate outside the originating FCR (Fault Containment Region) by an **erroneous message** of the faulty node to the environment.

- A propagated error **invalidates** the independence assumption.
- The error detector must be in a **different FCR** than the faulty unit.
- Distinguish between error detection in the **time-domain** and error detection in the **value domain**.
- In the TTA, **error detection in the time-domain is performed by the architecture** and error detection in the value domain must be done by the application (TMR).
- TMR requires replica determinism at all levels.

TMR Structure for Safety-Critical Tasks



State

In order to flush out *quasi-permanent state errors* caused by a transient fault, the state must be periodically subject to voting.

Fault Hypothesis in the TTA w.r.t. Physical Faults

i. **A Node Computer** forms a single FCR that can fail in an arbitrary failure mode (it is not possible to implement two independent FCRs on the same die).
ii. **A communication channel** including the central guardian forms a single FCR that can fail to distribute messages but cannot generate messages on its own.
iii. **A central guardian in the communication system** transforms (SOS) failures to fail-silent failures in the temporal domain.
iv. **Error detection** is performed by a membership and clique avoidance algorithms.
v. **The system can recover** from a single failure within two TDMA rounds.

Assumption about the Frequency of Faults of SoCs:[22]

Assumed Behavioral Hardware Failure Rates (Orders of Magnitude):

Type of Failure	Failure Rate in Fit	Source
Transient Node Failures (fail silent)	<1 000 000 Fit (MTTF > 1000 hours)	Neutron bombardment Aerospace
Transient Node Failure (non-fail silent)	<10 000 Fit (MTTF> 100 000)	Fault Injection Experiments
Permanent Hardware Failures	<100 Fit (MTTF> 10 000 000)	Automotive Field Data

Tendency: Increase of Transient Failures

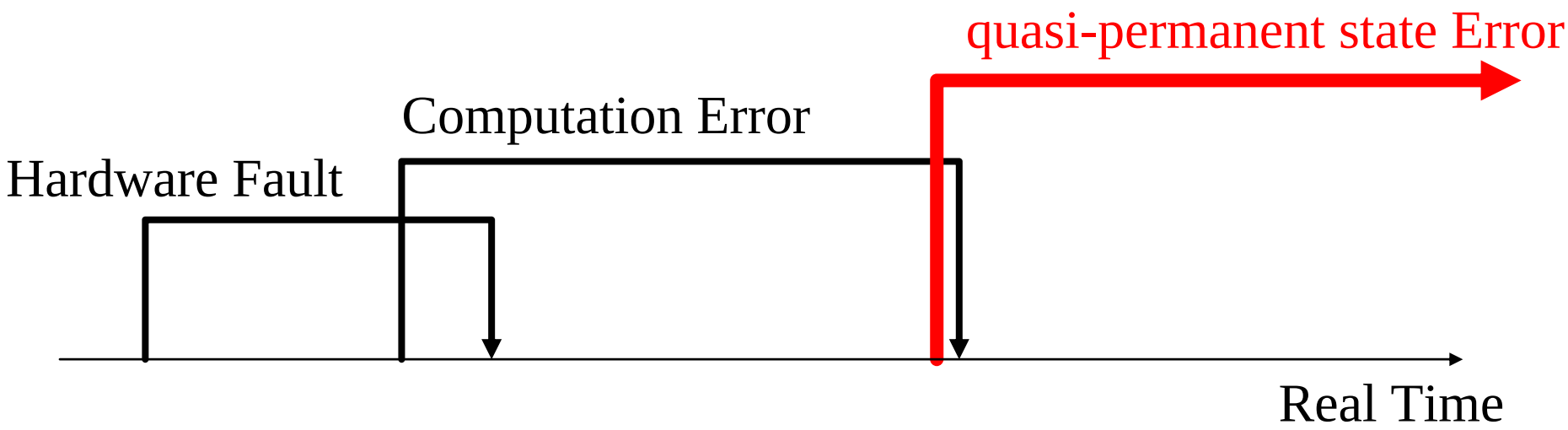
Experimental Evidence

Millions of fault injection experiments have been carried out on the TTA over a period of more than ten years with the support of the EU:

- Software based (TU Vienna, Austria)
- Alpha Particle (Chalmers University, Sweden)
- VLSI-model based (Univ. of Valencia, Spain, Carinthia Tech, Austria)
- Pin Level (LAAS, Toulouse,France, Univ. of Valencia, Spain)

Error Detection in the temporal domain	**Ratio of fail-silent to non-fail-silent failures**	**Experimental Evidence**
No Error Detector	**50:1**	FI Measurements in PDCS Project
Local Guardian	**1000: 1**	Fault Injection FIT Project
Autonomous Central Guardian	**no non-fail silent failure observed so far**	Fault Injection in FIT/NEXT TTA

Transient Faults may cause Permanent State Errors



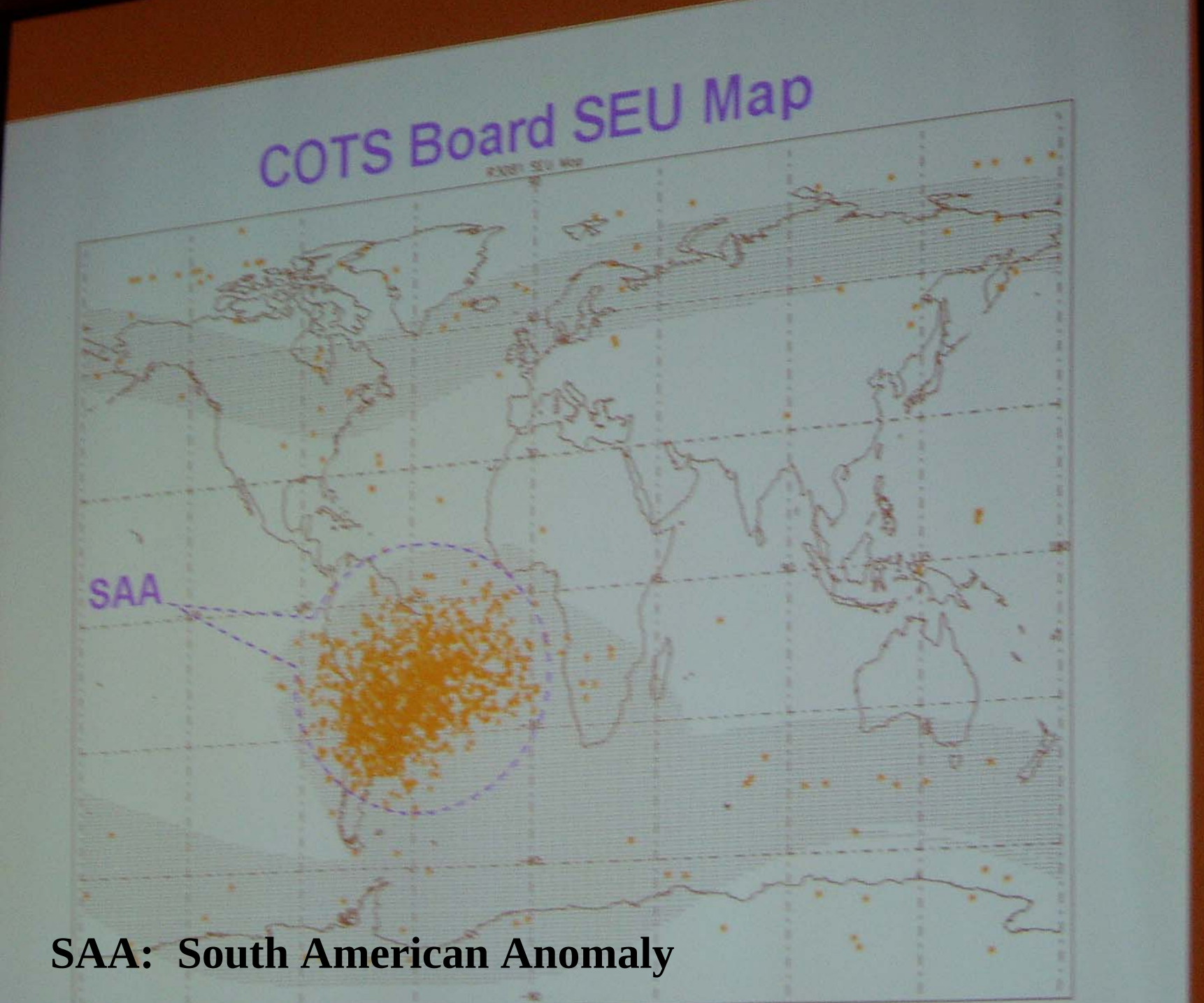
The interaction of a transient hardware fault with the state an cause a quasi-permanent state error: state erosion

Transient failures MTTF: 1000 hours
Permanent failures MTTF: > 1 000 000 hours

The Cause of a Transient Fault

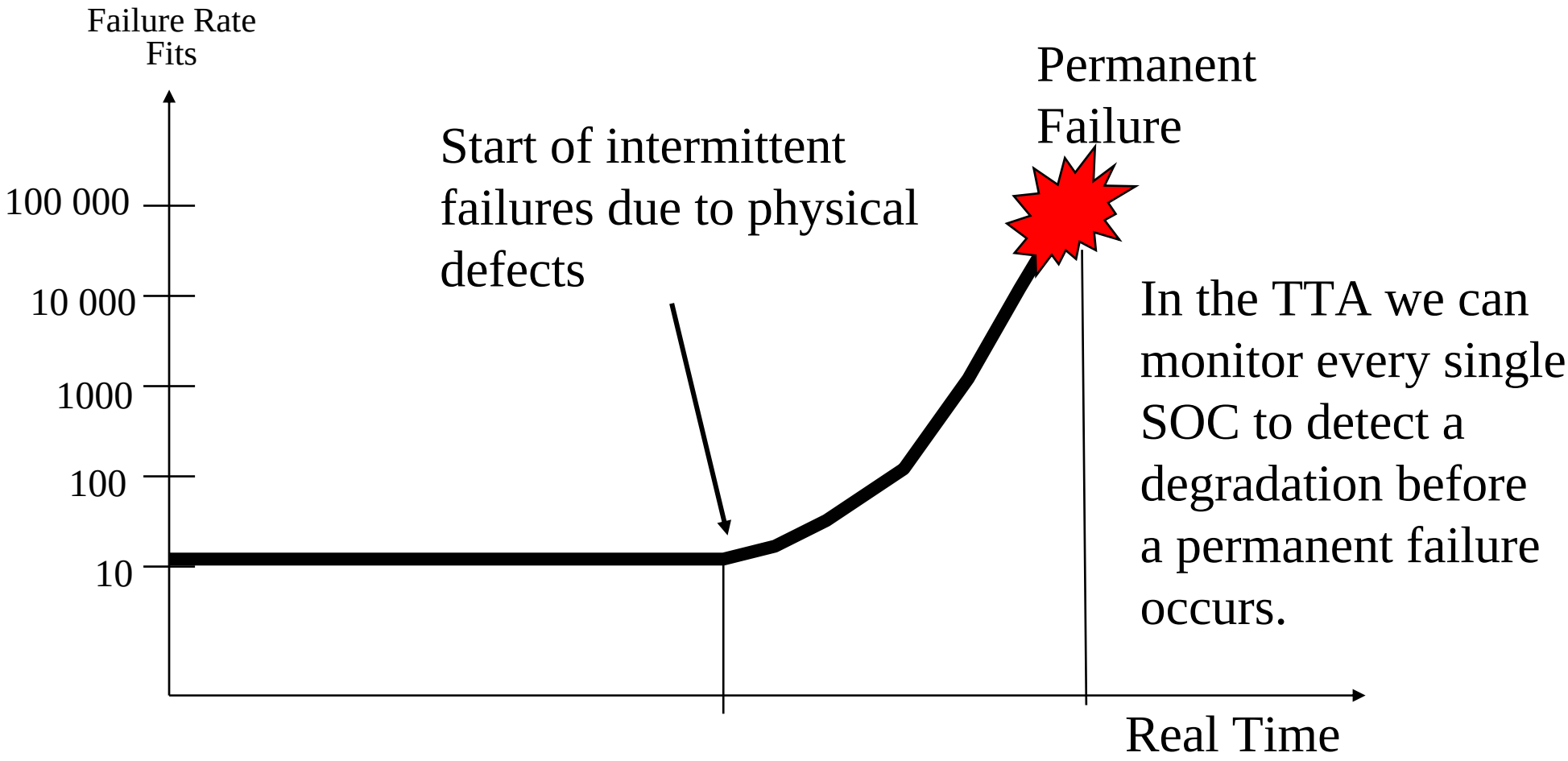
We have identified the following possible causes of a transient fault

- **External Disturbances**, e.g., high energy radiation (hardware)
- **Internal Degradation of the chip hardware**: e.g., corrosion of a PN junction (hardware)
- **Heisenbugs**, e.g., design error in the synchronization of processes (software)



SAA: South American Anomaly

Intermittent Failures of a Chip causes Transients



More than half of the transients may be caused by intermittents.

The Distinction between *Bohrbugs* and *Heisenbugs**

- *Bohrbugs* are design errors in the software that cause reproducible failures. E.g., a logic error in a program.
- *Heisenbugs* are design errors in the software that seem to generate quasi-random failures. E.g., a synchronization error that will cause the occasional violation of an integrity condition.
- From a phenomenological point of view, a failure that is caused by a *Heisenbug* cannot be distinguished from a failure caused by transient hardware malfunction.
- Experience shows that it is much more difficult to find and eliminate the *Heisenbugs* than it is to eliminate the *Bohrbugs* from a large software system.

*J. Gray, "Why do Computers Stop and What can be done about it?," Proc. 5th Symp. on Reliability in Distributed Software and Database Systems, Los Angeles, CA, USA, 1986

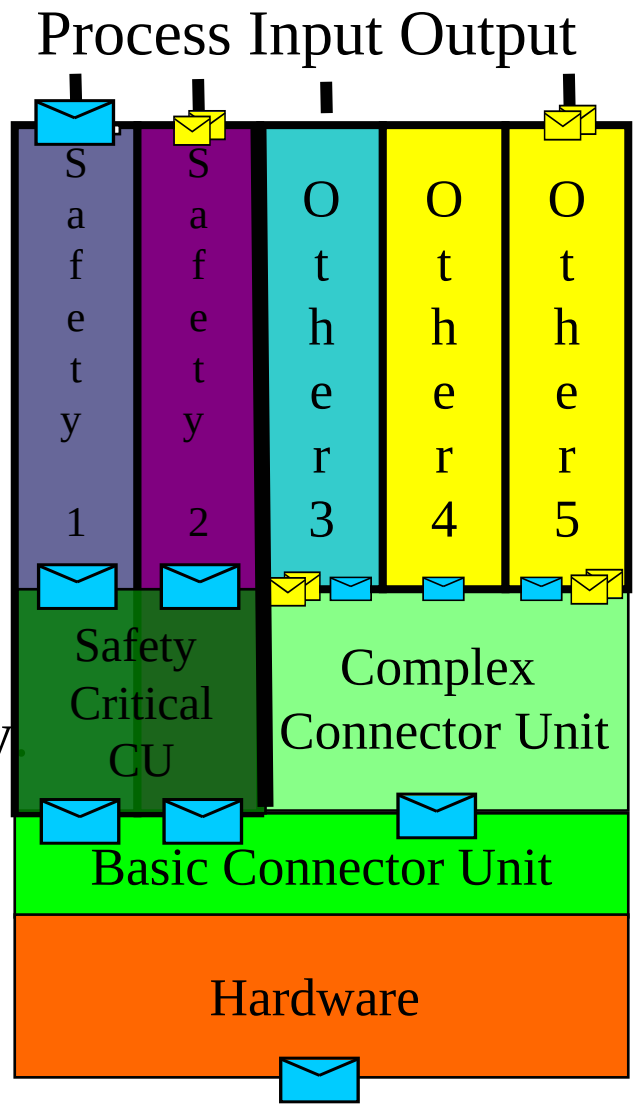
The Replacement Strategy

- From the observation of a transient failure of a node, it *impossible* to identify in a *single function node* the cause of the transient.
- It is possible to reason about the cause of the transient if a *population of nodes* is observed over time.
- It is also possible to reason about the cause of the transient if the malfunctions of a *single multifunction* node are observed over time.

Mixed-Criticality TTA Node

Mixed-Criticality Node with 6 Partitions, controlled by connector units.

The two safety-critical partitions depend on the correctness of the Basic Connector Unit only.



malign failures

DAS 1 Safety Critical

DAS 2 Safety Critical

benign failures

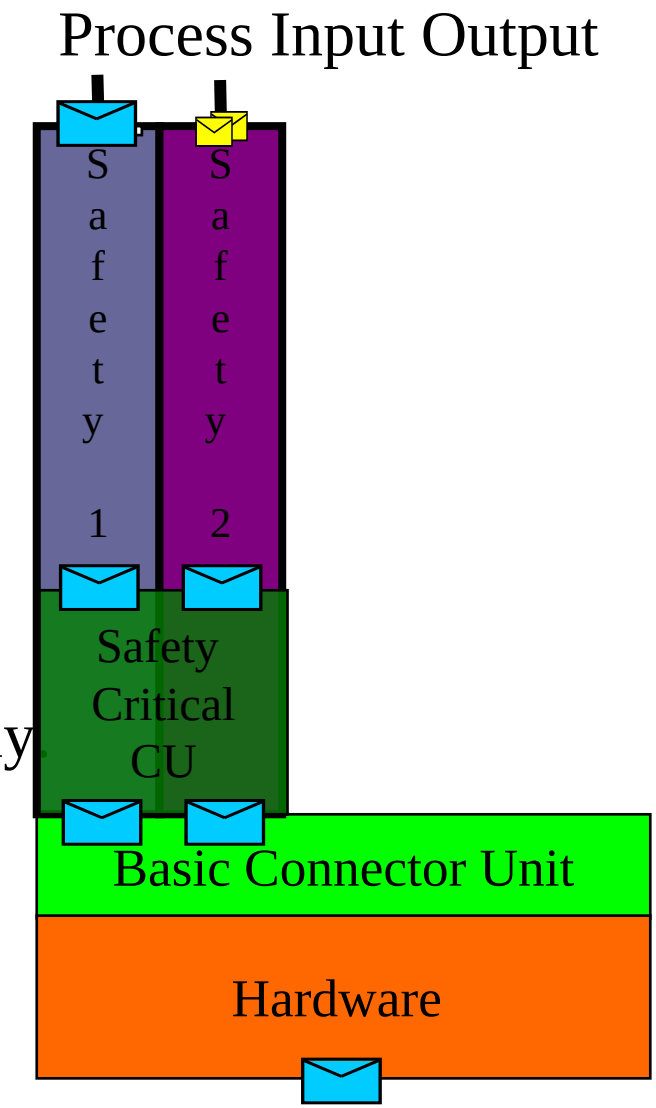
DAS 3

DAS 4

Critical Parts of a Mixed-Criticality TTA Node

Mixed-Criticality Node with 6 Partitions, controlled by connector units.

The two safety-critical partitions depend on the correctness of the Basic Connector Unit only.



malign failures

DAS 1 Safety Critical

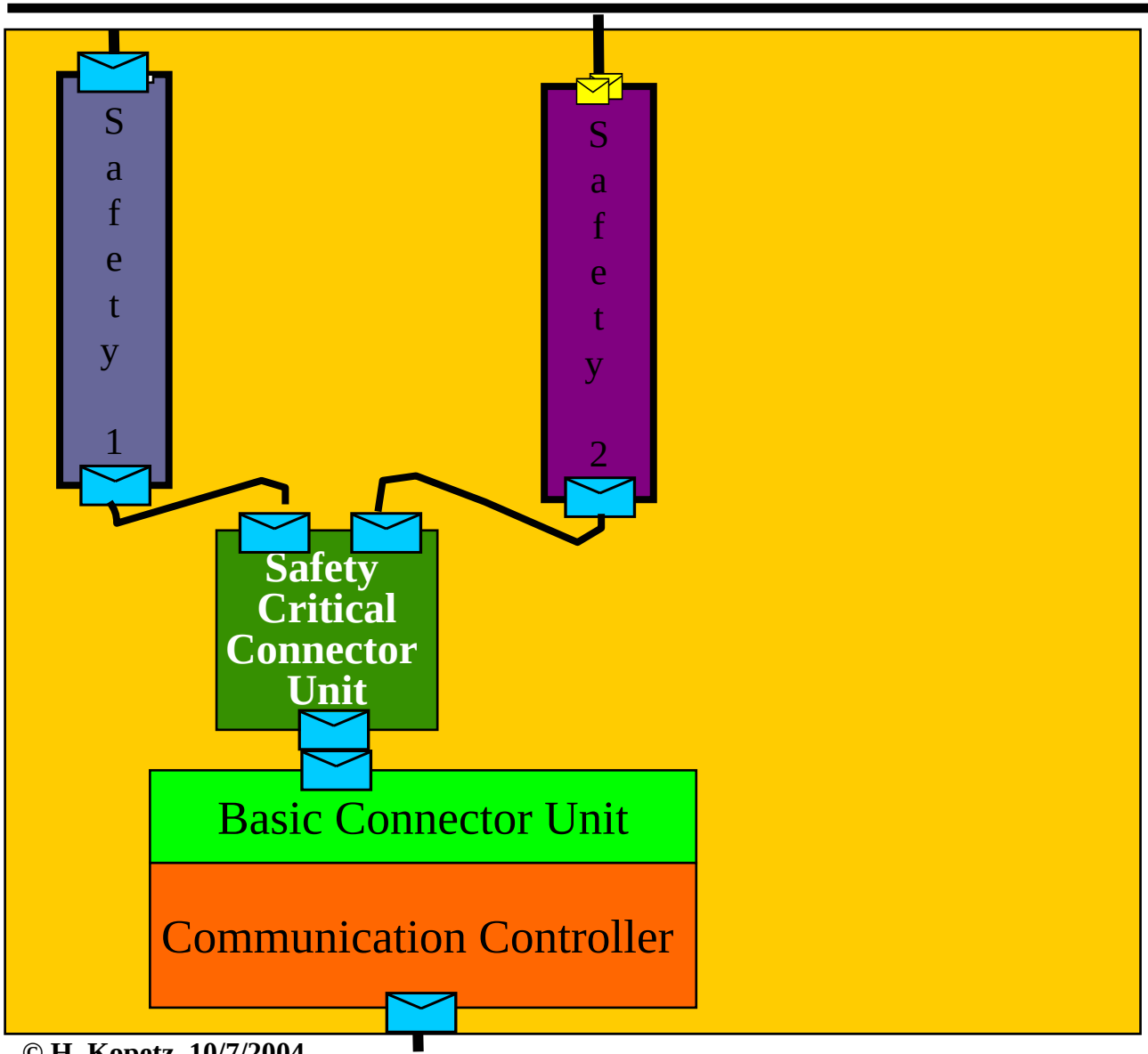
DAS 2 Safety Critical

benign failures

DAS 3

DAS 4

Modular Certification of the Critical Parts



Each unit can be certified in isolation form each other unit.

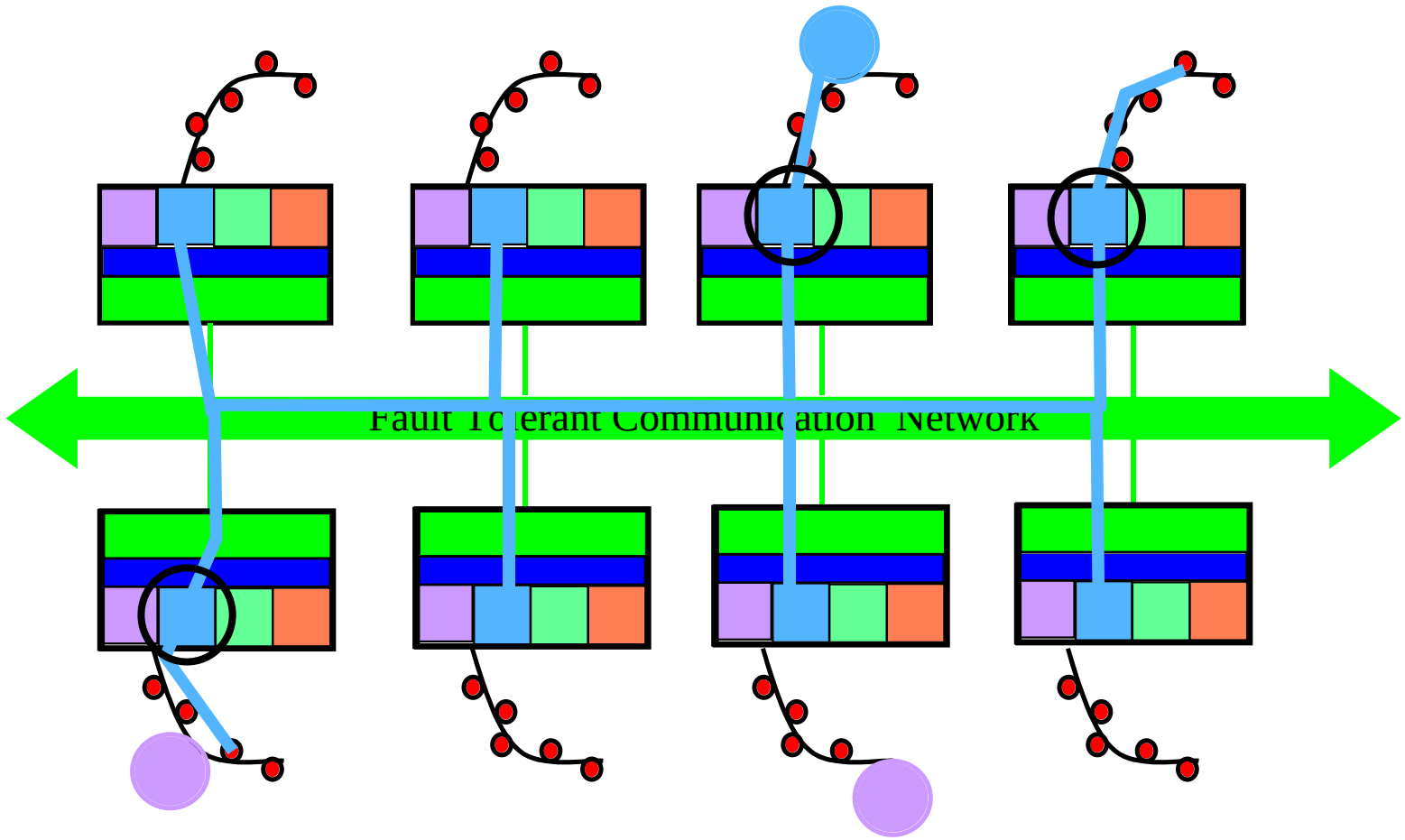
Unintended interactions are avoided by design.

Integration in the TTA

The TTA provides a distributed execution environment for the different DASes with the following properties:

- A job is encapsulated in a partition of a node. A node can support many partitions.
- A job communicates to its environment by the network or by a private I/O interface.
- One or more separated virtual communication channel with specified temporal properties is allocated to each DAS to link the ports of the DAS.
- A physical wire can host many different virtual networks with *a priori* known temporal properties.
- Different DASs can interact via a virtual gateway.

Example: Private Semivirtual CAN of a DAS



Blue nodes are connected by a virtual CAN networks

Hidden physical gateway

Two-level Design Methodology in the TTA

A two level design methodology that is supported by tools from TTTech, a spinoff company from the TU Wien (www.tttech.com):

System Level specifies the interactions among components by designing the Temporal Firewall Interfaces:

- Data items that are exchanged among the subsystems
- Instants when the TT communication system accesses the data
- Abstract model of the meaning of the data.

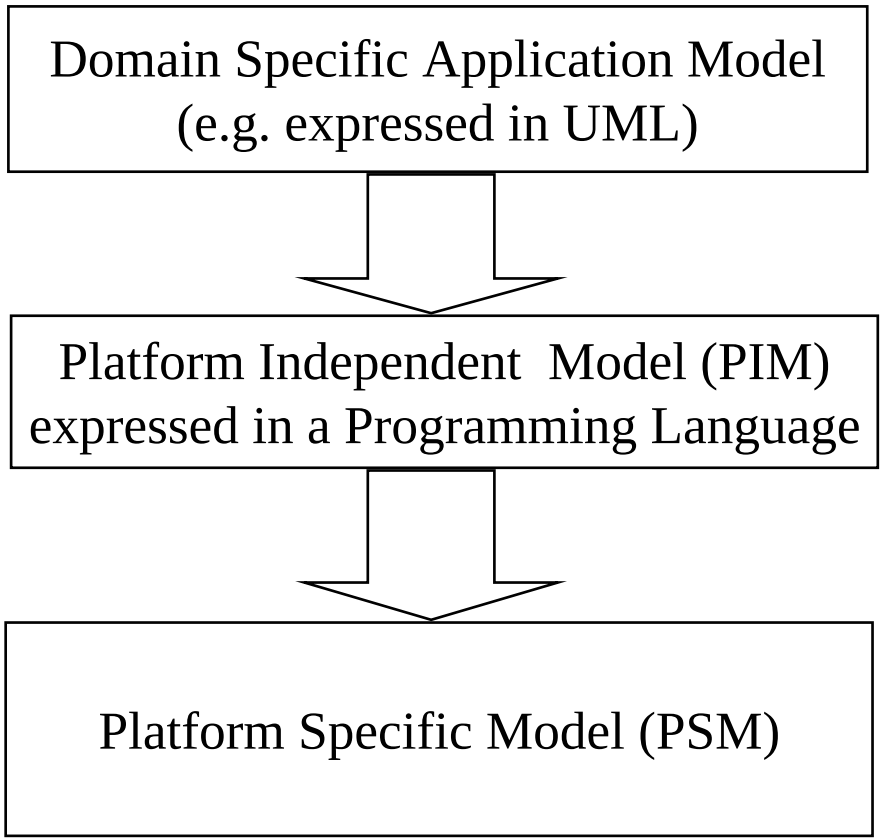
Component Level is concerned with the detailed Software Design:

- The host computer provides the intended function, taking the available temporal firewall specifications as constraints.
- Validation of a component with respect to the temporal firewalls can be performed in isolation.

PIM vs. PSM of a DAS

- The the platform independent model of a DAS consists of a set of *Jobs* that communicate via *interfaces* containing *ports*, connected to a *virtual communication channel*.
- The platform specific model (PSM) of a DAS is a model where the jobs have been assigned to partitions of nodes and the virtual channels to physical TT channels.
- The development of the PSM is constrained by
  - Dependability requirements (replicated jobs must be in partitions of independent FCRs)
  - The resource constraints of the nodes
  - The resource constraints of the physical TT channels

Model-Driven Design



Partition the application into modules and specify the message interfaces among the modules

Map the model to the selected target platform considering Quality of Service (QoS) Properties (e.g., Timeliness, Replication)

Diagnostic Subsystem of the TTA

- Independent DAS that collects and evaluates diagnostic information on line
- Two sources of diagnostic information
  - **System Based**: out of norm assertions, message loss, restart, difficulties in agreement, failures that are masked by fault tolerance
  - **Application based**: jobs generate diagnostic information based on their application know how (e.g., sensor behavior)
- Diagnostic information is evaluated on-line by a diagnostic job

Future Developments

At present, the TTA uses the TTP/C protocol chips which support transmission speeds of up to 25 Mbits/second.

We are extending the TTA to higher speeds on TT Ethernet:

- Distinguishes between two traffic classes: TT and ET
- ET is fully compatible with standard Ethernet
- TT traffic is standard Ethernet format but routed through the switch with constant delay (< 5 μsec) and minimal jitter (< 1 μsec)
- First version of TT Ethernet switch available before end of 2004.
- Gigabit Ethernet System under consideration.

Conclusion: Benefits of the TTA

The TTA is an *integrated architecture* for the implementation of large distributed real-time control system in high-dependability applications. The TTA realizes the *positive aspects of integration* such as

- significant reduction of the software cost by the strong support for composability, diagnosis, and the reuse of services.
- significant reduction of the hardware cost and of the wiring points (reliablity improvement)
- architectural support of the implementation of fault tolerance
- increased potential and flexibility of function integration

while minimizing the *negative impact of integration*

- error propagation from one DAS to another DAS
- blurring of responsibility during system integration
- difficulties in diagnostics.