# Diagnosis in the Time-Triggered Architecture

H. Kopetz

June 2010

# *Embedded Systems*

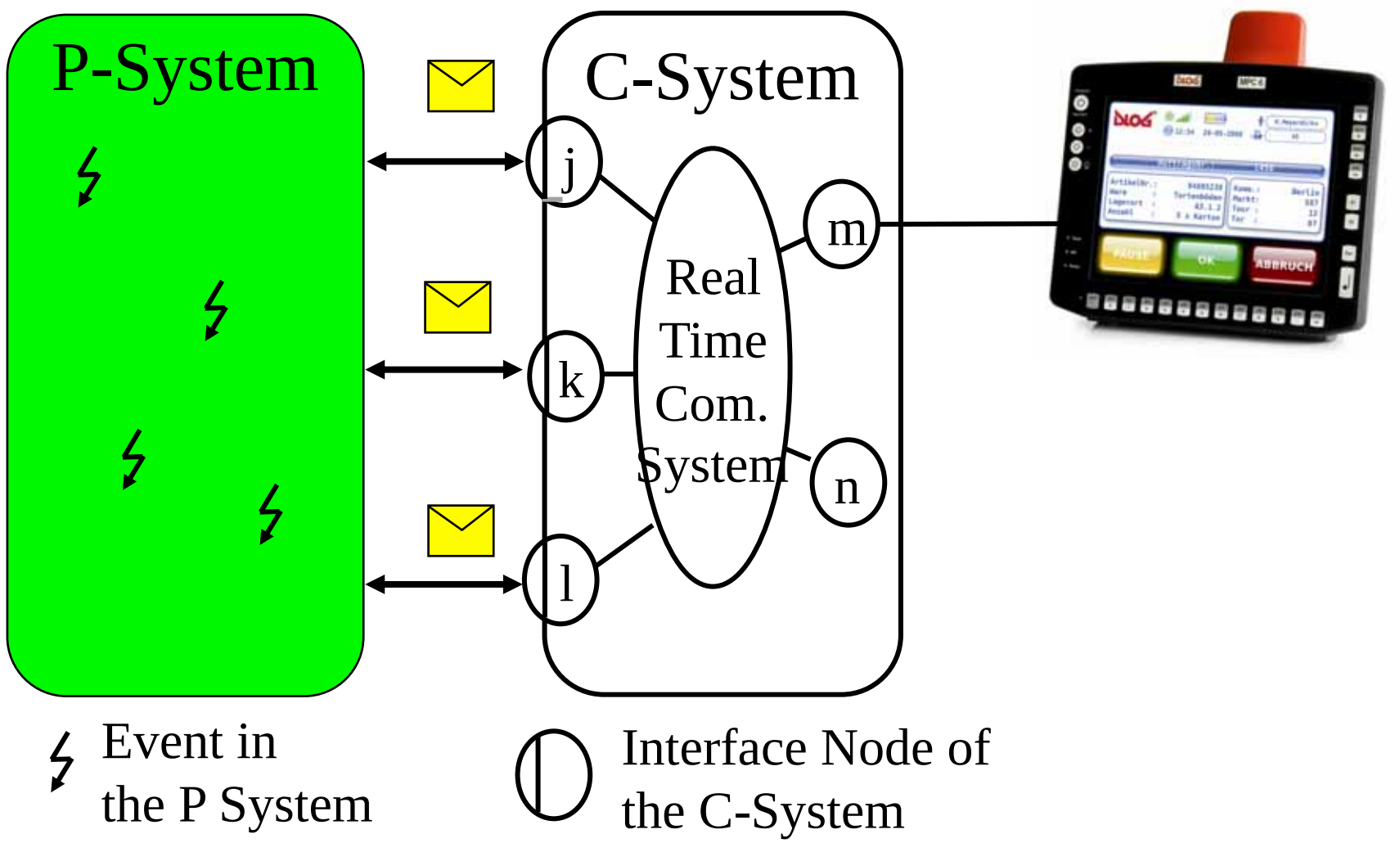
An *Embedded System* is a *Cyber-Physical System (CPS)* that consists of two subsystems:

♦A *physical subsystem –the P-System*--that is controlled by the laws of physics and is based on a *dense model of time*, and.

♦A (distributed) *computer system—the C-system*-- that is controlled by *computer programs* and is based on a *discrete model of time*.

There are different models of time in these two subsystems: *dense time* in the physical system and *discrete (or sparse)* time the cyber system.

# Embedded System: *Physical World* meets *Cyber World*



# *System Boundaries* are not *Clear-Cut*

- In a large plant, the boundaries between the *physical system* and *cyber-system* cannot be established easily.
- If we partition a plant into components, many components will belong to both worlds, e.g., a *smart sensor* or a *smart actuator*.
- We have to take a *system view*, where the behavior of components is the concern, irrespective of whether they belong to the physical world or the cyber world.

# The *Grand* Vision

- Reduce unscheduled maintenance in industrial plants to *zero*.
- Produce cars that will *never* fail on the road.
- Deliver embedded systems with close to
  *100% availability*
  for 24 hours per day and 7 days per week.

# *Reality* is not *Technology Paradise*

- Every piece of hardware in the plant and in the computer system will *eventually* fail.
- Failures will occur sporadically (*unforeseen*) or can, to some extent, be anticipated (*foreseen*).
- The design (software) is not free of design errors
- A successful system will evolve which leads to changes in the specification.

# Six Steps to Take

(1) Identify and design the planned field-replaceable units (FRU) with appropriate properties.

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

(5) Replace the faulty FRU

(6) Restart the replaced FRU

# Outline of the Talk

We will analyze each one of these six steps and show which mechanisms of the the Time-Triggered Architecture support each one of these steps.

# *Complexity Management*

In the Time-triggered Architecture (TTA) the design of the *diagnostic subsystem* is driven by the principle

**Divide and Conquer**

meaning that the diagnostic subsystem is, as far as possible, completely independent from the rest of the system. There should be no *unintended interdependence*, at the hardware or software level, of these two subsystems.

# Six Steps to Take

(1) Identify and design the planned field-replaceable units with appropriate properties (FRU).

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

(5) Replace the faulty FRU

(6) Restart the replaced FRU

# (1) FRU-Design

The size and accessibility of FRUs is determined by the maintenance strategy

- Both the physical system and the control system of an embedded system must be partitioned into FRUs
- The size of an FRU has an influence on the *reliability* of an FRU.
- FRUs with software, where design faults can be corrected by reloading a new version of the software, need special attention.

# *Size* versus *Reliability* of an FRU

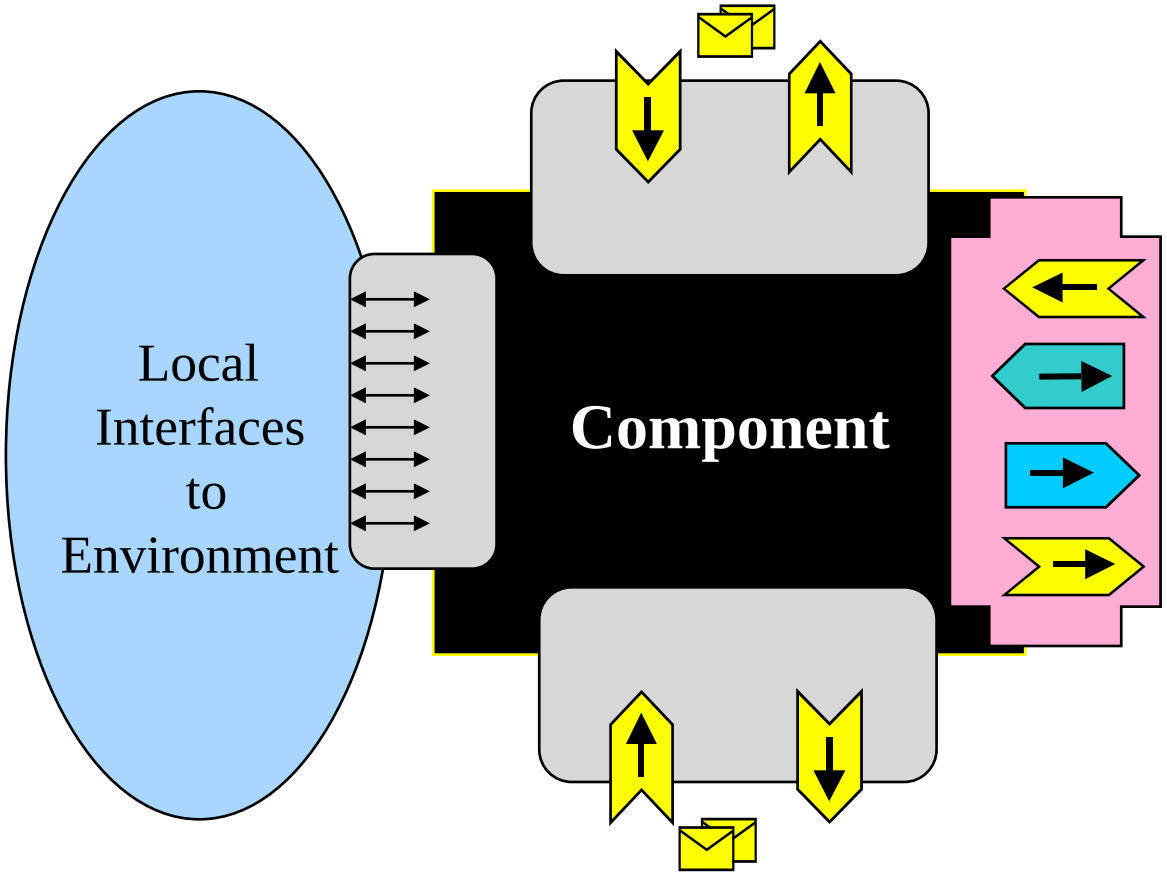
- An FRU must have maintainable interfaces such that it can be replaced easily within a short time.
- A maintainable interface is *less reliable* than a solid interface, e.g., *plug* vs. *solder* connections.
- A *non-maintainable subsystem*, where all connection are solid, is the most reliable subsystem.
- The size of an FRU is determined by the tradeoff between spare part costs, reliability and maintenance effort.

# Contribution of the TTA

- ✓ The TTA is based around a crystal clear *component model*—A component is a hardware software unit with precisely specified behavior across the message interfaces.
- ✓ A TTA component is a Fault Containment Unit (FCU).
- ✓ Temporal Error Propagation of a Faulty FCU to FCUs that are not affected by the fault is not possible due to the error-containment boundaries established by the time-triggered communication system.
- ✓ An FRU consists of one or more TTA components.

# Message Interfaces of a Component

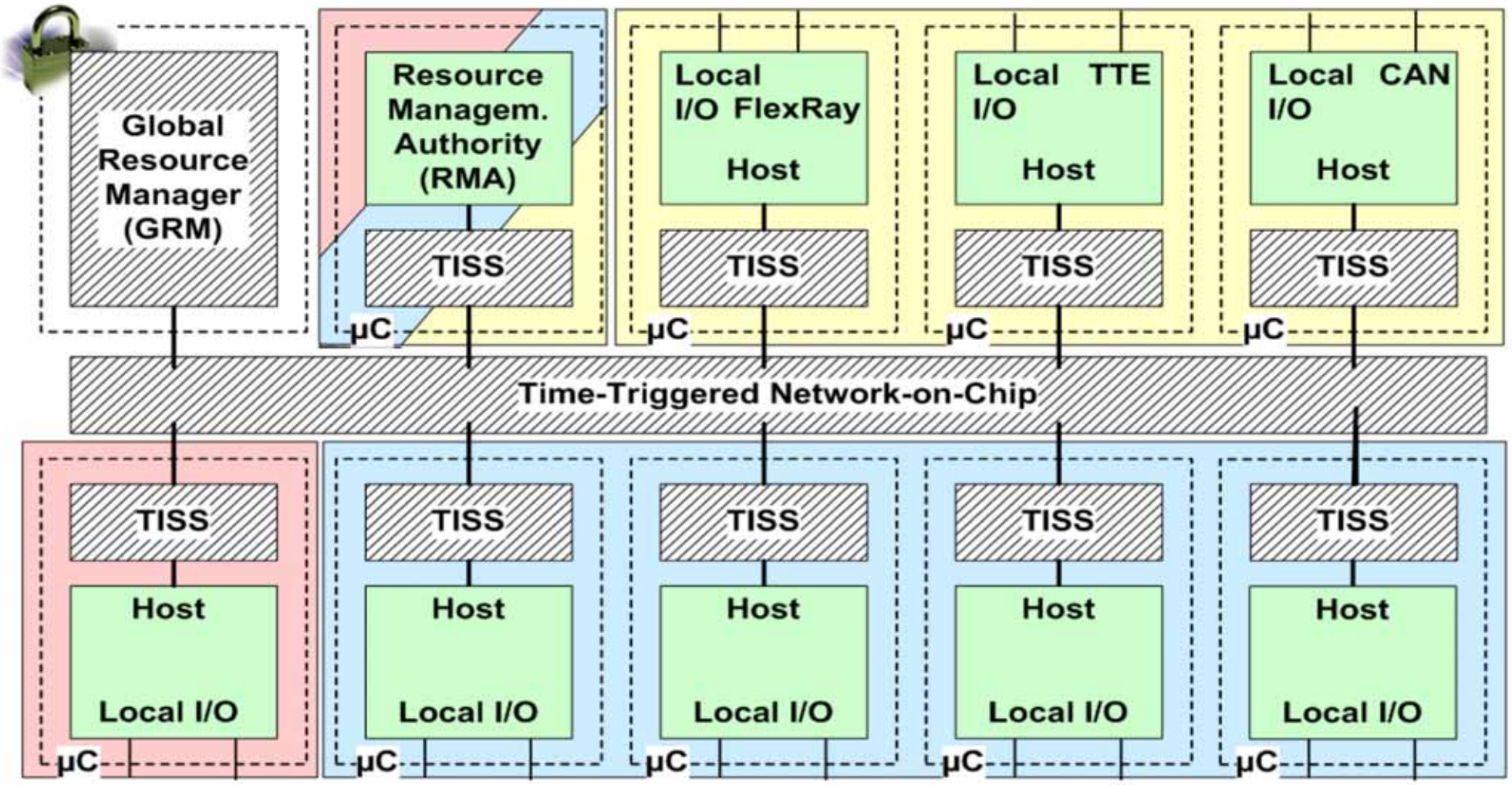
**TII Technology Independent Control Interface**
(Configuration and Execution Control)




**Linking Interface (LIF)**
Offers the services of a component.
Relevant for Composability

**TDI Technology Dependent Control Interface**

# On an MPSoC a TTA Component is an IP-Core



# Six Steps to Take

(1) Identify and design the planned field-replaceable units with appropriate properties (FRU).

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

(5) Replace the faulty FRU

(6) Restart the replaced FRU

# (2) Design Faults versus Physical Faults

- Only the impact of a fault on the behavior of a component can be observed.
- From the behavioral point of view, it is not possible to decide, whether a single failure was caused by a *transient hardware fault* or a *Heisenbug* in the software.
- In order to make this distinction—*which is very relevant from the point of view of maintenance*—a *set of transient failures* in a *population of devices* must be analyzed.

# (2) *Transient* versus *Permanent*

- *Transient Failures*: the behavior is incorrect, but the hardware is not damaged. A fast restart of the component with a relevant restart state will eliminate the error.
- *Permanent Failure:* the hardware is permanently broken und must be replaced.
- At first, a failure of an electronic component is assumed to be transient. If the restart is not successful, the assumption must be revised and a permanent hardware failure must be assumed.

# *Foreseen* versus *Unforeseen* Failures

It is distinguish between foreseen failures and unforeseen failures:

- *Foreseen Failures (wear out)*: FRUs with an increasing failure rate, where a failure can be foreseen: provide sensors to measure the parameters that are linked to an increase in the failure rate: e.g., temperature of a bearing.
- *Unforeseen Failures*: It is difficult to predict the failure of an electronic component.

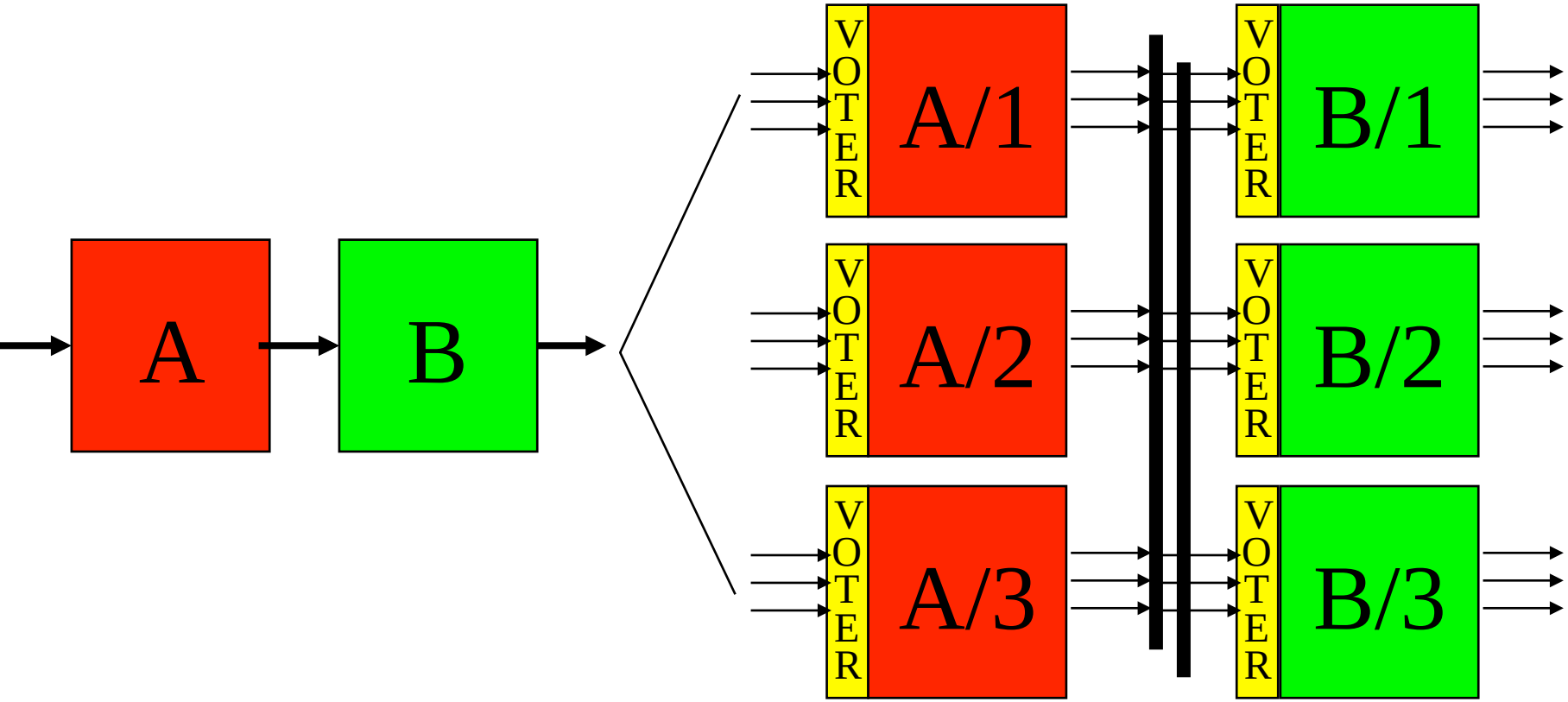
# Transform *Unforeseen* Failures to *Foreseen* Failures

What architectural services are needed to implement Triple Modular Redundancy (TMR) at the architecture level?

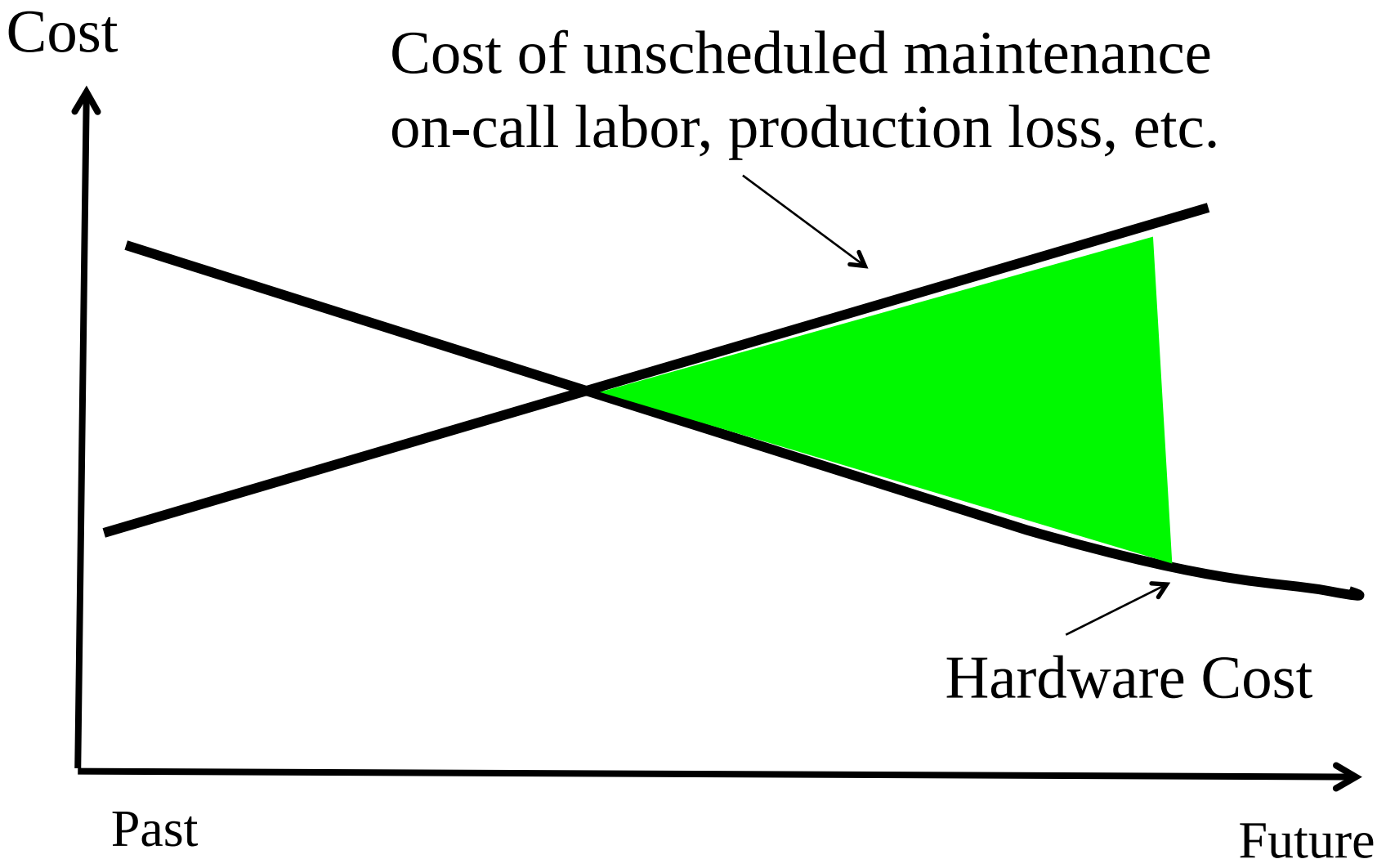
- Provision of an Independent Fault-Containment Region for each one of the replicated components
- Synchronization Infrastructure for the components
- Predictable Multicast Communication
- Replicated Communication Channels
- Support for Voting
- Deterministic (*which includes timely*) Operation
- Identical state in the distributed components

# Triple Modular Redundancy (TMR)

*Triple Modular Redundancy (TMR)* is the *generally accepted technique* for the mitigation of component failures at the system level:



# Costs-Effectiveness of TMR



# Contribution of the TTA

The TTA provides the mechanism needed for the implementation of fault tolerance by active redundancy, such as TMR:

- ✓ components are independent FCUs
- ✓ predictable deterministic communication
- ✓ synchronized global time
- ✓ multi-cast communication
- ✓ fault-tolerant clock synchronization

After a permanent fault of a FCU, the FCU can be replaced at the next routine scheduled maintenance interval.

# Six Steps to Take

(1) Identify and design the planned field-replaceable units with appropriate properties (FRU).

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

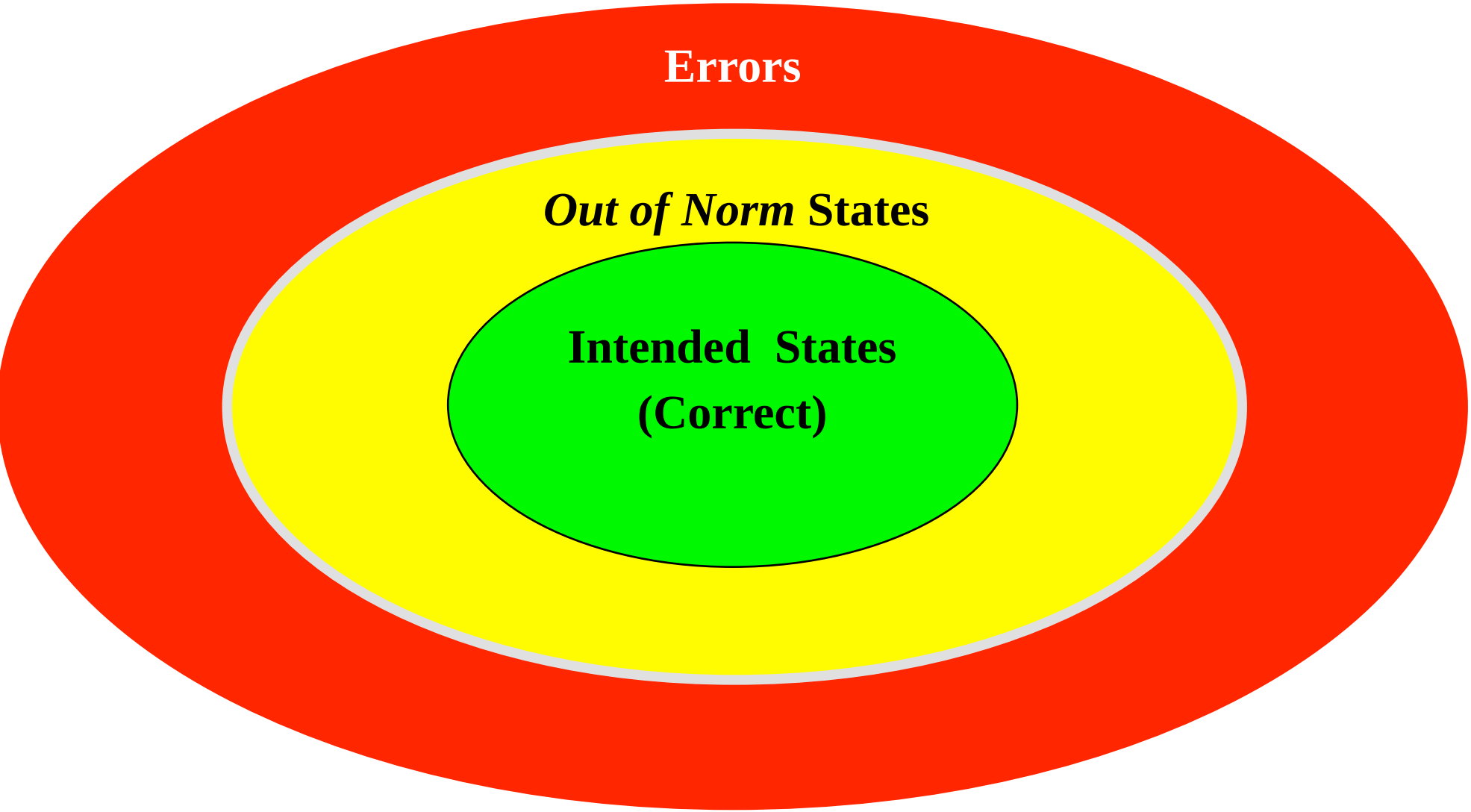
(5) Replace the faulty FRU

(6) Restart the replaced FRU

# (3) Observe all Anomalies and Failures

- An *anomaly* is a deviation from the normal behavior. It is in the grey zone between correct behavior and failure.
- It is sometimes difficult to distinguish between an anomaly and an outright failure.
- An increasing number of anomalies indicates an approaching failure: example *intermittent fault*
- Independent observation of the behavior must be supported at the level of the architecture.

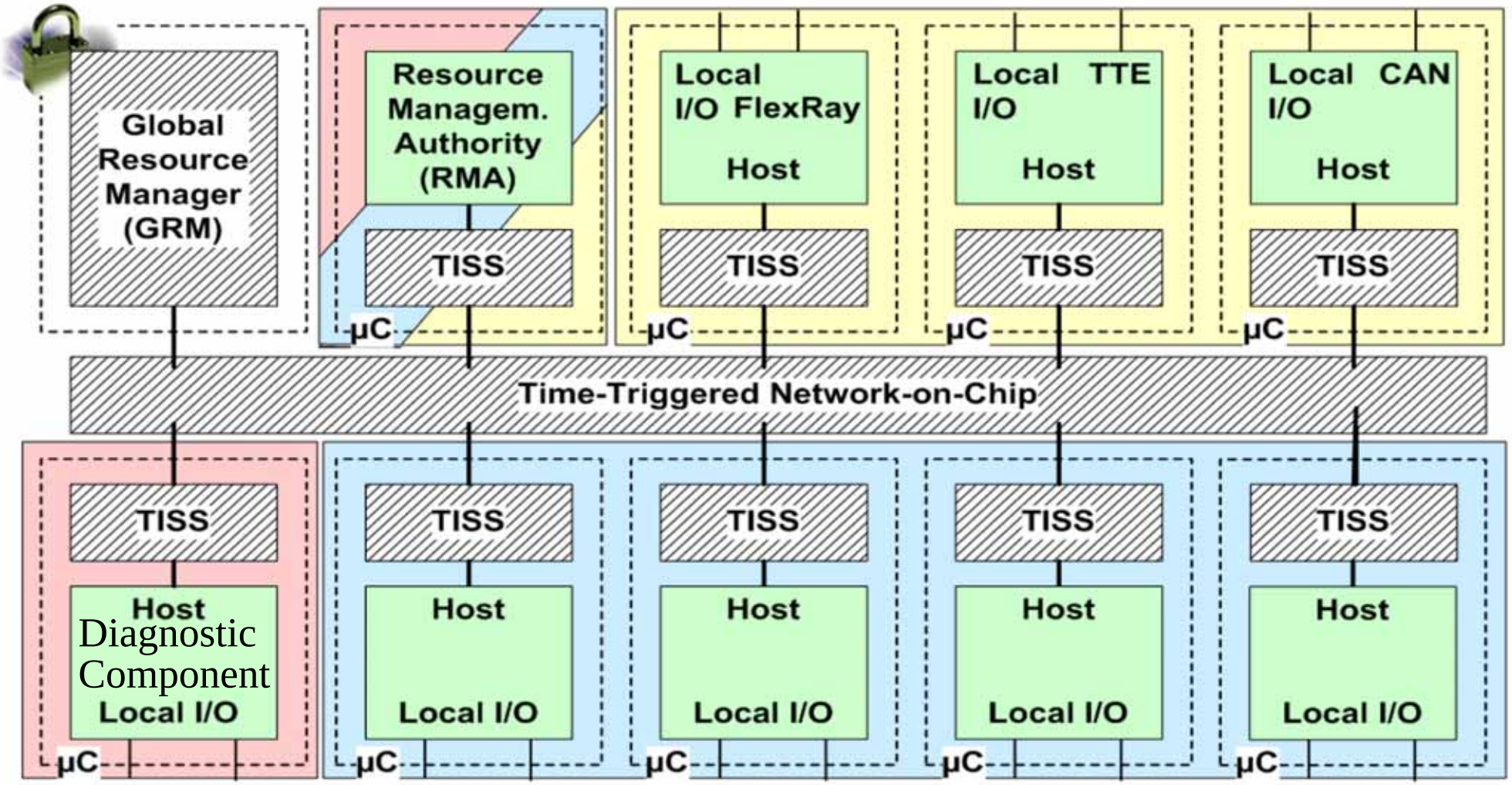
# The World of States is not *Black* and *White*



# Contribution of the TTA

- ✓ The (external) behavior of a TTA component consists of the messages a component exchanges with its environment.
- ✓ The availability of the global time makes it possible to timestamp every anomaly precisely.
- ✓ The basic core communication service is multicast, such that any message can be observed by an independent observer without the *probe effect*.
- ✓ At the chip level, a dedicated diagnostic component can look at all relevant messages.
- ✓ Every component is designed to output its ground-state periodically in order that it can be evaluated.

# Chip Level Prototype of the TTA



# Six Steps to Take

(1) Identify and design the planned field-replaceable units with appropriate properties (FRU).

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

(5) Replace the faulty FRU

(6) Restart the replaced FRU

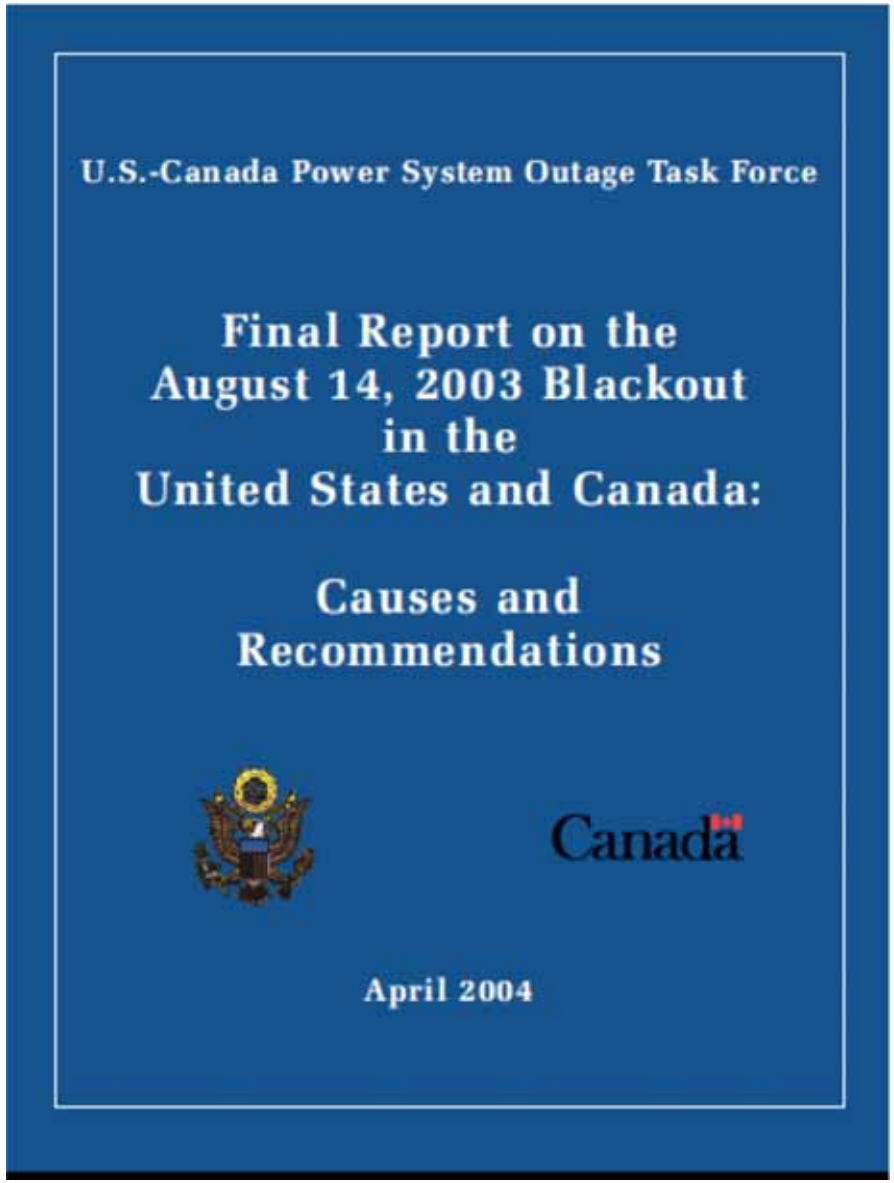
# (4) Analyze the Anomalies

- The collected information about the anomalies must be analyzed either on-line or off-line in a diagnostic subsystem.
- The establishment of a potential causality link between events should be supported.
- Failures in the diagnostic subsystem should not propagate into the operational subsystem.
- *Reproducibility* of failures needed.

# Example (Causality): *August 14, 2003 Power Outage*[31]

On August 14, 2003 a power-outage caused a blackout in in the North-Eastern United States and Canada.

This power outage has been examined and documented in an extensive report.



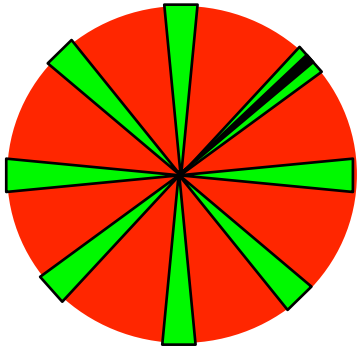
# On the *US-Canada Power Outage*, August 14, 2003



*A valuable lesson from the August 14 blackout is the importance of having time-synchronized system data recorders. The Task Force's investigators* ***labored over thousands of data items to determine the sequence of events, much like putting together small pieces of a very large puzzle****. That process would have been significantly faster and easier if there had been wider use of synchronized data recording devices.*

From the *Final Report on the August 14, 2003 Blackout* from the *US-Canada Power Outage Task Force*, p. 162 (bold added).

# Models of Time in the Cyber World

**Dense**

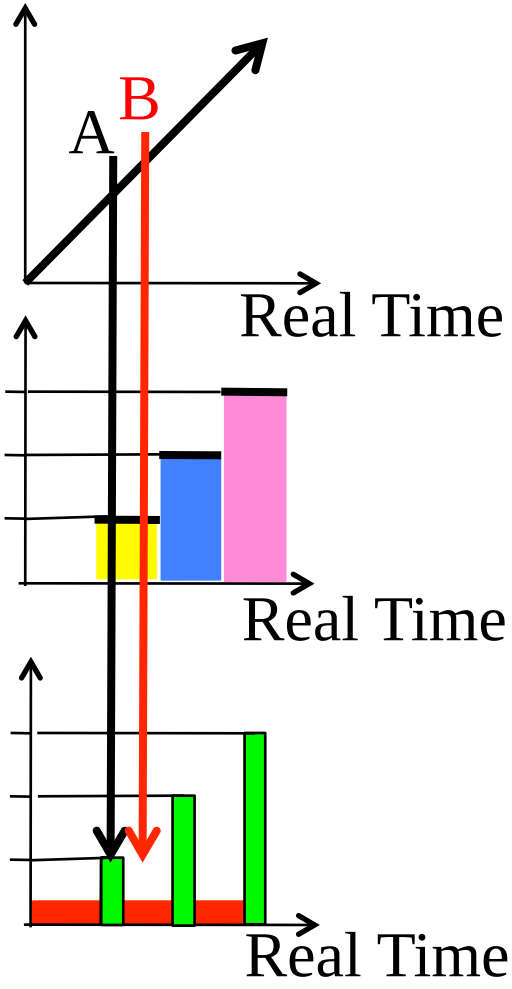
Physics

**Discrete**

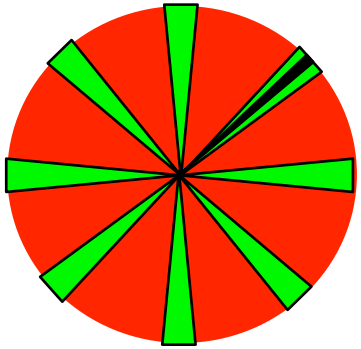
Central Computer

**Sparse**

Distributed Computer



# Models of Time in the Cyber World

**Dense**

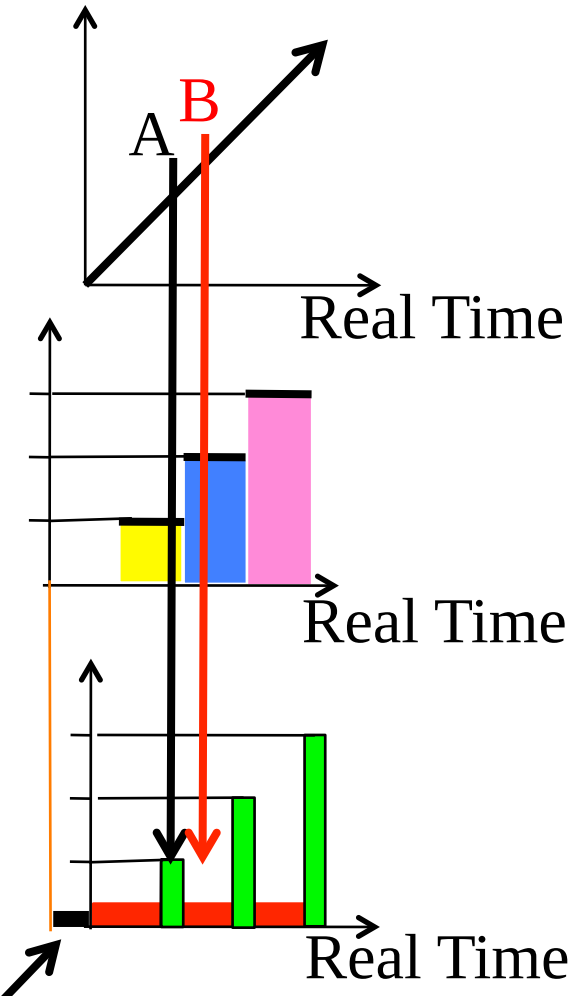
Physics

**Discrete**

Central Computer

**Sparse**

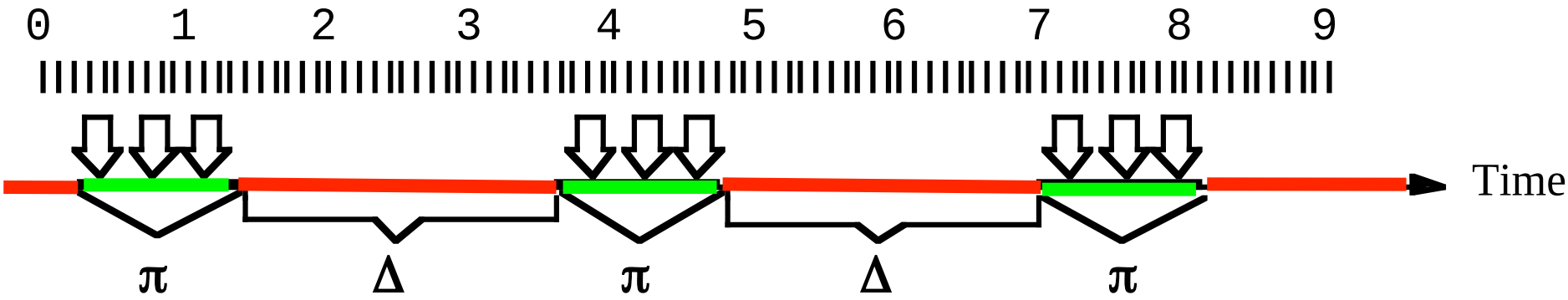
Distributed Computer




Precision of the *Global Time*

# Fault-Tolerant Sparse Time Base

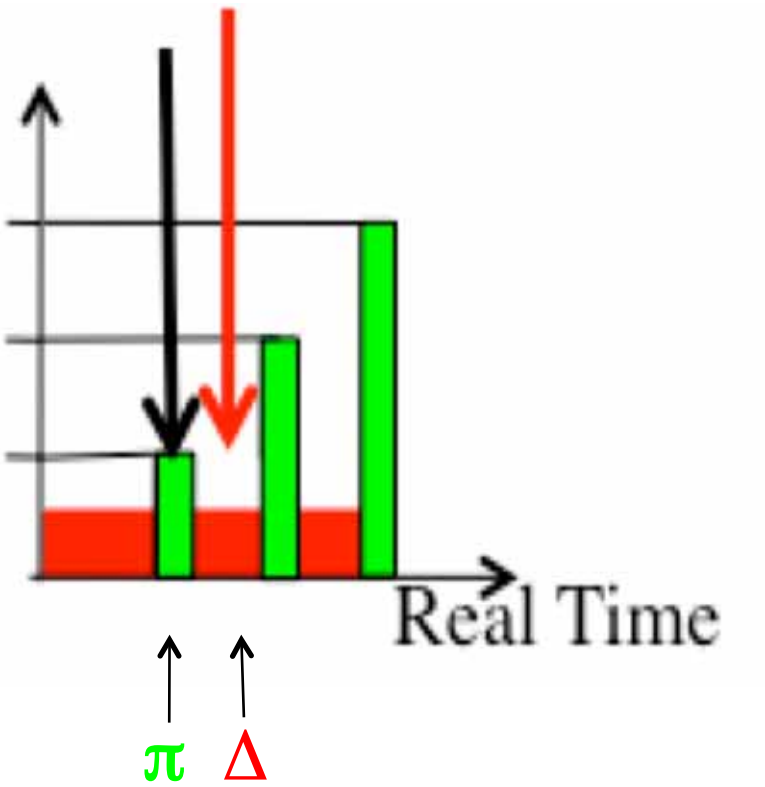
If the occurrence of events is restricted to some active intervals with duration $\pi$ with an interval of silence of duration $\Delta$ between any two active intervals, then we call the time base $\pi/\Delta$-sparse, or sparse for short.




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

Establish a static order among all involved partners to resolve simultaneity.

# The Intervals π and Δ in a *Sparse* Timebase




- Depend on the precision P of the clock synchronization.
- In reality, the precision is always larger than zero—in a distributed system clocks can never be fully synchronized.
- The precision depends on the stability of the oscillator, the length of the resynchronization interval and the accuracy of interval measurement.
- On a discrete time-base, there is always the possibility that the same external event will be observed by a tick difference.

# Contribution of the TTA

- ✓ Establishment of potential causality is supported by the sparse time base of the TTA—*determinism* helps.
- ✓ The unidirectional core communication service separates the diagnostic subsystem from the operational subsystem.
- ✓ The analysis of the anomalies is performed in an independent diagnostic component (an FCU) of the TTA, with no low-level feedback to the operation
- ✓ Diagnostic components can be supported at different levels of the architecture (e.g., chip level, device level).
- ✓ It is left up to the system designer to decide, whether a high level feedback (reconfiguration) of the results of the diagnosis is automatic or manual.

# Six Steps to Take

(1) Identify and design the planned field-replaceable units with appropriate properties (FRU).

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

(5) Replace the faulty FRU

(6) Restart the replaced FRU

# (5) Replace the Faulty FRU

- The physical replacement of a faulty FRU is a problem of mechanical and electrical design.
- High availability applications require replicated components to support the on-line replacement of FRUs.

# Contribution of the TTA: Reconfiguration

Dynamic reconfiguration poses special challenges:

- ✓ Failures of the reconfiguration system are of utmost criticality, because a correct configuration can be destroyed.
- ✓ The TTA enables the implementation of a *fault-tolerant (TMR)* reconfiguration architecture.
- ✓ Downloading new software versions dynamically requires an appropriate secure infrastructure, built on top of the TTA.

# Six Steps to Take

(1) Identify and design the planned field-replaceable units with appropriate properties (FRU).

(2) Characterize the FRUs depending on their expected failure rate.

(3) Observe the behavior of FRUs and collect data about *anomalous behavior*.

(4) Analyze the data about anomalous behavior, either online or offline to locate faulty FRUs

(5) Replace the faulty FRU

(6) Restart the replaced FRU

# (6) Restart the Faulty FRU

- Before a component can continue its service, the relevant ground state at the next reintegration instant must be loaded.
- Backward recovery to a previous state is not reasonable in a real-time environment.

# Contribution of the TTA

- ✓ The TTA requires that a periodic reintegration point (the ground cycle) is designed into the behavior of a component and published at the TII interface.
- ✓ The ground state (g-state) at the reintegration point can be captured periodically by a diagnostic component that is an independent FCU.
- ✓ A component can be reset and restarted by the diagnostic component with a restart message that contains an estimation of the relevant g-state at the next reintegration instant.

# Contribution of the TTA

- ✓ In a fault-tolerant configuration, the ground state is *voted into* a new component autonomously.
- ✓ In non fault-tolerant configurations, state estimation of the ground-state that is relevant at the next reintegration instant must be performed by the diagnostic component.

# Conclusion

The TTA supports diagnosis and maintainability by

- its clear component concept that forms a strong basis for the diagnosis.
- the non-intrusive observeability of component behavior
- the provision of error containment boundaries that limit the propagation of errors from faulty components.
- a sparse global time base that support the causal analysis of events in a distributed application.