GUARDS Architecture Overview

GUARDS

- **Generic Upgradable Architecture for Real-time Dependable Systems**
- The definition of an architecture and its development and validation environment
  - To meet a wide spectrum of dependability and real-time requirements
- Developed in the context of the European Esprit Project 20716 GUARDS
- **Partners:**
- Technicatome, Aix-en-Provence (France),
- Ansaldo Transporti, Genova (Italy),
- Matra Marconi Space France, Toulouse (France),
- Intecs Sistemi, Pisa (Italy),
- Siemens AG Osterreich PSE, Wien (Austria),
- LAAS-CNRS, Toulouse (France),
- PDCC, Pisa (Italy), (IEI/CNR and CNUCE/CNR),
- University of York (UK).

Key Non-Functional Requirements

- **Railway Applications** (fail-safe control system)
  - Catastrophic failure rate (wrt HW faults) < 10^{-13} / hour
  - Safe shutdown rate < 10^{-9} / hour
  - Single instances support both vital and non-vital functions
- **Nuclear Submarine Applications** (secondary protection functions)
  - Full test only once per year: Pr {unrevealed dormant fault} < 10^{-4}
  - Physical segregation of redundant hardware
  - High degree of evolutivity due to long deployment time
  - Must use unmodified COTS operating system(s)
- **Space Applications** (autonomous spacecraft with critical phases)
  - Reconfiguration to optimize phases
  - 15-year mission reliability = 0.985
  - Payload is unreliable

Considered fault classes

- permanent internal physical faults
  - requires physical redundancy
- permanent external physical faults (damage)
  - requires physical separation of redundancy
- temporary external physical faults (transients)
  - requires state restoration to avoid redundancy attrition
- temporary internal physical faults (intermittents)
  - requires filtering to decide type of fault treatment (permanent or transient)
- permanent design faults (“Bohrbugs”)
  - requires diversification of design or of specification
- temporary design faults (“Heisenbugs”)
  - requires at least diversification of activation conditions
- design faults in non-critical software ?
  - Not tolerated but requires confinement of effects

Real-Time Models

- Genericity towards supporting a range of real-time computational and scheduling models
- Computational model: defines the form of concurrency and any restriction that must be imposed to application programs to facilitate their timing analysis (e.g., bounded recursion)
  - Applications supported by GUARDS may conform to time-triggered, event-triggered or mixed computational models
- Scheduling models:
  - Cyclic, as typified by the traditional cyclic executive
  - Cooperative, where an application-defined scheduler and the prioritized application tasks explicitly pass control between one another to perform the required dispatching
  - Preemptive priority scheme - the most flexible

Real-Time

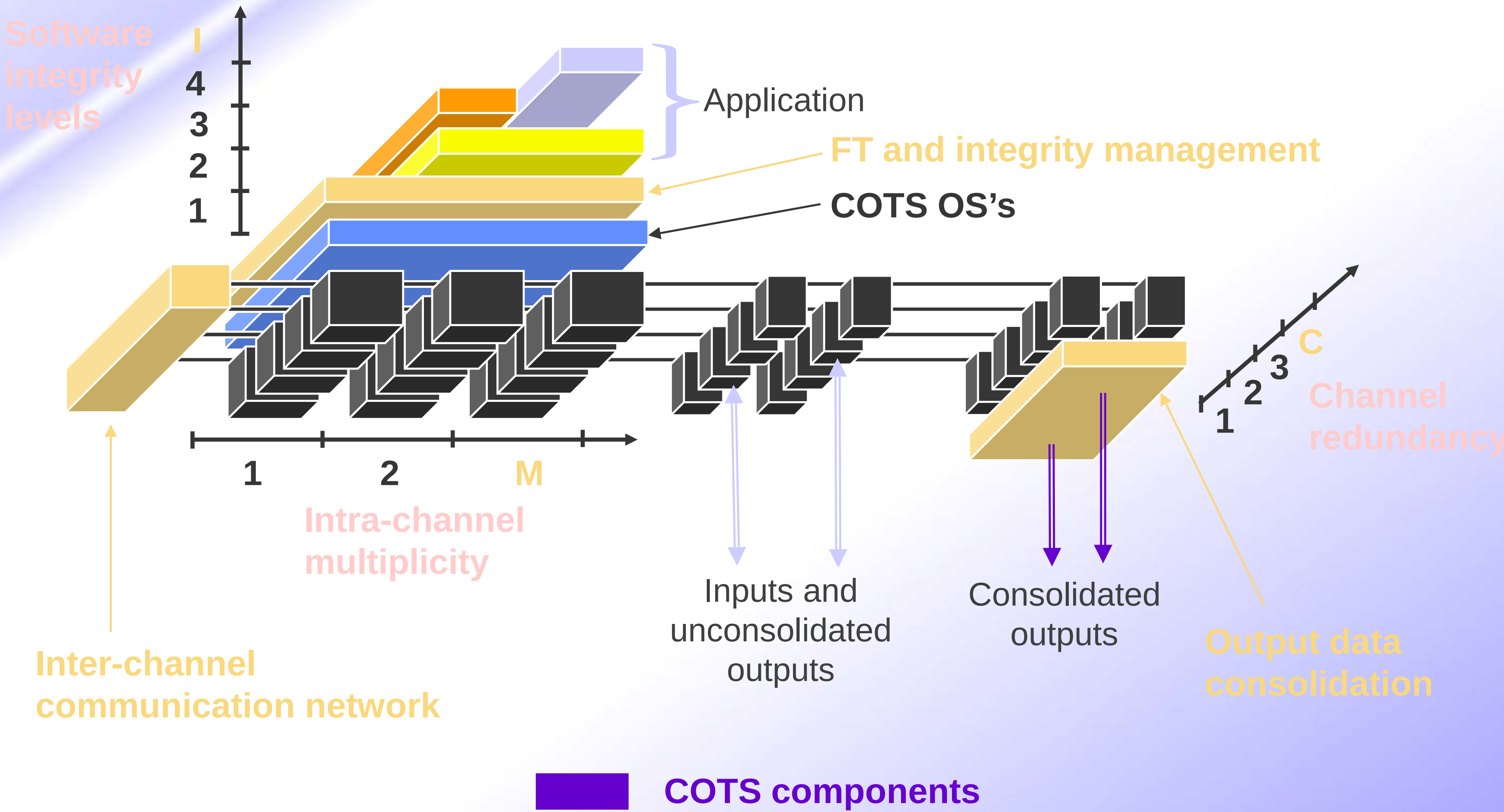
Computational model	Function release	Cyclic	Cooperative	Pre-emptive
		Scheduling model		
Time-triggered	Periodic	By construction	Response time analysis	Rate monotonic analysis
Event-triggered	Sporadic	N/R	Response time analysis	Response time analysis
Mixed	Periodic & sporadic	N/R	Response time analysis	Response time analysis
		Timing analysis		

- Choice depends on:
  - performance
  - certification constraints
  - maintainability

The Generic Architecture

- Defined along three dimensions of fault containment:
  - Integrity Levels, or design-fault containment regions
  - Lanes, or secondary physical-fault containment regions
  - Channels, or primary physical-fault containment regions
- An instance of the architecture is characterized by:
  - the dimensional parameters {C, M, I}
  - A reconfiguration strategy
  - An appropriate selection of generic hw and sw GUARDS components implementing
    - Interchannel communication
    - Output data consolidation
    - Fault tolerance and integrity management

A Generic Architecture



The Lane Dimension

- Multiple processors or lanes can be used to:
  - Improve the capabilities for fault diagnosis within a channel (e.g., by comparison)
  - Improving coverage wrt design faults by using interchannel diversification
  - Improve the availability of a channel
  - Improve performance by parallel processing
  - Isolation of software of different integrity levels

The Channel Dimension

- Channels provide the primary fault containment regions for physical faults that affect a single channel
- Fault tolerance is based on active replication --> it must be ensured that replicas get the same inputs in the same order
- Interesting configurations:
  - C=2: self-checking pair (or duplex if exist intra-channel test or diagnosis)
  - C=3: TMR for masking of one quasi-arbitrary fault
  - C=4: masking of one arbitrary fault or to allow off-line channel testing

Multi-Level Integrity

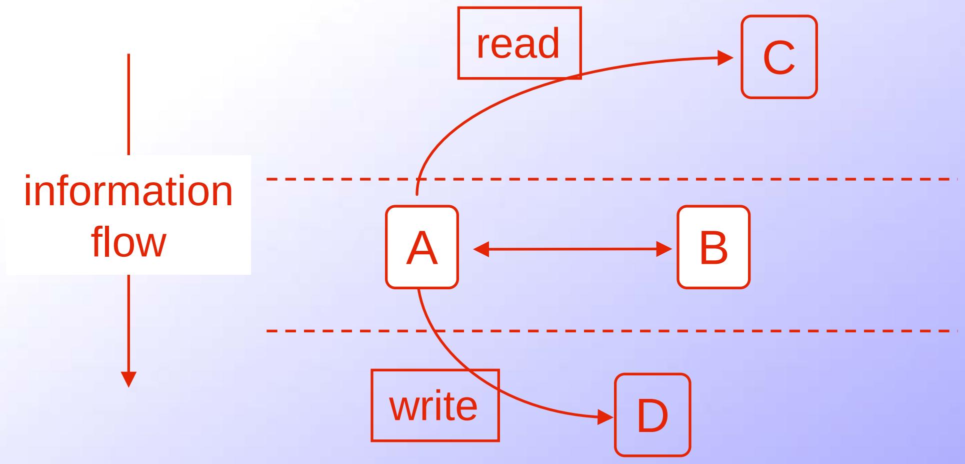
Objective

To allow software of different criticality to share common resources

- **criticality is linked to consequences of potential failures**
- **non-critical software may contain bugs**

Means

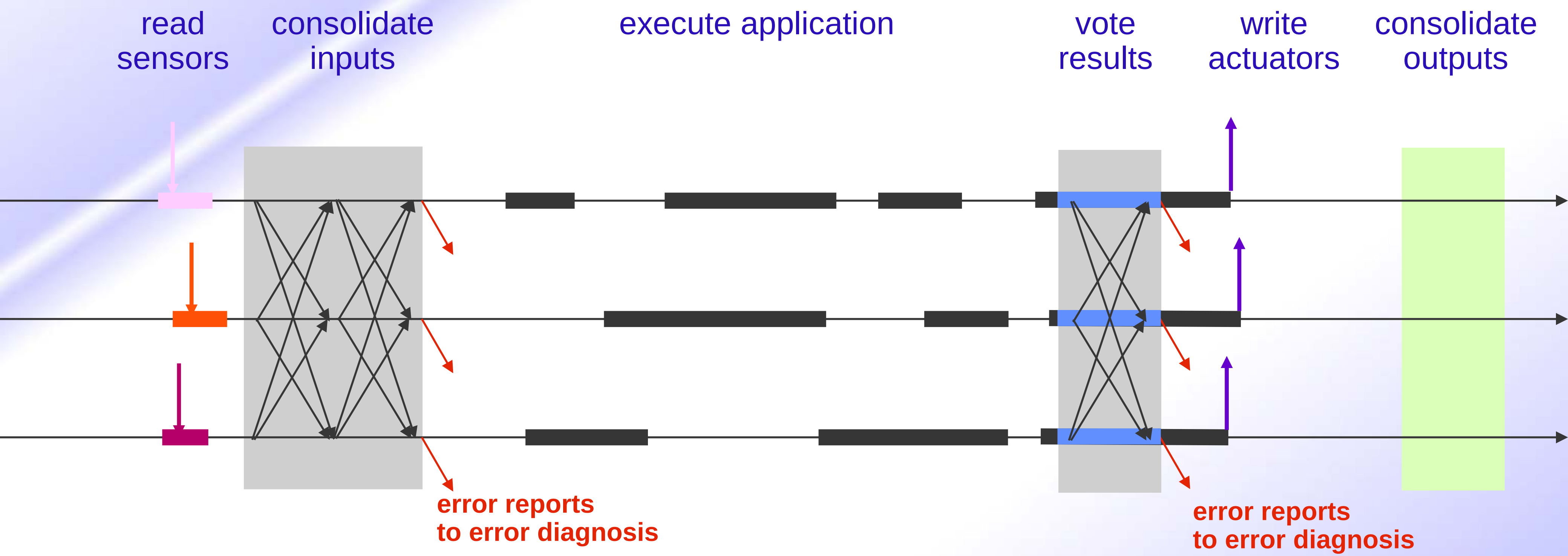
- Isolation by Firewalls
  - Spatial
    - prevent illegal memory access
    - supported by a Memory Management Unit (MMU)
  - Temporal
    - prevent resource hogging
    - supported by Budget Timers and Watchdogs
- Mediation according to a Biba-like Integrity Policy



Interchannel Error Processing

- active replication (or diversification)
  - C=2 : self-checking pair (or duplex if ∃ intra-channel test or diagnosis)
  - C=3 : masking of one quasi-arbitrary fault
  - C=4 : masking of one arbitrary fault or to allow off-line channel testing
- primarily by N-modular redundancy to detect disagreeing channels
- Reading the replicated sensors
- Input values consolidated across all the channels through the interactive consistency algorithm
- Application tasks executed asynchronously -> some diversification in the execution allows many residual design faults to be tolerated as if they were intermittents
- it is application transparent and managed by software

Inter-Channel Error Processing



- **replica scheduling uncertainty:**
  - sensor reading and result voting: split threads
  - shared data: timestamp mechanism

Inter-Channel Error Diagnosis

- collect error reports from inter- and intra-channel error detection mechanisms
- α-count filtering mechanism:
  - if channel *i* perceives channel *j* as faulty
    then $\alpha_i(j) = \alpha_i(j) + 1$
    else $\alpha_i(j) = k*\alpha_i(j)$ $(0<k<1)$
  - consolidate α-counts from each channel => $\alpha(j)$
- case:
  - ($\forall j, \alpha(j) <= threshold\_1$) <no damage> $(threshold\_1 > threshold\_2)$
  - do nothing
  - ($\exists j, \alpha(j) > threshold\_1$ & $\forall i°j, \alpha(i) <= threshold\_2$) <single channel damage>
    attempt restoration of channel *j*
  - otherwise <multiple channel damage>
  - application-specific forward
    recovery (e.g., switch to safe state)

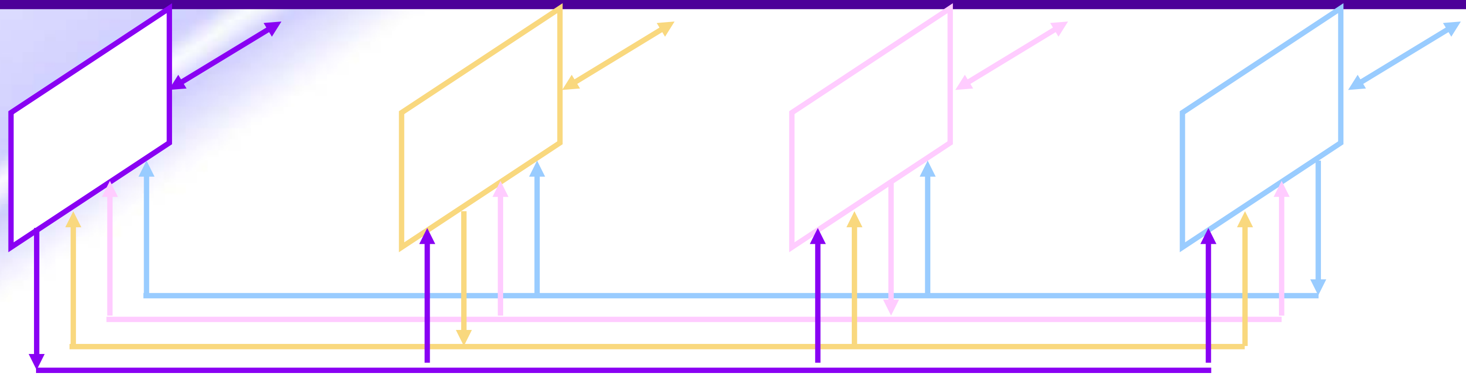
Inter-Channel Fault Treatment

- isolate channel (disconnect from outside world)
- reset channel, reinitialize operating system structures
- carry out channel self-test
  - if successful
    then <fault was soft>
    else <fault is hard> switch off channel
- channel re-integration (after soft fault or repaired hard fault)
  - join pool of channels (clock resynchronization)
  - ask to enter “running context transfer” mode
    - maintain (small but vital) application running
    - modifications to global variables are propagated to all channels
    - execute sweep and copy to transfer global variables from non-faulty channels
  - NB: requires explicit identification of a “context object” encapsulating all global variables

Inter-channel Communication Network

- Functions of the ICN:
  - it provides a global clock to all the channels
  - it allows channels to achieve consensus on non-replicated data
- ICN consists of:
  - an ICN-manager for each channel
  - unidirectional serial links to interconnect ICN-managers
- Clock synchronization
  - Each node has a physical clock, and computes a global logical clock through a fault tolerant synchronization algorithm satisfying the agreement condition, i.e., the skew between any non-faulty logical clocks is bounded, and the accuracy condition, i.e. all non-faulty logical clocks have a bounded drift wrt real-time
- Interactive Consistency
  - The exchange of private data among channels and agreeing on a common value in the presence of arbitrary faults with the properties of:
    - Agreement - p and q non faulty agree on the same value for r
    - Validity - p non faulty agrees on the actual private value of non-faulty q

Inter-Channel Communication Network (ICN)



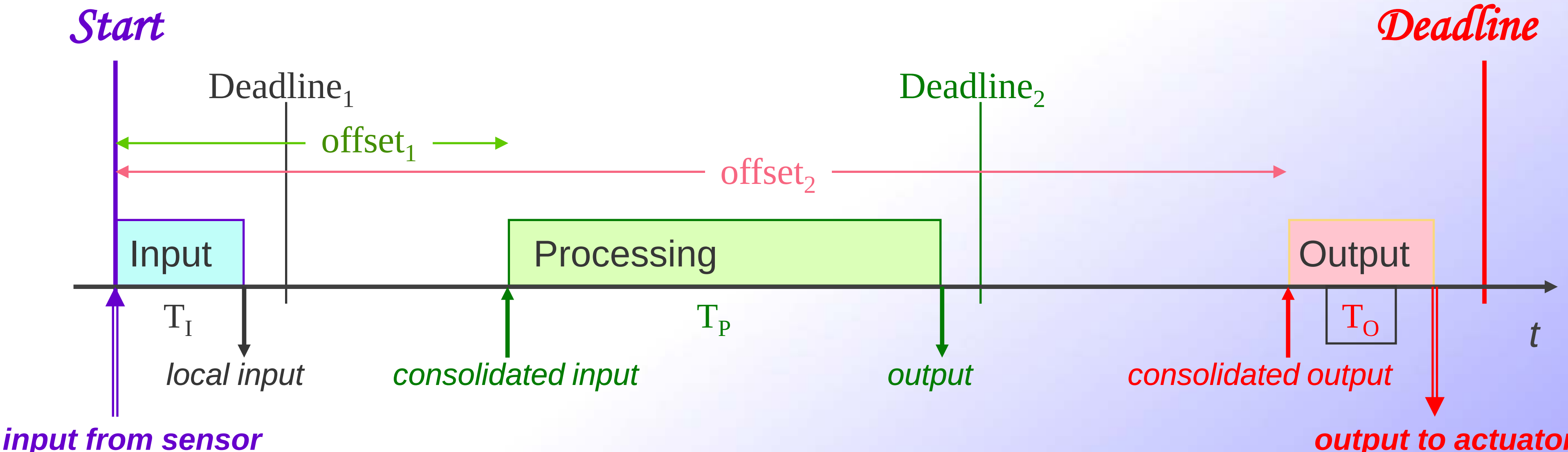
- Clock synchronization
- Convergence averaging algorithm [Lundelius-Welch & Lynch 1988]
- but convergence function depends on number of active channels (n)
- Tolerance:
  - n=4: arbitrary fault
  - n=3: quasi-arbitrary fault
  - n=2: must assume crash faults or just detect by comparison

Interactive consistency algorithm

- ZA algorithm (hybrid fault model) [Gong *et al.* 1995]
- Relies on cryptographic checksum to authentify relayed messages
- Tolerance:
  - n=4: arbitrary fault or 2 non-arbitrary
  - n=3: arbitrary fault
  - n=2: must assume crash faults or just detect by comparison

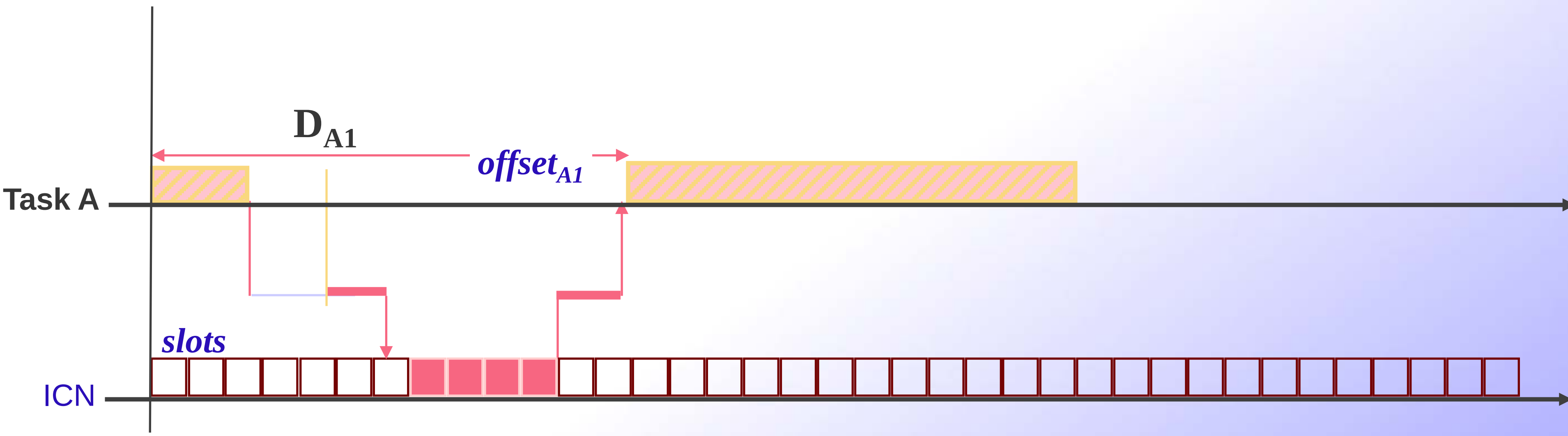
Inter-Channel Schedulability model

- Consolidation of input and output values
  - Threads split in three parts (or transactions consist of three threads):
    - The input value is acquired on each channel
    - The "local" input values are consolidated (by interactive consistency)
    - The ouput values are calculated on each channel
    - The calculated output values are consolidated (by majority voting)
    - The consolidated output is actually produced



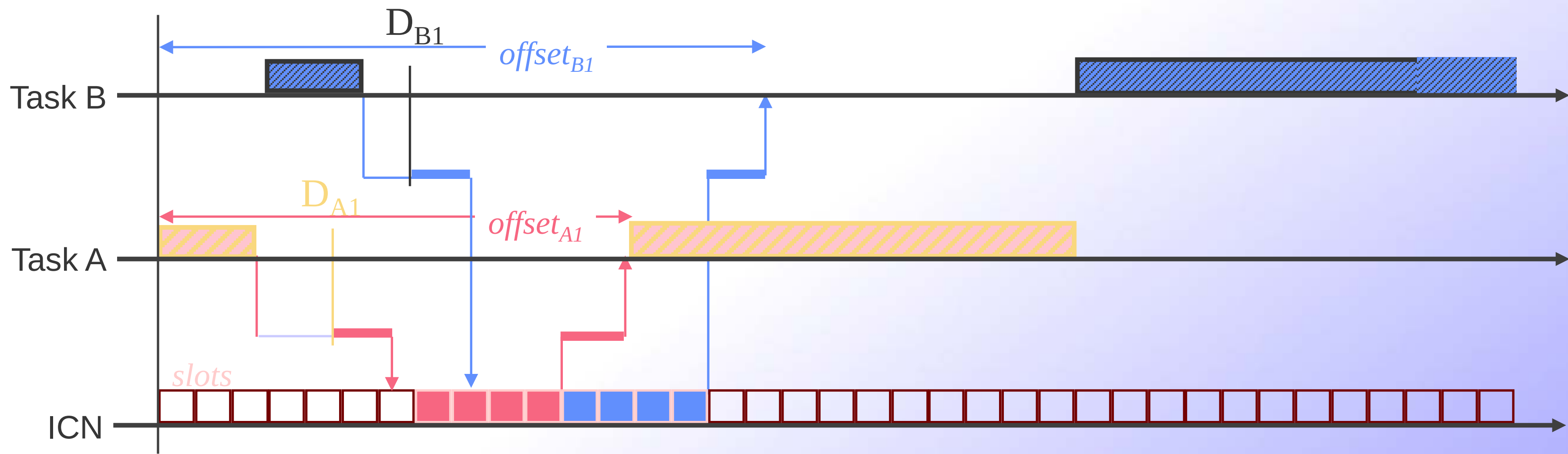
Inter-Channel Schedulability-1

- *Deadline* is set by the requirements and defines the time by which the final value has to be sent to the actuator
- Intermediate Deadlines$_i$ are introduced by the design and define the time by which the value is ready for being transferred through the ICN

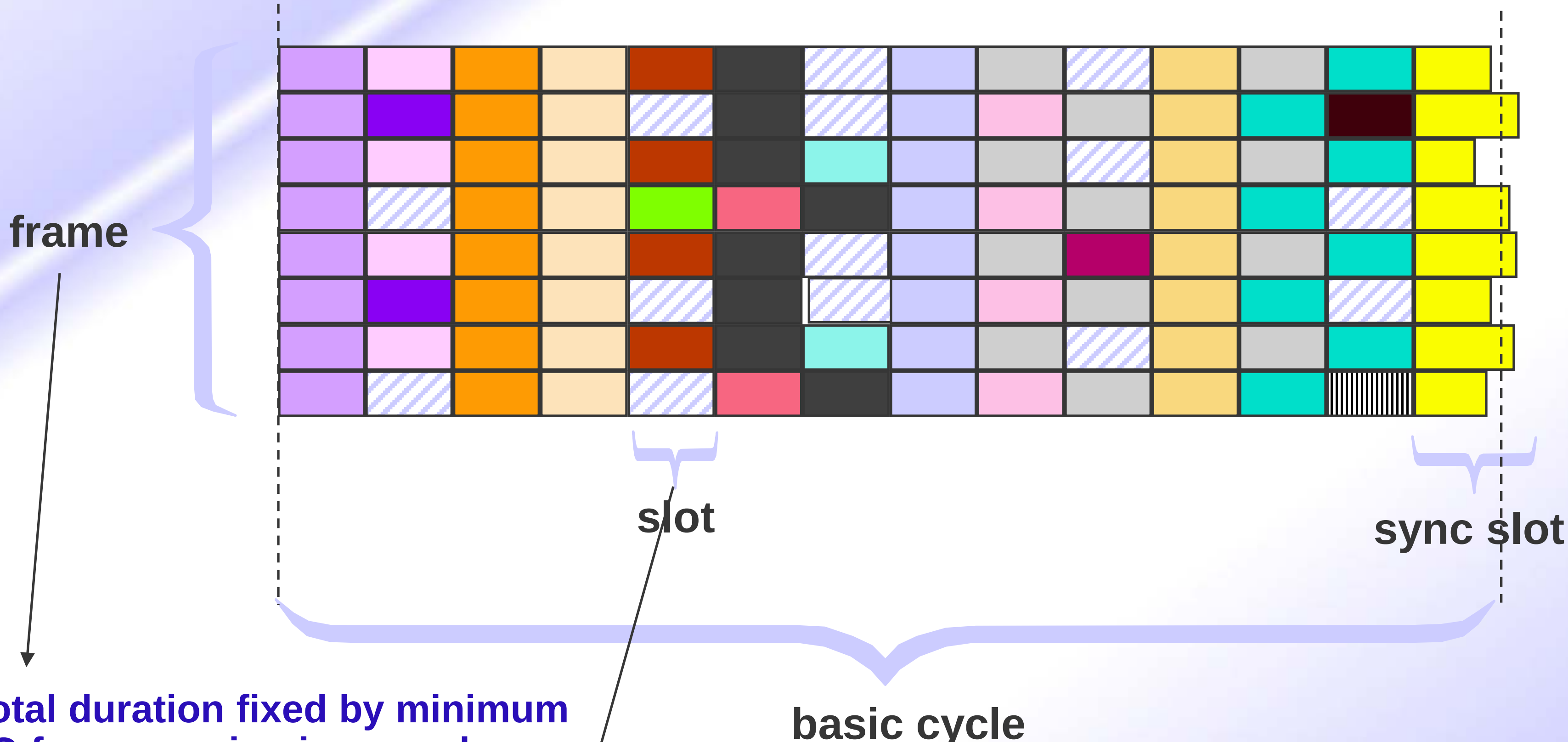


Inter-Channel Schedulability-2

- **Deadline** is set by the requirements and defines the time by which the final value has to be sent to the actuator
- Intermediate Deadlines$_i$ are introduced by the design and define the time by which the value is ready for being transferred through the ICN



ICN Scheduling



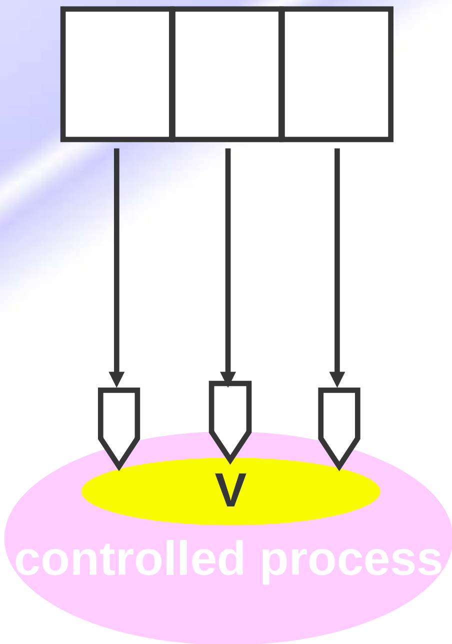
Total duration fixed by minimum I/O frequency in given mode

Time window for 1 message transmission (and up to 3 message receptions)

Duration fixed by resynchronization interval or by maximum I/O frequency in given mode

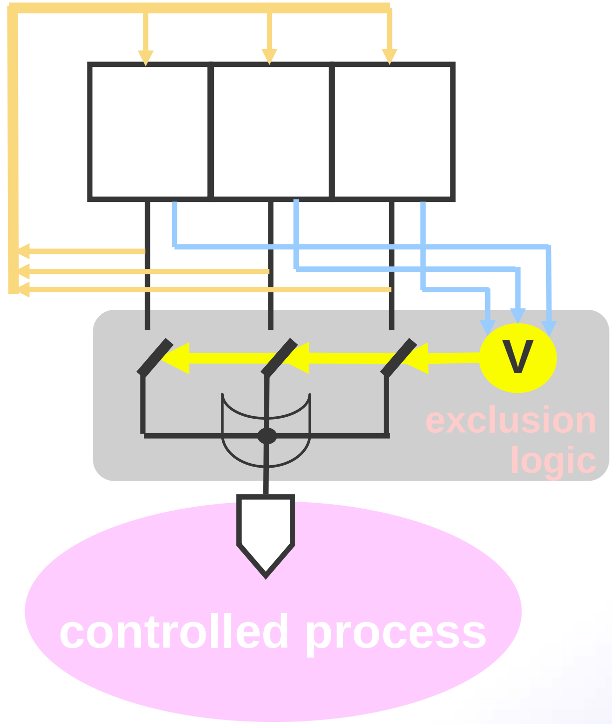
Output Consolidation

application-specific consolidation



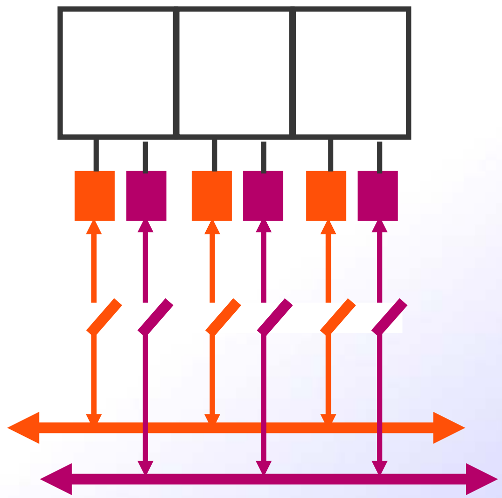
- relay network
- voting actuators
- arm and fire
- actuator state readback

discrete I/O



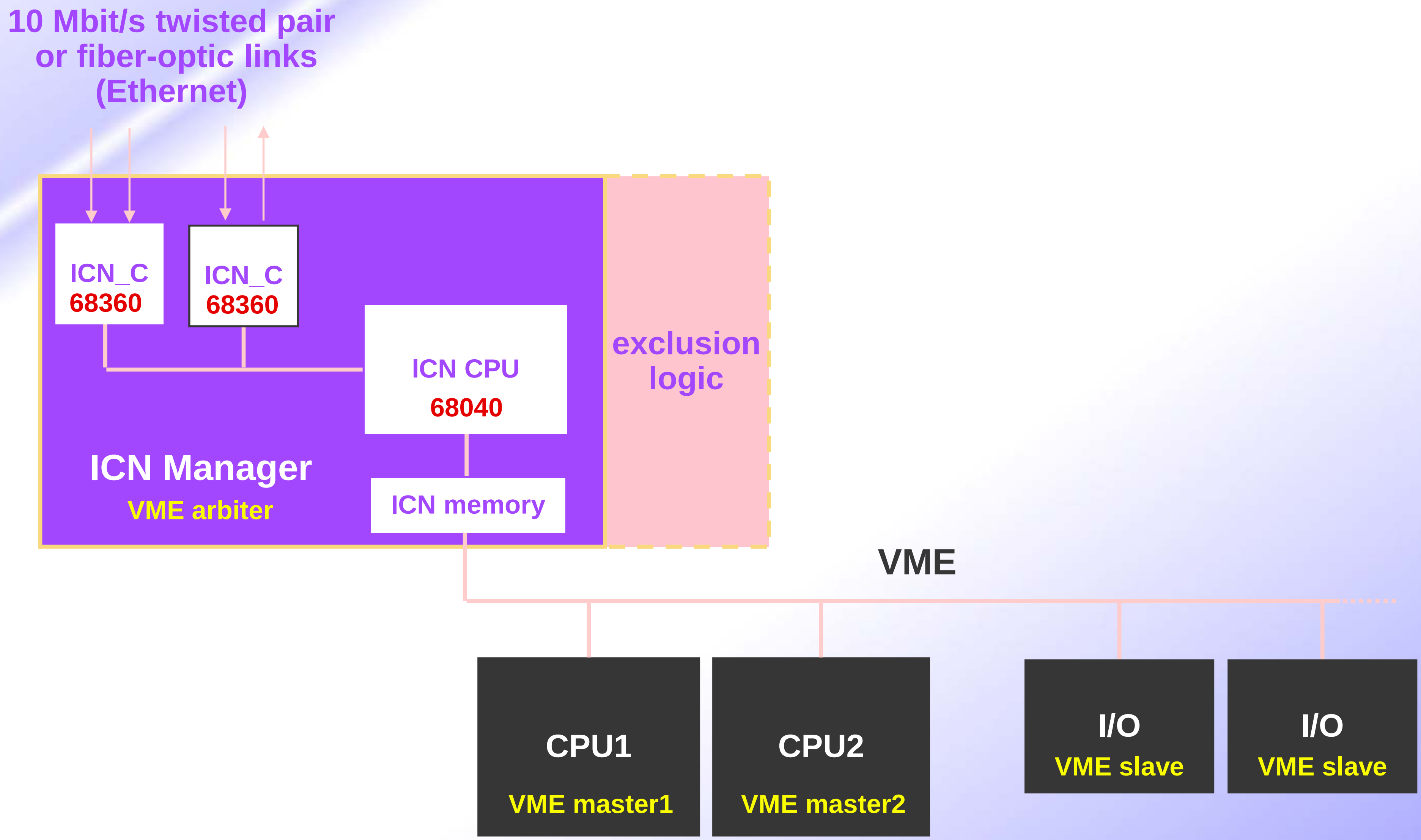
- intra-channel readback
- inter-channel readback
  - serial link
  - discrete wires

networked I/O



- one network per channel
- pre-adaptor voting
  - dependable network
- master channel
  - SW or HW exclusion logic
- multiplexed channels
  - remote voting

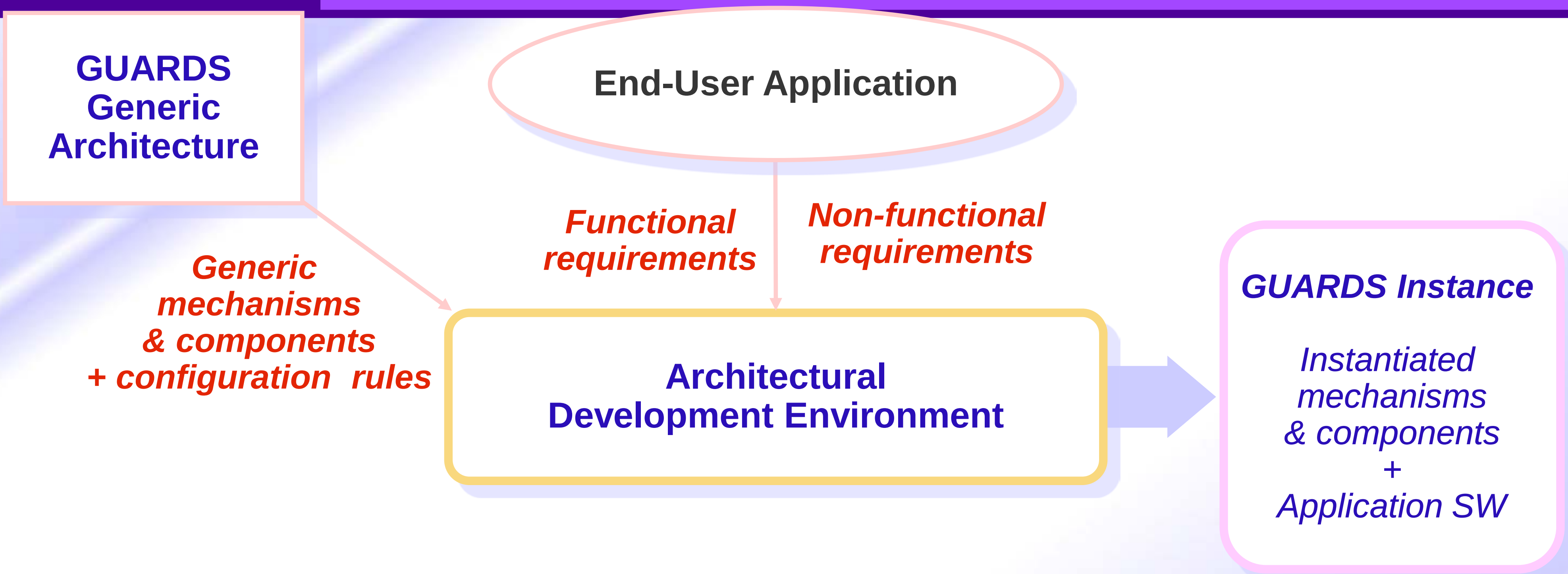
Channel Hardware Architecture



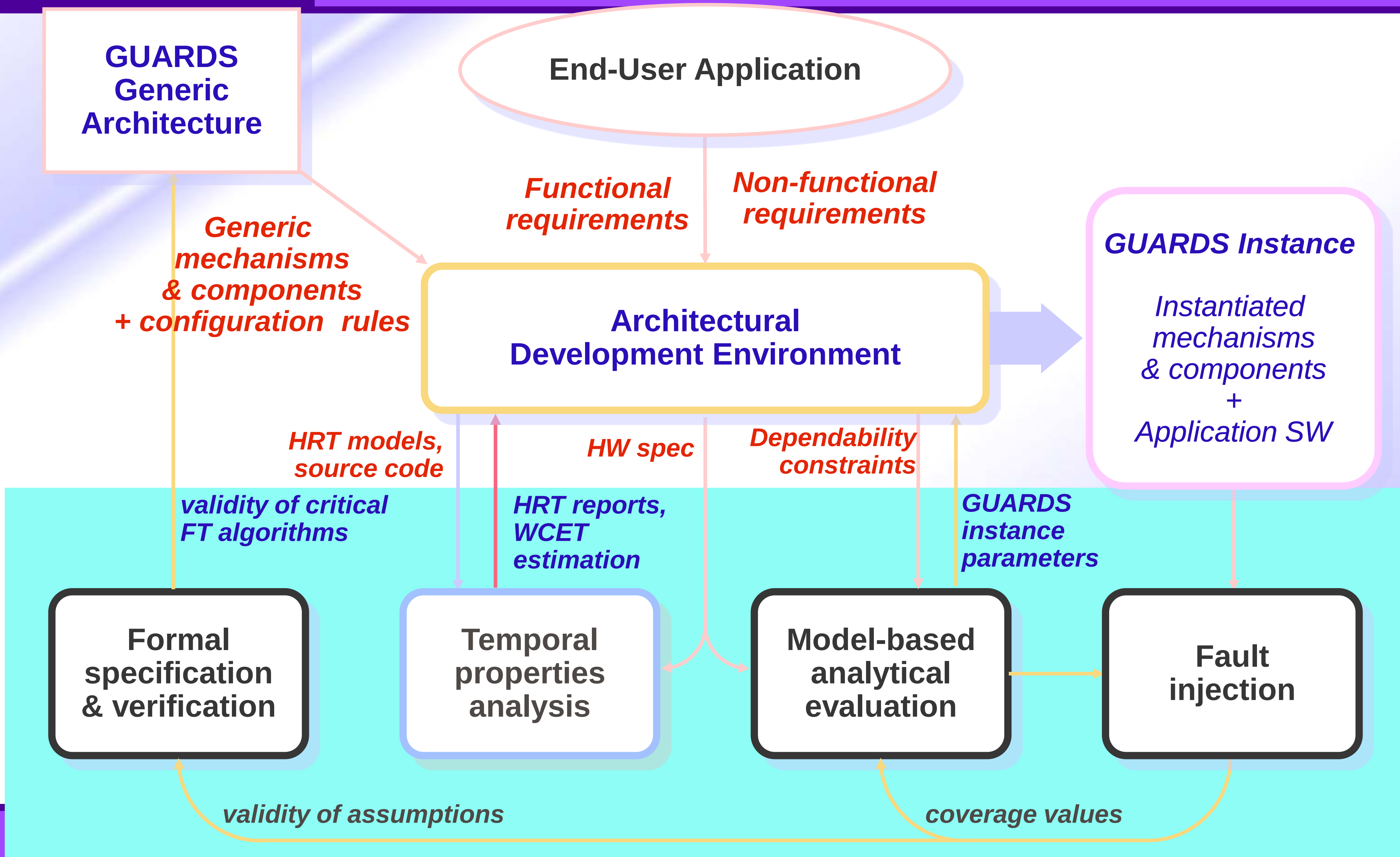
Validation

- Objectives of the validation process:
  - Validation of the design principles of the generic architecture, including both real-time and dependability mechanisms
  - Validation of the development of instances of the architecture implementing specific end-user requirements
- Compound usage of:
  - Formal verification - mainly applied to validate critical FT algorithms (clock synchronization, interactive consistency, ..)
  - Model-based analytical evaluation - applied both to validate generic dependability mechanisms and to configure instances meeting specific application dependability requirements
  - Fault injection - carried out on prototypes, to provide means for:
    - I) assessing the validity of the necessary assumptions made by formal verification;
    - ii) estimating the coverage parameters included in the analytical models.

Validation Environment-1



Validation Environment-2

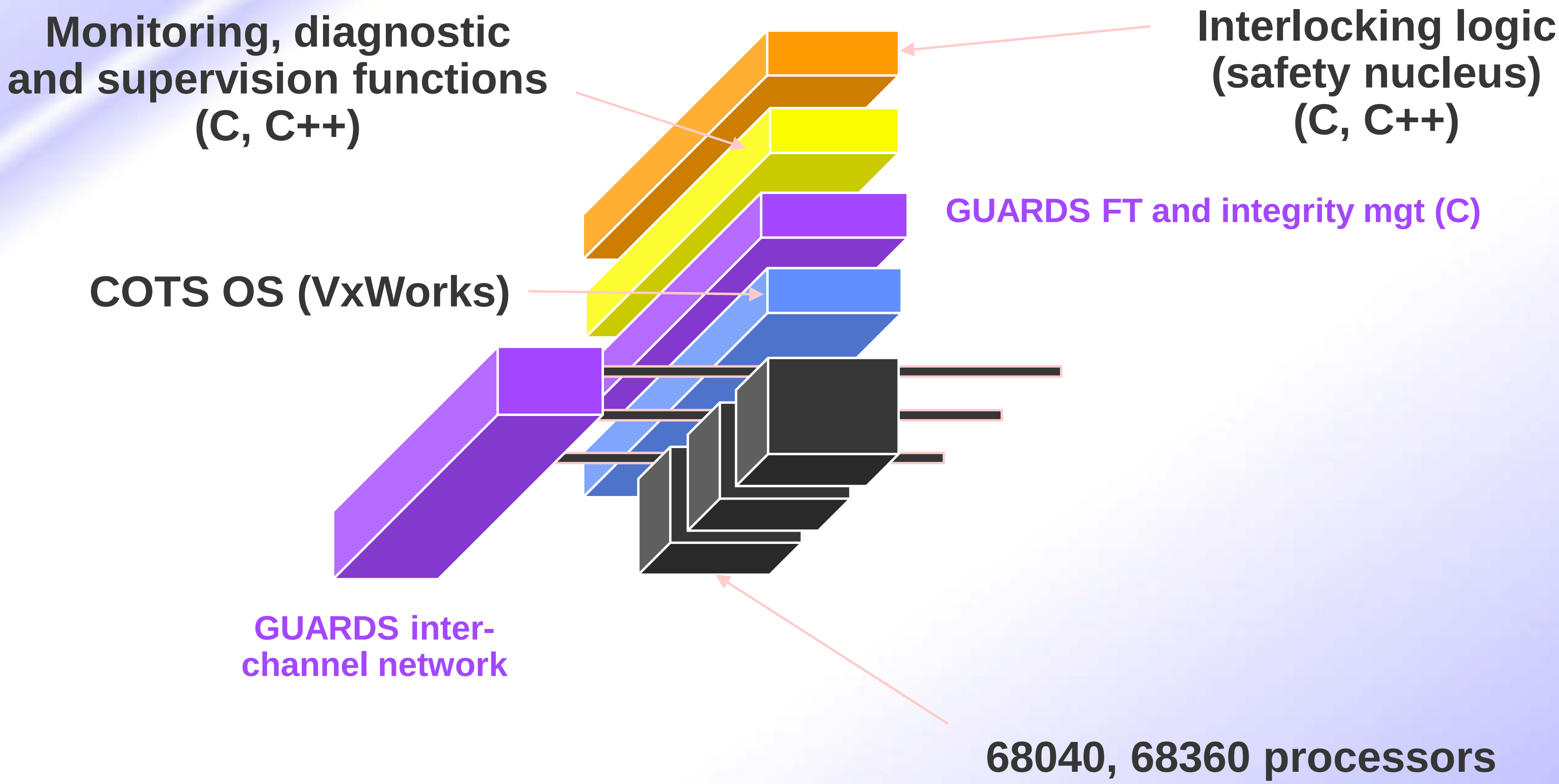


Railway Prototype

- TMR architecture, with 1 processor per channel - if a channel is diagnosed as permanently faulty, the system degrades to a 2-out-of-2 mode. A detected additional fault leads the system to a safe state.
- Novelty wrt current practice:
  - Coexistence of 2 levels of application sw integrity corresponding to very different degrees of criticality
    - Highly critical interlocking logic
    - Noncritical monitoring, diagnostic and supervision functions
- A second prototype has adopted a duplex fail-safe configuration

The Railway Prototype Instance

C=3 M=1 I=2

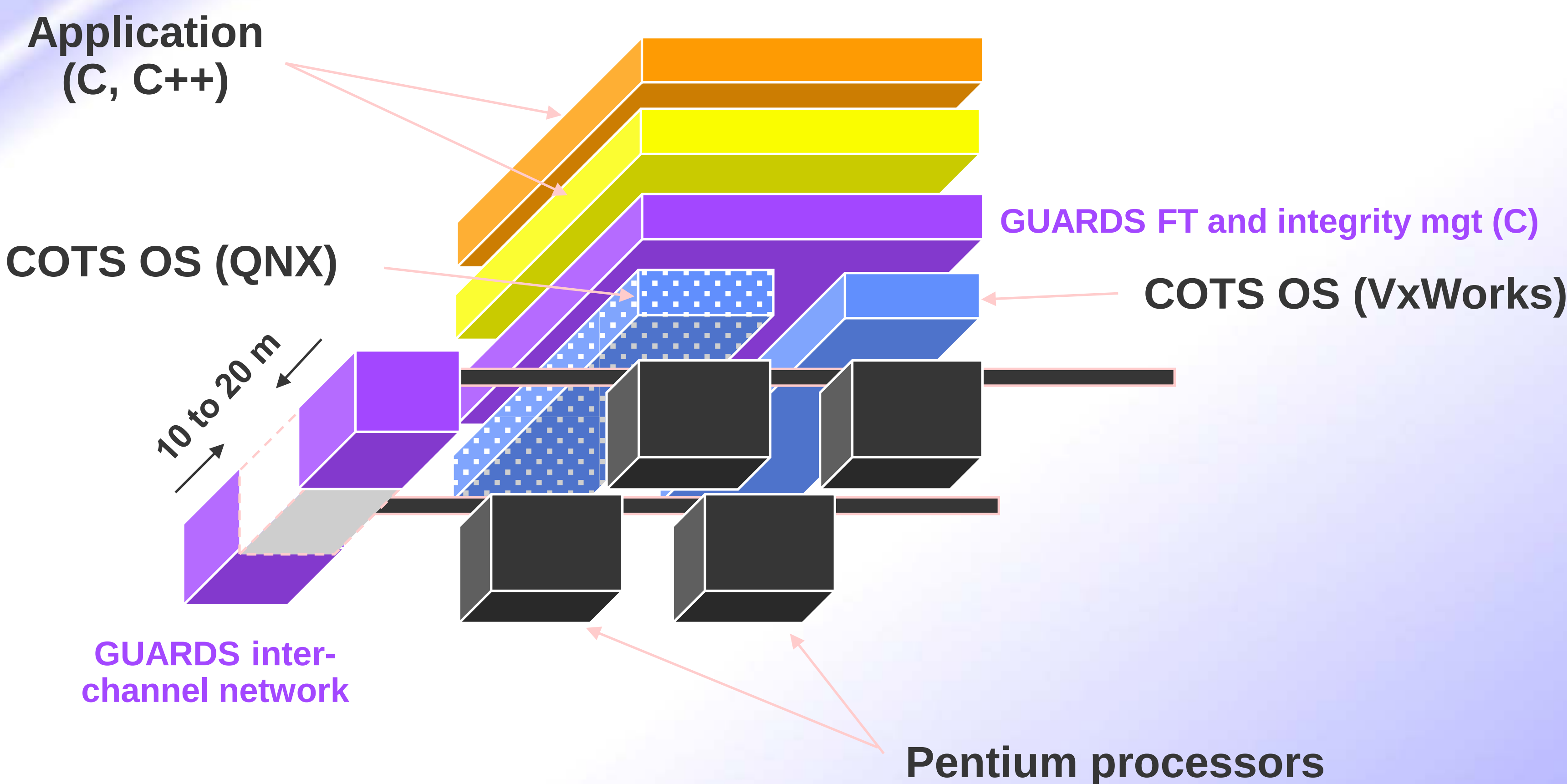


Nuclear Submarine Prototype

- The targeted nuclear submarine application is a secondary protection system
- Dual-channel architecture, with 2 processors for channel (a self-checking pair)
- Prevention of physical damage by geographical separation of the channels (several meters)
- Two levels of integrity
- Both channels operational --> 2-out-of-2 mode
  - Results produced by the 2 channels are compared if they are declared error-free by the intra-channel mechanisms
  - If the comparison between channels reveals disagreement, the instance is put into a safe state
  - If errors are detected inside a channel, that channel declares itself to be faulty and the instance switches to a single channel operation

The Nuclear Submarine Prototype Instance

C=2 M=2 I=2

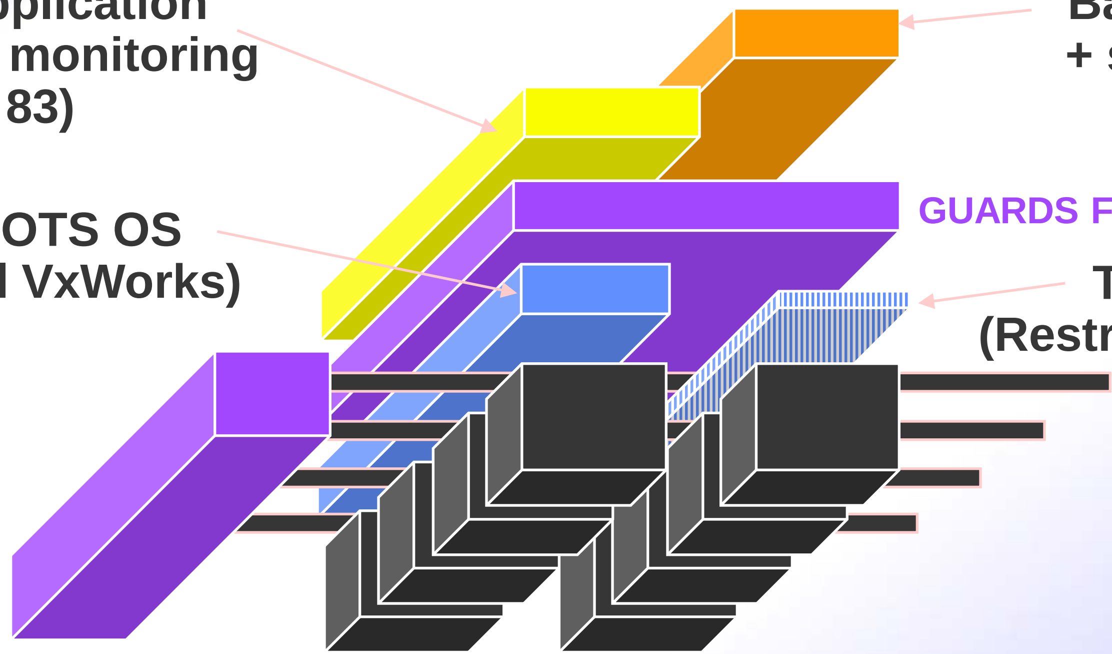


Space Prototype

- 4 channel instance, with possible degradation to 3-, 2-, and 1-channel operation
- 2 levels of integrity
- 2 processors in a channel - one primary and a secondary (back-up)
  - The primary runs a full version of the OS and a nominal application that provides full control of the spacecraft
  - The secondary runs a restricted version of the OS and safety-monitoring and simple back-up application (trusted to be free of design faults, to provide control of the spacecraft in a very limited (degraded) survival mode
  - The nominal application is assigned to the lower integrity level; the back-up application is placed at the higher integrity level

The Space Prototype Instance

C=4 M=2 I2



Conclusion

- Tri-dimensional generic architecture
  - C-dimension: 1 to 4 primary fault containment domains
  - M-dimension: secondary fault containment domains
  - I-dimension: segregation and mediation between multiple integrity levels
- Tolerance of physical faults
- Tolerance or confinement of design faults
- Consequence of black-box OS
  - imposes high-granularity implementation of fault tolerance
  - OS cannot be protected from errors so even a transient fault may require re-boot and re-integration
  - FT management cannot access internal OS data structures
    - non-trivial channel re-integration
    - application-level context object(s)