Resilience Modeling and Model-based Design for CPS

Gabor Karsai, Daniel Balasubramanian, Abhishek Dubey, Will Emfinger, Tihamer Levendovszky, Nag Mahadevan, Subhav Pradhan, Will Otte

Vanderbilt University/ISIS

Berkeley
UNIVERSITY OF CALIFORNIA

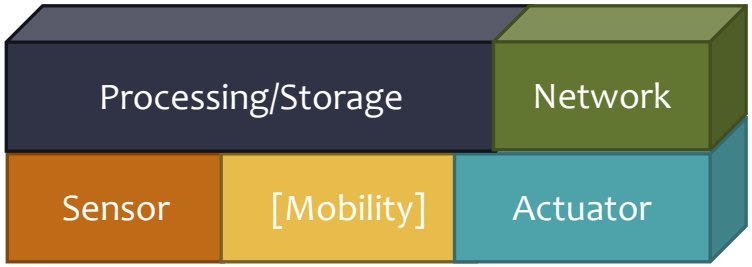
Massachusetts Institute of Technology

MICHIGAN

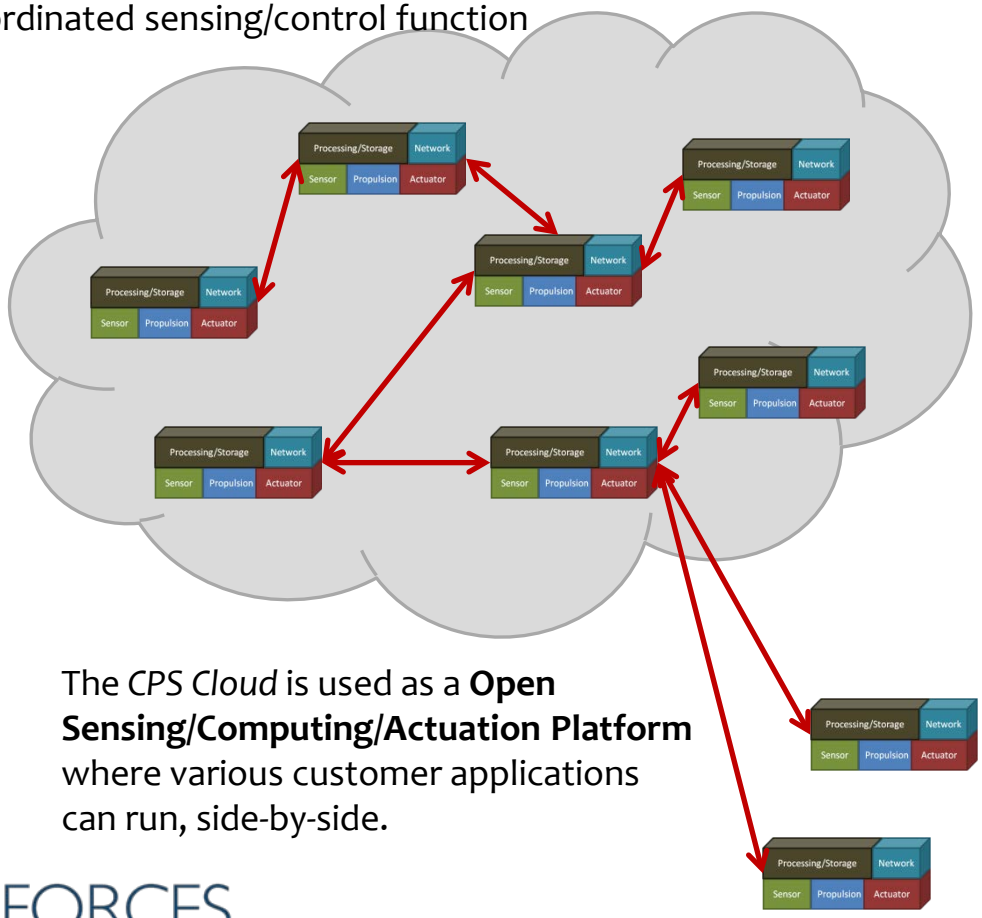
VANDERBILT UNIVERSITY

CPS Cloud:
A Distributed Sensor/Control Network Platform

Network node with local processing and storage, sensors, actuators, and mobility:



Nodes of an **ad-hoc** network that has 1+ ground-link and performs a coordinated sensing/control function



Examples:
- Smart grid devices and systems
- Swarm of UAVs performing tornado damage assessment
- Fleet of UUVs performing collecting climate data from oceans

Challenges:
- Networked, distributed monitoring and control
- Fault resilience, security
- Applications with different trust and security levels must run side-by-side

The *CPS Cloud* is used as a **Open Sensing/Computing/Actuation Platform** where various customer applications can run, side-by-side.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Cyber-Physical Systems
Faults, cyber effects, and resilience

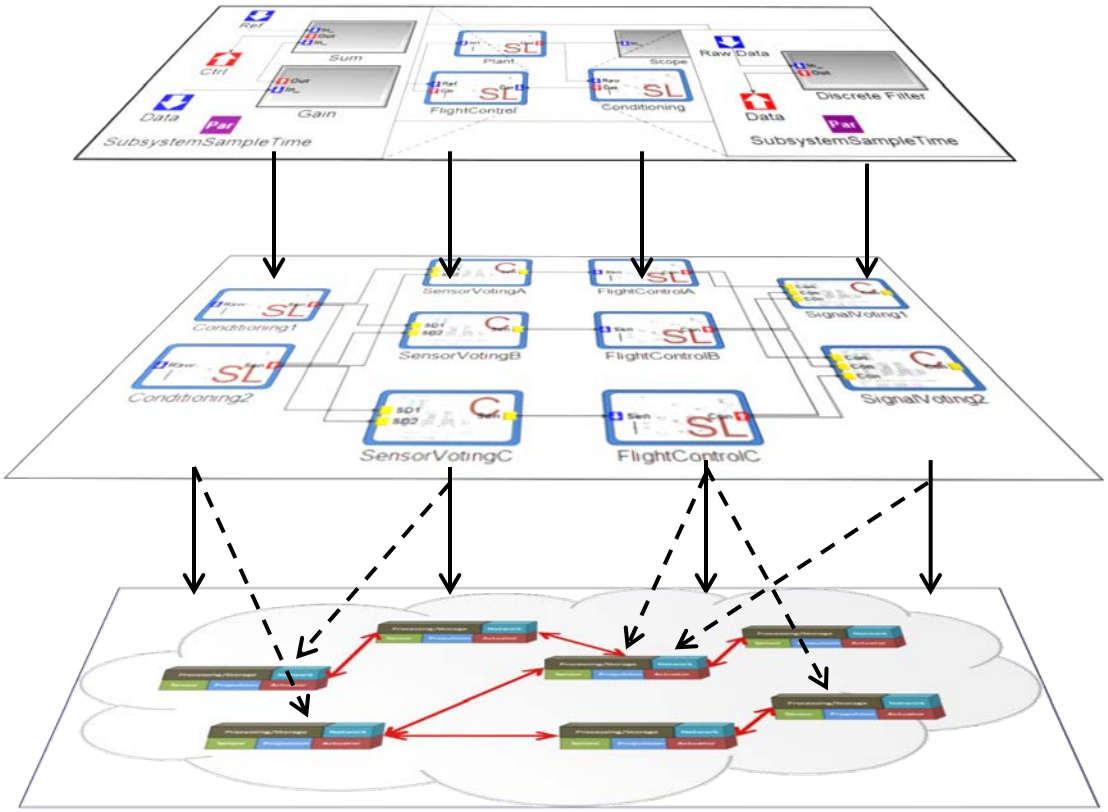
- In CPS *faults* can develop in and/or cascade to
  - Physical system
  - Hardware (computing and communication) system
  - Software (application and platform) system

Vulnerable to cyber effects

- In CPS physical and cyber elements are integrated
  - Many interaction pathways: P2P, P2C, C2C, P2C2P, C2P2P2C
  - System-level resilience *must* consider these interactions
- In CPS resilience can be facilitated to restore function via
  - Physical action, including reconfiguration
  - Cyber restart (node, network, …)
  - Software adaptation

1.Resilience is a system-level property
2.Desirable to reuse resilience techniques
3.A resilient software platform is needed

CPS Applications deployed on the Platform



Applications

- **Built from Processes**

Processes

- **Built from Components**

Network nodes

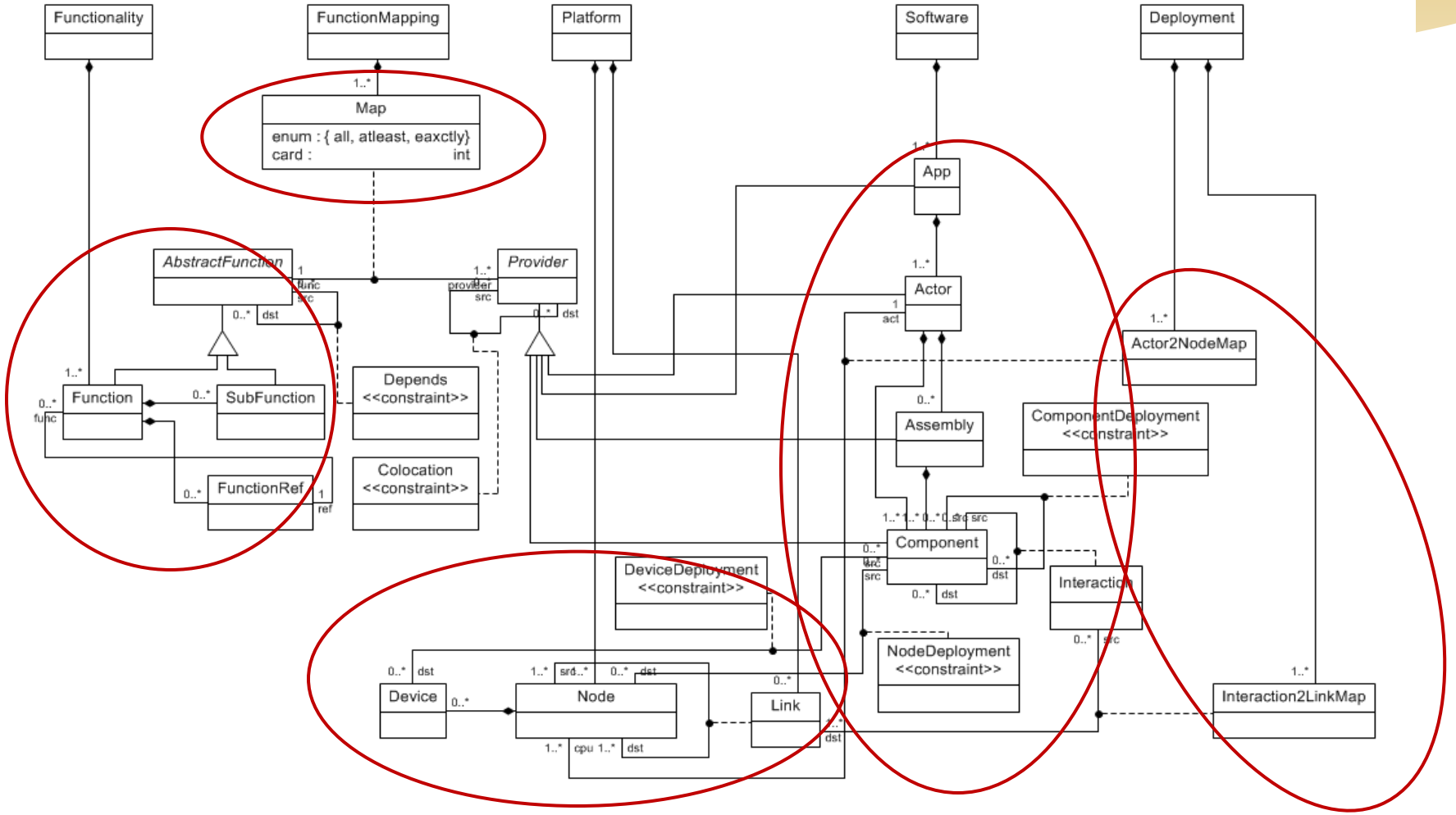
- **Host processes**

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

CPS Cloud Application Platform Architecture 'Language'

Concept	Definition	Notes
Node	Computing node	Hosts processes
Link	Network link	Facilitates interactions
Component	Software component	Unit of concurrency
Process	Software process	Hosts components
Application	Software applications	Consists of processes
Interaction	Component interactions	Pub/sub or service (sync/async)
Device	Physical device	Assigned to node
Function	Functionality	Decomposable with dependencies

Resilient CPS Platform Concepts Metamodel as UML class diagram



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Why is it resilient?

* A **System function** *can be* allocated to various (combinations of) providers: Applications / Processes / Components
* Processes / Components *can be* allocated to various (combinations of) platform Nodes
* When a Node / Link / Process / Component **fails** (compromised), functionality can be restored by an
  * Change allocation of *functions* to *providers*, or
  * Change allocation of *providers* to *platform* nodes

Evaluating CPS Architectures

Architecture: set of related designs.

* Architecture: a collection of architectural variants captured via configuration constraints that define a *configuration space*.
* **Resilience of an architecture:**
  * Capability of eventual recovery from loss of functionality
* Evaluation via a **resilience metric** defined as pair of integers
  1. Measures the margin of recovering capabilities in the *worst case* (redundancy of the system along the most vulnerable/critical path)
  2. Measures the margin of recovering capabilities in the *optimistic case* (overall redundancy of the system)

Definitions

* Component availability refers to the availability of a software or hardware component for use at any time during operation. I.e., it is deployed and is active.
* Function availability refers to the availability of a function. For a function to be available, all the components required for the realization of this function should be available.
* If a function can be supported even when a component becomes unavailable, then there is active redundancy in the function with respect to that component.
* If a function can be carried out even when a component becomes unavailable by activating/deploying another set of components or migrating the affected component to another node, then there is deployment redundancy in the function with respect to that component.

Resilience metric [m1, m2]

* Definition
  * m1: The *worst case resilience* is defined as the *least* number of failures that will make supporting the mission infeasible.
  * m2: The *best case resilience* is defined as the *maximum* number of failures that can be sustained while supporting the mission remains feasible
* Measure the level of *active* and *deployment* redundancy in the system
* The set of deployment constraints imposed on the system and the number of alternative choices affect these numbers.
* Examples of deployment constraints:
* High-res imaging component requires a node with HR camera device.
* A node cannot host more than one instance of a high performance computing component.
* Application A's processes must never be collocated on a node with application B's.

Calculating the resilience metric

* Encode the metric calculation problem as a Satisfiability Modulo Theory (SMT) problem over integers
  * SMT problems are Boolean satisfiability problems (SAT) where some of the binary variables are replaced by predicates over a suitable set of non-binary variables. A predicate is a binary-valued function of non-binary variables that relies on a 'theory'.
* Use an SMT solver to compute the solutions
  * SMT solution = valuation of binary variables
* Metric: number of solutions (for the worst/best case)

Modeling the resilience problem
Domain model

Concept	Description
Component (instances)	Deployed software component instances that operate at the same time. The smallest software unit that can fail. Pre-deployed, inactive backups are *not* modeled as component instances. Binary resources can be modeled as component instances, too.
Nodes	Hardware nodes that are capable of executing component instances.
Links	Interaction links between the nodes. Provided as an adjacency matrix of the graph consisting of nodes and links between them.
Countable component resources	Provided as an integer matrix: ***ccr[i,j]*** means the resource **requirement** of component ***i*** of resource ***j***
Countable node resources	Provided as an integer matrix: ***cnr[i,j]*** means the **availability** of resource ***j*** on node ***i***.
Actors	Defines component groups. If an actor fails, all included component fails.

Modeling the resilience problem
Constraints

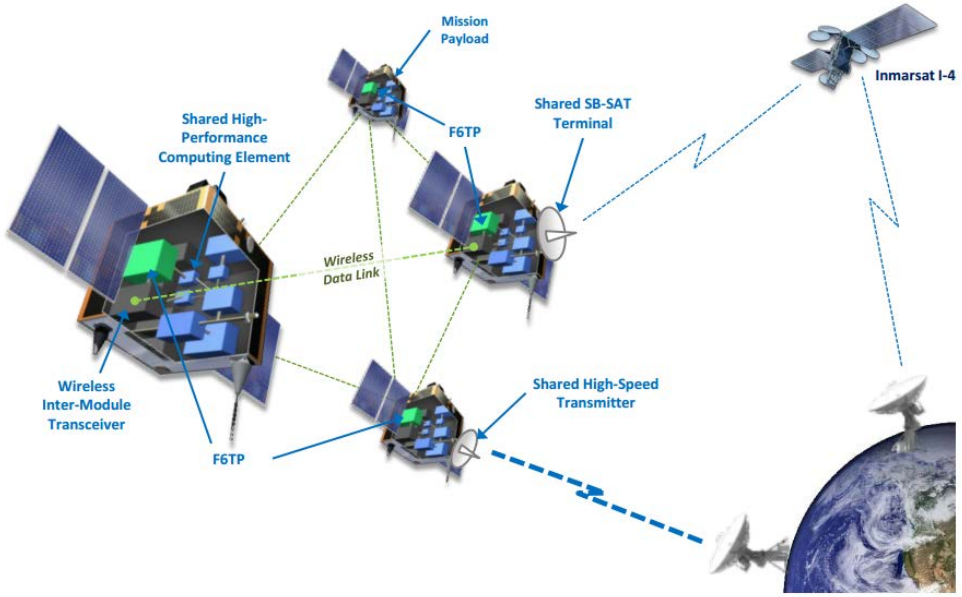
Constraints	Description	Examples
Component to node constraints	Describes the conditions under which a component can be deployed onto a node	Collocation constraints (C_1 and C_2 must/cannot be deployed on the same node); C_1 must be deployed on N_1.
Component to component dependencies	A component's availability is dependent on the availability of another component.	An image processor component C_{im} needs a camera component C_c.
Interaction constraints	In order for two components to interact, there must be a link between the nodes they are deployed on.	An image processor component C_{im} needs a connection to communicate with camera component C_c if they are deployed on a separate node.
Function realization	Describes the dependency between a function and the components it uses.	Function F_1 is realized by components C_1 and C_2.
Functional decomposition	Describes the dependency between functions.	Function F_1 uses functions F_2 or F_4.

An example

Resilience metric calculations

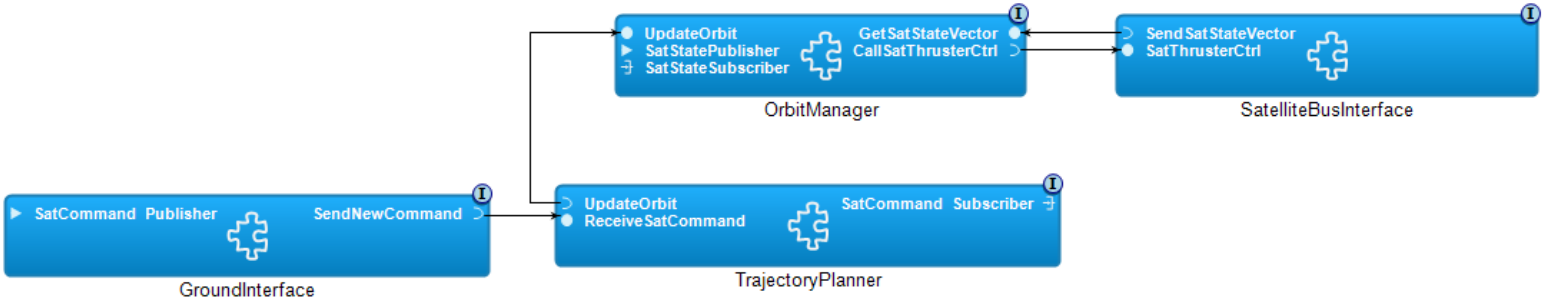
Satellite Resources

- Three similar satellites
- Sat 1
  - HR Camera
  - LR Camera
  - GPU
  - Ground link
- Sat 2
  - HR Camera
  - GPU
  - Ground link
- Sat 3
  - LR Camera
  - Ground Link
- All satellites have wireless links that they can use to communicate with each other
- Some satellites have a ground link



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

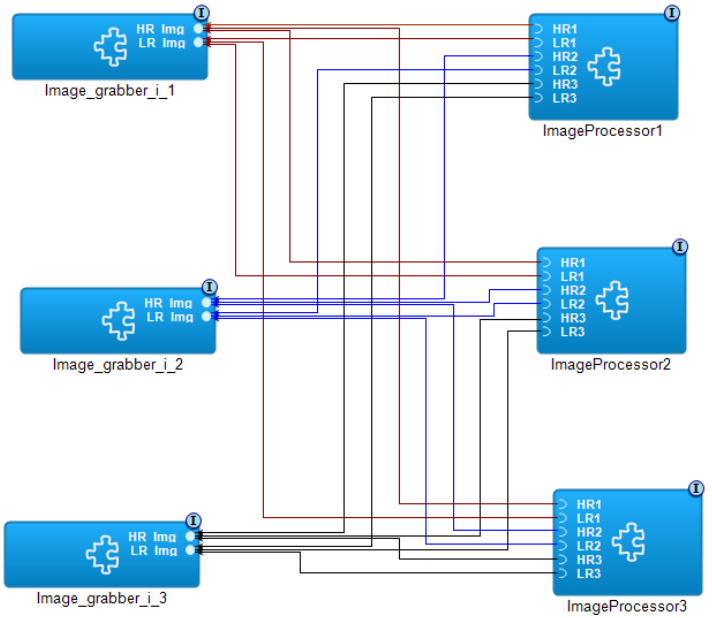
Applications



* Cluster Flight Application (CFA) – Flight Control Software
  * `GroundInterface`: An actor that provides access to the ground station. The ground uses this actor to send commands to the cluster.
  * `SatelliteBusInterface`: An actor that provides access to the satellite bus hardware
  * `TrajectoryPlanner`: Runs the trajectory planning service. It receives the commands from ground and then updates the orbit.
  * `OrbitManager`: Runs the control loop. Disseminates position to other satellites and commands the satellite thruster via the bus interface to adjust the orbit as required.

Applications (continued)

- Wide area imaging application – Sensor Software
  - Uses high resolution and/or low resolution cameras different nodes to create a combined image.
  - Each satellite runs an image grabber component.
    - It can provide service either through the high resolution service or low resolution service or both, depending on the hardware available on the satellite.
  - Only one instance of image processor component runs in the cluster at any time.
    - It can be redeployed as required.

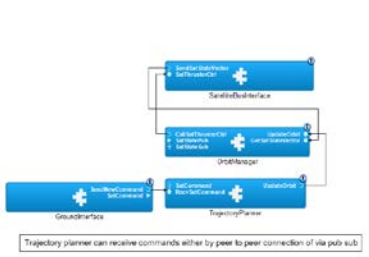


FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Architectural and Resource Requirements

Physical Resource Requirements

* `GroundInterface` **requires** `GroundLink`
* `ImageGrabber`
  * LR_Img: LR_CAMERA (LR_img port **requires** LR camera)
  * HR_Img: HR_CAMERA (HR_img port **requires** HR camera)
* `ImageProcessor` 1,2 **requires** a GPU
* One instance of the CFA runs on each node that **requires** the `OrbitManager` and `SatelliteBusInterface` from the same node.
* Every node is **required** to have a camera
* `TrajectoryPlanner` **requires** 40MB



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Functional requirements

* Represent the functional decomposition for the mission
  * Cluster flight function
  * Wide area imaging function
* Map functions to application/component instances

* Failure of one component/hardware resource/network link is used to compute whether the mission function is unavailable.
* Thereafter an alternative configuration (if available) can be chosen to recover the functionality.

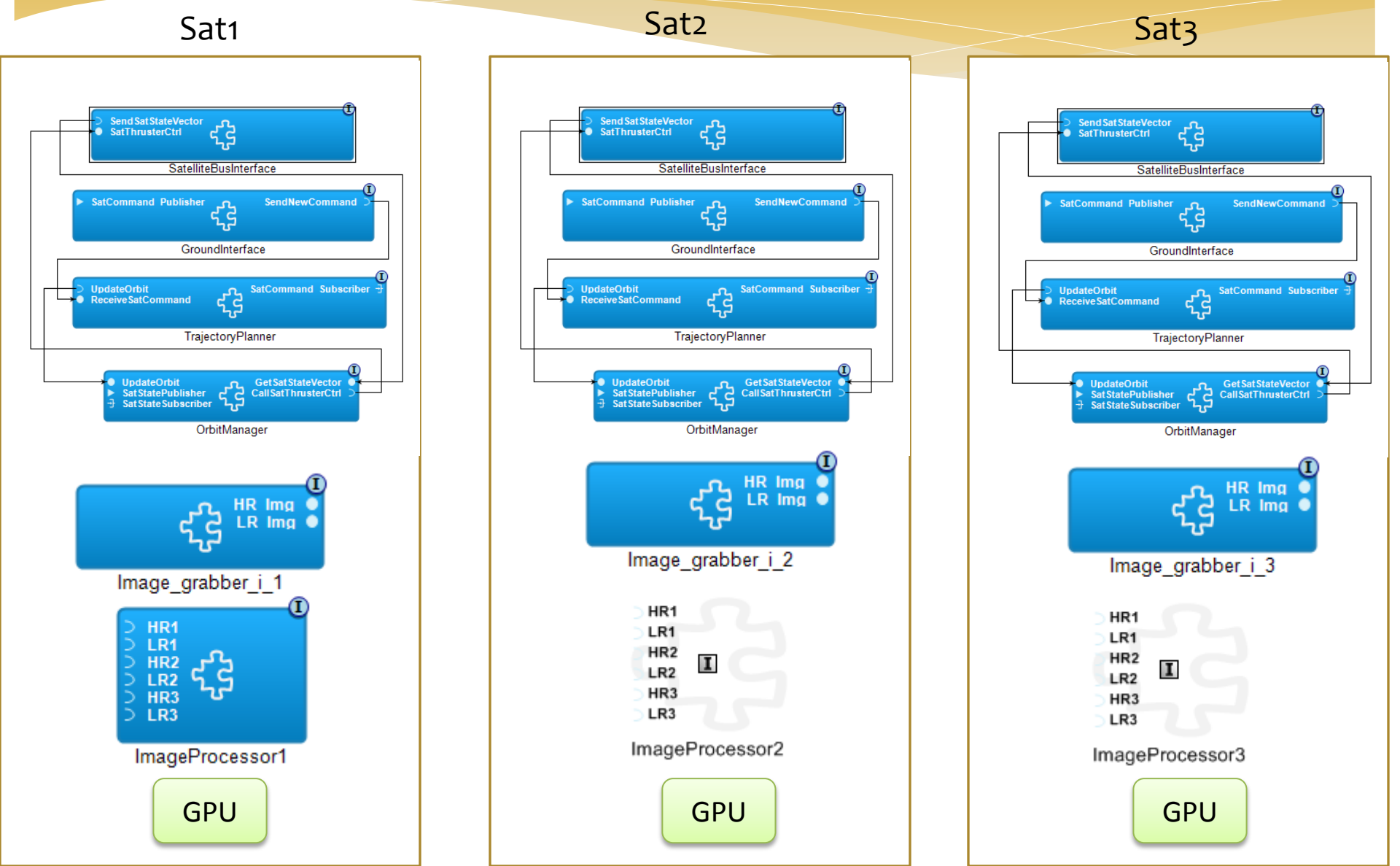
Results: Resilience Metric and Reconfiguration Scenarios

* Metric = [2,23]
  * Assumption: all functions are required
  * The system is 2-fault tolerant, but can operate as many as 23 faults

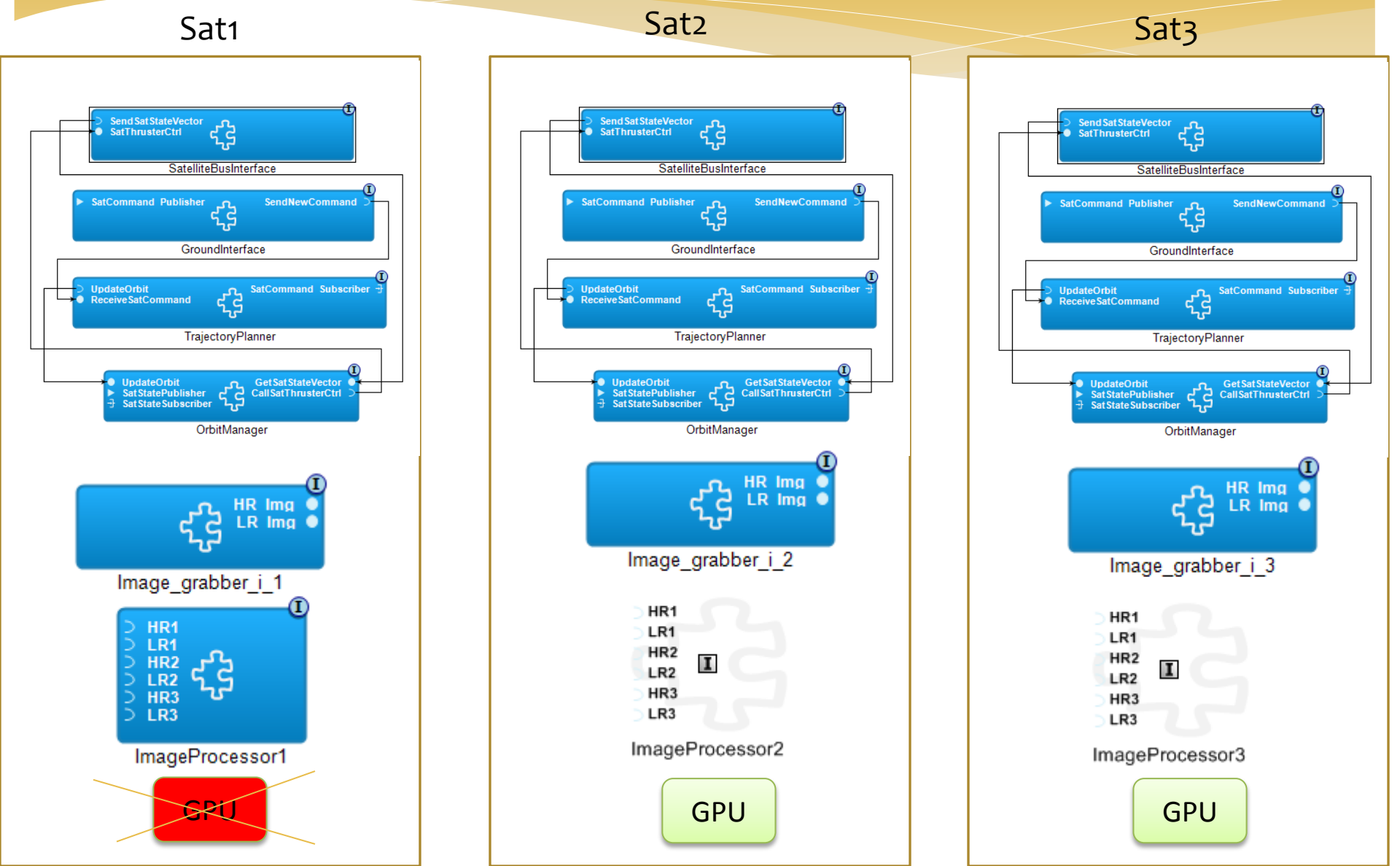
Reconfiguration Scenarios:

* Complete failure of Sat2
  * `ImageProcessor` on Sat2 is out, another `ImageProcessor` on Sat1 or Sat3 should be activated.
* Failure of GPU on Sat1
  * GPU is required by the `ImageProcessor`
  * Therefore, a reconfiguration is required which activates image processor on Sat3
* Failure of `GroundLink` on Sat 1
  * No reconfiguration is required. The ground command is disseminated by either Sat2 or Sat3 via pub/sub ports via the network

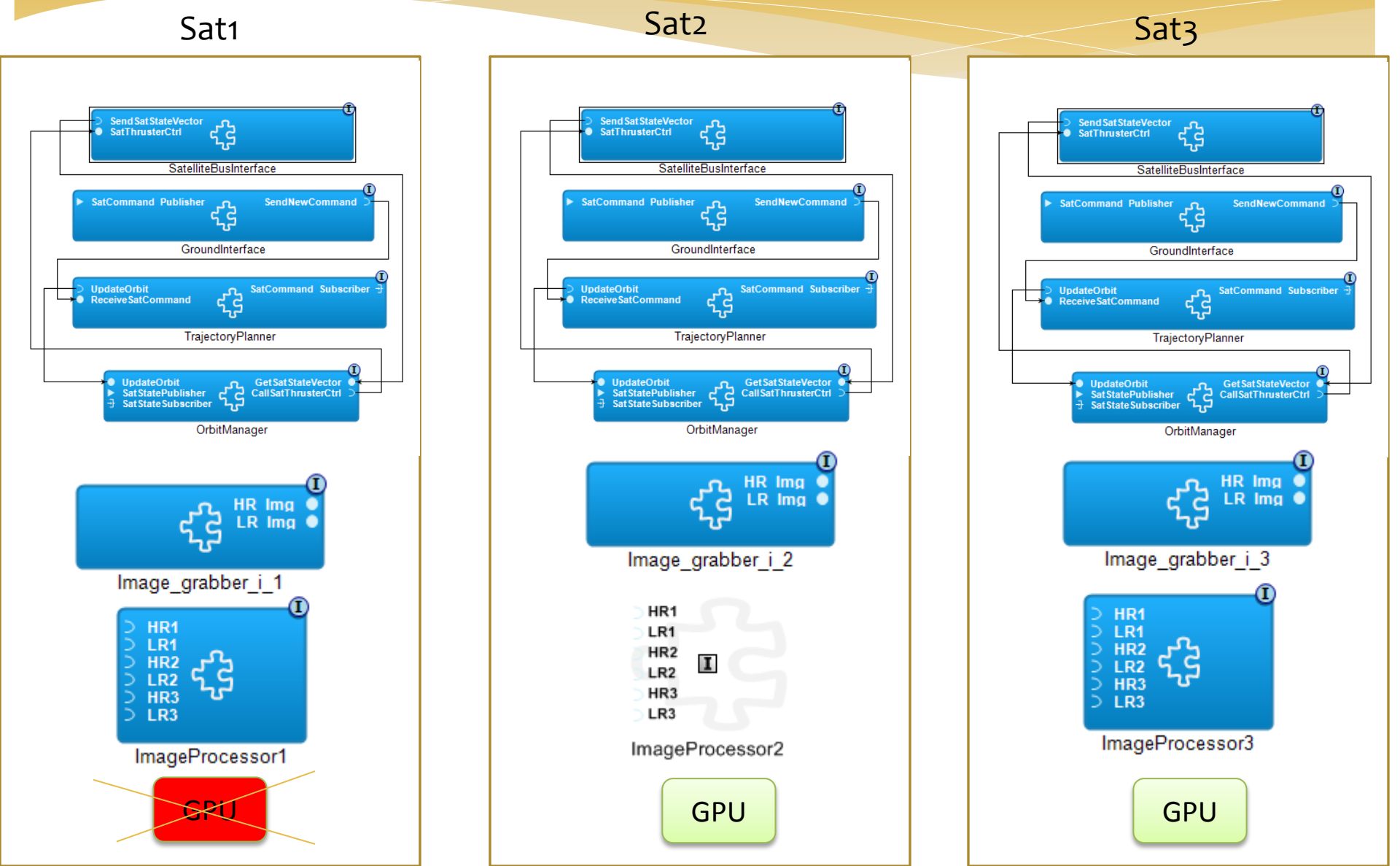
Example: Initial Configuration



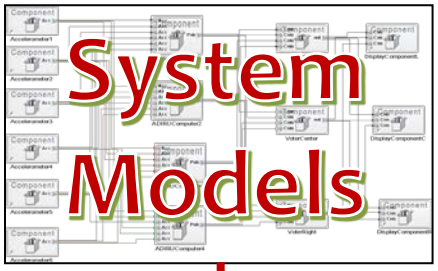
Example: Fault Appears in GPU of Sat1



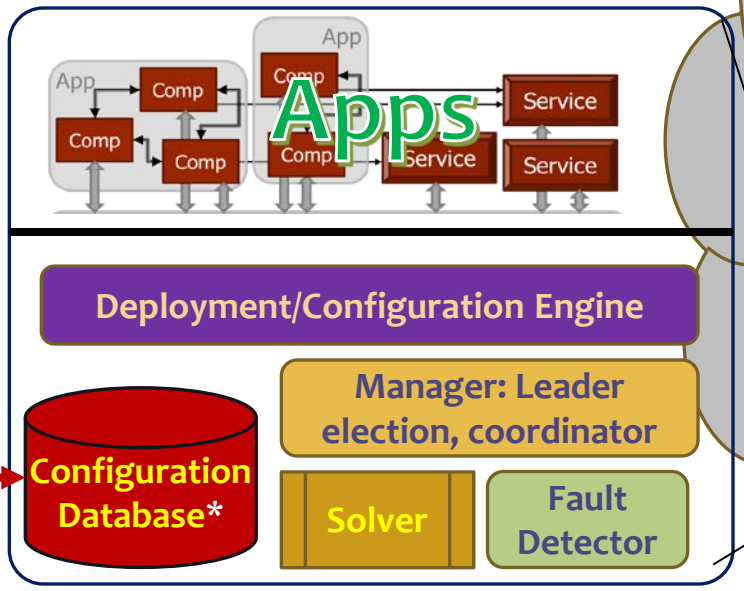
Example: Reconfigure to ImageProcessor3

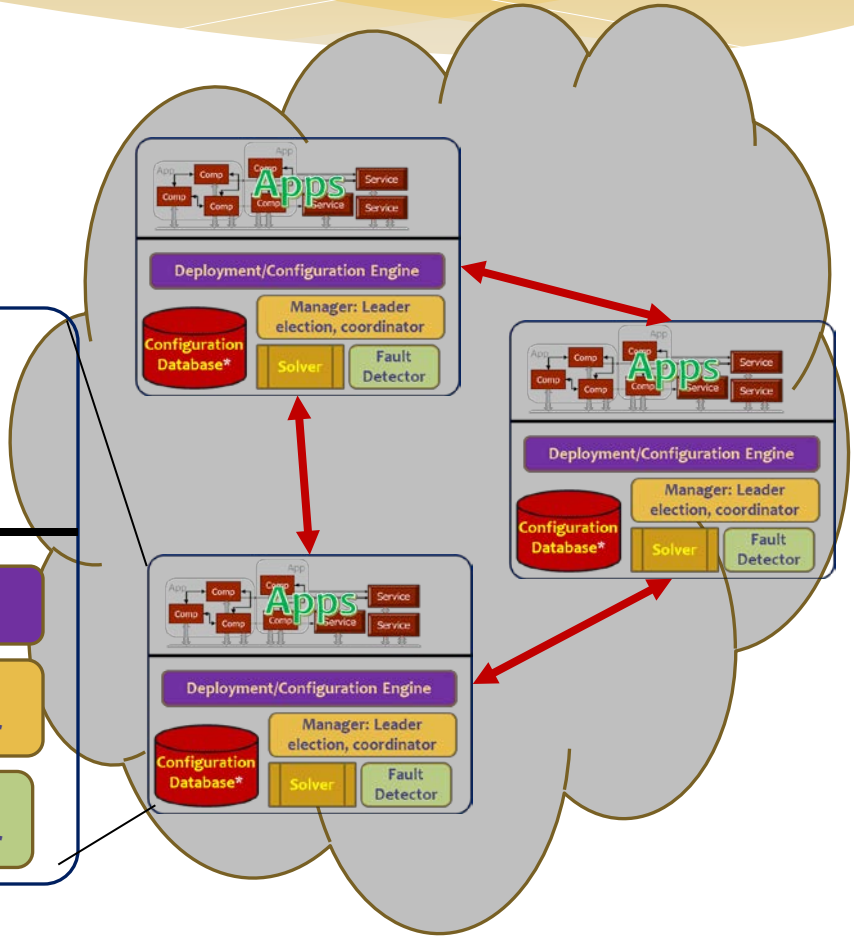


Towards implementing resilience



Resilient
CPS Node:





FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Towards implementing resilience

* Models are translated into deployment plans and configuration spaces that are loaded into a configuration database
* The database is *fault tolerant* (via active replication)
* Each node includes:
  * Deployment and configuration engine (software manager)
  * Fault detector (detects local anomalies and remote node loss)
  * Solver (to re-compute configuration solution)
  * Manager (to do leader / controller election and coordination)
* Any node/process/component/link can fail (become compromised) → the system recovers

Summary: Resilience Modeling and Implementation

- A resilient architecture can be modeled as: software components and architecture + hardware platform + functions + constraints
- A constraint solver can calculate
  - The resilience metric (to compare architectures)
  - Novel configuration/s for the system (to do reconfiguration)
- Resilience implementation relies on:
  - Robust supervisory layer / database/ platform that manages system reconfiguration

Resilience challenges

* Effective, rich, and usable *modeling paradigm/s* for engineering resilient CPS systems
  * Modeling resilience in the physical system through redundancy
* Verification
  * Scalable analysis approaches to compute resilience metrics and properties
* Prototyping specific platform services needed for cyber-resilience: detection, diagnosis, mitigation -- automation
  * Detecting anomalies and cyber effect signatures
  * Isolating effected processes, nodes, links
  * Selecting an appropriate mitigation strategy
* Implementation/experimentation in testbeds: C2WT, DREMS, ROS,…