FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Towards Resilience in CPS Software Platforms

Gabor Karsai, Daniel Balasubramanian, Abhishek Dubey, Will Emfinger, Pranav Kumar, Will Otte

ISIS/Vanderbilt University

Berkeley UNIVERSITY OF CALIFORNIA

Massachusetts Institute of Technology

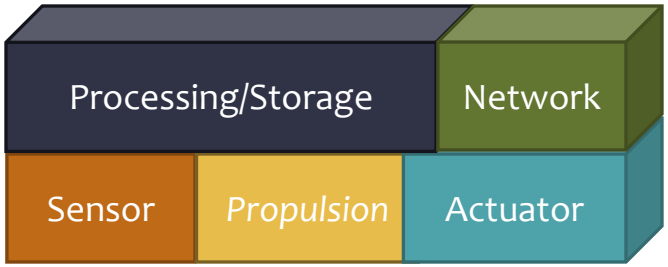
NSF

MICHIGAN

VANDERBILT UNIVERSITY

A 'CPS Cloud': A Distributed Sensor/Control Network Platform

Networked node with local processing and storage, sensors, actuators (and propulsion) system:



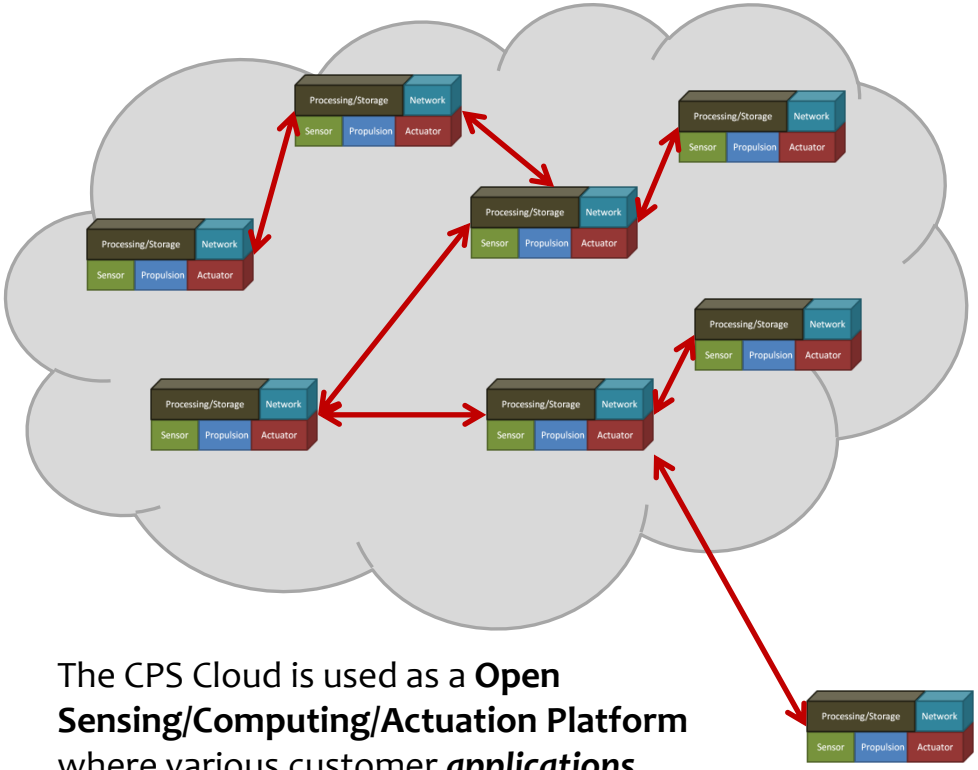
Examples:

- Coordinated swarm of UAVs for tornado damage surveillance running sensor fusion ***apps***
- Fleet of UUVs collecting climate change data from oceans running sensor fusion and motion control ***apps***
- Nodes on the Smart Electric Grid running distributed monitoring and control ***apps***

Challenges:

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

Nodes for an ad-hoc network that has 1+ link to a 'base station' perform coordinated sensing/control functions



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

Goal: Platform as a Service

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Concepts for a CPS Platform

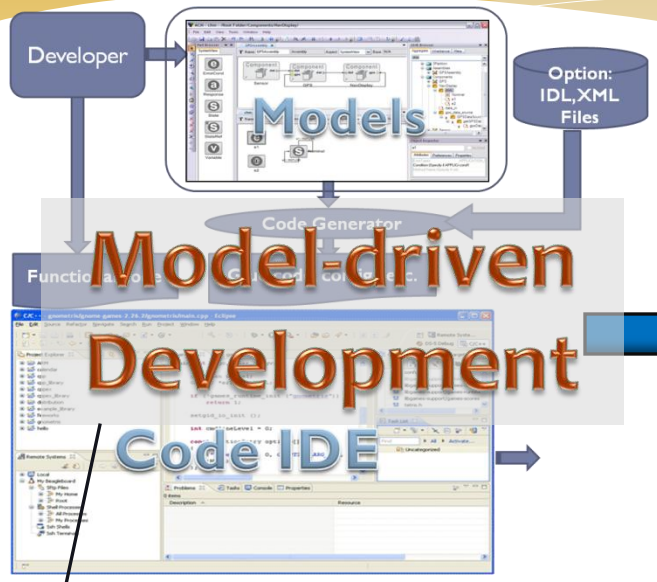
Property	Characteristics
Distributed	Distributed applications are deployed and managed on multiple hosts connected to a network Diverse applications may interact with each other Resources (including I/O devices) are network-accessible and can be shared among applications Some applications are 'open' – interact with the external network Most applications interact only via the internal network
Real-time	Real-time requirements are imposed on the system: response time, message latency, time-triggered activities, guaranteed processor and network bandwidth/timeliness Platform provides real-time guarantees Applications must interact with the physical world through I/O devices
Embedded	System is embedded in a physical environment with its constraints on dynamics Some highly specialized/restricted user interface with constrained interactions

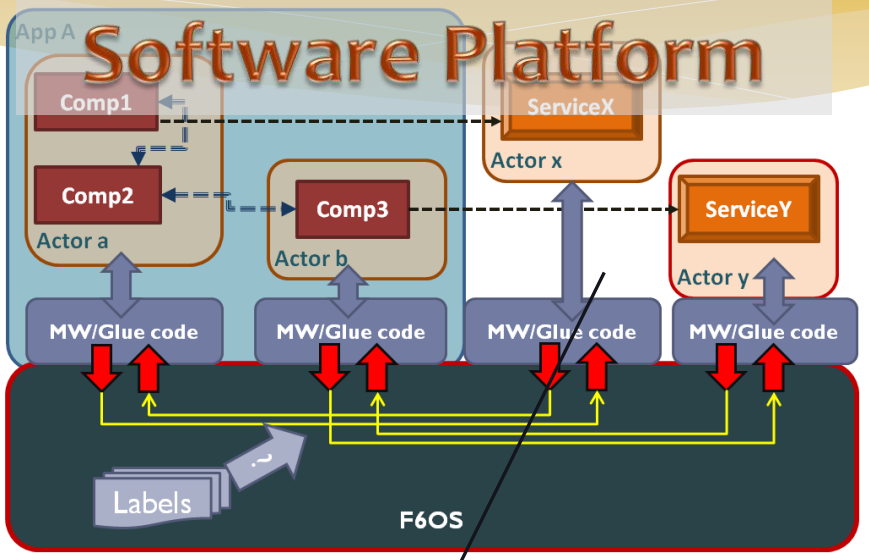
Concepts for a CPS Platform

Property	Characteristics
Resilience	System must be resilient to **faults**, **changes**, and **security attacks** System is functional even under partial fault conditions Anything can fail at any time – platform provides *mechanisms* to implement fault tolerance for the applications and the system Some faults are handled autonomously by the platform Deliberate updates to the system are expected, system is resilient to faults in updates Anything can change at any time – environment, system configuration, executables – platform provides services to detect and mitigate such changes Anything can be attacked at any time, apps are trusted to a varying degree Applications are informed about fault conditions and can mitigate their effects
Secure	Untrusted applications are not to interfere with system functions or other apps Apps with different security level are isolated and operate side-by-side

Definition of 'resilience':
Capable of withstanding shock without permanent deformation or rupture. Tending to recover from or adjust easily to misfortune or change. [Webster]
The persistence of the avoidance of failures that are unacceptably frequent or severe, when facing changes. [Laprie, '04]

DREMS:
An Experimental Toolchain and Platform





Software *toolchain* for modeling, synthesis, analysis, and verification

Software *platform* with support for resource sharing, security, and fault tolerance

Platform prototype development supported by: DARPA

The software will be released under the MIT/X Open Source License.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Platform Prototype Testbed Setup

Physics simulator (Orbiter)

Model-driven Development Environment

Networked Embedded Processors

Not shown: Network emulator

Foundations: Linux Kernel modified to implement new, secure system calls, CORBA/DDS middleware, novel component framework, privileged system management services

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Resilience as addressed currently

Principle	Details
Platform protection	System service entry interfaces are *protected* by auto-generated guards
Application isolation	Applications are spatially and temporally *isolated* from each other, all information flows are managed (including covert channels)
Managed network	Network is strictly *controlled* – apps cannot communicate arbitrarily All information flows on the network are pre-configured before the applications starts
Managed applications	Application deployment and configuration is performed by *trusted entity* Applications can be started and managed only by a privileged process
Component-based development	Apps are *integrated* from reusable, configurable, verified components rather than being built from scratch
Modeling and model-driven development	Software: interfaces, components, architecture, etc. is *modeled*, glue code and infrastructure configuration is *generated*, models can be *analyzed*

Dubey, Abhishek, William Emfinger, Aniruddha Gokhale, Gabor Karsai, William R. Otte, Jeffrey Parsons, Csanád Szabó, Alessandro Coglio, Eric Smith, and Prasanta Bose. "A software platform for fractionated spacecraft." In Aerospace Conference, 2012 IEEE, pp. 1-20. IEEE, 2012.

Resilience framework: Mechanisms
Research problems

Core platform capabilities

- Anomaly detection - health monitoring
  - Run-time monitoring of application components, system interfaces, kernel integrity, shared resources, network interfaces
  - Group membership verification
  - Integrated fault tolerant leader election
- Diagnostics
  - Root cause determination
  - Distributed reasoning / no central authority
  - System 'state' estimation
- Mitigation
  - Local / reactive mitigation
  - Coordinated, global, distributed recovery
  - Autonomous vs. managed response

Core platform services

- Resilient software deployment
  - Robust deployment of software applications, with transactional/roll-back capabilities
- Multi-level fault management
  - Each level: (1) *assumes* properties of lower levels, and (2) *guarantees* properties for higher levels
  - Each level detects/diagnoses/recovers (while it can)
  - Component framework must provide resilience services
  - Aspects of fault management are customizable for the applications

Modeling/Analytical capabilities

- *Given a model of the architecture and assumed failure/threat models, is the system resilient?*