Next-Generation Embedded Systems: Functional Reactive Programming and Real-Time Virtual Resources

Prof. Albert M. K. Cheng

- Outline
  - Embedded Real-Time Systems
  - Functional Reactive Systems (FRS)
  - Cyber-Physical Systems (CPS)
  - Response Time Analysis
  - Real-Time Virtual Resources

* Supported in part by the National Science Foundation under Awards No. 0720856 and No. 1219082.

Real-Time Systems Group

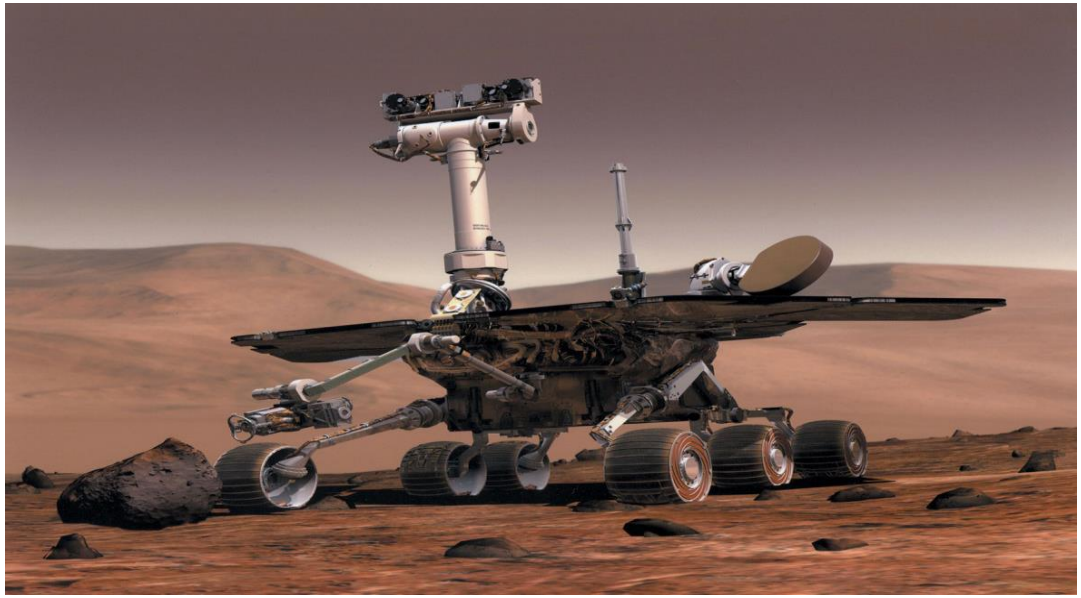
- **Director** Prof. Albert M. K. Cheng
- **PhD students**
  Yong Woon Ahn, Yu Li, Xingliang Zou, Behnaz Sanati, Sergio Chacon, Zeinab Kazemi, Carlos Rincon, Xin Liu, Qiong Lu, Seyed Mohsen Alavi (arriving in spring 2015)
- **MS students**
  Daxiao Liu, Chonghua Li
- **Undergraduate students (NSF-REU)**
  Mozahid Haque, Rachel Madrigal
- **Visiting scholars**
  Yu Jiang (Heilongjiang U.), Qiang Zhou (Beihang U.), Yufeng Zhao (Xi'an Tech. U.)
- **Recent graduates and their positions**
  Yuanfeng Wen (MS, Microsoft), Chaitanya Belwal (PhD, Visiting Assistant Professor, UHCL), Jim Ras (PhD), Jian Lin (PhD, Assistant Professor, UHCL)

Yu Li (Best Junior PhD Student Awardee and Friends of NSM Graduate Fellow) and Prof. Albert Cheng visit the NSF-sponsored Arecibo Observatory after their presentation at the flagship RTSS 2012 in Puerto Rico.

Real-time systems research group at Yuanfeng Wen's graduation party in May 2013. Yuanfeng is now at Microsoft.

Fall 2014 (9/3) group meeting - from left to right: Dr. Qiang Zhou, Qiong Lu, Carlos Rincon, Chonghua Li, Prof. Yu Jiang, Xin Liu, Prof. Yufeng Zhao, Prof. Albert Cheng, Xingliang (Jeffrey) Zou, Daxiao Liu, Yu Li, Yong Woon Ahn, and Behnaz Sanati. Zeinab Kazemi in class.

Real-Time Systems Theory



Pathfinder mission to Mars: best known Priority Inversion problem.

Failure to turn on priority Inheritance (PI) - Most PI schemes complicate and slow down the locking code, and often are used to compensate for poor application designs.

http://research.microsoft.com/en-us/um/people/mbj/mars_pathfinder/mars_pathfinder.html

http://www.windriver.com/announces/curiosity/

Real-Time Systems Theory

- **The more components a real-time system has, the more difficult it is to build and maintain.**
  - In such systems, **preemptive scheduling** may not be suitable, since it is likely to create runtime overheads which can result in worst-case task execution times of up to **40%** greater than fully **non-preemptive** execution.
    - Yao G., Buttazzo G., Bertogna M., "Feasibility analysis under fixed priority scheduling with limited preemptions," Real-Time Systems, Volume 47 Issue 3, pages: 198-223, May 2011.

Real-Time Systems Theory

- However, **preemptive** scheduling allows for more feasible schedules than **non-preemptive** scheduling.
- **Non-preemptive** scheduling automatically prevents unbounded priority inversion, which avoids the need for a concurrency control protocol, leading to a less complex scheduling model.
- However, fully **non-preemptive** scheduling is too inflexible for some real-time applications, and has the added disadvantage of potentially introducing large blocking times that would make it impossible to guarantee the schedulability of the task set.

Real-Time Systems Theory

- Simplify the design and scheduling
  - Avoid priority inheritance
  - Use functional programming
  - Use abort-and-restart
  - Use harmonic task sets
    - However, harmonic tasks sets may be too restrictive for some situations. For example, one sensor needs to be serviced every 9 seconds and another (because of its design / physical characteristics) 10 seconds.

Embedded Real-Time Systems

- An embedded system is a computer system designed for specific control functions within a larger system

  (**A** is embedded into **B** for control)

- Often with such systems there are constraints such as deadlines, memory, power, size, etc.

Embedded Real-Time Systems

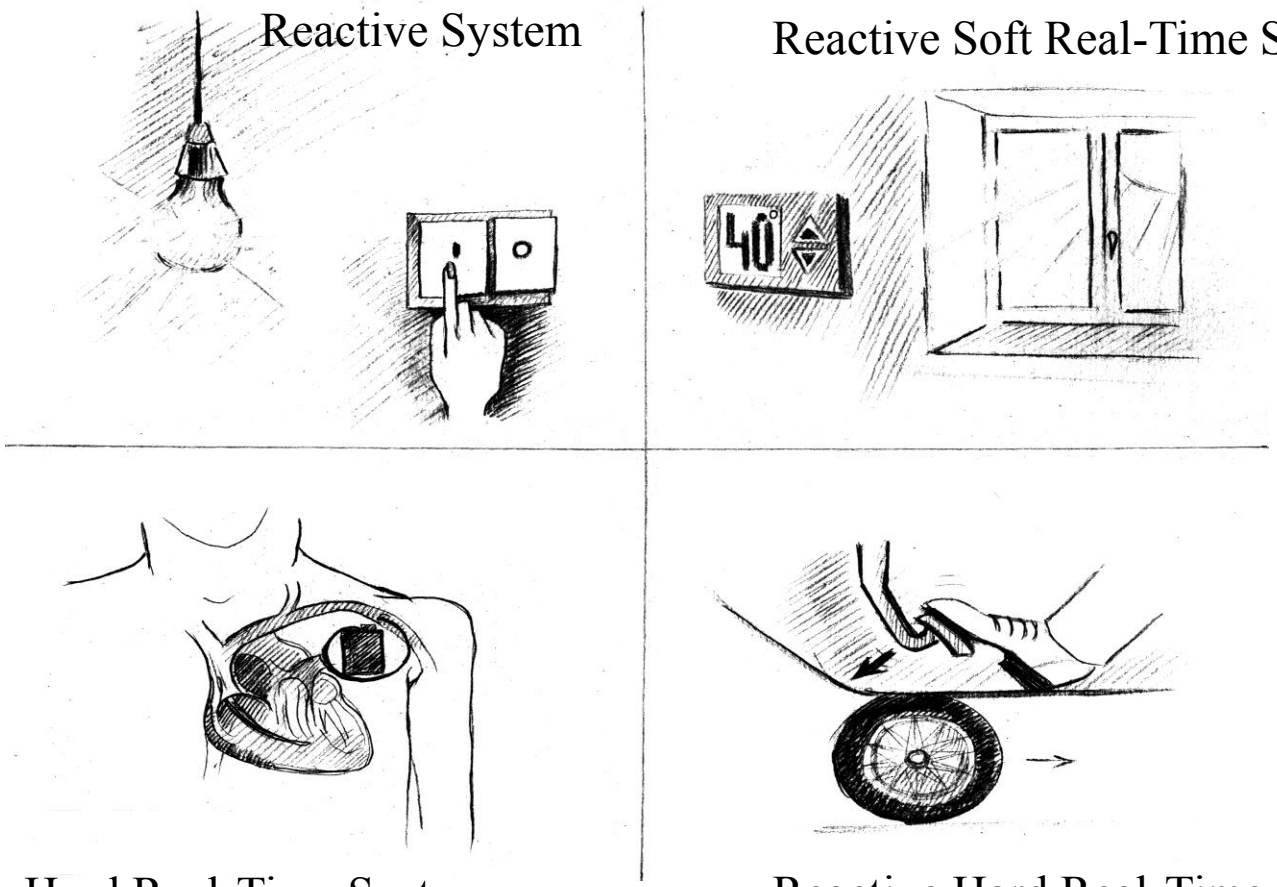
- **Real-time systems (RTS)** are reactive systems that are required to respond to an environment in a bounded amount of time.
- **Functional reactive systems (FRS)**
- **Cyber-physical systems (CPS)**
  - Challenges
    - Complexity
    - Reliability
      - Fault-tolerant design
      - Meeting deadlines (Response Time Analysis (RTA))
    - Security/Privacy

Functional Reactive Systems (FRS)

Systems that react to the environment being monitored and controlled in a timely fashion using functional (reactive) programming are known as Functional Reactive Systems (FRS).

These systems can range from small devices (which are not a CPS) to distributed and complex components (similar to a CPS).

Functional Reactive Systems (FRS)



Cyber-Physical Systems (CPS)

- Systematic integration of computation/information processing and physical processes and devices.
- Communication and sensing are components of CPS

Cyber-Physical Systems (CPS)

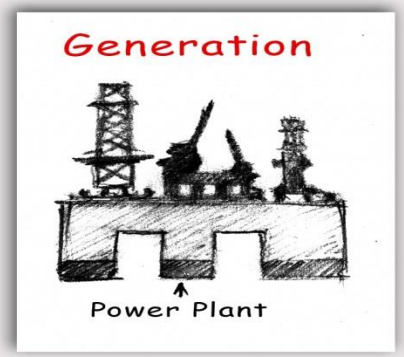
The current set of tools available for analysis cannot handle the complexity of CPS and thus are unable to predict system behavior with high degree of accuracy.

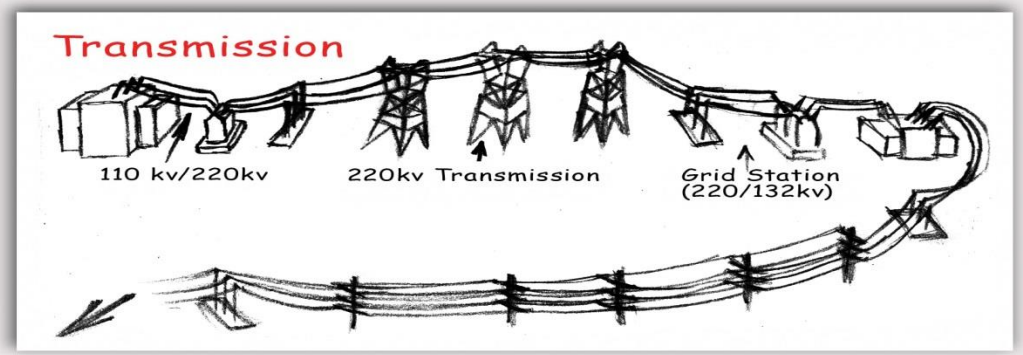
The consequences of these shortcomings:

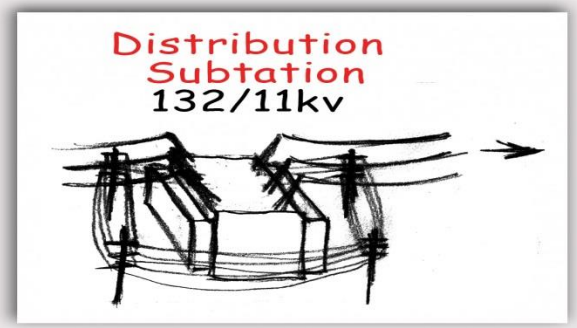
Consider the electric power grid -- Massive failures leading to blackouts can be triggered by minor events.

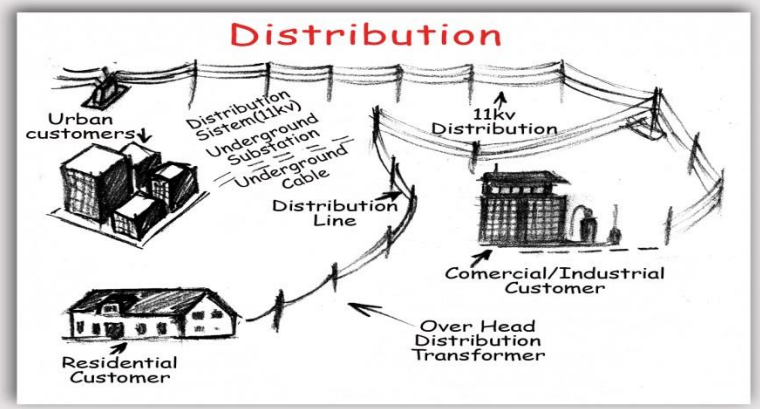
Cyber-Physical Systems (CPS)

Classic (non-CPS) electric grid system/behavior









Cyber-Physical Systems (CPS)

- In a CPS, wireless/wired smart meters measuring real-time electricity usage and historical data (state) feedback (communication) to the generation station to better manage and distribute electricity.

- Current and predicted weather condition data can also further inform the decision-making in where to distribute electricity (very hot or very cold weather increase electricity demand).

- There is also a need to guard against intrusion into the system.

- Advocate formal verification to ensure satisfaction of safety properties.

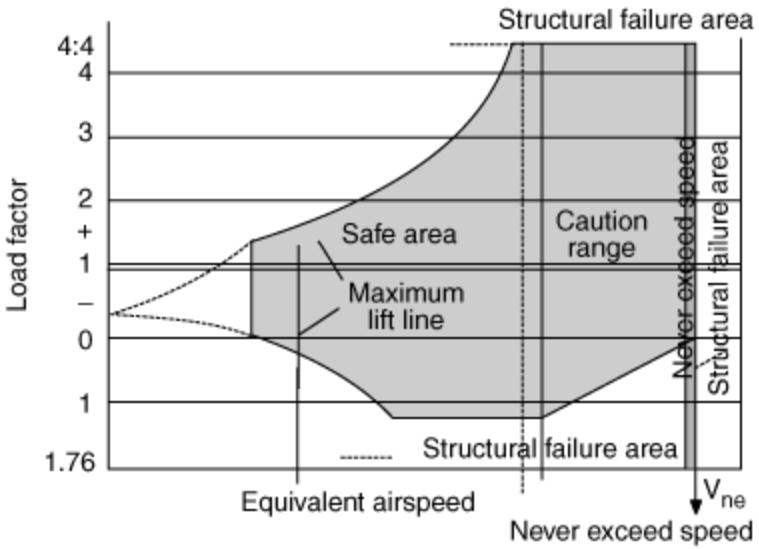
Cyber-Physical Systems (CPS)

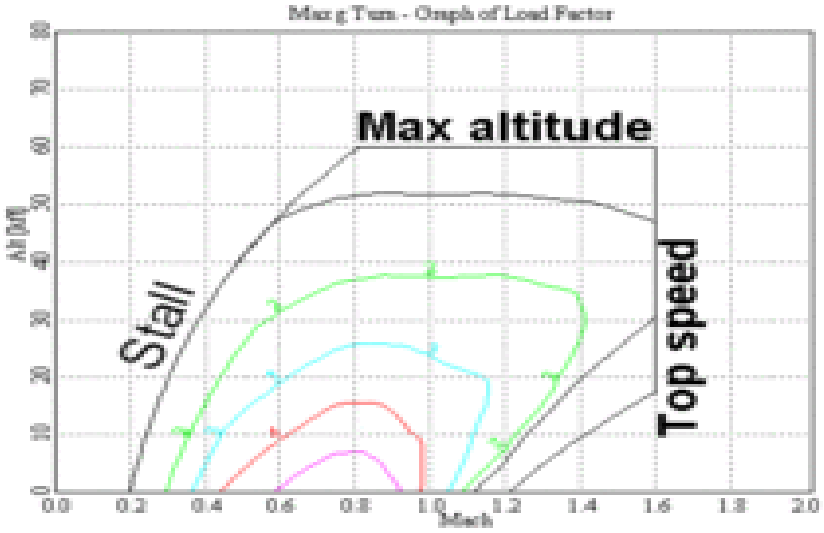
Cyber-Physical Systems (CPS)

Imagine an airplane that refuses to crash. While preventing all possible causes of a crash is not possible, a well-designed flight control system can prevent certain causes. The systems that do this are good examples of cyber-physical systems.

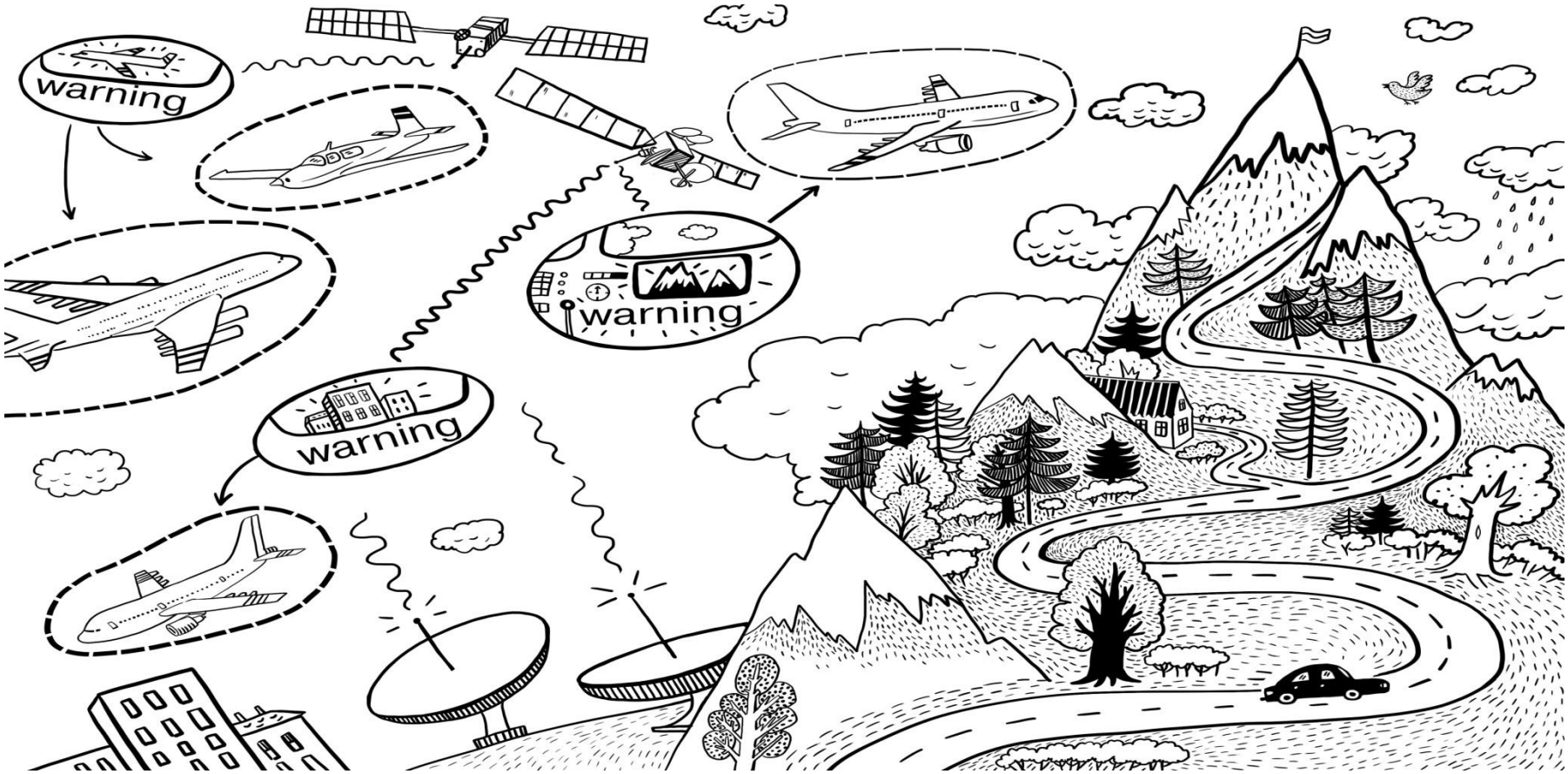
Cyber-Physical Systems (CPS)

For example, some airplanes use a technique called flight envelope protection to prevent a plane from going outside its safe operating range, and prevent a pilot from causing a stall.





Cyber-Physical Systems (CPS)



Cyber-Physical Systems (CPS)

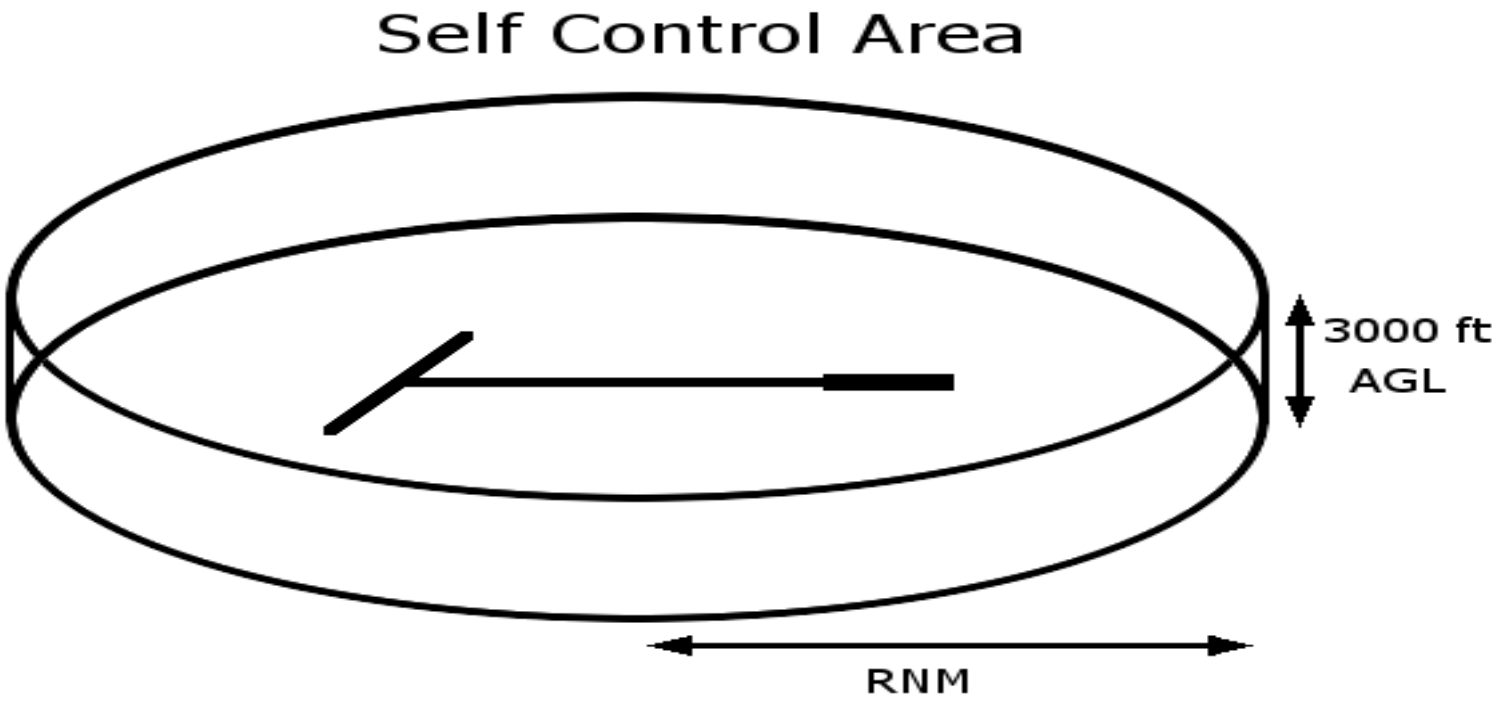
- One of the key goals in our research is to develop the core tools that can be used to facilitate the analysis, design and engineering of highly-complex systems.

- With such tools, we can ensure that these systems are reliable, predictable, efficient, secure and resilient to multiple points of failure, and hence that their operation and safety can be depended upon with a high degree of confidence.

- We advocate formal verification to ensure safety of CPS's, but their complexity requires further research in verification tools.

Cyber-Physical Systems (CPS)

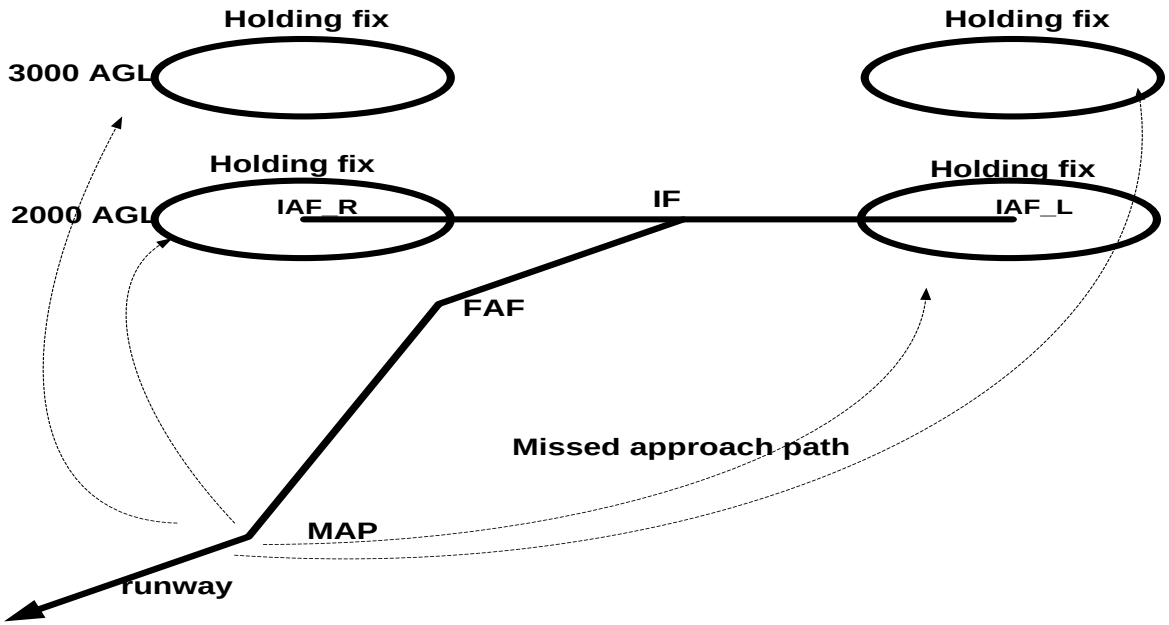
Small Aircraft Transportation System (SATS)



Self Control Area Airspace Volume

Cyber-Physical Systems (CPS)

Small Aircraft Transportation System (SATS)



3-D View of the SCA

Functional Reactive Programming

- Priority-based Functional Reactive Programming (P-FRP)
- P-FRP provides real-time guarantees using static priority assignment
- Higher-priority tasks preempt lower-priority ones; preempted tasks are aborted
- Multi-version commit model of execution
- Atomic execution – “all or nothing” proposition
- Execution different from ‘standard’ models

Other Examples of Functional Programming (FP) Languages:

- Haskell
- Atom - Domain Specific Language in Haskell
- Erlang - Developed at Ericsson for programming telecommunication equipment
- Esterel - Designed for reactive programming
- F# - Developed by Microsoft; available as a commercial platform

Functional Reactive Programming (FRP)

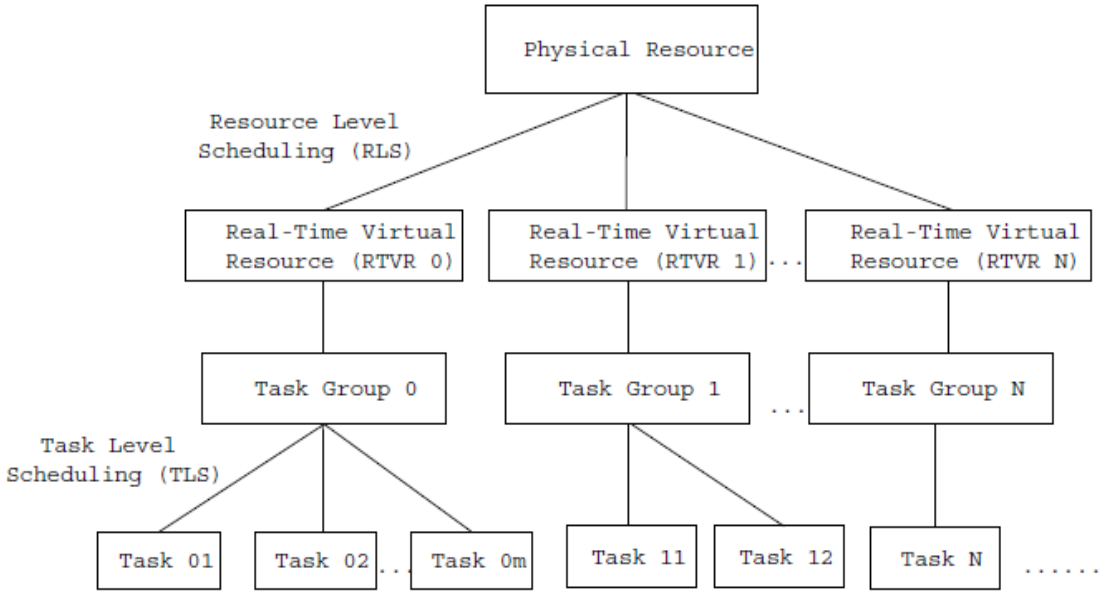
- Functional reactive programming (FRP) is a style of functional programming where programs are inherently stateful, but automatically react to changes in state.

- FRP allows intuitive specification and formal verification of safety-critical behaviors, thus reducing the number of defects during the design phase, and the stateless nature of execution avoids the need for complex programming involving synchronization primitives.

- Therefore, the program remains an algebraic description of system state, with the task of keeping the stated (unidirectional) relationships in sync left to the *language*.

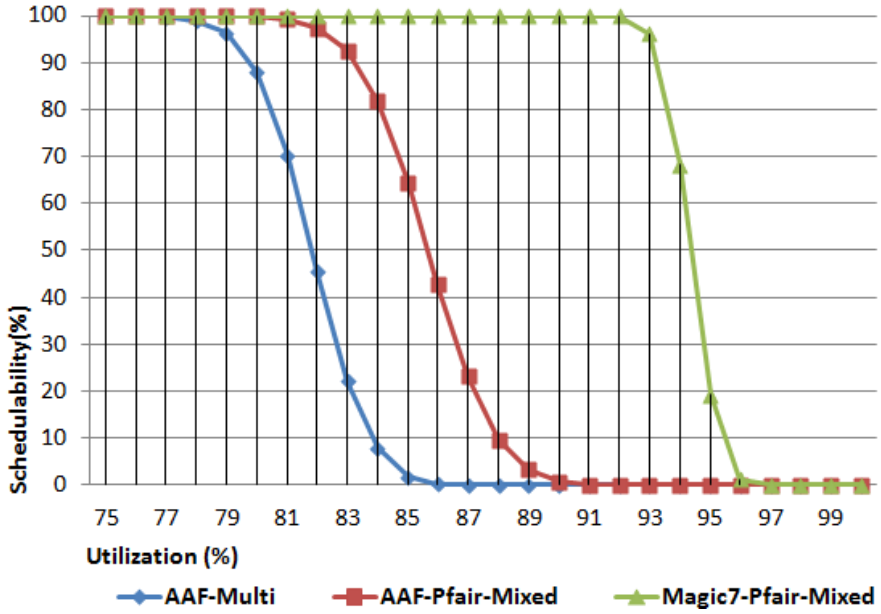
Hierarchical Real-Time Scheduling (HRTS) – Virtual Resources

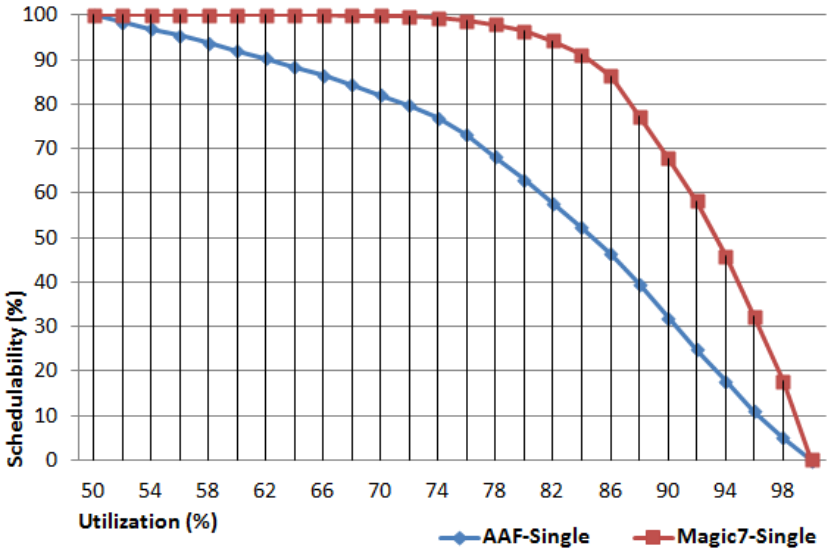
- Motivation: deploying real-time systems on powerful modern hardware causes **low resource utilization**
- Solution: **integrating** multiple real-time systems into one single platform



Magic7: Experimental Results

64-resource ➔





← Single-resource

Concluding Remarks

- Our goal: Enhance the safety and performance of a physical system controlled by an embedded controller consisting of single or networked control components with functional reactive programming (FRP) and real-time virtualization.
- FRP allows intuitive specification and formal verification of safety-critical behaviors, thus reducing the number of defects injected during the design phase, and the stateless nature of execution avoids the need for complex programming involving synchronization primitives.
- Accurate response time analysis tools (accounting for CPU execution, memory access, I/O, and sensor processing times), novel scheduling techniques, and new power-conserving methods are needed.
- Research impact: Facilitate the design and update of the embedded controller (or network of controllers) as well as its (their) timing and safety verification.
- Enhance and update embedded systems with real-time virtual resources.