PTIDES: A Discrete-Event-Based Programming Model for Distributed Embedded Systems

John C. Eidson
Edward A. Lee
Slobodan Matic
Sanjit A. Seshia
Jia Zou

UC Berkeley

Tutorial on Modeling and Analyzing Real-Time Multiprocessor Systems

ESWEEK 2010
Scottsdale, AZ, USA, October 24, 2010
Cyber-Physical Systems (CPS):
*Orchestrating networked computational resources with physical systems*

**Automotive**
- E-Corner, Siemens
- Daimler-Chrysler

**Military systems:**

**Power generation and distribution**

**Building Systems**

**Telecommunications**

**Avionics**

**Transportation**
- (Air traffic control at SFO)

**Instrumentation**
- (Soleil Synchrotron)

**Factory automation**

*Courtesy of Kuka Robotics Corp.*

*Lee, Berkeley 2*
Focus of this Talk: Distributed CPS
Example – Printing Press

- **Distributed**
  - 100s of microcontrollers.
  - Ethernet with time synchronization (IEEE 1588).
  - Requires distributed fault handling.

- **High-speed, high precision**
  - Speed: 1 inch/ms.
  - Precision: 0.01 inch
  -- Time accuracy: 10us.
Approaching the CPS Challenge

*Physicalizing the cyber (PtC)*: to endow software and network components with abstractions and interfaces that represent their physical properties, such as dynamics in time.

*Cyberizing the Physical (CtP):* to endow physical subsystems with cyber-like abstractions and interfaces
Timing challenges for distributed applications: Networks with “quality of service” are insufficient. Need “correctness of service.”

Traditionally, “faster is better.”

This is like saying that for a roller coaster, “stronger is better.”

We have to change the mindset to “not fast enough is an error! So is too fast!”
For distributed cyber-physical systems,

Timing needs to be a part of the network *semantics*, not a side effect of the implementation.

Technologies needed:
- Time synchronization
- Bounds on latency
- Time-aware fault isolation and recovery
- Time-aware robustness
Background - Domain-Specific Networks with Timed Semantics

- **WorldFIP**: Factory Instrumentation Protocol
  - Created in France, 1980s, used in train systems

- **CAN**: Controller Area Network
  - Created by Bosch, 1980s/90s, ISO standard

- Various **ethernet** variants
  - PROFInet, EtherCAT, Powerlink, …

- **TTP/C**: Time-Triggered Protocol
  - Created around 1990, Univ. of Vienna, supported by TTTech

- **MOST**: Media Oriented Systems Transport
  - Created by a consortium of automotive & electronics companies
  - Under active development today

- **FlexRay**: Time triggered bus for automotive applications
  - Created by a consortium of automotive & electronics companies
  - Under active development today
Services in Time-Aware Networks

- Frequency locking
  - E.g., **synchronous ethernet**: ITU-T G.8261, May 2006
  - Enables integrating circuit-switched services on packet-switched networks
  - Can deliver performance independent of network loading.

- Time synchronization
  - E.g., **IEEE 1588** standard set in 2002.
  - Synchronized time-of-day across a network.
Time Synchronization on Ethernet with TCP/IP: IEEE 1588 PTP

Press Release October 1, 2007

Clocks on a LAN agree on the current time of day to within 8ns, far more precise than older techniques like NTP.
An Extreme Example: The Large Hadron Collider

The WhiteRabbit project at CERN is synchronizing the clocks of computers 10 km apart to within about 80 psec using a combination of IEEE 1588 PTP and synchronous Ethernet.
The question we address:

If you assume that computers on a network can agree on the current time of day within some bounded error, how does this change how we develop distributed real-time software?

Our answer: It changes everything!

Our approach: Model-based design based on distributed discrete-event (DE) models.
Our Approach is based on Discrete Events (DE)

- Concurrent actors
- Exchange time-stamped messages ("events")

A correct execution is one where every actor reacts to input events in time-stamp order.

Time stamps are in "model time," which typically bears no relationship to "real time" (wall-clock time). We use superdense time for the time stamps.
DE Director specifies that this will be a DE model
Building a DE Model (in Ptolemy II)

Model of regularly spaced events (e.g., a clock signal).
Building a DE Model (in Ptolemy II)

Model of irregularly spaced events (e.g., a failure event).
Building a DE Model (in Ptolemy II)

Model of a subsystem that changes modes at random (event-triggered) times
Building a DE Model (in Ptolemy II)

Model of an observer subsystem
Events on the two input streams must be seen in time stamp order.
Aside:

Superdense Time Enables Better Conjunction of Computation and Physical Processes

![Graph showing the relationship between index and time with points and lines indicating different time intervals.](image)
This is a Component Technology Model of a subsystem given as an imperative program.
This is a Component Technology

Model of a subsystem given as a state machine.
This is a Component Technology

Model of a subsystem given as a modal model.

More types of components:
- Modal models
- Functional expressions.
- Submodels in DE
- Submodels in other MoCs
Using DE Semantics in Distributed Real-Time Systems

- DE is usually a simulation technology.
- Distributing DE is traditionally done for acceleration.
- Hardware design languages (e.g. VHDL) use DE where time stamps are literally interpreted as real time, or abstractly as ticks of a physical clock.

- We are using DE for distributed real-time software, binding time stamps to real time only where necessary.
- *PTIDES*: Programming Temporally Integrated Distributed Embedded Systems
PTIDES: Programming Temporally Integrated Distributed Embedded Systems

Distributed execution under discrete-event semantics, with “model time” and “real time” bound at sensors and actuators.
PTIDES uses static causality analysis to determine when events can be safely processed (preserving DE semantics).
Schedulability analysis incorporates computation times to determine whether we can guarantee that deadlines are met.

Deadline for delivery of event with time stamp $t$ here is $t - c_3 - d_2$

Assume bounded computation time $c_1$

Assume bounded computation time $c_2$

Assume bounded computation time $c_3$
... and being explicit about time delays means that we can analyze control system dynamics...

*Feedback through the physical world*
Example Application: Power Plant Controller

Top-level view of a power plant. The Supervisory Control block provides the signal to start the power plant and sets the target output levels. The Local Control block implements a simple control law to achieve the target output levels. It also detects failures to receive sensor data from the network. Upon the first such failure, it notifies the Supervisory Control block with a warning. Upon the second such failure, it locally changes the target output level to shut down the power plant, and also notifies the Supervisory Control block.

With the test data shown here, a warning occurs at time 8. The plant resumes normal operation at time 20. At time 22, an emergency occurs, and the plant shuts down.
Example Application: Power Plant Controller

The actors represent the transfer function between fuel valve setting and electrical output.

The submodel at the left represents sensors monitoring plant functions. For example, this sensor monitors the output of the turbine blades.

With the test data shown here, a warning occurs at time 8. The plant resumes normal operation at time 20. At time 22, an emergency occurs, and the plant shuts down.
Local control is a PTides component.
Leveraging Network Time Synchronization to Detect Faults
Our Experimental Setup

Analysis
- Schedulability Analysis
- Causality Analysis
- Program Analysis

HW Platform
- Software Component Library
- Code Generator
- Ptides Model
- Code
- PtidyOS

Mixed Simulator
- Plant Model
- Network Model

HW in the Loop Simulator
- Luminary Micro 8962
- IEEE 1588 Network time protocol

Ptides Domain
- Ptolemy II Ptides domain

Discrete-event, Continuous, and Wireless domains
- Ptolemy II Discrete-event, Continuous, and Wireless domains

Code
- PtidyOS

Platform
- Luminary Micro 8962

IEEE 1588 Network time protocol

Lee, Berkeley 32
Summary

- Network time synchronization is a potentially game-changing advance for distributed embedded systems.

- The PTIDES model of computation offers an attractive possible programming model for distributed cyber-physical systems.