PTIDES: Programming Temporally Integrated Distributed Embedded Systems

Yang Zhao*, Edward Lee*, Jie Liu† (*EECS, UC Berkeley, †Microsoft Research)

This work extends discrete-event models with the capability of mapping certain events to physical time and proposes them as a programming model, called PTIDES. We seek analysis tools and execution strategies that can preserve the deterministic behaviors specified in DE models without paying the penalty of totally ordered executions.

Discrete-Event (DE) Systems

- Typically used for modeling physical systems where atomic events occur on a time line. Examples:
  - VHDL
  - OPNET Modeler
  - NS-2
  - VisualSense

- Time is only a modeling property, DE systems are primarily used in performance modeling and simulation.

Time Synchronization

- Provides a convenient coordination mechanism for coordinated actions over distances.
  - NTP (standard networks, ~ms)
  - IEEE1588 (Ethernet, ~ns)
  - RBS (wireless network)

- A key question that arises in the face of such technologies is how they can change the way software is developed.

Motivating Example

- At two distinct sensor nodes A and B we need to generate precisely timed samples under the control of software. Moreover, the devices that generate these samples provide some sensor data to the software after generating the event.

- A distributed DE model to be executed on a two-sensor, time-synchronized platform A and B is shown in the right figure.

PTIDES

- Uses model time to define execution semantics, and constraints that bind certain model time events to physical time.
  - PTIDES programs are constructed as networks of actors.
  - The interface of actors contains ports.
  - Designate a subset of the input ports to be real-time ports. Time-stamped events must be delivered to these ports before physical-time exceeds the time stamp.

- The global notion of time that is intrinsic in DE models is used as a binding coordination agent.
- The focus here is not about speed of execution but rather about timing determinism.

Relevant Dependency Analysis

Relevant dependency analysis gives a formal framework for analyzing causality relationships to determine the minimal ordering constraints on processing events. The key idea is that events only need to be processed in time-stamp order when they are causally related.

Causality Interface

- Declares the dependency that output events have on input events.
  \[ \delta : P \times P_o \rightarrow D \]

- D is an ordered set associated with the \( \min (\text{\( \square \)}) \) and plus \( (\text{\( \boxplus \)}) \) operators.

- The dependencies between any two ports in a composition can be determined by using \( (\text{\( \boxplus \)}) \) for serial composition and \( (\text{\( \square \)}) \) for parallel composition.

Towards Deployability Analysis

- A key requirement for preserving runtime determinism of PTIDES programs is that each event with model time \( t \) at a real-time port must be received before the physical time exceeds \( t = \tau \).

- When the execution time is negligible comparing to the network delays and setup time, deployability checking becomes straightforward by using the relevant order.

- A full analysis, when the execution time is not negligible is ongoing.