Efficient Schedulability Testing for PTIDES

Christos Stergiou

10th Biennial Ptolemy Miniconference
Berkeley, CA
November 7, 2013
Motivation

- Programming CPS is flawed
- Timing affects behavior & correctness
- Insufficient software abstractions
- Lack of temporal semantics
- Thesis: time has to be a first-class citizen in CPS programming
Bosch Rexroth Printing Press

Photos from www.boschrexroth.com
Discrete-Event Systems

\[ (t_0 + P, v) \]
Discrete-Event Systems

\[ (t_0 + 2P, v) \]
Discrete-Event Systems

$D_1(t_0 + 3P, \nu)$

$D_2$

$C$

Display
Discrete-Event Systems

Clock

$\text{Display}$

$D_1$

$D_2$

$C$

$e(t_0, v)$
Discrete-Event Systems

Clock

$D_1$

$D_2$

$C$

Display

$e(t_1 = t_0 + D_1, v)$
\[ C(e(t_1 = t_0 + D_1, v)) \]

\[ C(e'(t_2, v')) \]
Discrete-Event Systems

Clock

$D_1$

$D_2$

$C$

Display

e(t_1, v'')
e'(t_2, v')
PTIDES: Sensors and Actuators

Ptides Platform

Sensor \( D \) \( C \) Actuator

real-time \( t \)
PTIDES: Sensors and Actuators

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

$t_1$ Sensor fires

real-time $t$
PTIDES: Sensors and Actuators

Ptides Platform

Sensor \( e_1(t_1) \) → \( D \) → \( C \) → Actuator

Sensor fires

\( t_1 \)

real-time \( t \)
PTIDES: Sensors and Actuators

Ptides Platform

Sensor $\rightarrow D \rightarrow e_2(t_1 + D) \rightarrow C \rightarrow$ Actuator

$t_1$

Sensor fires

real-time $t$
PTIDES: Sensors and Actuators

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow$ Actuator

Sensor fires at $t_1$ in real-time $t$. 
PTIDES: Sensors and Actuators

Ptides Platform

Sensor \rightarrow D \rightarrow C \rightarrow e_3(t_1 + D) \rightarrow \text{Actuator}

Sensor fires at $t_1 \rightarrow t_2$ in real-time $t$
PTIDES: Sensors and Actuators

Ptides Platform

Sensor $\rightarrow D \rightarrow C \rightarrow e_3(t_1 + D)$ Actuator

Sensor fires $t_1 \rightarrow t_2 \rightarrow t_1 + D$ real-time $t$

Actuator actuates
If $t_2 > t_1 + D$ then $e_3$ misses its deadline.
PTIDES: Timestamp Order

Sensor S1 → $D$ → $e(t_1 + D)$ → $C$ → Actuator A

When should $e$ be processed?
PTIDES: Timestamp Order

- When should $e$ be processed?
- After $t_1 + D$ any event that arrives at $S_2$ will have timestamp $> t_1 + D$
PTIDES: Timestamp Order

When should $e$ be processed?

- After $t_1 + D$ any event that arrives at $S_2$ will have timestamp $> t_1 + D$
- Safe-to-process analysis
PTIDES: Scheduling

\[ e_1(t_1) \rightarrow D_1 \rightarrow C_1 \rightarrow e_2(t_2) \rightarrow C_2 \rightarrow A1 \]

\[ S1 \]

\[ e_3(t_3) \rightarrow C_3 \rightarrow D_3 \rightarrow A3 \]

\[ S3 \]

\[ \text{deadline}(e_2) = t_2 \]

\[ \text{deadline}(e_3) = t_3 + D_3 \]

\[ \text{EDF with preemption} \]
PTIDES: Scheduling

S1 → $D_1$ → $C_1$ → $C_2$ → A1

S2 → $D_2$ → $C_1$ → $e_2(t_2)$ → A2

S3 → $e_3(t_3)$ → $C_3$ → $D_3$ → A3

> $\text{deadline}(e_2) = t_2$
> $\text{deadline}(e_3) = t_3 + D_3$
PTIDES: Scheduling

- deadline(e_2) = t_2
- deadline(e_3) = t_3 + D_3
- deadline(e) = t + (delay to actuators)
PTIDES: Scheduling

- deadline(e_2) = t_2
- deadline(e_3) = t_3 + D_3
- deadline(e) = t + (delay to actuators)
- EDF with preemption
Schedulability Problem

- Worst-case execution time per actor
- Models for sensor and network inputs
  - Periodic, sporadic (min. inter-arrival time)
- Schedulability problem:
  *Do all the events meet their deadlines?*
Challenges

- Difficult to identify worst-case scenario
- Two dependent objectives:
  - Processor demand
  - Safe-to-process waiting
- Expressiveness of programming model
Maximize Computation Demand

When both sensors fire at time 0 there is a deadline miss.

When $S_2$ fires so as to maximize the safe-to-process delay, no deadline is missed.
Maximize Safe-to-Process Delay

When both sensors fire at time 0 there is no deadline miss.

When $S_2$ fires so as to maximize the safe-to-process delay, a deadline is missed.
Maximize Safe-to-Process Delay

When both sensors fire at time 0, there is no deadline miss.

When $S_2$ fires so as to maximize the safe-to-process delay, a deadline is missed.

It is not easy to find a worst-case scenario for schedulability.
In previous work, we reduced the problem to reachability in TA.
Hard Real-Time Theory

- In previous work, we reduced the problem to reachability in TA
- Can we leverage traditional hard real-time theory for more efficient schedulability tests?
In previous work, we reduced the problem to reachability in TA

Can we leverage traditional hard real-time theory for more efficient schedulability tests?

We described two dependent objectives:

- Processor demand
- Safe-to-process waiting
In previous work, we reduced the problem to reachability in TA.

Can we leverage traditional hard real-time theory for more efficient schedulability tests?

We described two dependent objectives:
- Processor demand
- Safe-to-process waiting

We will try to factor the latter in the real-time task system.
Periodic and Sporadic Task Systems

- **Periodic Task**
  
  \[ P, W, D \]

  - Execution time: \( W \)
  - Absolute deadline: \( D \)
  - Start time: \( 0 \)
Periodic and Sporadic Task Systems

- **Periodic Task**

  \[ P \]
  \[ W, D \]

  Execution time: \[ W \]
  Absolute deadline: \[ D \]
  
  \[ 0 \] \[ P \] \[ P + D \]

- **Sporadic Task**

\[ I_0 \]
Periodic and Sporadic Task Systems

- **Periodic Task**

  - Execution time:
    - \( W \)
  - Absolute deadline:
    - \( D \)
  - Period:
    - \( P \)
  - Start time:
    - \( 0 \)
  - Completion time:
    - \( P + D \)

- **Sporadic Task**

  - Execution time:
    - \( W \)
  - Absolute deadline:
    - \( D \)
  - Start time:
    - \( 0 \)
Periodic and Sporadic Task Systems

- **Periodic Task**

  \[ P \quad W, D \]

  Execution time \( W \)
  Absolute deadline \( D \)
  \( W \) \( P + D \)

- **Sporadic Task**

  \[ I \quad W, D \]

  Execution time \( W \)
  Absolute deadline \( D \)
  \( W \) \( t + D \)

  \( t \geq I \)
Periodic and Sporadic Task Systems

- **Periodic Task**

  \[ W, D \]

  - Execution time: \( W \)
  - Absolute deadline: \( D \)

  - Start time: 0
  - Period: \( P \)
  - Completion time: \( P + D \)

- **Sporadic Task**

  \[ W, D \]

  - Execution time: \( W \)
  - Absolute deadline: \( D \)

  - Start time: 0
  - Interval: \( t \geq l \)
  - Completion time: \( t + D \)
Multiframe Tasks

$I$

$W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \ldots \xrightarrow{S_{N-1}} W_{N-1}, D_{N-1}$
Multiframe Tasks

\[ I \]

- \( W_0, D_0 \) \( \xrightarrow{S_1} \) \( W_1, D_1 \) \( \xrightarrow{S_2} \) \( \ldots \) \( \xrightarrow{S_{N-1}} \) \( W_{N-1}, D_{N-1} \)

\[ W_0 \]

\[ D_0 \]

\[ 0 \]
Multiframe Tasks

\[
I \quad W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \ldots \xrightarrow{S_{N-1}} W_{N-1}, D_{N-1}
\]

\[
W_0 \quad D_0 \quad W_1 \quad S_1 + D_1
\]

\[
0 \quad S_1
\]
Multiframe Tasks

\[ I \]

\[ W_0, D_0 \rightarrow W_1, D_1 \rightarrow \ldots \rightarrow W_{N-1}, D_{N-1} \]

\[
\begin{align*}
W_0 & \uparrow & D_0 & \uparrow & W_1 & \uparrow & S_1 + D_1 & \uparrow & W_2 & \uparrow & S_1 + S_2 + D_2 & \uparrow & \ldots
\end{align*}
\]

\[
\begin{align*}
0 & \uparrow & S_1 & \uparrow & S_1 + S_2 & \uparrow & \ldots
\end{align*}
\]
Multiframe Tasks

\[ I \xrightarrow{S_1} W_0, D_0 \xrightarrow{S_2} W_1, D_1 \xrightarrow{S_{N-1}} W_{N-1}, D_{N-1} \]

Below the diagram:

\[
\begin{align*}
W_0 & \rightarrow D_0 & W_1 & \rightarrow S_1 + D_1 & W_2 & \rightarrow S_1 + S_2 + D_2 ^{\ldots} \\
0 & \rightarrow S_1 & S_1 + S_2 & \ldots \\
\end{align*}
\]

\[
\begin{align*}
W_0 & \rightarrow t + D_0 & W_1 & \rightarrow t + S_1 + D_1 ^{\ldots} \\
t \geq l & \rightarrow t + S_1 & \\
\end{align*}
\]
Multiframe Tasks

\[ I \]

\[ W_0, D_0 \xrightarrow{S_1} W_1, D_1 \xrightarrow{S_2} \cdots \xrightarrow{S_{N-1}} W_{N-1}, \]

\[ D_{N-1} \]

\[ W_0 \]

\[ D_0 \]

\[ W_1 \]

\[ S_1 + D_1 \]

\[ W_2 \]

\[ S_1 + S_2 + D_2 \]

\[ \cdots \]

\[ 0 \]

\[ S_1 \]

\[ S_1 + S_2 \]

\[ t \geq I \]

\[ t + D_0 \]

\[ t + S_1 + D_1 \]

\[ \cdots \]
PTIDES as Multiframe Tasks

\[ S \rightarrow D \rightarrow W \rightarrow A \]
PTIDES as Multiframe Tasks

Easy for parallel chains of actors
PTIDES as Multiframe Tasks

- Easy for parallel chains of actors
- What about merging and splitting paths?
Merging Paths

\[ S_1 \rightarrow D_1 \rightarrow W_1 \rightarrow e(t+D_1) \rightarrow D_3 \rightarrow W_3 \rightarrow A_1 \]

\[ S_2 \rightarrow W_2 \]

- Event \( e \) is safe to process at \( t' \geq t + D_1 \)
Merging Paths

\[ S_1 \rightarrow D_1 \rightarrow W_1 \rightarrow e(t+D_1) \rightarrow D_3 \rightarrow W_3 \rightarrow A_1 \]

\[ S_2 \rightarrow W_2 \]

\[ I_1 \]

\[ W_1 \]

\[ D_1 + D_3 \]

\[ W_3 \]

\[ D_3 \]

\[ D_1 \]

Event \( e \) is safe to process at \( t' \geq t + D_1 \).

If it is not safe at \( t + D_1 \), \( W_2 \) is executing an event with smaller deadline than \( e \).

Under EDF, "task" can be released at \( t + D_1 \).
Merging Paths

- $S_1 \rightarrow D_1 \rightarrow W_1 \rightarrow e(t+D_1) \rightarrow D_3 \rightarrow W_3 \rightarrow A_1$

- $S_2 \rightarrow W_2$

- Under EDF, "task" can be released at $t+D_1$.

- $W_2 = W_1 + D_3$

- $D_1 + D_3$ is the Deadline of $W_1$.
Splitting Paths

\[ S_1 \rightarrow D_1 \rightarrow W_1 \rightarrow D_2 \rightarrow W_2 \rightarrow A_1 \]

\[ D_2 < D_3 \]

\[ D_1 + D_2 \rightarrow 0 \rightarrow D_1 + D_3 \]

\[ W_1 + W_2 \rightarrow W_3 \]
Acyclic Graphs

- Construct one multiframe task per source
- Release-time for each path to an actor
  - $\text{delay}(\text{path}) - \text{offset}(\text{actor})$
  - Lower bound of actual release-time
  - An event of smaller or equal deadline is available if task is not released at that time
- Traverse graph starting from source
- Process channels in increasing release-time
Schedulability with Multiframe Tasks

- Schedulability for multiframe tasks
  - Sporadic input sources
  - EDF
  - Pseudo-polynomial time
- Limitations of reduction
  - Input-agnostic safe-to-process analysis
  - Simultaneous events
  - Loops
Ptolemy Implementation
Ptolemy Implementation
Ptolemy Implementation
Summary

Discrete-Event Systems

PTIDES

Platform

Multiframe Tasks

Schedulability Analysis
Thank you

Questions?