Schedulability and Verification of Real-Time Discrete-Event Systems

Christos Stergiou

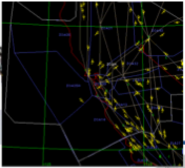
University of California, Berkeley

September 16, 2013

Cyber-Physical System(CPS):

Orchestrating networked computational resources with physical systems

Avionics

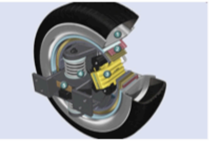


Transportation (Air traffic control at SFO)

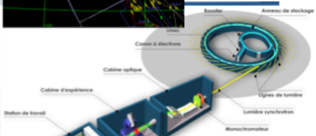
Building Systems

Telecommunications

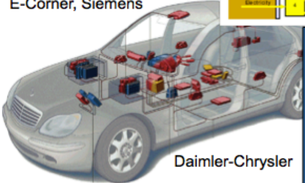
Automotive



E-Corner, Siemens



Instrumentation (Soleil Synchrotron)

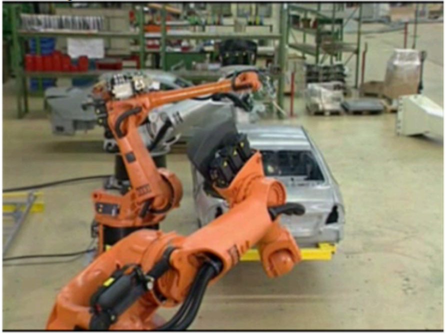


Daimler-Chrysler

Power generation and distribution

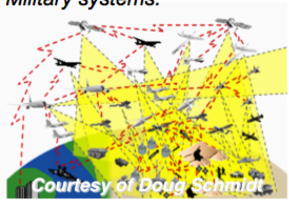
Courtesy of General Electric

Factory automation



Courtesy of Kuka Robotics Corp.

Military systems:

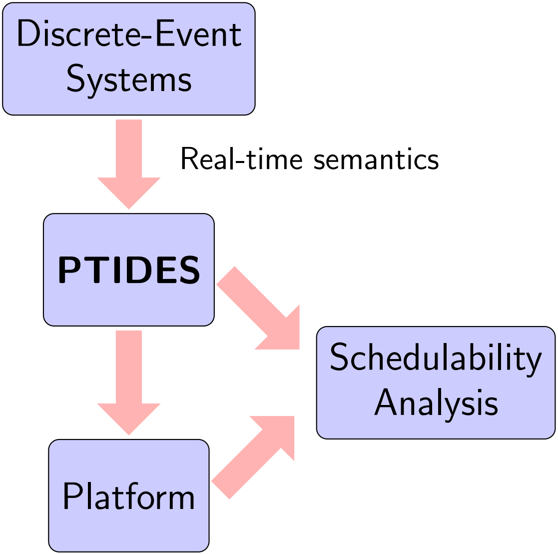


Courtesy of Doug Schmidt

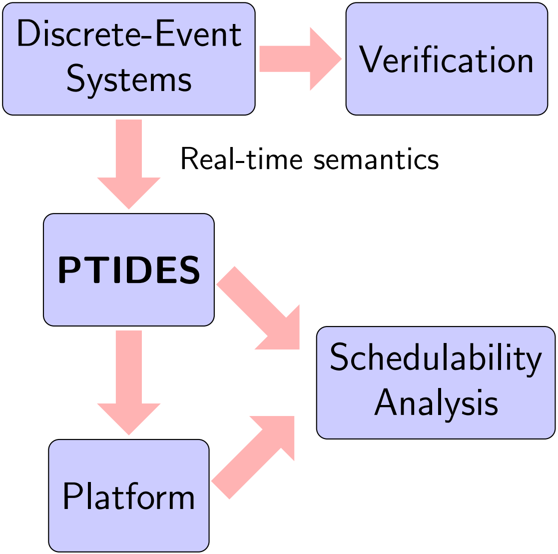
Motivation

- 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

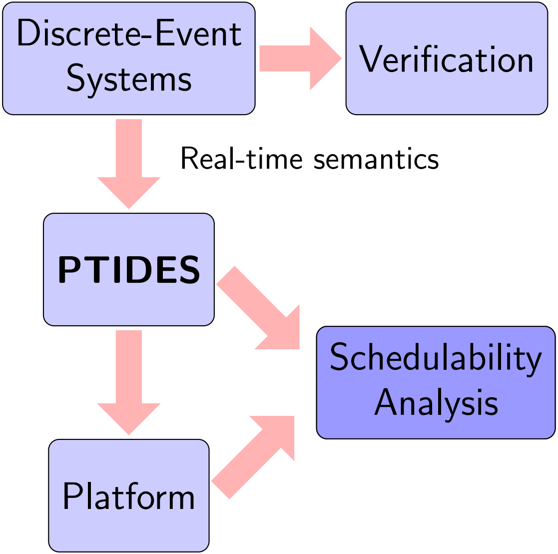
PTIDES


Discrete-Event Systems
Real-time semantics
PTIDES
Schedulability Analysis
Platform

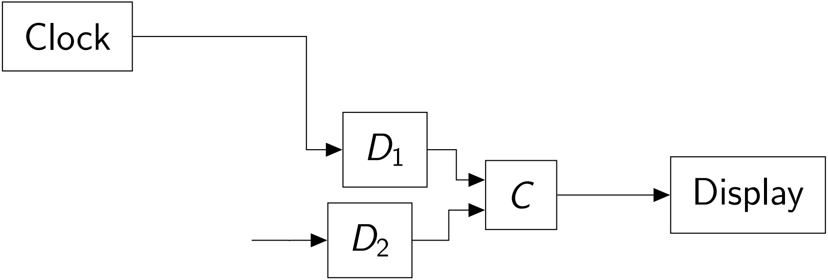
PTIDES


Discrete-Event Systems
Verification
Real-time semantics
PTIDES
Schedulability Analysis
Platform

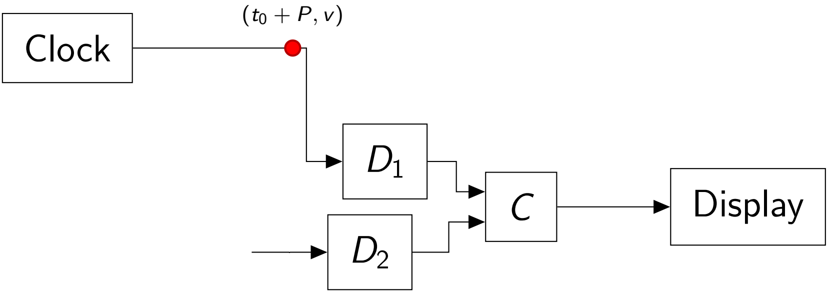
PTIDES


Discrete-Event Systems
Verification
Real-time semantics
PTIDES
Schedulability Analysis
Platform

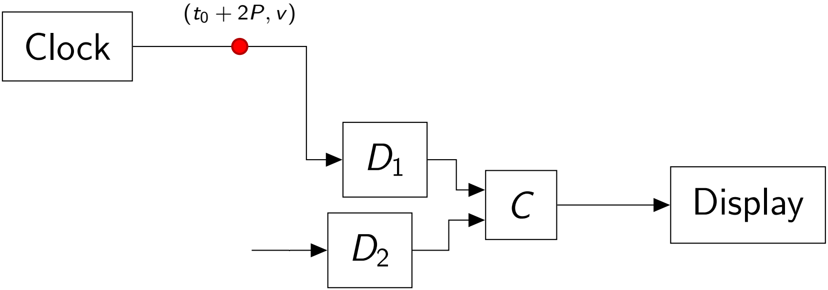
Discrete-Event Systems



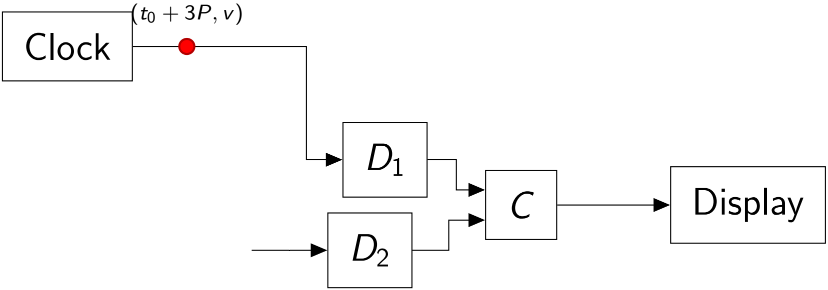
Discrete-Event Systems



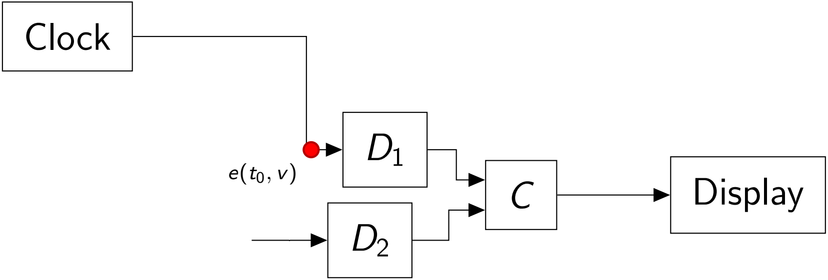
Discrete-Event Systems



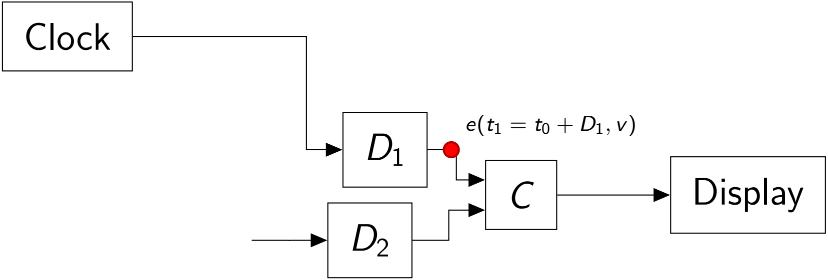
Discrete-Event Systems



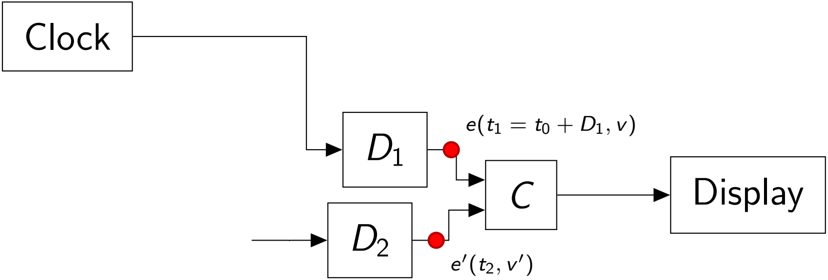
Discrete-Event Systems



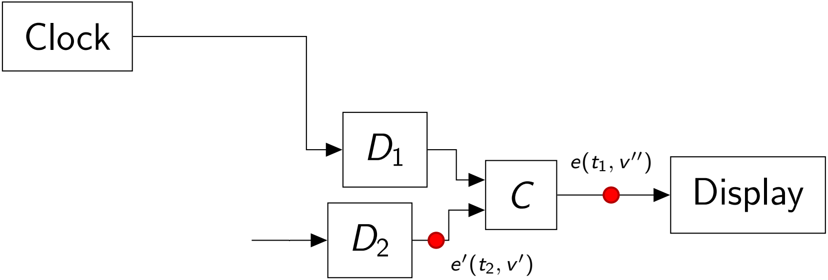
Discrete-Event Systems



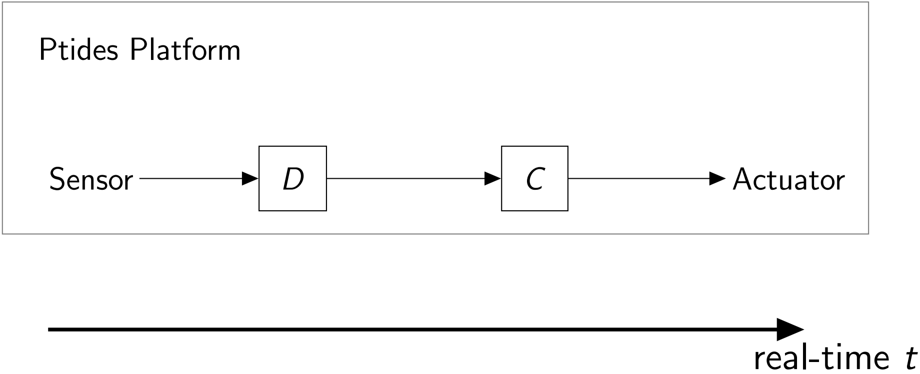
Discrete-Event Systems



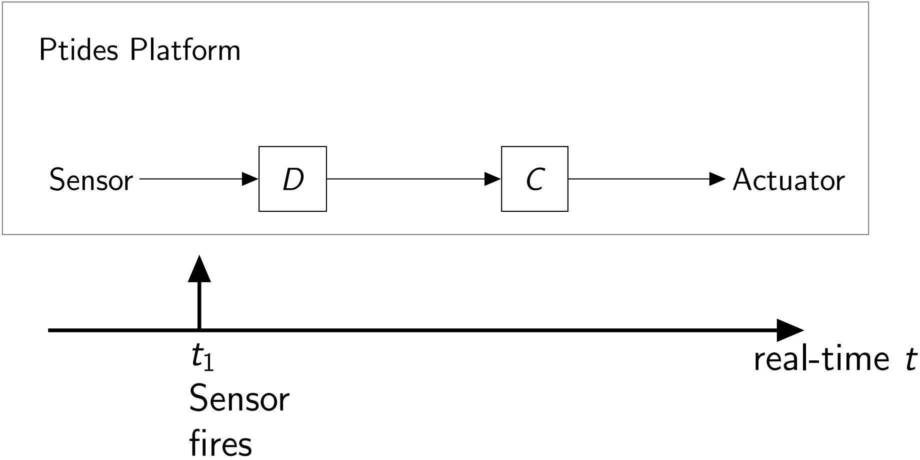
Discrete-Event Systems



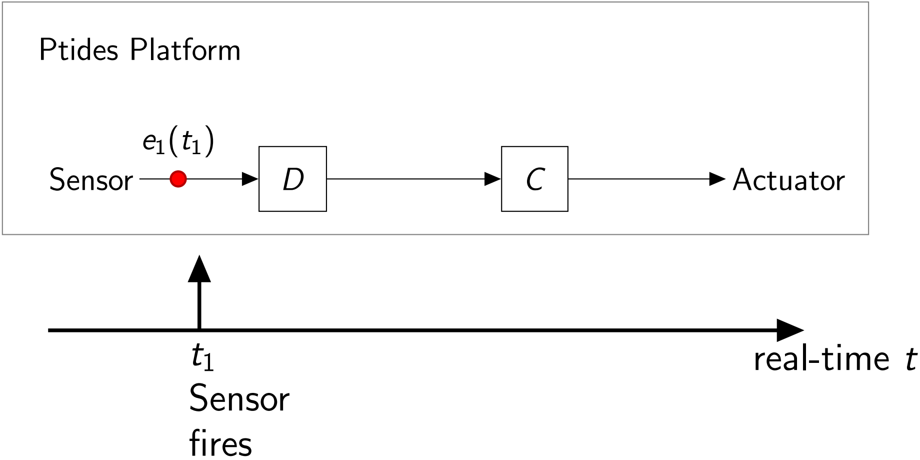
PTIDES: Sensors and Actuators



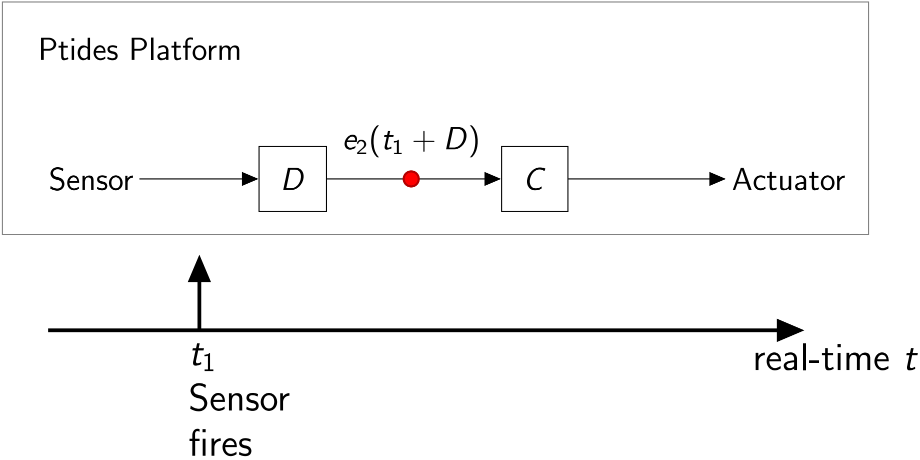
PTIDES: Sensors and Actuators



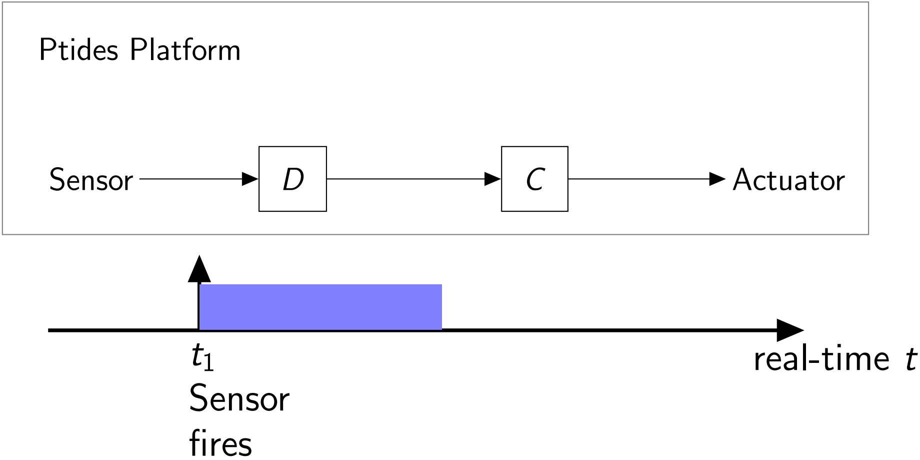
PTIDES: Sensors and Actuators



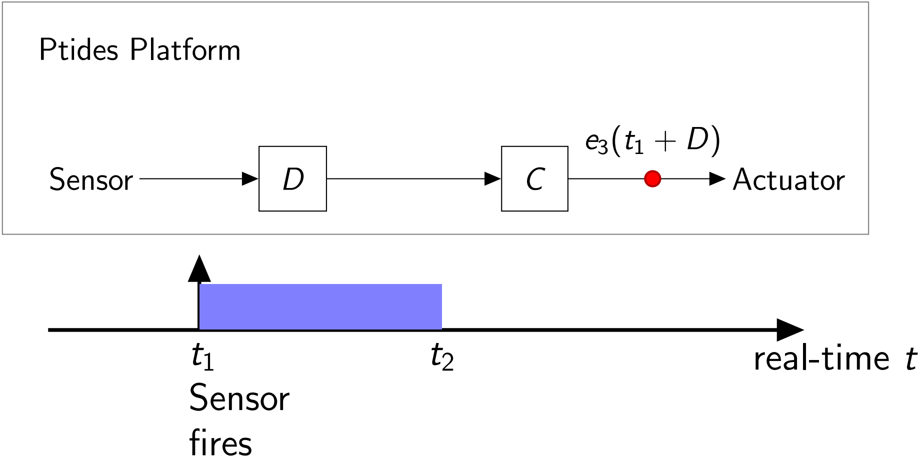
PTIDES: Sensors and Actuators



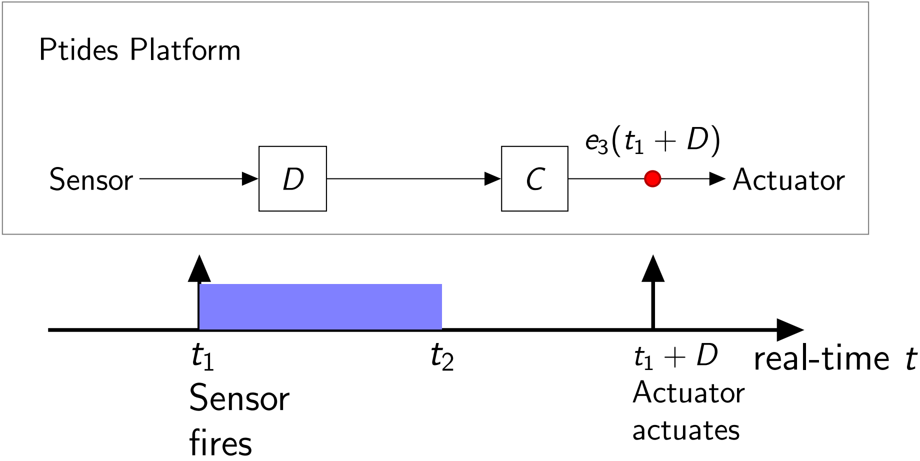
PTIDES: Sensors and Actuators



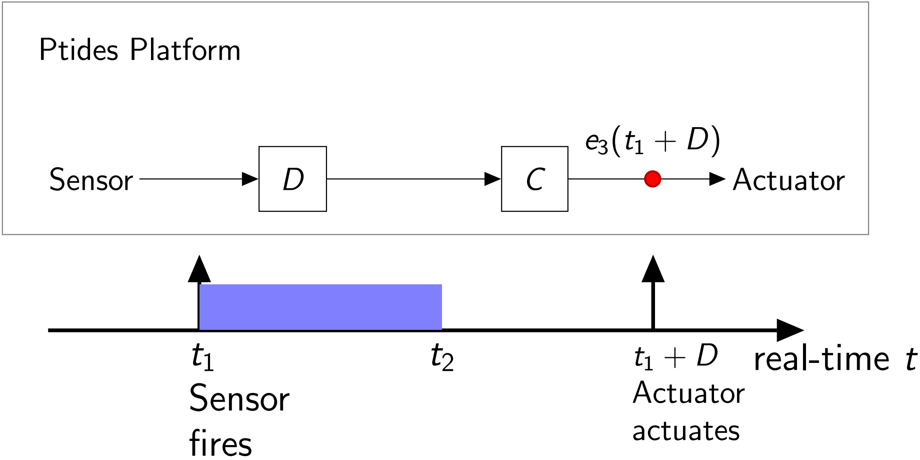
PTIDES: Sensors and Actuators



PTIDES: Sensors and Actuators

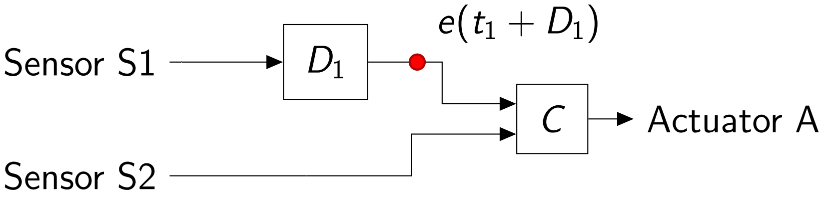


PTIDES: Sensors and Actuators



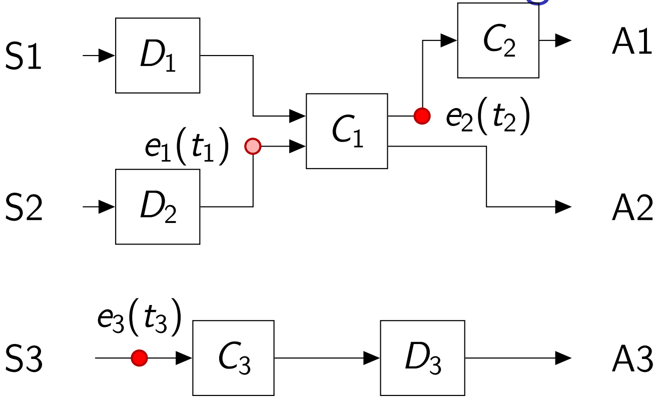
If $t_2 > t_1 + D$ then e_3 misses its *deadline*

PTIDES: Timestamp Order

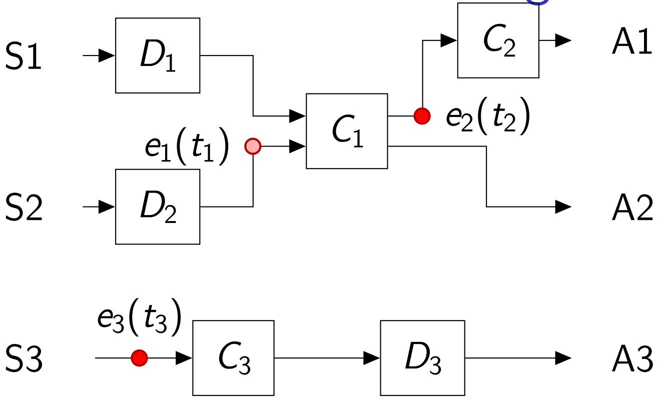


- When should e be processed?
- After $t_1 + D_1$ any event that arrives at S_2 will have timestamp $> t_1 + D_1$
- Safe-to-process analysis

PTIDES: Scheduling

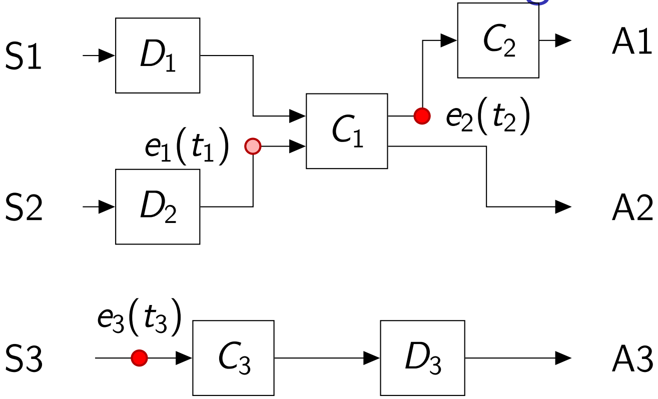


PTIDES: Scheduling



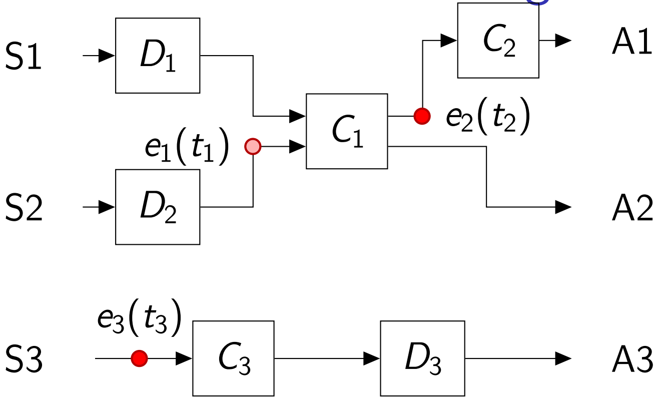
- $\text{deadline}(e_2) = t_2$
- $\text{deadline}(e_3) = t_3 + D_3$

PTIDES: Scheduling



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

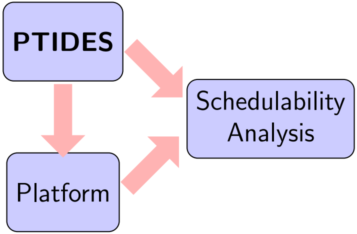
PTIDES: Scheduling



- $\text{deadline}(e_2) = t_2$
- $\text{deadline}(e_3) = t_3 + D_3$
- $\text{deadline}(e) = t + (\text{delay to actuators})$
- 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:
  Does the program always meet its deadlines?



Challenges

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

Our Approach

- Address infinite state space
  - Real-time and timestamps
  - Number of events
- Reduce schedulability to reachability in timed automata
  - Implement DE semantics
  - Simulate EDF with preemption

Real-time & Timestamps

- Real-time and timestamps can grow without bound
- Their absolute value is not necessary for execution
- Difference between timestamp and real-time is sufficient for PTIDES semantics
  - Discrete-event semantics and safe-to-process
  - EDF scheduling, i.e., compare deadlines
  - Deadline misses

Relative Timestamps

- Relative timestamp, timestamp - real-time: $\tau - t$
- Starts at 0
- Decreases continuously as real-time advances
- Makes discrete jumps when an event is processed by delay actor
- Has to be ≥ 0 when an event reaches an actuator

Bounding relative timestamps

- Find L, U such that $L \leq \tau - t \leq U$

Bounding relative timestamps

- Find L, U such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time t grow unboundedly relative to a timestamp τ?

Bounding relative timestamps

- Find L, U such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time t grow unboundedly relative to a timestamp τ?
  - No: $t \leq \tau +$ (delay to actuators)
    or else deadline miss

Bounding relative timestamps

- Find L, U such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time t grow unboundedly relative to a timestamp τ?
  - No: $t \leq \tau +$ (delay to actuators)
    or else deadline miss
- $\tau - t \leq U$
  - Can a timestamp grow unboundedly relative to real-time?

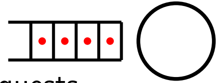
Bounding relative timestamps

- Find L, U such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time t grow unboundedly relative to a timestamp τ?
  - No: $t \leq \tau +$ (delay to actuators)
    or else deadline miss
- $\tau - t \leq U$
  - Can a timestamp grow unboundedly relative to real-time?
  - No: $\tau \leq t +$ (delay from sensors)
    or else we could violate timestamp order

Bounding relative timestamps

- Find L, U such that $L \leq \tau - t \leq U$
- $L \leq \tau - t$
  - Can real-time t grow unboundedly relative to a timestamp τ?
  - No: $t \leq \tau +$ (delay to actuators) or else deadline miss
- $\tau - t \leq U$
  - Can a timestamp grow unboundedly relative to real-time?
  - No: $\tau \leq t +$ (delay from sensors) or else we could violate timestamp order
- $-$(delay to actuators) $\leq \tau - t \leq$ (delay from sensors)

Queue-size bounds (1)



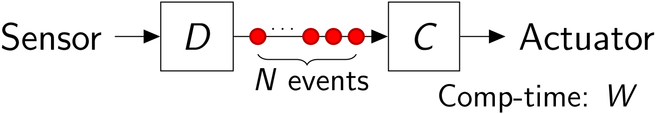
N requests

Execution time: W
Relative deadline: D

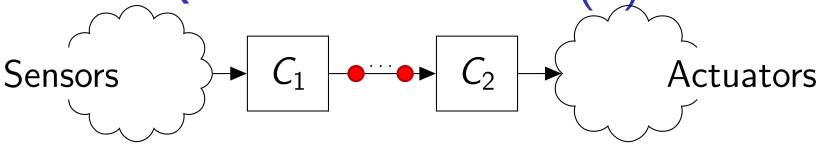
- How big can N be?
- If a request arrives at t, its deadline is $t + D$
- Total execution time of N events is $N \cdot W$
- $N \leq \left\lceil \frac{D}{W} \right\rceil$

Queue-size bounds (1)

- How big can N be?
- If a request arrives at t, its deadline is $t + D$
- Total execution time of N events is $N \cdot W$
- $N \leq \lceil \frac{D}{W} \rceil$



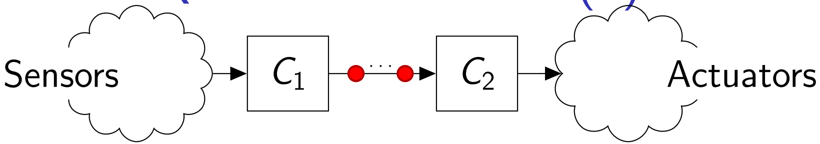
Queue-size bounds (2)



- Absolute deadline of event with timestamp τ:

$$\tau + \text{(delay to actuators)}$$

Queue-size bounds (2)



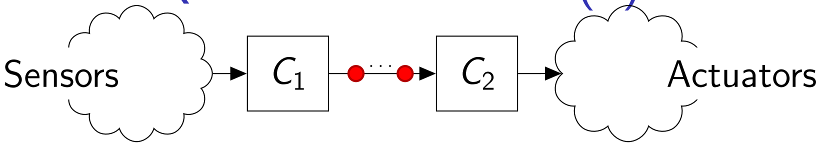
- Absolute deadline of event with timestamp τ:

$$\tau + \text{(delay to actuators)}$$

- Relative deadline associated with channel is:

$$\tau - t + \text{delay}(C_2, \text{actuators})$$

Queue-size bounds (2)



- Absolute deadline of event with timestamp τ:

$$\tau + (\text{delay to actuators})$$

- Relative deadline associated with channel is:

$$\tau - t + \text{delay}(C_2, \text{actuators})$$

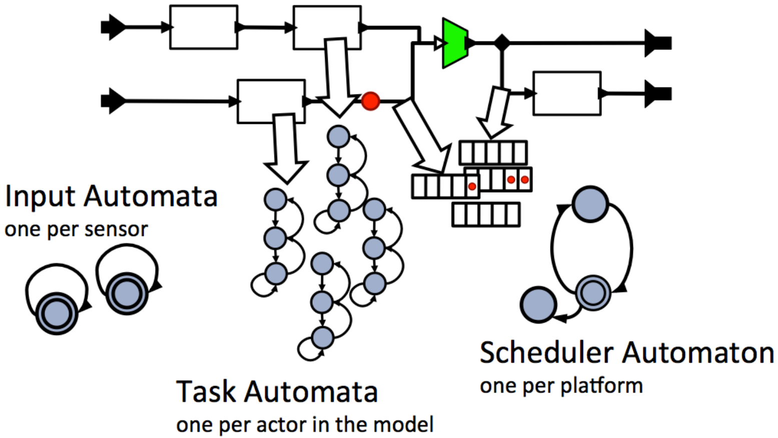
- Upper bound on relative deadline

$$(\text{delay from sensors}) + (\text{delay to actuators})$$

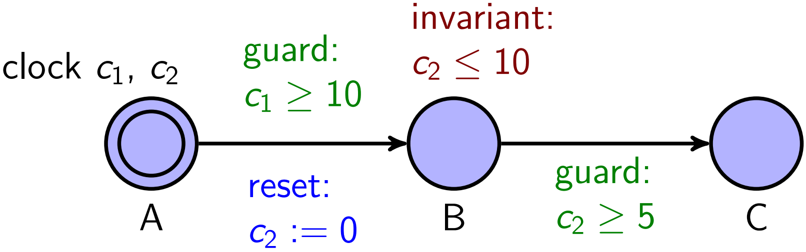
Our Approach

- Address infinite state space
  - Real-time and timestamps
  - Number of events
- Reduce schedulability to reachability in timed automata
  - Implement DE semantics
  - Simulate EDF with preemption

Schedulability using Timed Automata



Timed Automata



- Finite automata + finite set of real-valued clocks
- Time elapses at locations
- Clocks can be reset on transitions
- Guards: clock constraints on transitions
- Invariants: clock constraints on locations

TA example

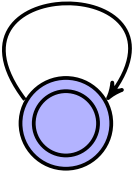
Periodic Source

clock c;

const period;

guard: $c = \text{period}$

reset: $c := 0$



invariant: $c \leq \text{period}$

TA example

Sporadic

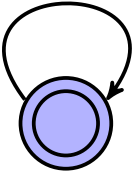
~~Periodic~~ Source

clock c;

const period;

guard: $c \cancel{=} \geq$ period

reset: $c := 0$

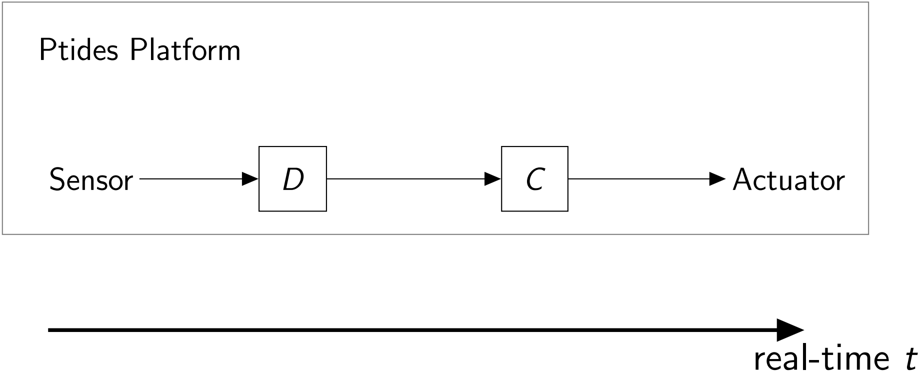


~~invariant: $c \leq$ period~~

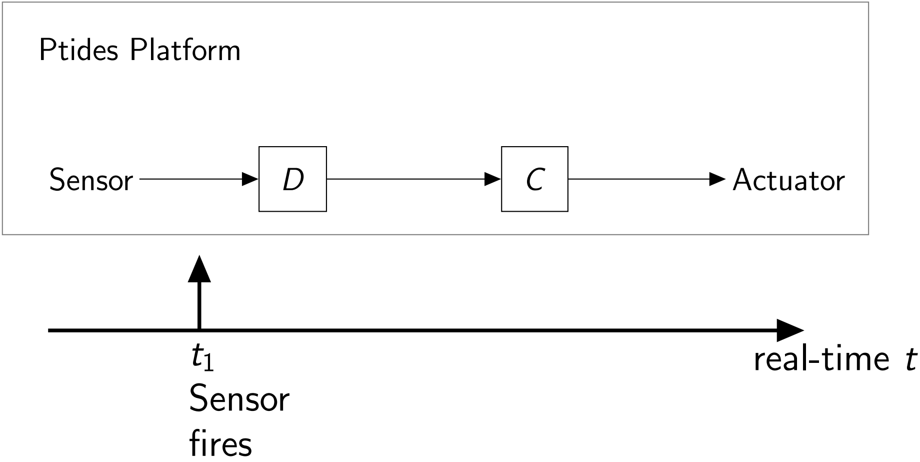
Timestamps with clocks (1)

- Associate a clock c and a discrete variable d with each event
- Reset clock when event enters platform
- c measures the relative time in the platform: $t - c =$ time the event entered platform
- d accumulates the delay added to the event: $t - c + d =$ timestamp of event

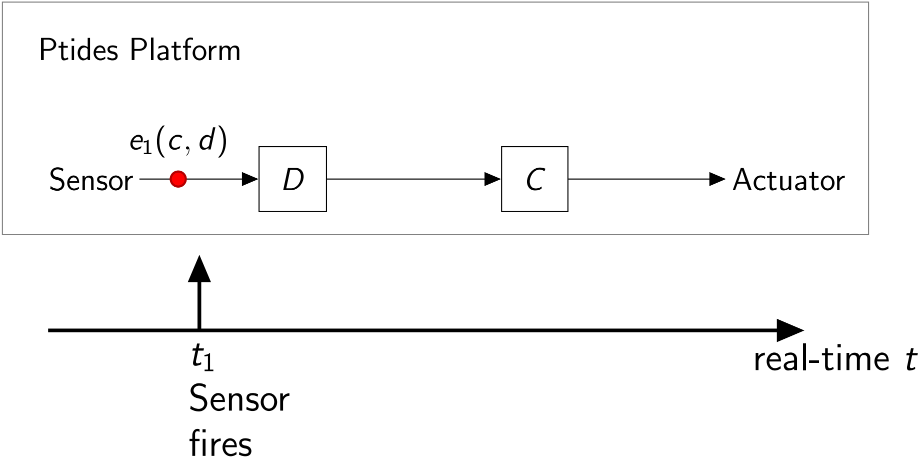
Timestamps with clocks (2)



Timestamps with clocks (2)

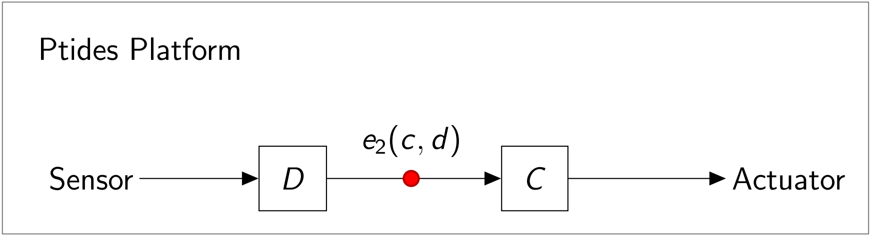


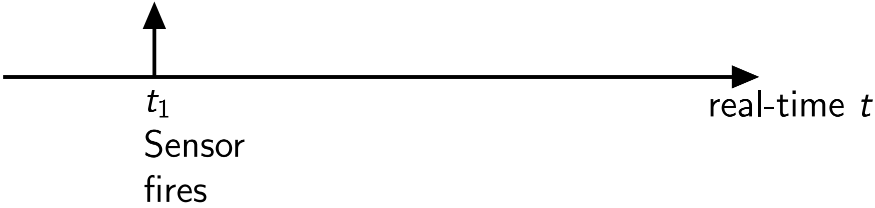
Timestamps with clocks (2)



$c := 0, d := 0$

Timestamps with clocks (2)

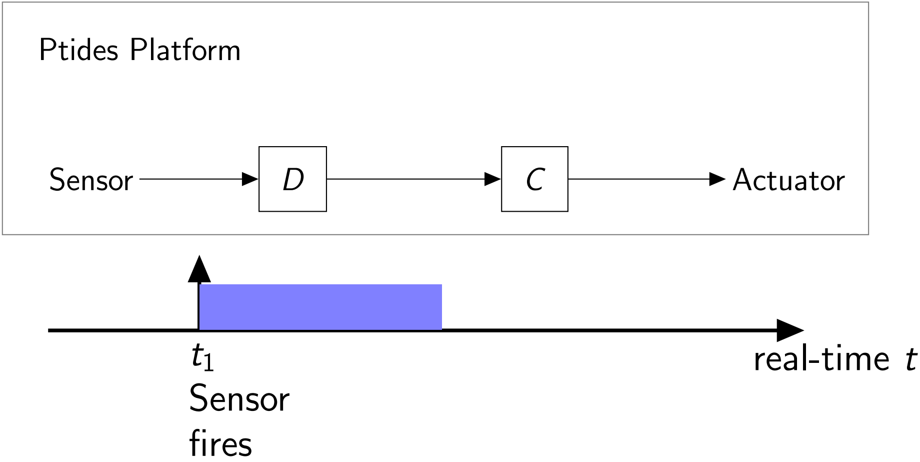




$c := 0, d := 0$

$d := d + D$

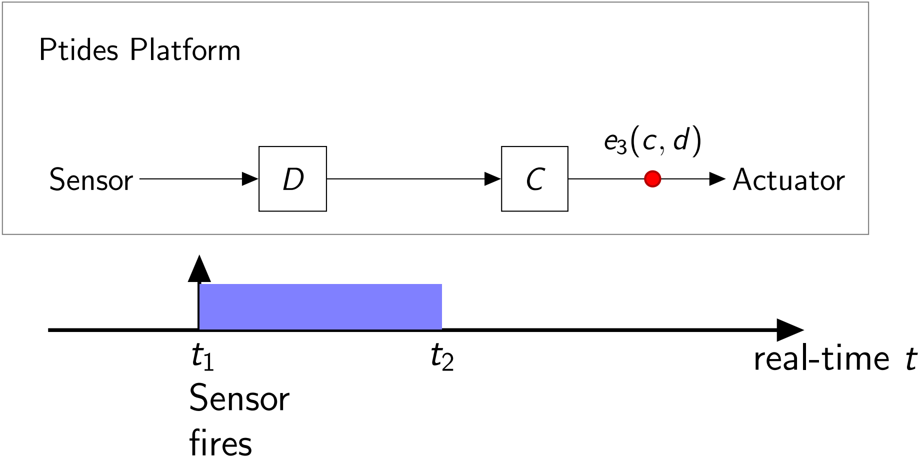
Timestamps with clocks (2)



$$c := 0, d := 0$$

$$d := d + D$$

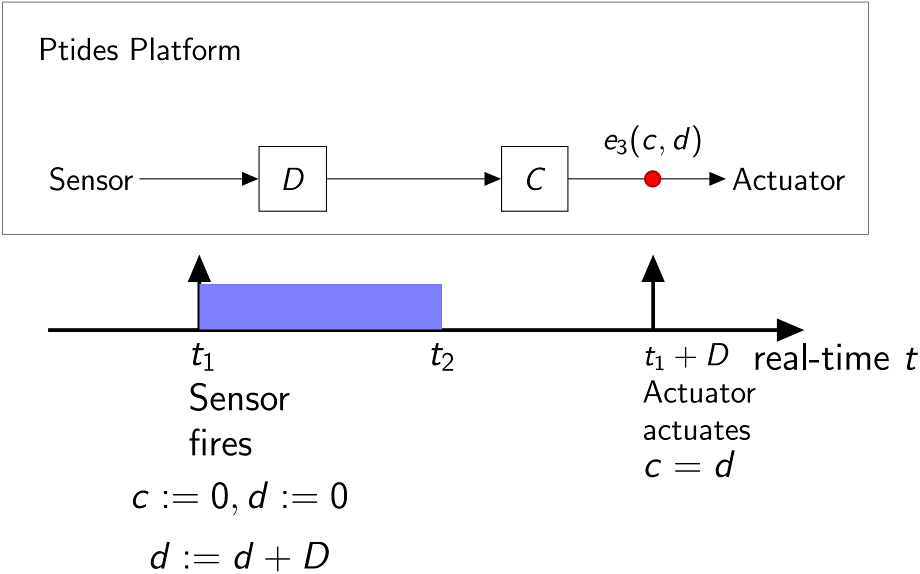
Timestamps with clocks (2)



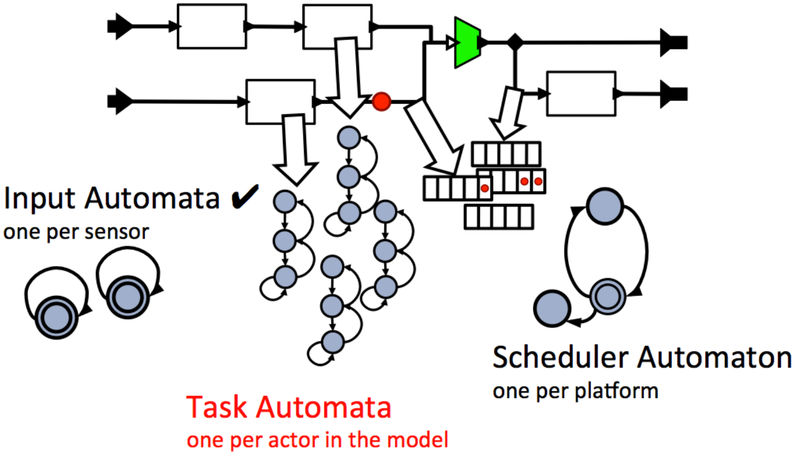
$c := 0, d := 0$

$d := d + D$

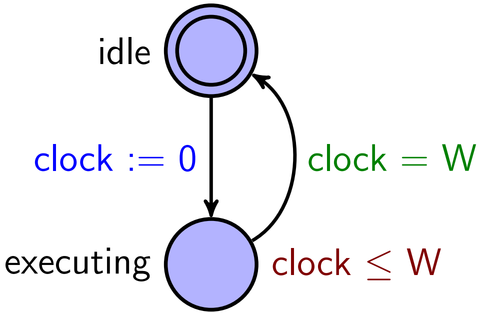
Timestamps with clocks (2)



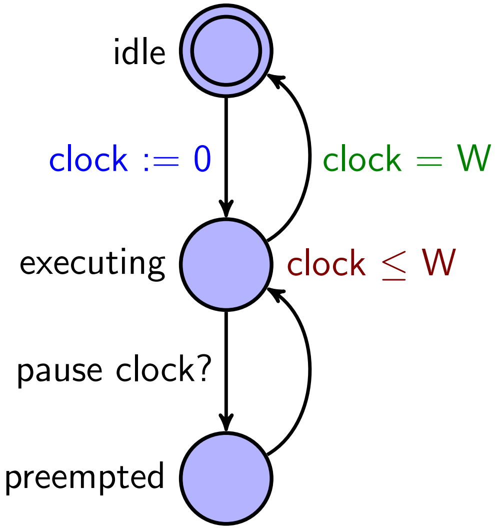
Schedulability using Timed Automata



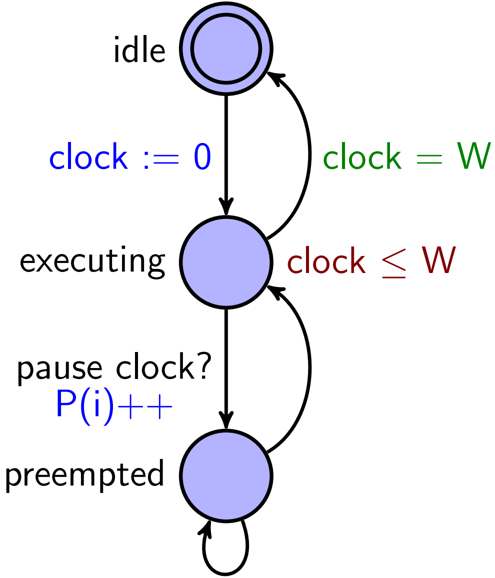
Task Automaton



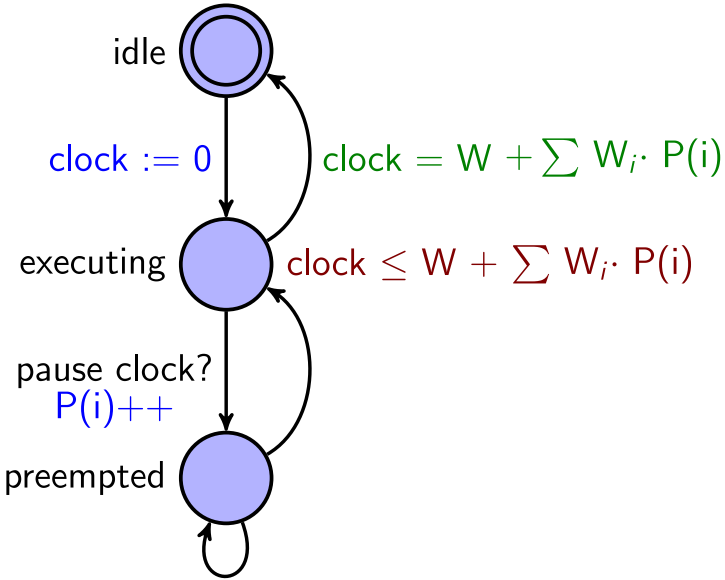
Task Automaton



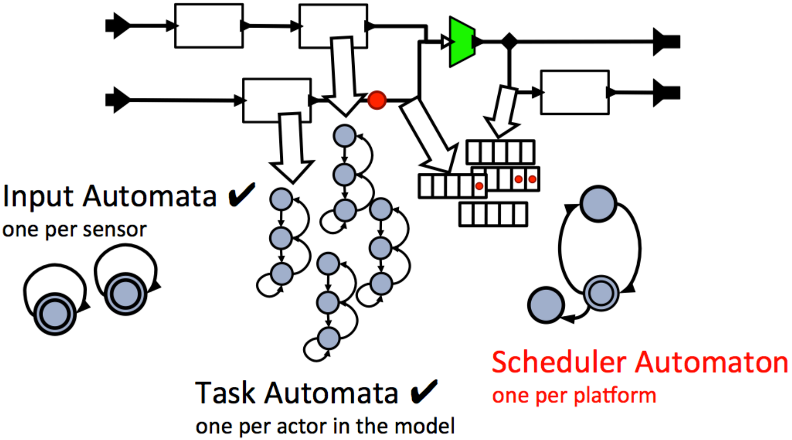
Task Automaton



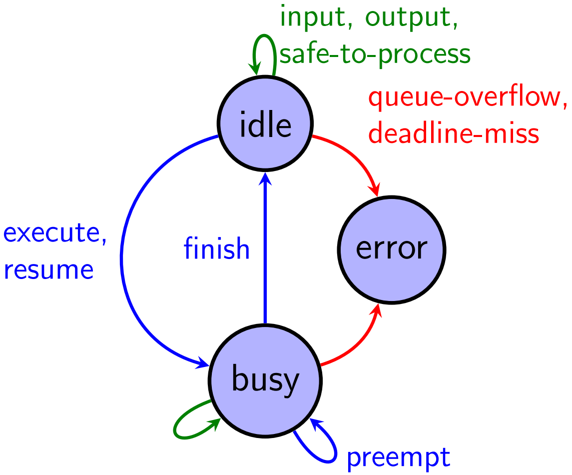
Task Automaton



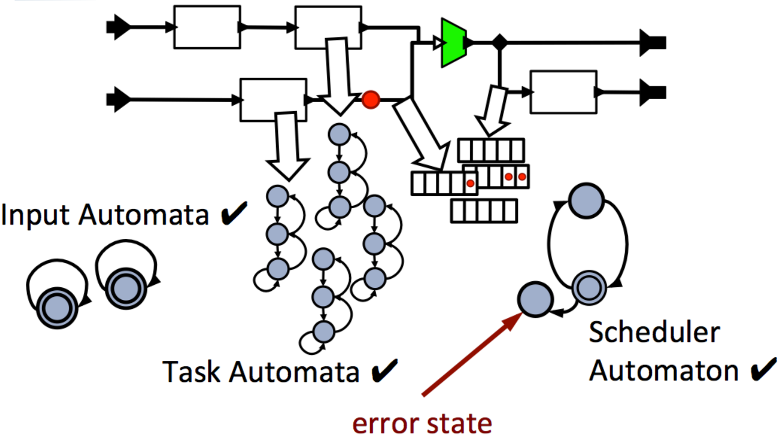
Schedulability using Timed Automata



Scheduler automaton



Schedulability using Timed Automata



E<> (Scheduler.deadline miss state)

Our Approach

- Address infinite state space
  - Real-time and timestamps
  - Number of events
- Reduce schedulability to reachability in timed automata
  - Implement DE semantics
  - Simulate EDF with preemption

Hard Real-Time Theory

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

Hard Real-Time Theory

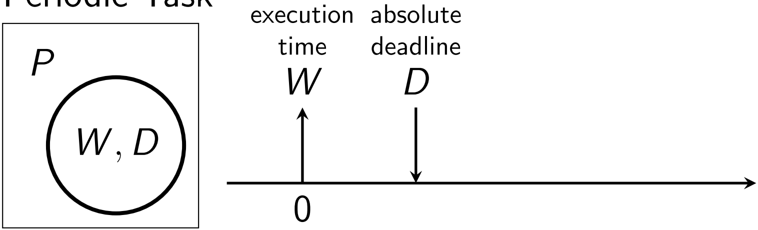
- Can we leverage traditional hard real-time theory for more efficient and sufficient schedulability tests?
- We described two dependent objectives:
  - Processor demand
  - Safe-to-process waiting

Hard Real-Time Theory

- Can we leverage traditional hard real-time theory for more efficient and sufficient 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

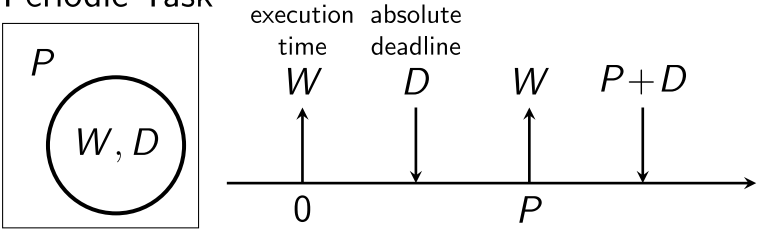
Periodic and Sporadic Task Systems

- Periodic Task



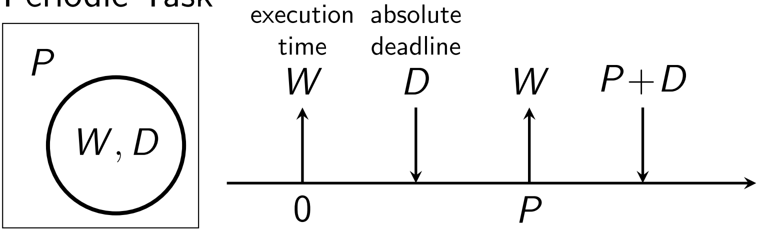
Periodic and Sporadic Task Systems

- Periodic Task

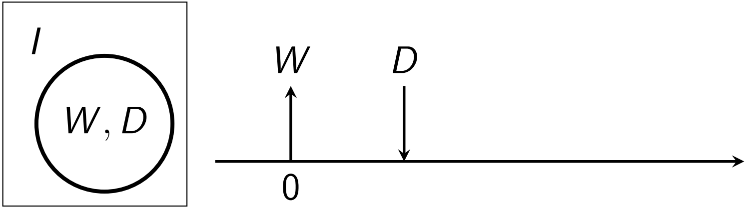


Periodic and Sporadic Task Systems

- Periodic Task

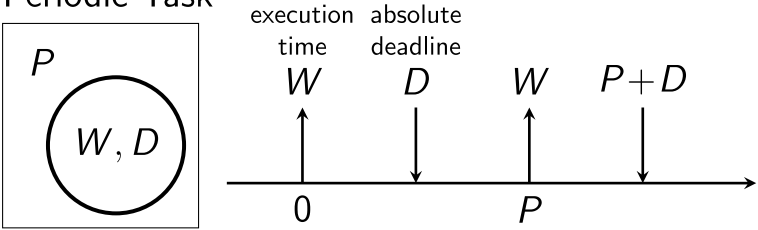


- Sporadic Task

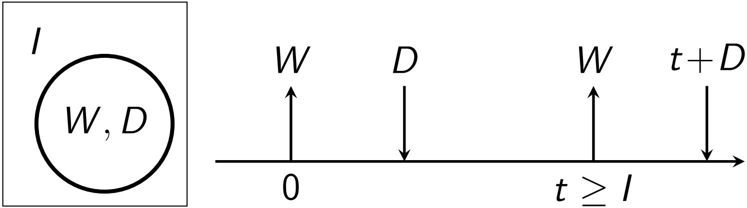


Periodic and Sporadic Task Systems

- Periodic Task

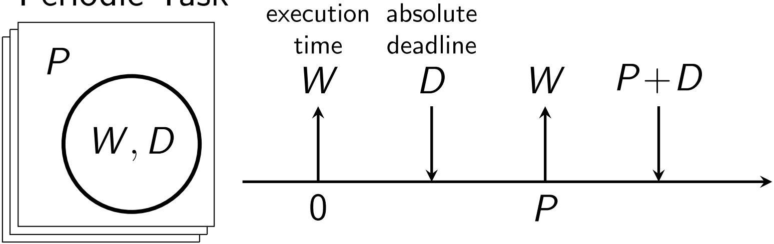


- Sporadic Task

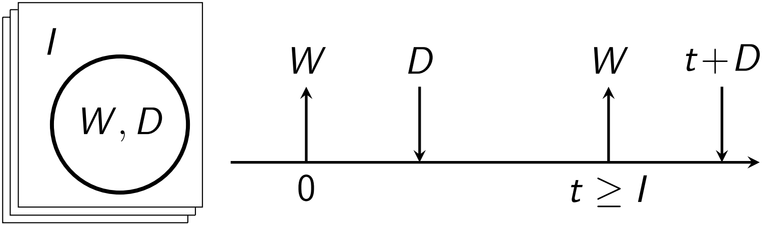


Periodic and Sporadic Task Systems

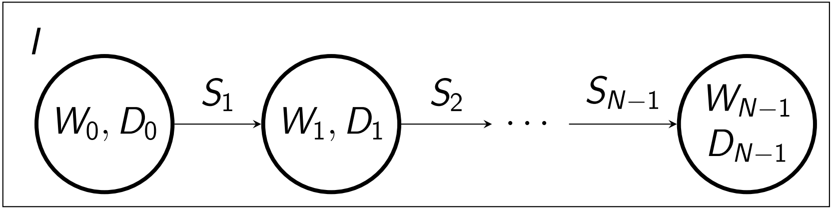
- Periodic Task



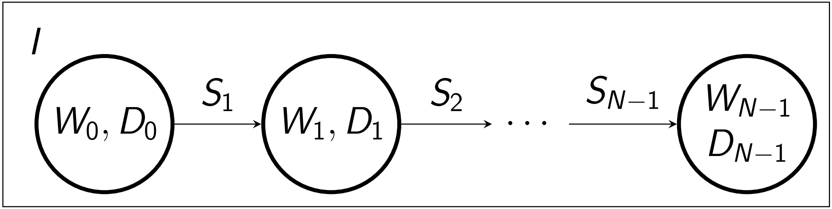
- Sporadic Task



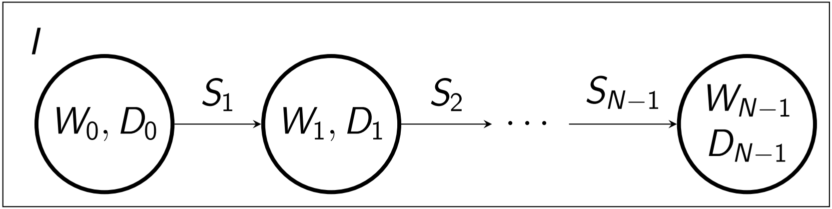
Multiframe Tasks

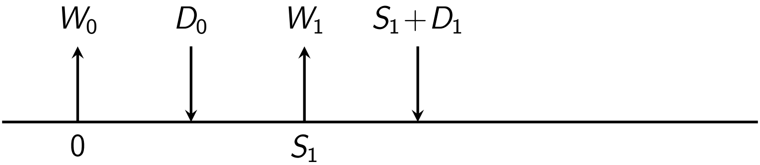


Multiframe Tasks

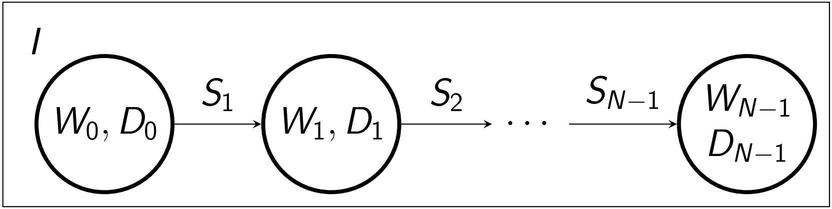


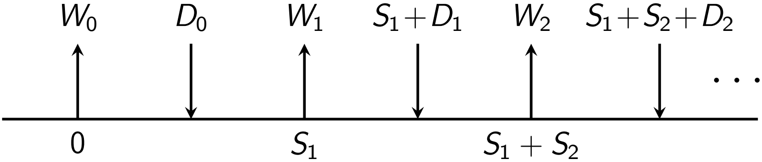
Multiframe Tasks



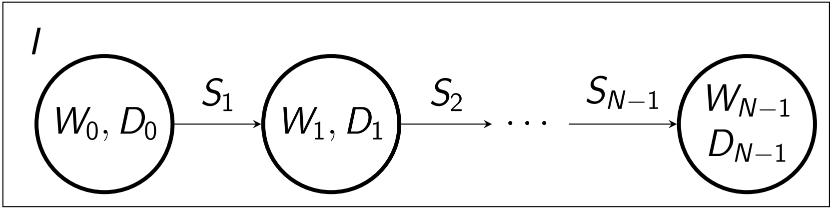


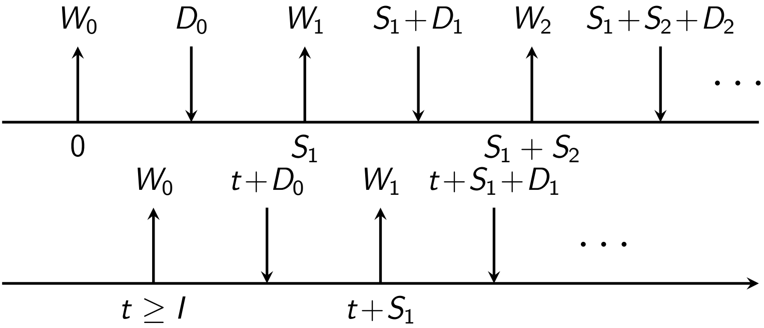
Multiframe Tasks



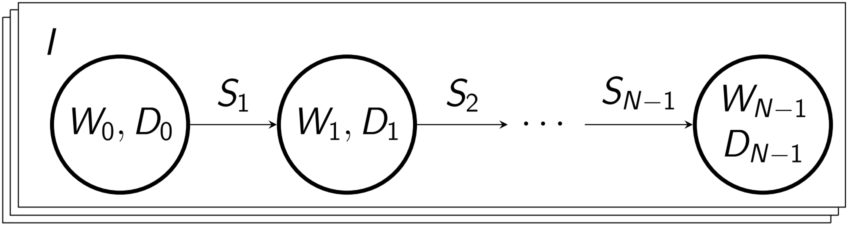


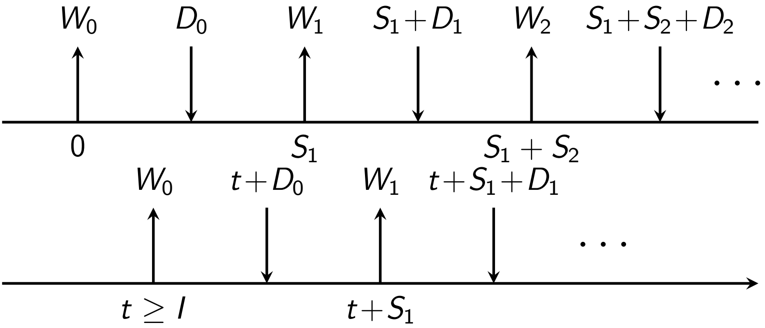
Multiframe Tasks



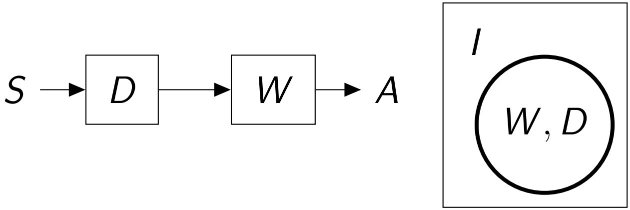


Multiframe Tasks



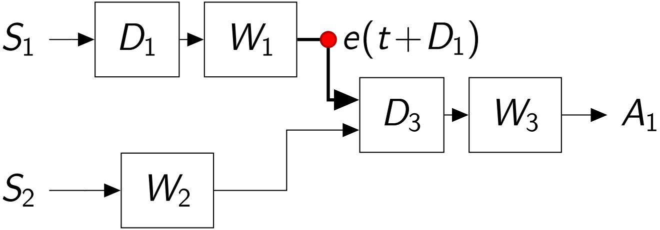


PTIDES as multiframe tasks



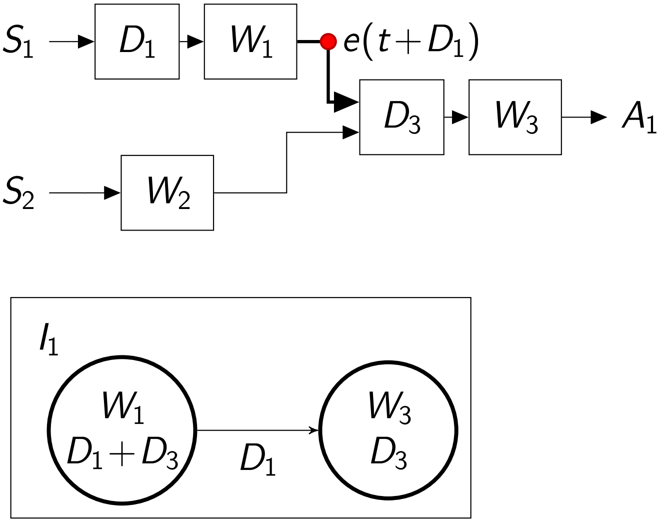
- Reduction to tasks is easy for parallel chains of actors
- What about merging and splitting paths?

Merging as multiframe tasks

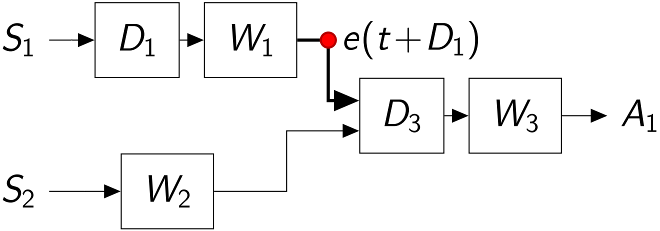


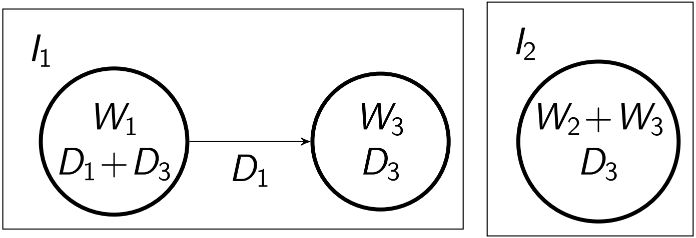
- Event e is safe to process at $t' \geq t + D_1$

Merging as multiframe tasks



Merging as multiframe tasks

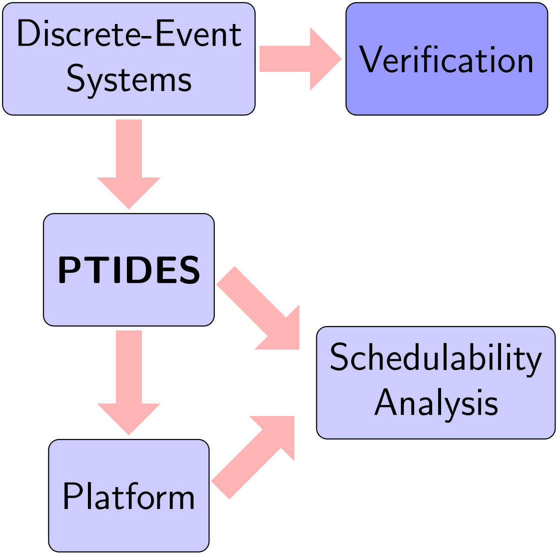




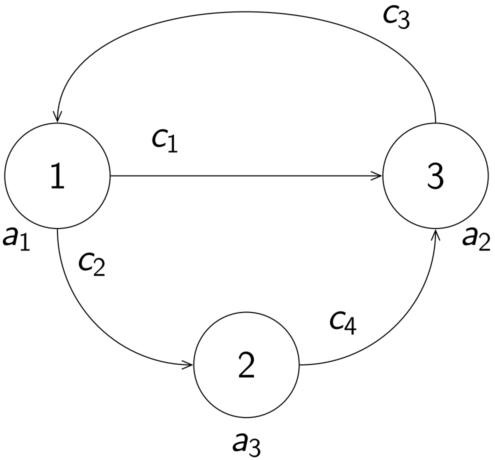
Schedulability as multiframe tasks

- Statically compute lower bounds for release time of tasks
- Reduce to schedulability of multiframe tasks:
  - EDF
  - Sporadic input sources
  - Input-agnostic safe-to-process analysis

PTIDES

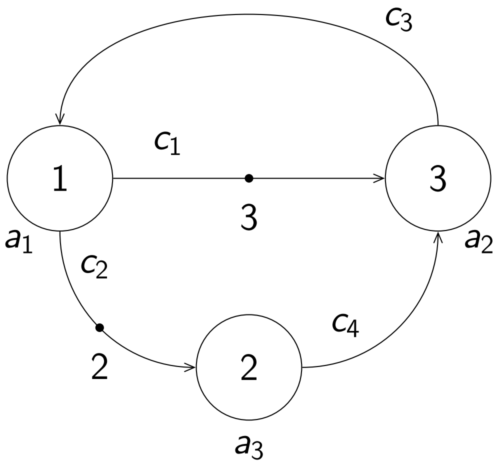

Discrete-Event Systems
Verification
PTIDES
Schedulability Analysis
Platform

Example DE



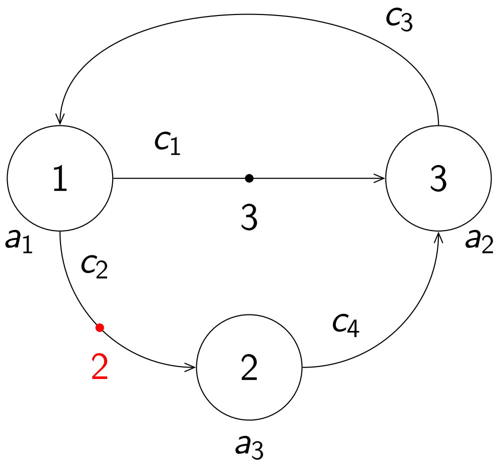
t: 0

Example DE



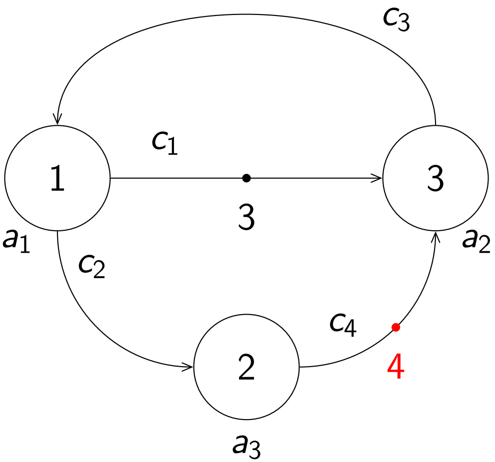
t: 0

Example DE



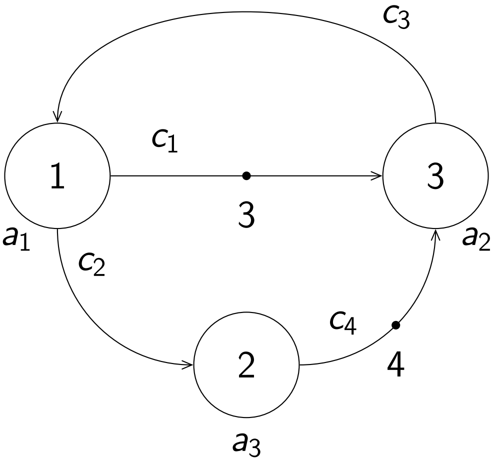
t: $0 \rightarrow 2$

Example DE



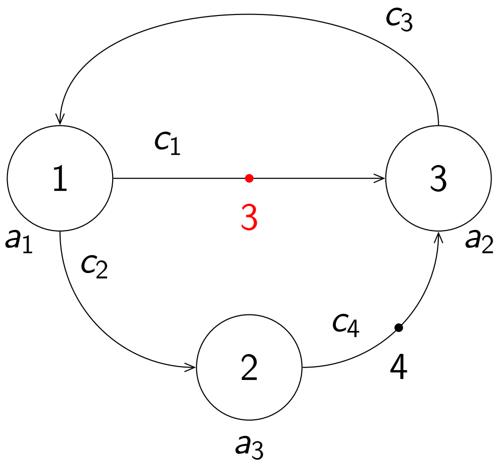
t: 0 $\rightarrow$ 2

Example DE



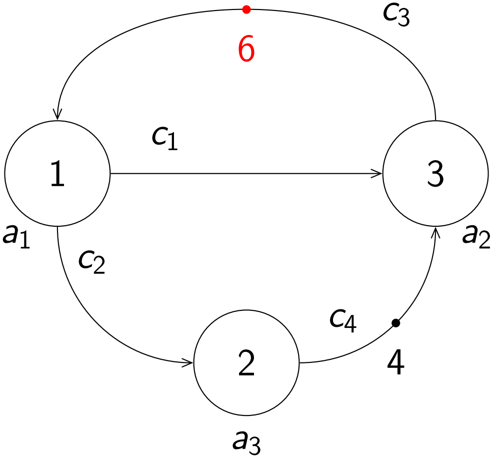
t: 0 $\rightarrow$ 2

Example DE



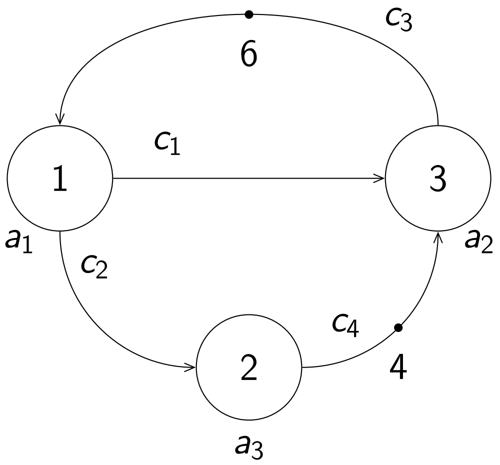
t: 0 $\rightarrow$ 2 $\rightarrow$ 3

Example DE



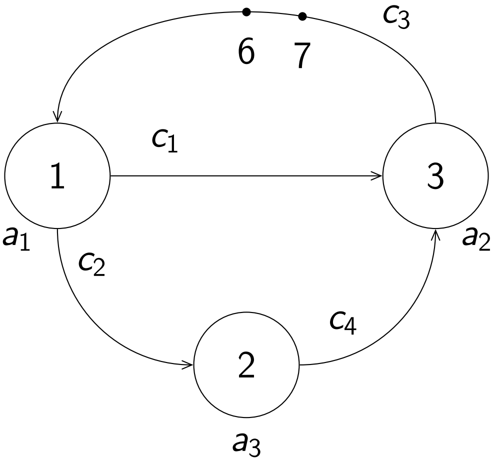
t: $0 \rightarrow 2 \rightarrow 3$

Example DE



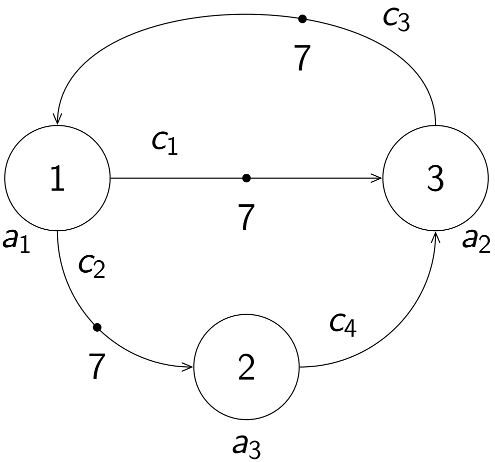
t: $0 \rightarrow 2 \rightarrow 3 \rightarrow 4$

Example DE



t: $0 \rightarrow 2 \rightarrow 3 \rightarrow 4$

Example DE

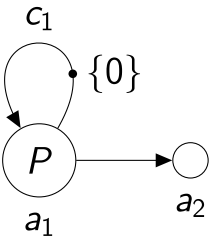


t: $0 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6$

Timed transition system

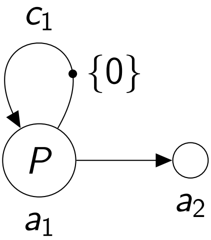
- State (r, t)
  - r is a map from channels to sets of timestampes
  - t is global time
- Two types of transitions
  - *delay* transitions: set global time equal to the smallest timestamp in the map
  - *discrete* transitions: fire actor with smallest timestamp in its input channels
- Actors fire in timestamp order
- Execution is deterministic

Boundedness of DE



- Events in c_1: $\{i \cdot P \mid i \in \mathbb{N}\}$
- Events in c_2: $\{(i+1) \cdot P \mid i \in \mathbb{N}\}$

Boundedness of DE



- Events in c_1: $\{i \cdot P \mid i \in \mathbb{N}\}$
- Events in c_2: $\{(i+1) \cdot P \mid i \in \mathbb{N}\}$
- Is the number of events in any channel at a state of the TTS bounded?
- Can we address the issue of timestamps and time being infinite?

Bounding number of events

- Let $\tau_{\min}(s)$ be the min. timestamp in state s

[1] initial events

Bounding number of events

- Let $\tau_{\mathsf{min}}(s)$ be the min. timestamp in state s
- if $s \to s'$ then $\tau_{\mathsf{min}}(s) \leq \tau_{\mathsf{min}}(s')$

[1] initial events

Bounding number of events

- Let $\tau_{\text{min}}(s)$ be the min. timestamp in state s
- if $s \to s'$ then $\tau_{\text{min}}(s) \leq \tau_{\text{min}}(s')$
- for any event τ in a state s,[1]

$$\tau \leq \tau_{\text{min}}(s) + \max\{delay(a) \mid a \in A\}$$

[1] initial events

Bounding number of events

- Let $\tau_{\min}(s)$ be the min. timestamp in state s
- if $s \to s'$ then $\tau_{\min}(s) \leq \tau_{\min}(s')$
- for any event τ in a state s,[1]

$$\tau \leq \tau_{\min}(s) + \max\{\mathit{delay}(a) \mid a \in A\}$$

- Lower and upper bound for all events in a state
- Fractional part of every timestamp is determined by fractional part of initial events

[1] initial events

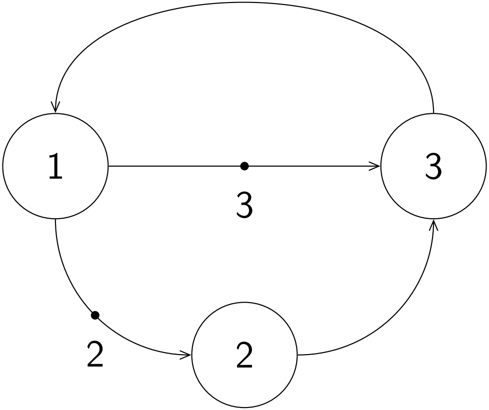
Bounding timestamps

- In *TTS* timestamps and global time can grow unbounded

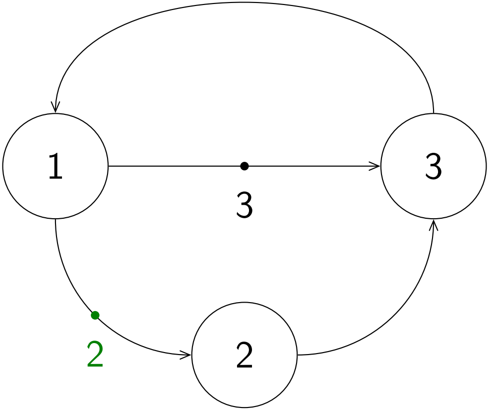
Bounding timestamps

- In TTS timestamps and global time can grow unbounded
- Bounded timed transition system $BTS(G, r_0)$
- Delay transitions: subtract minimum timestamp from all events
- Discrete transitions: process event with timestamp equal to 0
- BTS transitions:
  - delay transition: $r \xrightarrow{\delta}_b r'$ where $\delta = \tau_{\min}(r)$, $r' = r - \tau_{\min}(r)$
  - discrete transition: $r \xrightarrow{a}_b r'$ with $r' = f(a, r, D(a))$, $\tau_{\min}(a, r) = \tau_{\min}(r) = 0$

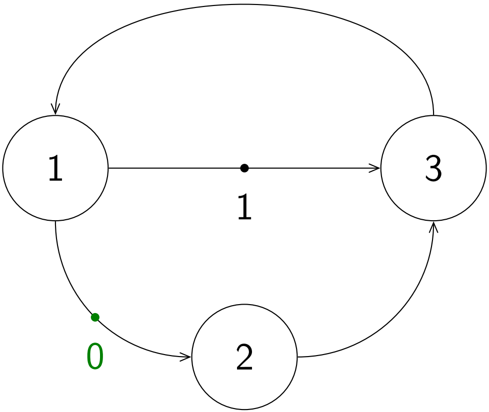
Example BTS



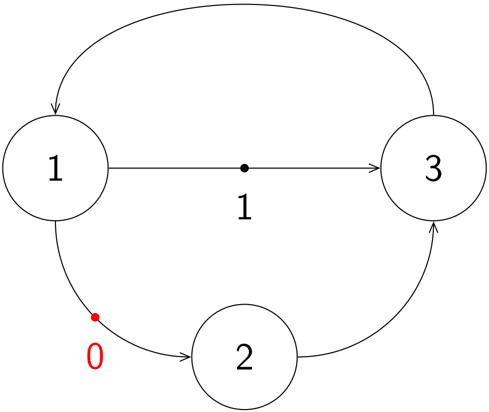
Example BTS



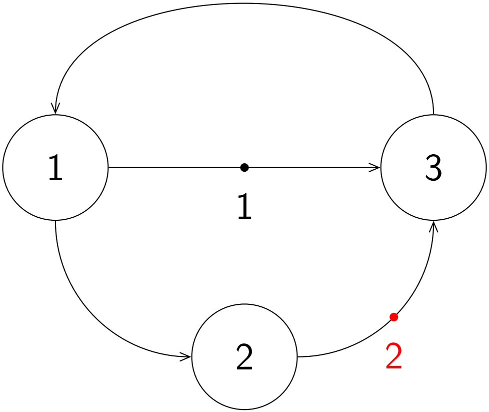
Example BTS



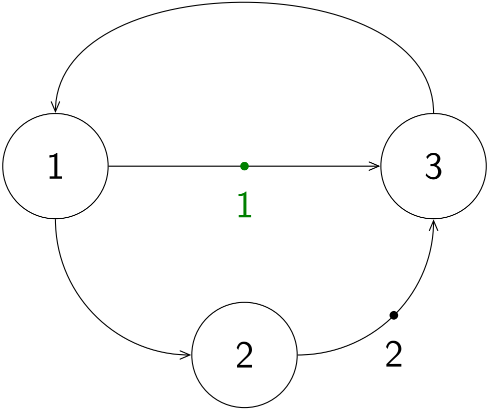
Example BTS



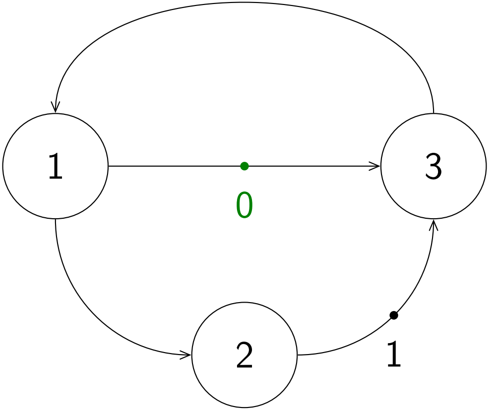
Example BTS



Example BTS



Example BTS



Bounded Transition System

- The set of reachable states of $BTS(G, r_0)$ is finite
  - Number of events bounded as in TTS
  - Possible timestamps finite, despite initial events $\in \mathbb{R}$
- A bisimulation exists between TTS and BTS
  - R contains all pairs $((r, t), r - t)$

Verification - Queries

- Signal queries
  - A channel signal denotes the set of all events that occur in a channel along an execution
  - "an event occurs in c", $\phi := \exists \tau : \tau \geq 0$
  - "two events occur in c separated by at most 1 time unit", $\exists \tau_1, \tau_2 : |\tau_1 - \tau_2| \leq 1$.

Verification - Queries

- Signal queries
  - A channel signal denotes the set of all events that occur in a channel along an execution
  - "an event occurs in c", $\phi := \exists \tau : \tau \geq 0$
  - "two events occur in c separated by at most 1 time unit", $\exists \tau_1, \tau_2 : |\tau_1 - \tau_2| \leq 1$.
- State queries

Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system

Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system
- Compute for every channel c, an affine expression that describes channel signal of c

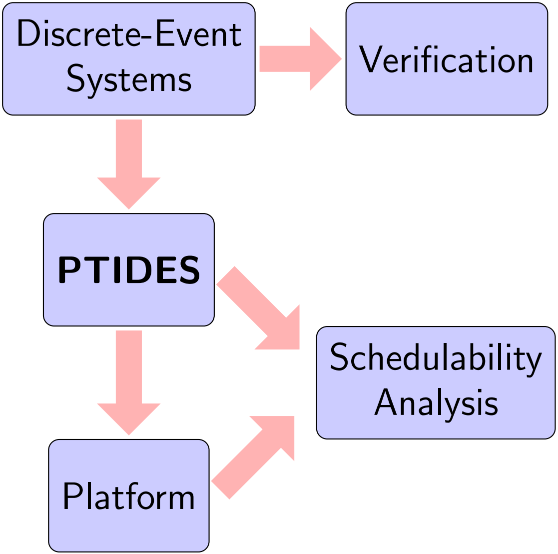
Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system
- Compute for every channel c, an affine expression that describes channel signal of c
- Reduce the problem of checking whether $\sigma_c \models \phi$ to an SMT problem
  - Affine expression: $i \cdot P + j \cdot D$
  - τ_1, τ_2 such that $\tau_1 - \tau_2 = 5$
  - $\tau_1 = i_1 P + j_1 D \wedge \tau_2 = i_2 P + j_2 D \wedge \tau_1 - \tau_2 = 5$

Verification - Algorithms for DE

- Construct lasso from BTS
  - Merge all enabled transitions in one
  - Finite and deterministic transition system
- Compute for every channel c, an affine expression that describes channel signal of c
- Reduce the problem of checking whether $\sigma_c \models \phi$ to an SMT problem
  - Affine expression: $i \cdot P + j \cdot D$
  - τ_1, τ_2 such that $\tau_1 - \tau_2 = 5$
  - $\tau_1 = i_1 P + j_1 D \wedge \tau_2 = i_2 P + j_2 D \wedge \tau_1 - \tau_2 = 5$
- Similarly for state queries

Conclusions



Thank you

Questions?