Towards a Generic Framework for Formal Verification and Performance Analysis of Real-time Scheduling Algorithms

Salwa Habbachi, Zhiwu Li, Mohamed khalgui

Macau University of Science and Technology, MUST
Macau Institute of System Engineering, MISE

VECoS, September 2020

Summary

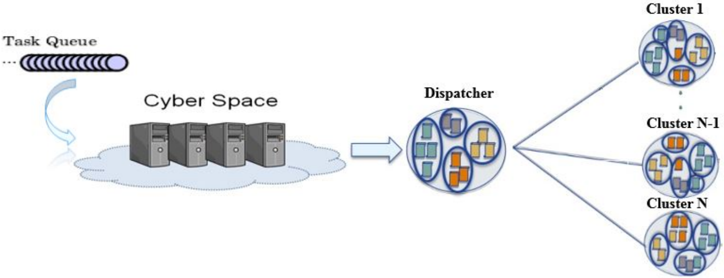
1. General Context
2. Preliminaries: UPPAAL framework
3. Proposed Formal Model
4. Experimental Results
5. Conclusion and Perspectives

Summary

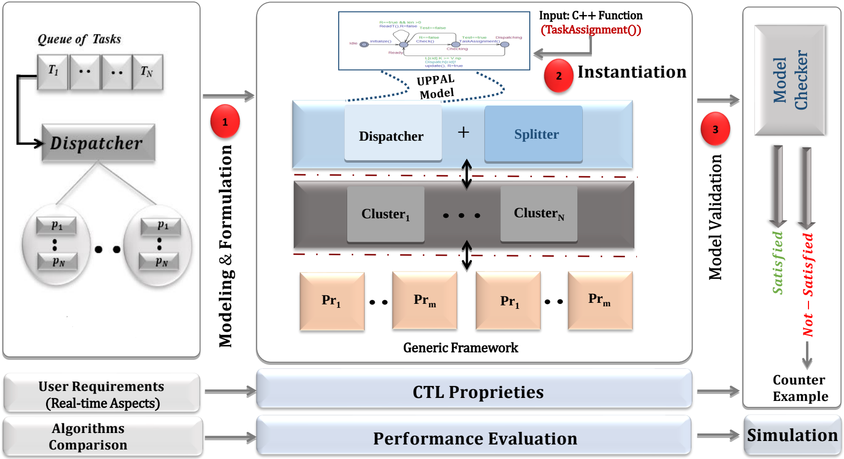
1. General Context
2. Preliminaries: UPPAAL framework
3. Proposed Formal Model
4. Experimental Results
5. Conclusion and Perspectives

Motivation

- Real-time systems ⇒ Widespread distributed systems ⇒ **Critical** systems ⇒ Time requirements.
- Variety of daily life applications ⇒ Human safety.
- Multiple components ⇒ **Multi-tasking operations**.
- Set of tasks to be executed ⇒ Real-time tasks scheduling algorithms.
- Scheduling algorithm ⇒ **System design** (correctness, performance).



Contribution



Queue of Tasks

Dispatcher

Modeling & Formulation

UPPAL Model

Input: C++ Function (TaskAssignment())

Instantiation

Dispatcher + Splitter

Cluster_1 • • • Cluster_N

Pr_1 • • Pr_m Pr_1 • • Pr_m

Generic Framework

Model Validation

Model Checker

Satisfied

Not – Satisfied

Counter Example

User Requirements (Real-time Aspects)

CTL Proprieties

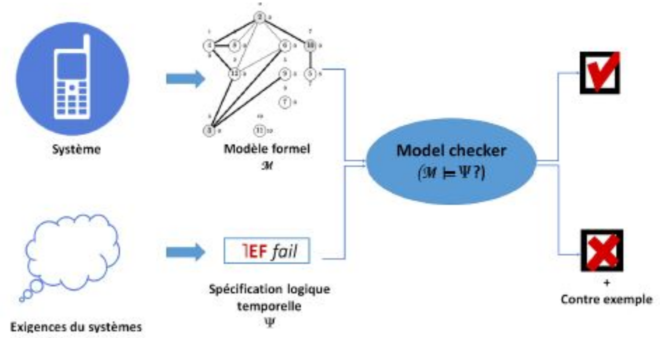
Algorithms Comparison

Performance Evaluation

Simulation

Model Checking

- Automatic verification technique of reactive systems.
- Algorithmic method to formally verify that a finite state system satisfies a logical property.



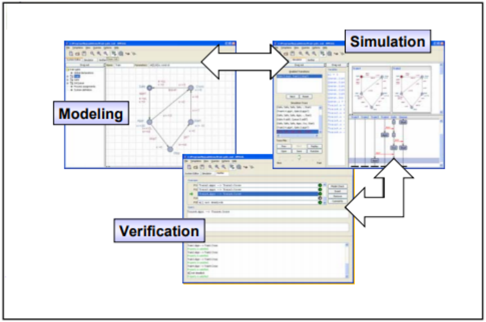
Summary

1. General Context
2. Preliminaries: UPPAAL framework
3. Proposed Formal Model
4. Experimental Results
5. Conclusion and Perspectives

UPPAAL Framework

UPPAAL framework

- A modeling and verification framework of real-time systems that can be represented as timed automaton.



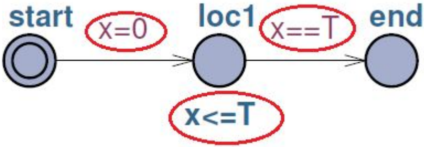
UPPAAL editor

Timed automata:

- Finite state machine with clocks.
- Clocks, x
- Invariant, $(x \leq T)$

UPPAAL Model:

- Locations
  - Initial
  - Urgent
  - Committed
  - Normal (all the rest)



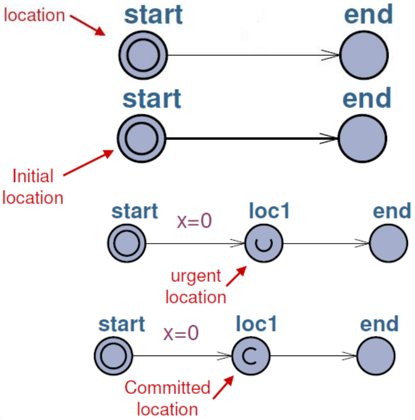
UPPAAL editor

Timed automata:

- Finite state machine with clocks.
- Clocks, x
- Invariant, $(x \leq T)$

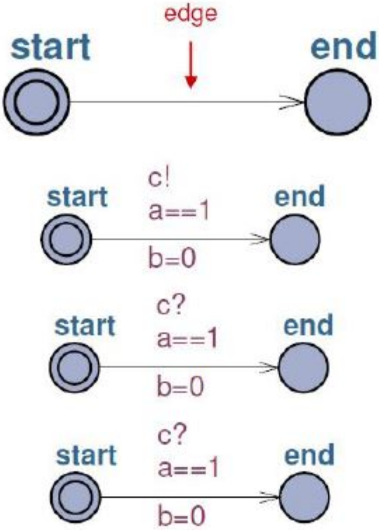
UPPAAL Model:

- Locations
  - Initial
  - Urgent
  - Committed
  - Normal (all the rest)



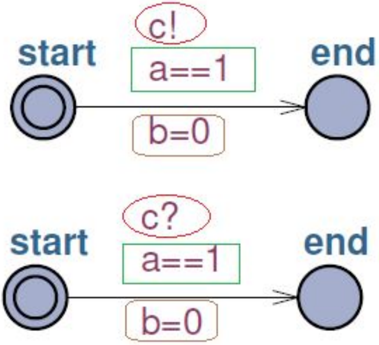
UPPAAL Model:

- Locations
  - Initial
  - Urgent
  - Committed
  - Normal (all the rest)
- Edges
  - Synchronizations (Channels)
    - Binary synchronization: *chan c*.
    - Urgent synchronization: *urgent chan c*.
    - Broadcast synchronization: *broadcast chan c*.
  - Guards
  - Update



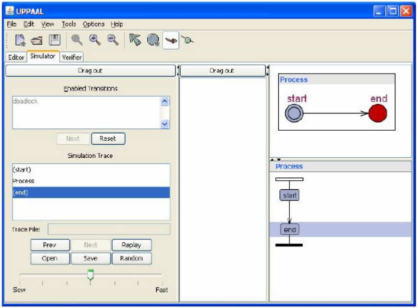
UPPAAL Model:

- Locations
  - Initial
  - Urgent
  - Committed
  - Normal (all the rest)
- Edges
  - Synchronizations (Channels)
    - Binary synchronization: *chan c*.
    - Urgent synchronization: *urgent chan c*.
    - Broadcast synchronization: *broadcast chan c*.
  - Guards
  - Update



UPPAAL simulator

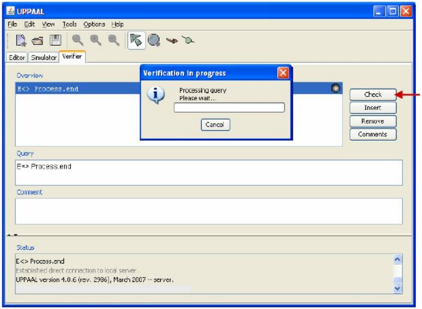
- Syntactically correct model ⇒ Behavioral simulation ⇒ simulation traces.



```
(start)
Process
(end)
```

UPPAAL model checker:

- Properties specification and verification.
  - Green light (Property satisfied)
  - Red light (Property not satisfied)



E<> Process.end

Verification in progress

Processing query
Please wait...

Query: E<> Process.end

Status:

E<> Process.end
Established direct connection to local server.
UPPAAL version 4.0.6 (rev. 2986), March 2007 -- server.

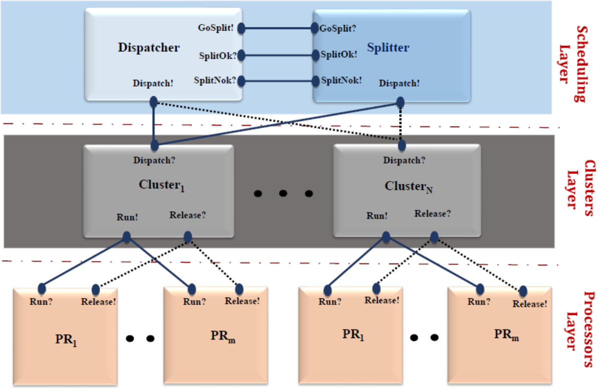
Click

Summary

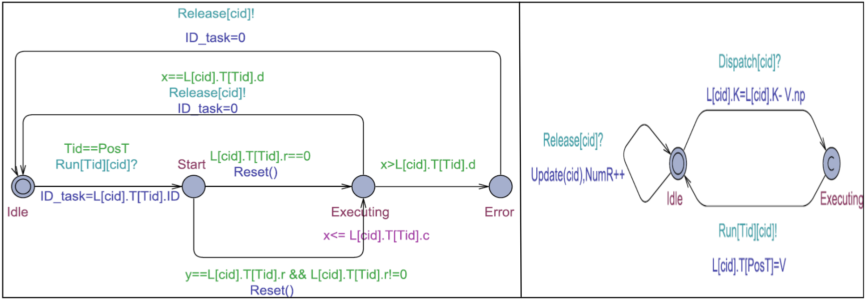
1. General Context
2. Preliminaries: UPPAAL framework
3. Proposed Formal Model
4. Experimental Results
5. Conclusion and Perspectives

Modeling and Formulation

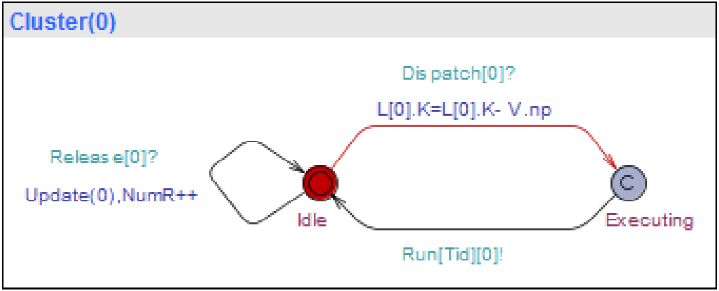
- Centralized architecture ⇒ Formal model ⇒ Superposition of 3 layers:
  - Scheduling layer: **Dispatcher**.
  - Clusters layer: Set of distributed clusters.
  - Processors layer: Set of processors.



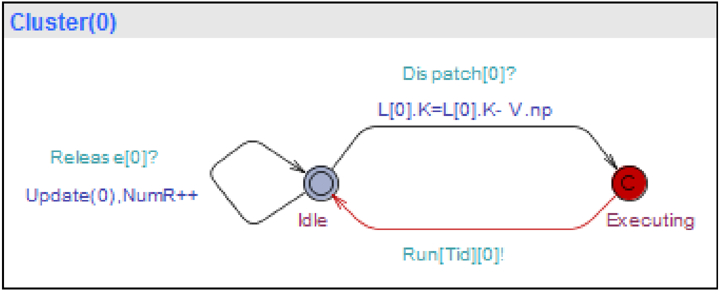
Modeling and formulation: Processors layer



Modeling and formulation: Clusters layer.

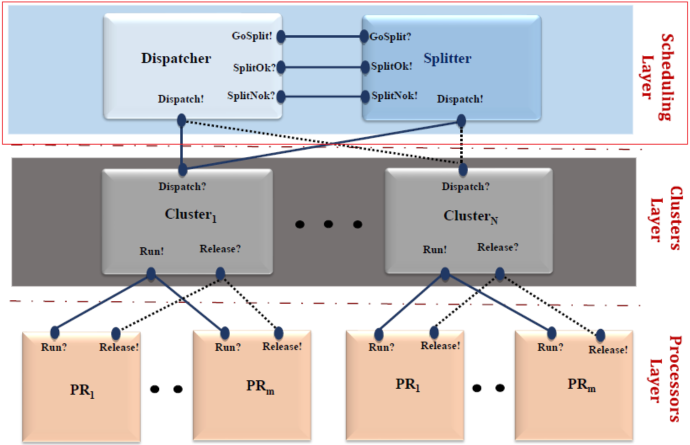


Modeling and formulation: Clusters layer.

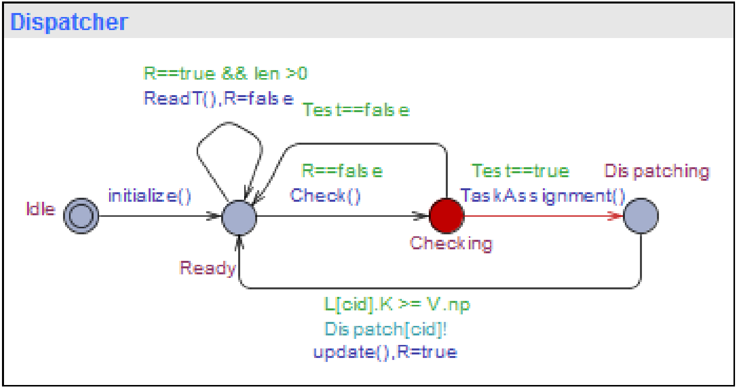


Instantiation: Scheduling layer

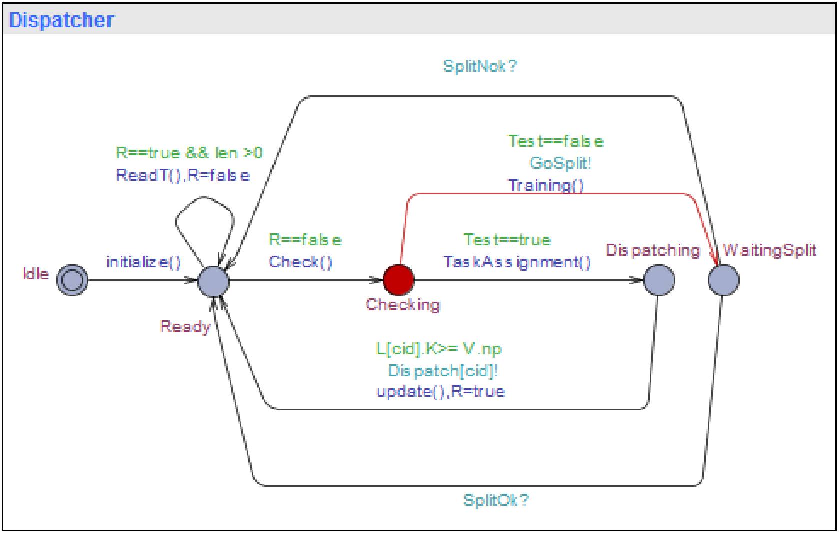
Scheduling layer ⇒ Task assignment policy
The other layers are identical regardless of the scheduling algorithm.



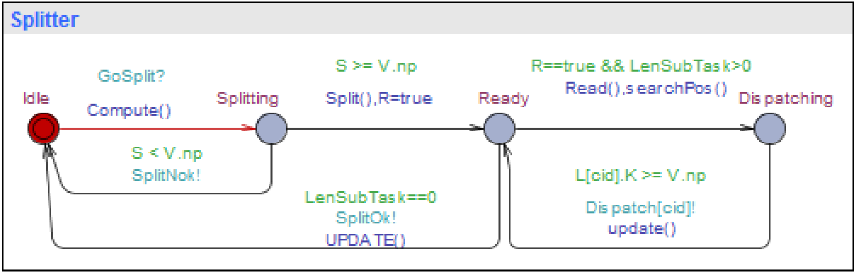
Instantiation: Scheduling Strategy



Instantiation: Task-Splitting Strategy



Instantiation: Task-Splitting Strategy



Summary

1. General Context
2. Preliminaries: UPPAAL framework
3. Proposed Formal Model
4. Experimental Results
5. Conclusion and Perspectives

Deadlock Freedom Verification

- UPPAAL Model-Checker → Deadlock Freedom

  A [] not deadlock

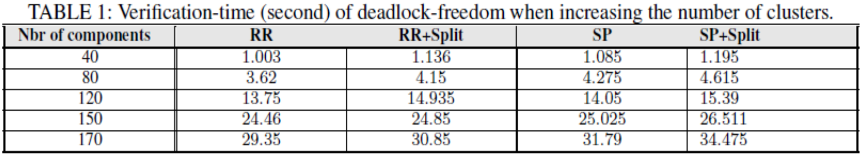


TABLE 1: Verification-time (second) of deadlock-freedom when increasing the number of clusters.

Nbr of components	RR	RR+Split	SP	SP+Split
40	1.003	1.136	1.085	1.195
80	3.62	4.15	4.275	4.615
120	13.75	14.935	14.05	15.39
150	24.46	24.85	25.025	26.511
170	29.35	30.85	31.79	34.475

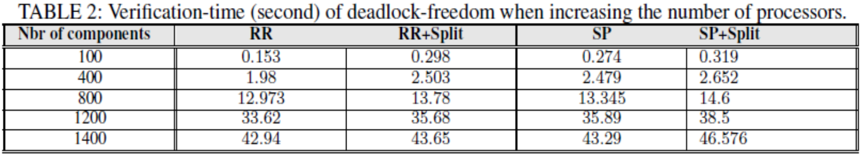


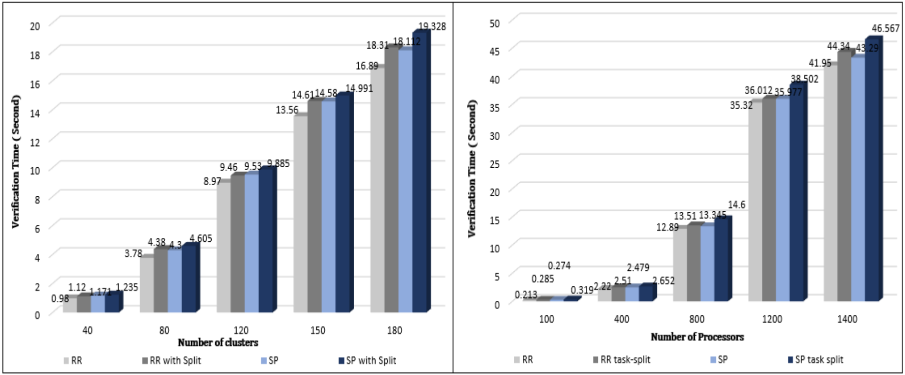
TABLE 2: Verification-time (second) of deadlock-freedom when increasing the number of processors.

Nbr of components	RR	RR+Split	SP	SP+Split
100	0.153	0.298	0.274	0.319
400	1.98	2.503	2.479	2.652
800	12.973	13.78	13.345	14.6
1200	33.62	35.68	35.89	38.5
1400	42.94	43.65	43.29	46.576

Invariant Verification

- UPPAAL Model-Checker → Invariant

$$A[]\ Pr(Tid, cid).Executing \Longrightarrow (Pr(Tid, cid).x \leq L[cid].T[Tid].c)$$



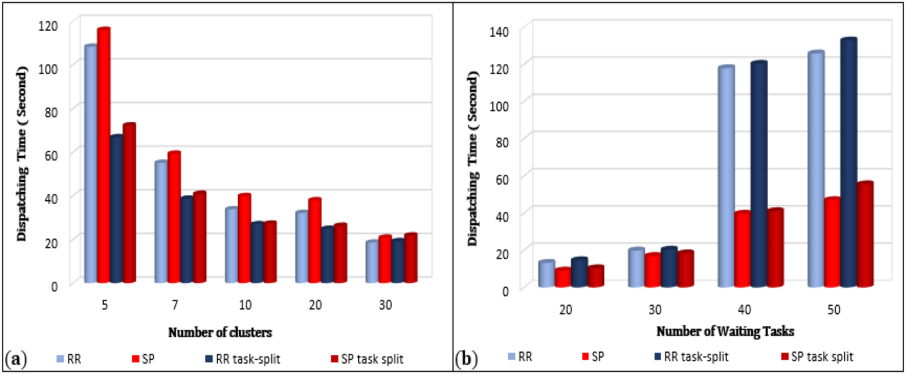
Timing constraints

- UPPAAL Model-Checker → Timing constraints

$$A[] not \ Pr(Tid, cid).Error$$

Performance Analysis

- Analysis with UPPAAL simulator ⇒ Measure the time required to distribute a set of tasks.



Summary

1. General Context
2. Preliminaries: UPPAAL framework
3. Proposed Formal Model
4. Experimental Results
5. Conclusion and Perspectives

Conclusion and Perspectives

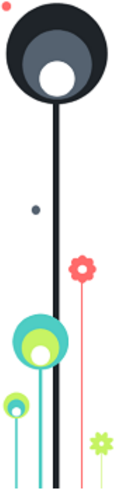
Conclusion

- Generic framework based on a formal model of task scheduling algorithms.
- Analysis and verification of different properties: deadlock-free, invariant property, etc.

Conclusion and Perspectives

Perspectives

- Propose distributed versions of our formal model.
- Model and analyze new task scheduling protocols based on the same architecture.
- . . .



Do you have any questions?

THANK YOU FOR YOUR ATTENTION !