A New Model for Measuring the Performance Cost of Deadline Misses

Authors:

Paolo Pazzaglia, Luigi Pannocchi, Alessandro Biondi, Marco Di Natale

Scuola Superiore Sant'Anna, Pisa

paolo.pazzaglia@santannapisa.it

Siena, IWES, September 14th, 2018

Introduction

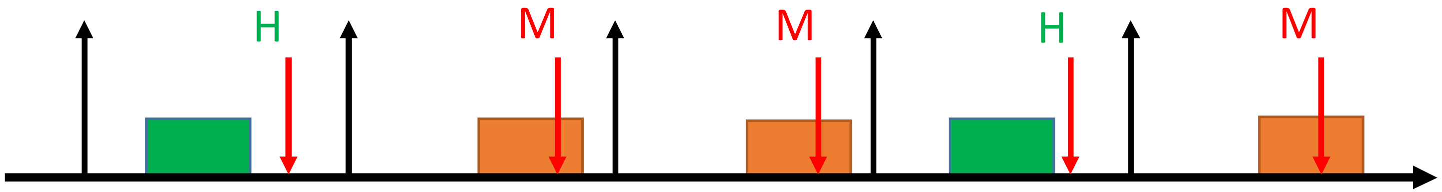
- Embedded systems with control tasks may face **overload** conditions (e.g. automotive)

- Common (practical) approach: running at a high rate and allowing some **deadline miss** is an acceptable compromise
- Missing (few) deadlines: not catastrophic!

How to study performance evolution under overload conditions?

- **Weakly Hard model**: limited number of deadline misses
  - **(m,k)**: at most **m** deadlines are missed every **k** activations

Ex: (3,5)



Weakly hard model limitations

- (m,k) constraint is not enough descriptive…

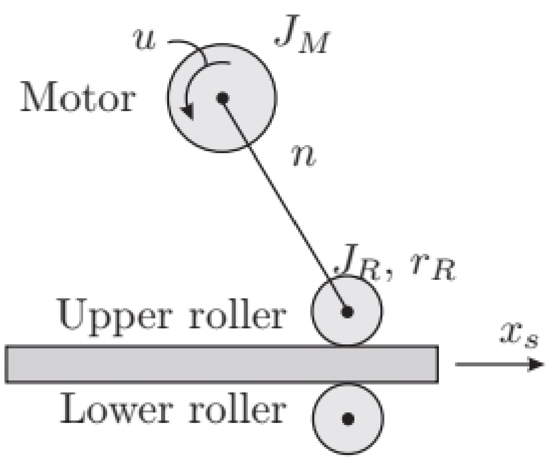
- (m,k) constraint leads to a **binary** model (either pass or fail)
  - Easy to define stability guarantees
  - No information about **performance** of different patterns
  - Difficult to extract an **ordering** between constraints

- No relation with the system state**:**
  - Deadline misses may have different effects (transients vs steady state)

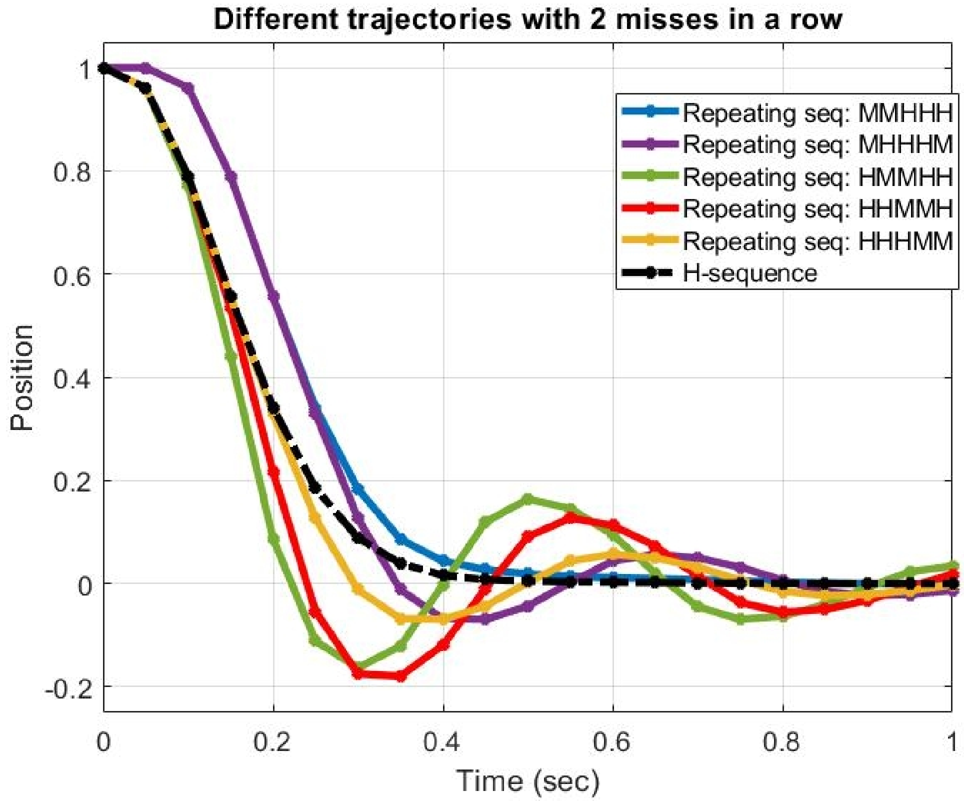
Weakly hard model limitations

Different patterns of H/M deadlines lead to different performance evolutions!

Assumption: When a deadline is missed, the control output is not updated



$T = 50\,ms;\quad D = 0.7 * T$



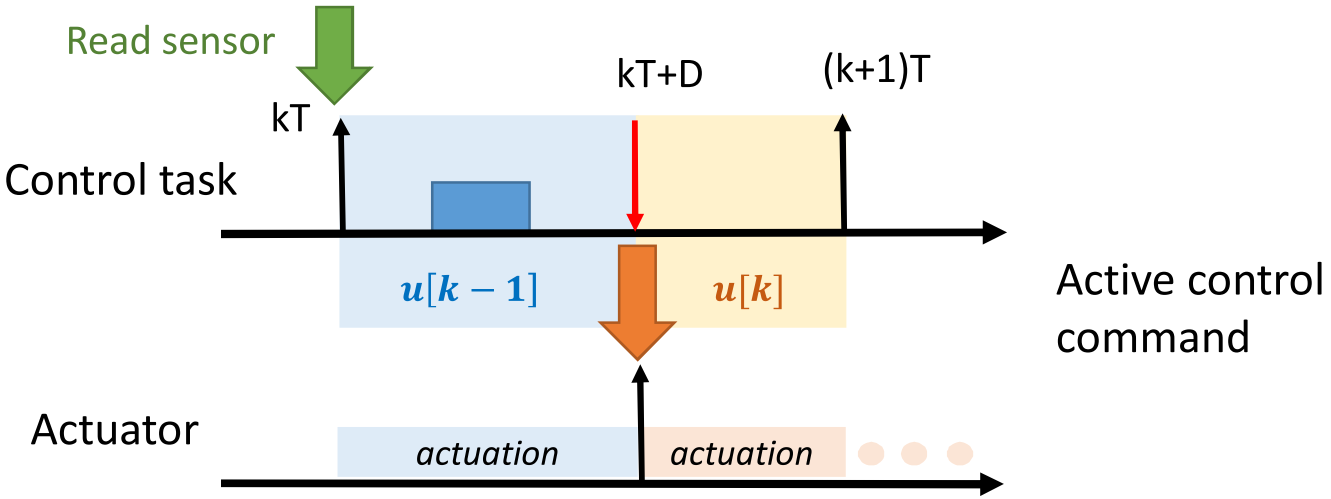
A new model for performance analysis

- <u>Goal:</u> Developing a new model for studying:
  - How the **performance** changes with different **patterns of missed deadlines** that satisfy a given (m,k) constraint
  - **Worst guaranteed performance**
  - Different policy at deadline miss (continue or kill?)

- Merging real-time analysis with control system dynamics and performance analysis

System model

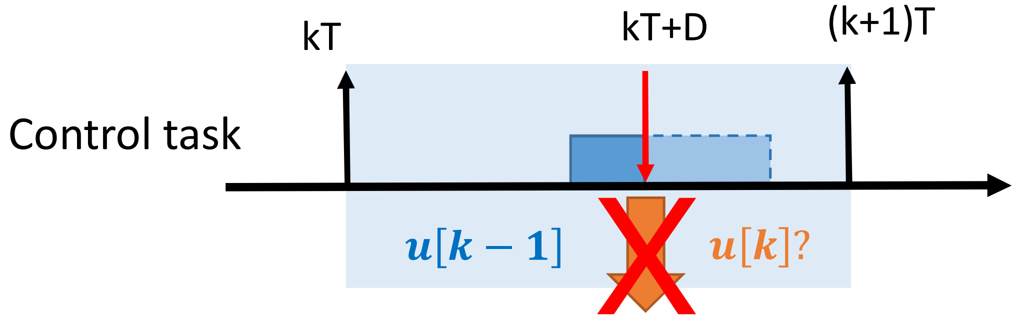
- **Linear Time Invariant** plant, MIMO
- Periodic control of period T_i and deadline $D_i \leq T_i$
- State-feedback control: $u[k] = K(r[k] - x[k])$



State update function: $\mathrm{x}[\mathrm{k}+1] = \mathrm{A_d}\mathrm{x}[\mathrm{k}] + \mathrm{B_{d1}}u[k-1] + B_{d2}u[k]$

- Similar to **LET** model: trading jitter for latency

Missing a deadline



- Missing a deadline means **missing** an actuator command **update:**

  <u>Keep the previous actuation value</u>

- **The system dynamics changes!**

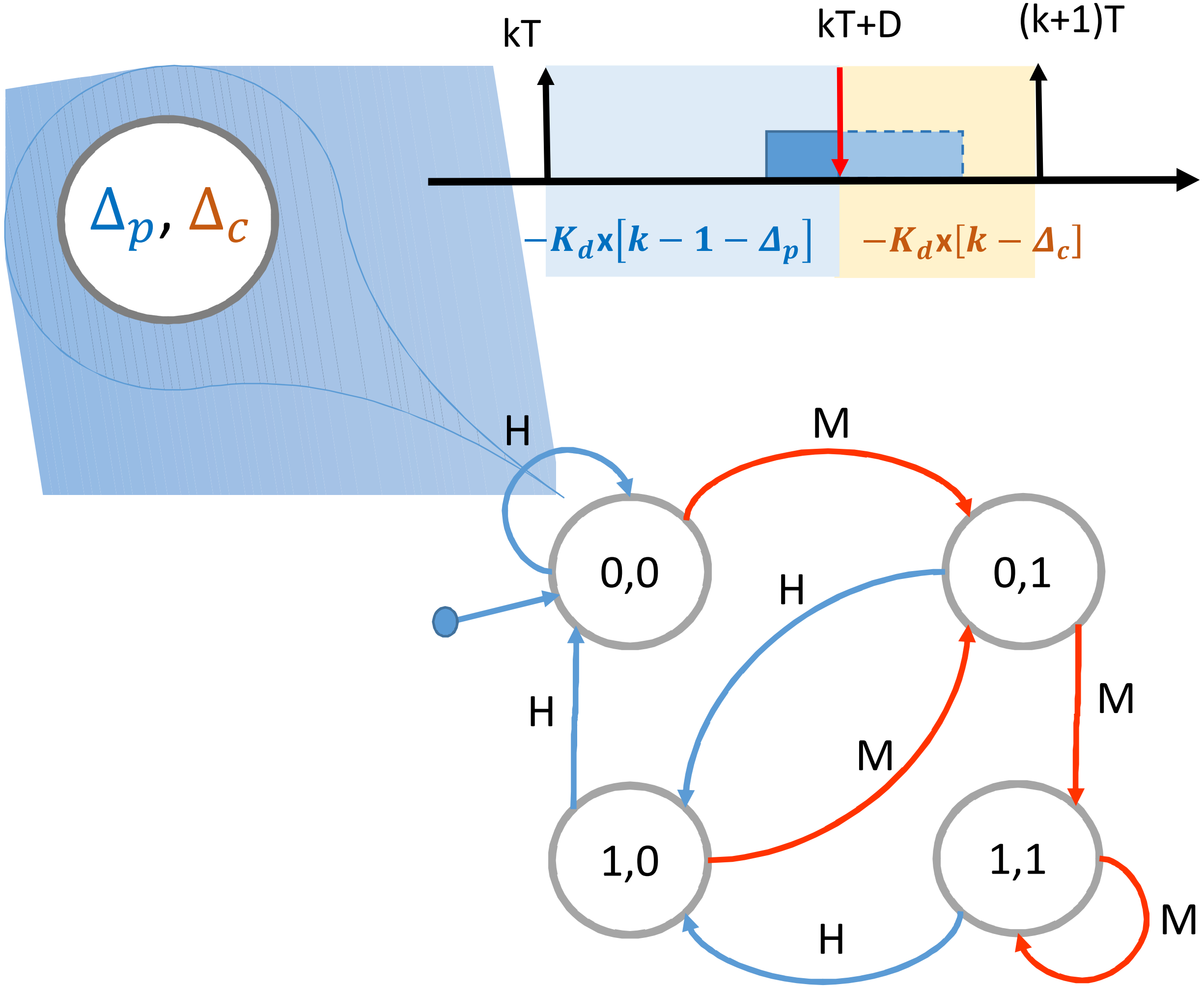
$$x[k+1] = A_d x[k] + B_{d1}\boldsymbol{u}[\boldsymbol{k}-\boldsymbol{1}] + B_{d2}\boldsymbol{u}[\boldsymbol{k}]$$

➢ $\boldsymbol{u}[\boldsymbol{k}-\boldsymbol{1}] = -K_d x[k-1-\Delta_p]$

➢ $\boldsymbol{u}[\boldsymbol{k}] = -K_d x[k-\Delta_c]$

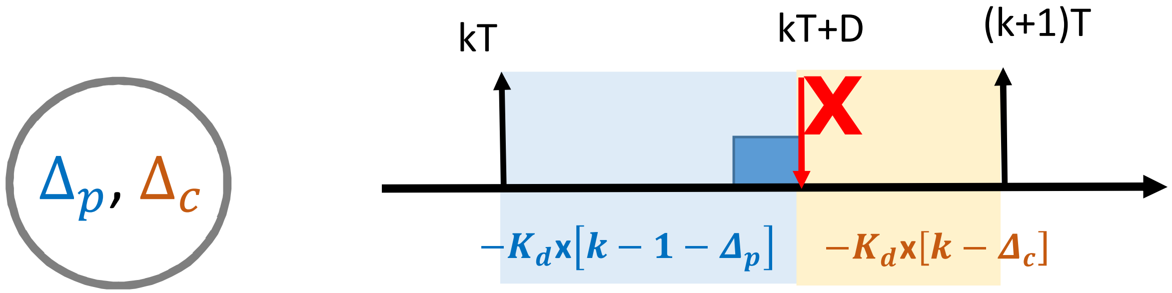
- **Update freshness** Δ (ageing steps) of the control output

Update freshness: Continue strategy

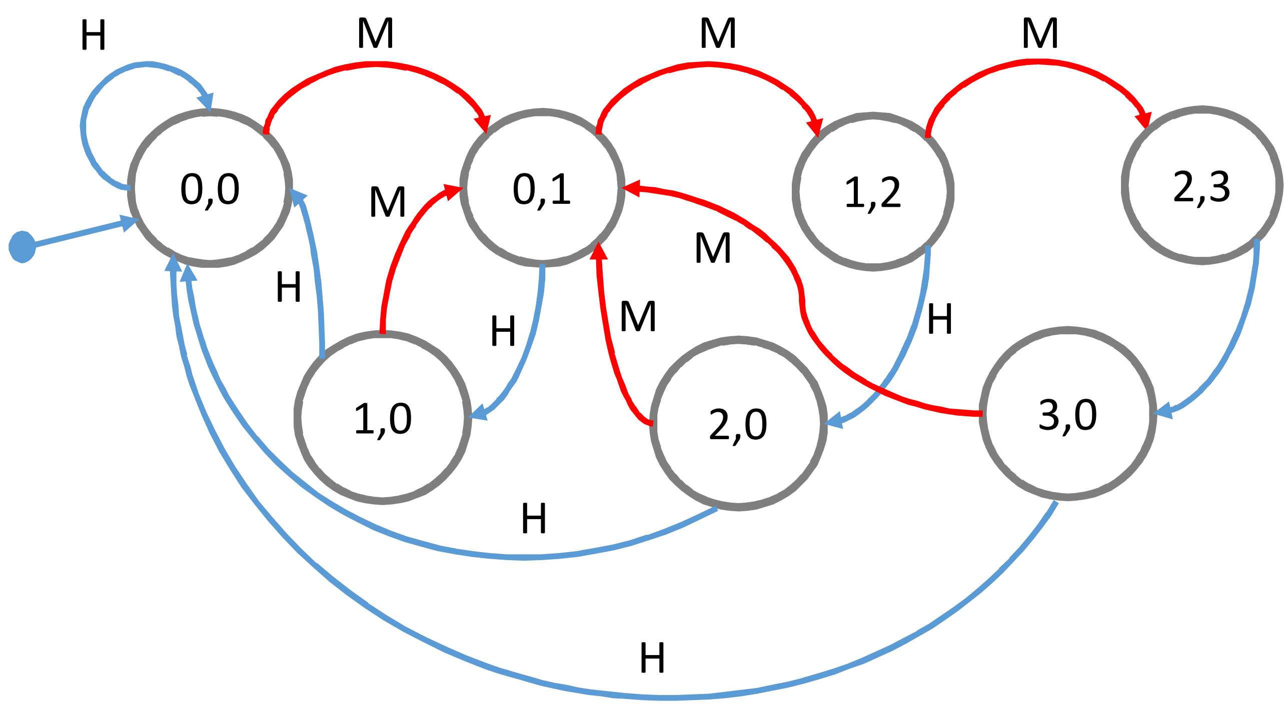


- $BCRT \leq D_i$
- $WCRT < T_i + D_i$

Update freshness: Kill strategy



$* \; BCRT \leq D_i$



In this example, maximum number of consecutive deadline misses is equal to 3

State update matrix

- System dynamics as a function of freshness pairs

$$x[k+1] = A_d x[k] - B_{d1} K_d x\left[k - 1 - \Delta_p\right] - B_{d2} K_d x[k - \Delta_c]$$

- Augmented state vector $\xi[k]$

$$\xi[k] = [x[k]; x[k-1]; \dots . x[k - \Delta_{max} - 1]]$$

$$\boxed{\xi[k+1] = \Phi(\Delta_p, \Delta_c)\, \xi[k]}$$

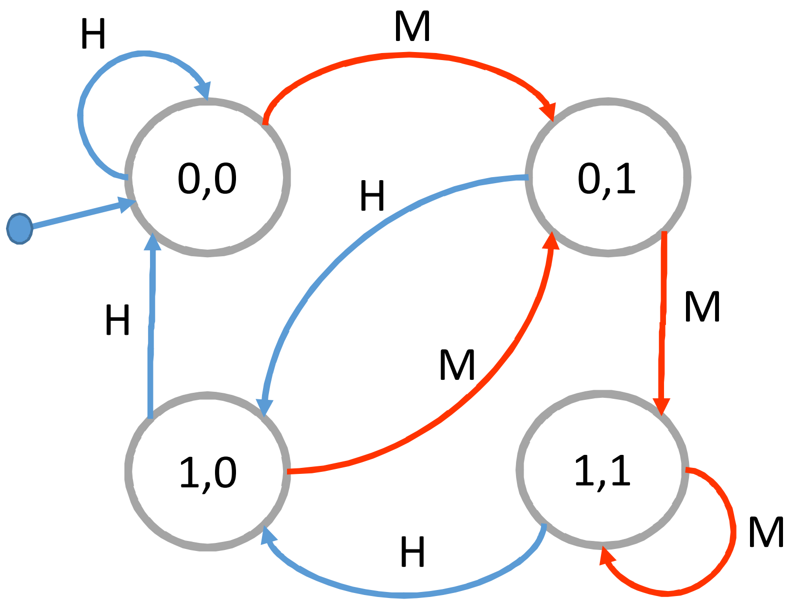
- **State update matrix** $\Phi(\Delta_p, \Delta_c)$

$$\Phi(\Delta_p, \Delta_c) = \begin{bmatrix} A_d & \cdots & -B_{d2}K_d & \cdots & -B_{d1}K_d & \cdots \\ I_n & 0_n & \cdots & \cdots & \cdots & \\ 0_n & I_n & 0_n & & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \cdots & \cdots \end{bmatrix}$$

State update matrix: an example

- Every combination of (Δ_p, Δ_c) is mapped to **a specific dynamic of the system** through the matrix $\Phi(\Delta_p, \Delta_c)$

Example:



$$\Phi(0,0) = \begin{bmatrix} A_d - B_{d2}K_d & -B_{d1}K_d & \mathbf{0}_n \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

$$\Phi(0,1) = \begin{bmatrix} A_d & -(B_{d1} + B_{d2})K_d & \mathbf{0}_n \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

$$\Phi(1,0) = \begin{bmatrix} A_d - B_{d2}K_d & \mathbf{0}_n & -B_{d1}K_d \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

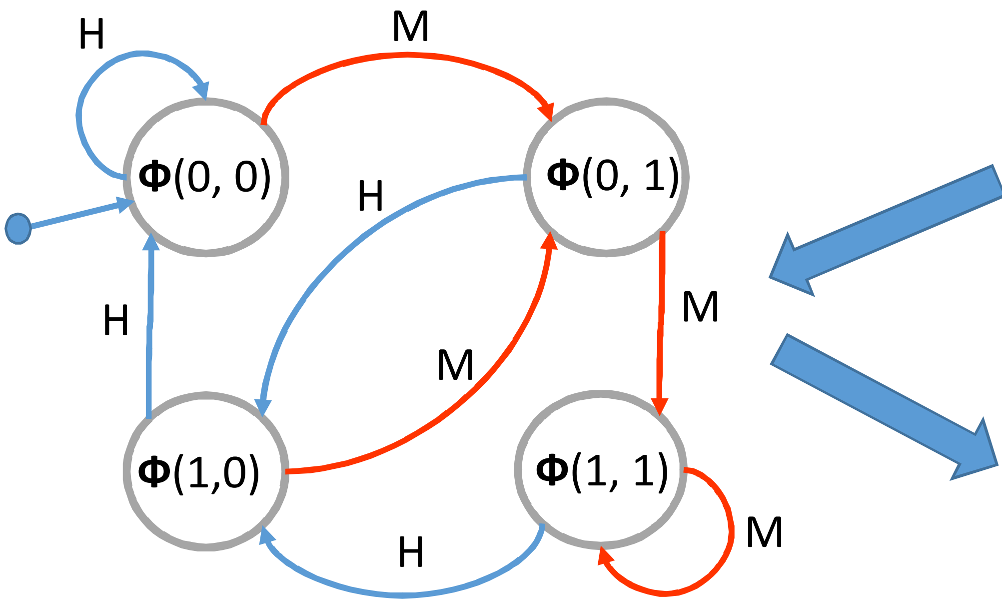
$$\Phi(1,1) = \begin{bmatrix} A_d & -B_{d2}K_d & -B_{d1}K_d \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}.$$

Constrained switched linear system

State update matrix: an example

- Every combination of (Δ_p, Δ_c) is mapped to **a** **specific dynamic** **of the system** through the matrix $\Phi(\Delta_p, \Delta_c)$

Example:



$$\left.\begin{array}{l}\text{HMMHHHMMH..}\\ \text{HHHMMHHM..}\\ \quad\ldots\end{array}\right\} \text{(m,K)}$$

Sequence of states

$$\xi[k+1] = \Phi(\Delta_p, \Delta_c)\,\xi[k]$$

Constrained switched linear system

Performance analysis

- Assign a **performance value** for each sequence of N jobs

- **Sum of quadratic error**

$$P(s) = \sum_{i=0}^{N-1} \xi[i]^T \xi[i]$$
$$= \xi[0]^T \left(\mathbf{I} + \mathbf{\Phi}_0^T \mathbf{\Phi}_0 + \mathbf{\Phi}_0^T \mathbf{\Phi}_1^T \mathbf{\Phi}_1 \mathbf{\Phi}_0 + \ldots + \mathbf{\Phi}_0^T \mathbf{\Phi}_1^T \cdots \mathbf{\Phi}_{N-1}^T \mathbf{\Phi}_{N-1} \cdots \mathbf{\Phi}_1 \mathbf{\Phi}_0 \right) \xi[0]$$
$$= \xi[0]^T \mathbf{\Psi}(s) \xi[0]$$

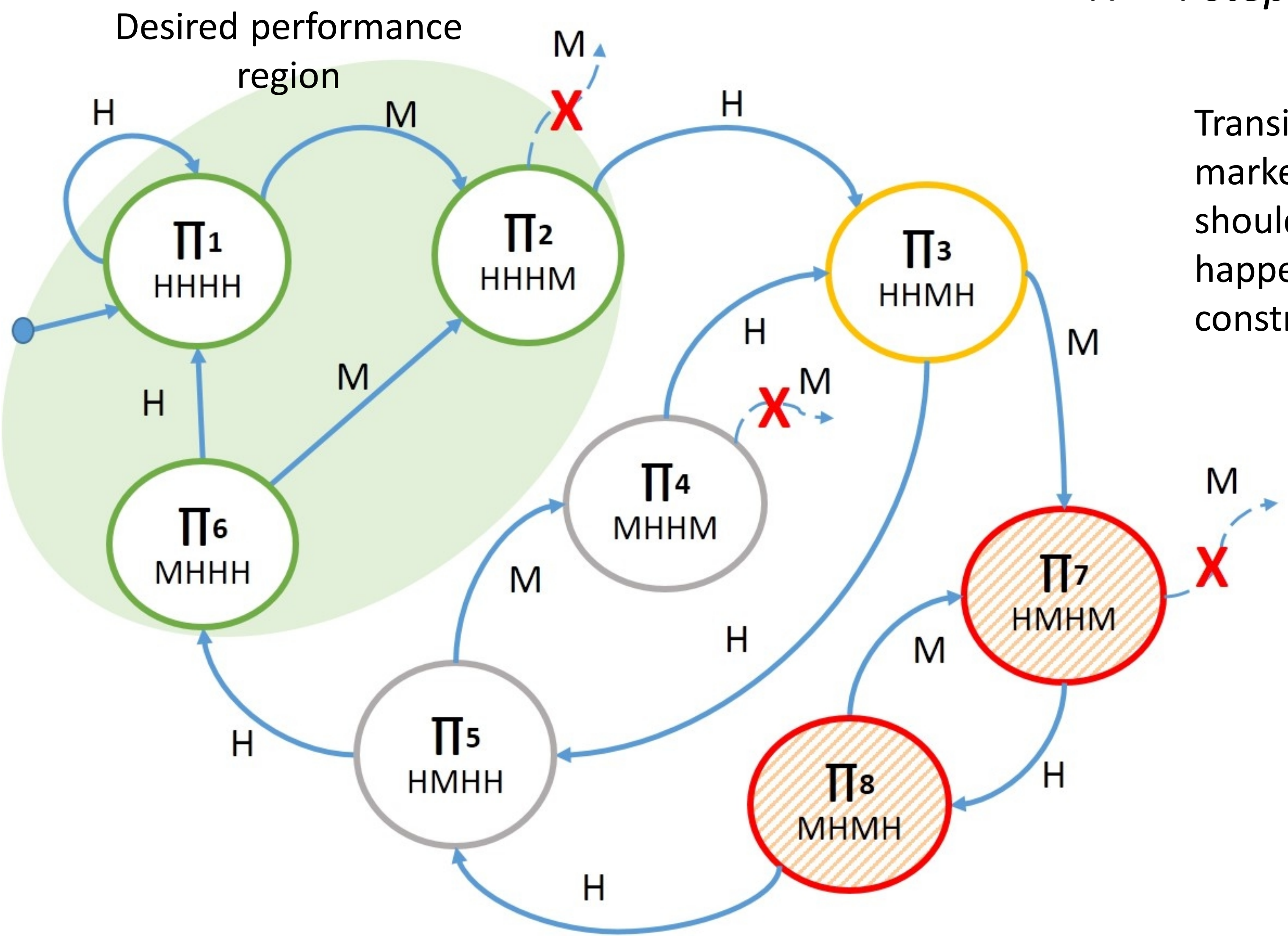
- Matrix elements of $\Psi(s)$ depends on the **ordered** sequence of H/M

- $\prod(s) = ||\Psi(s)||_2$

- **Worst Case Normalized Performance:** $WCPn = \dfrac{max_s \prod(s)}{\prod(all\ hits)}$

Performance state machine

WH constraint ***(1,2)***
N = 4 steps

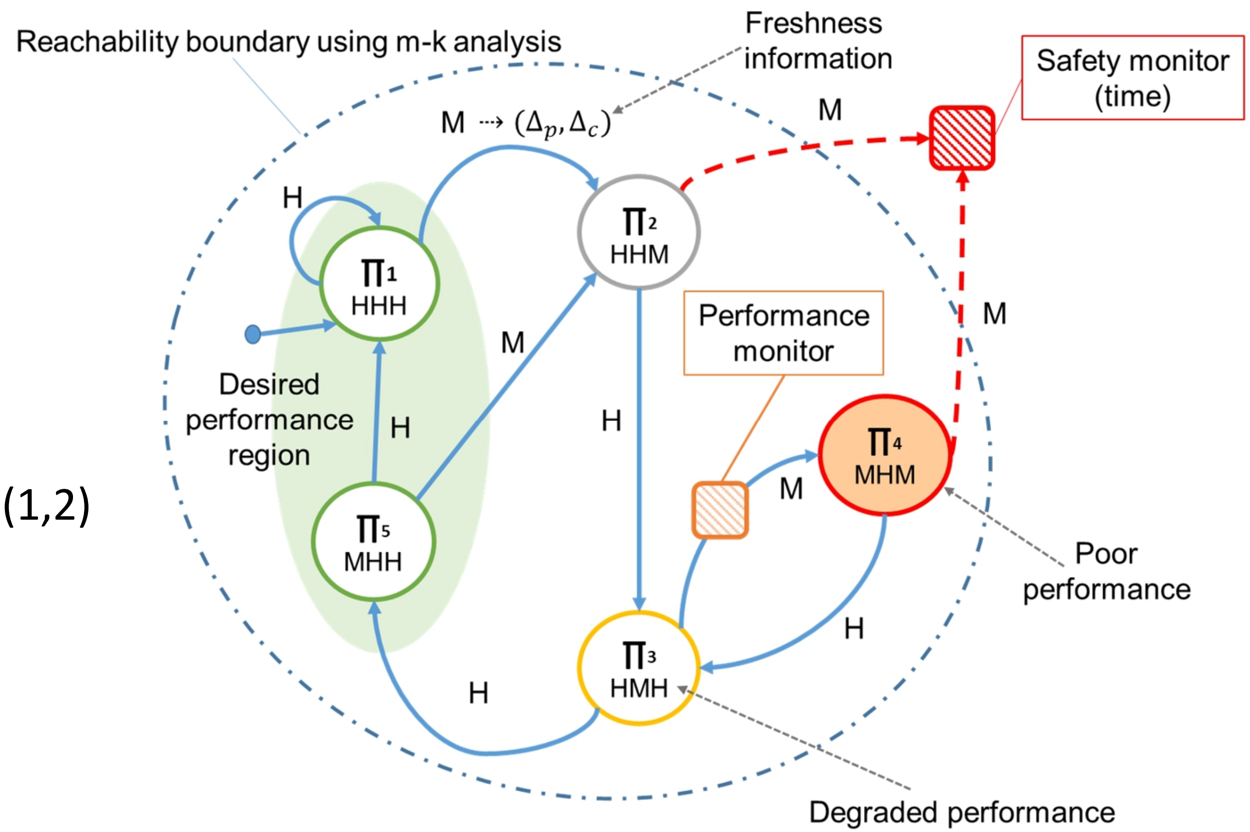


Transitions marked with **X** should never happen for (m,k) constraints

Possible applications

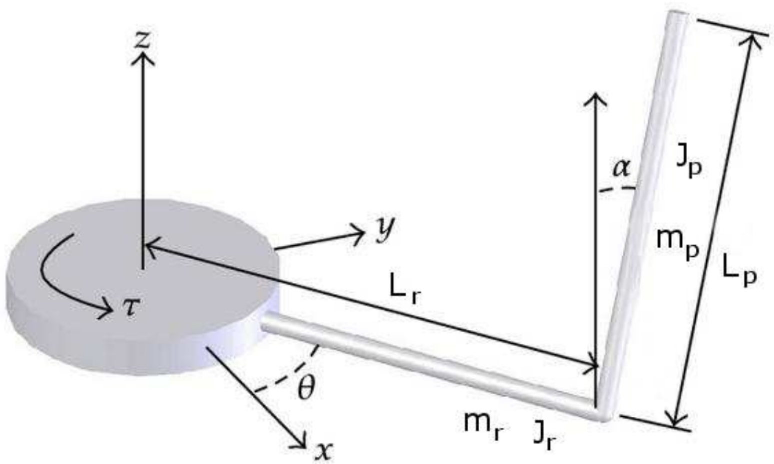
- This new model can be used as a **time contract** between software designers and control engineers
- Possibility of inserting **run-time monitors**

(m,k) = (1,2)



Case study: Furuta pendulum

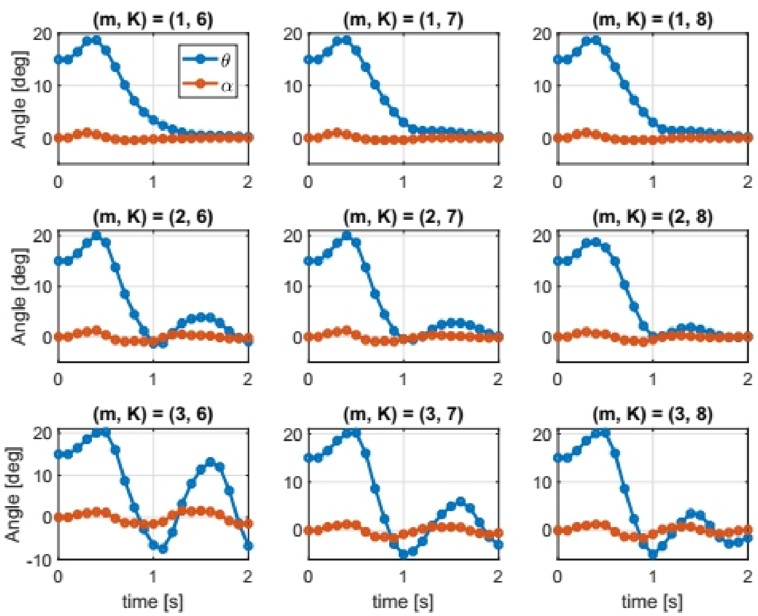
- **Furuta pendulum**: rotary inverted pendulum

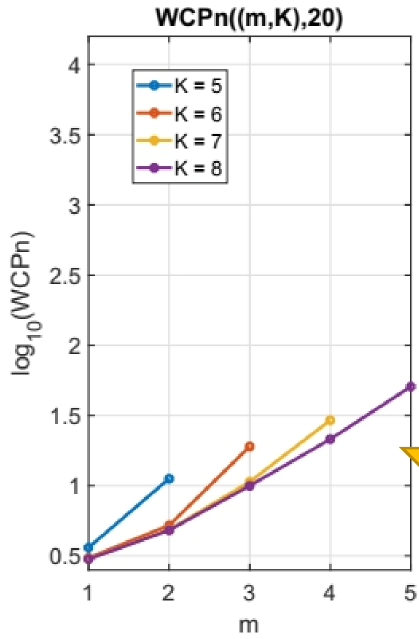


- **Linearized** model in the neighbourhood of the upward position
- Feedback control with $T_i = 0.1sec$ and $D_i = 0.2 * T_i$

- Testing different (m,K) values and studying how Worst Case performance changes

Case study: Furuta pendulum

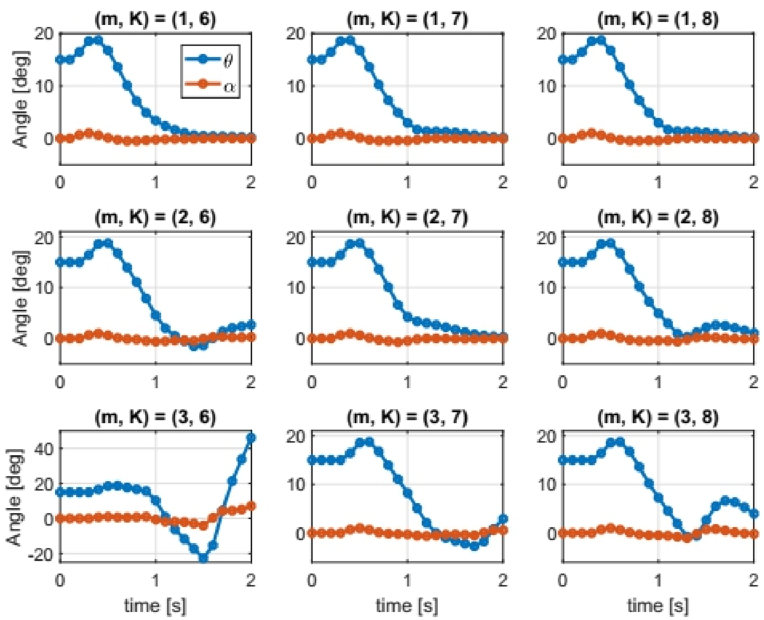


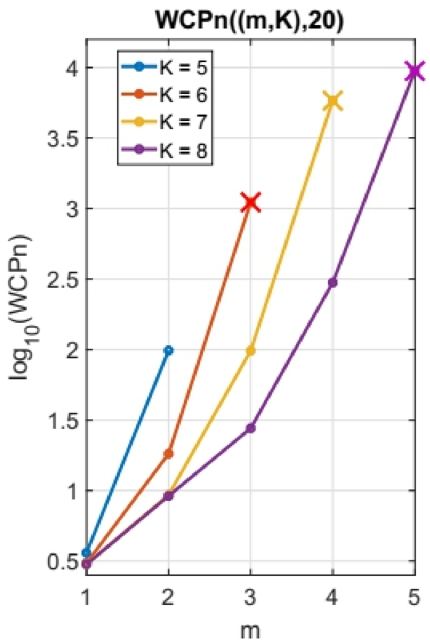


Continue job strategy

The lower the better

Kill job strategy





Summary

- New model for studying performance evolution under overload conditions

1. Creating a state machine for computing **freshness** of outputs, applicable to different patterns and handling of deadline misses
2. Intergrating freshness information with state evolution of the controlled system: different **operating modes**
3. Creating a state machine for computing **performance** values realted to patterns of H/M deadlines
   - Worst case performance guarantees
   - Runtime monitors for performance evolution

- **Case study**: Furuta pendulum

Future work

- Extensions:
  - Including additional performance metrics
  - Extending the case study to WCRT>T+D, allowing multiple pending jobs at deadline
- Finding **optimal controller** for a system under (m,K) constraints, for achieveing a given performance
- **Adaptive control** when deadline misses occur
- More complex case studies:
  - Testing non linear systems performance by simulation
  - More complex deadline miss handlings

Any questions?

Thank you!

paolo.pazzaglia@santannapisa.it

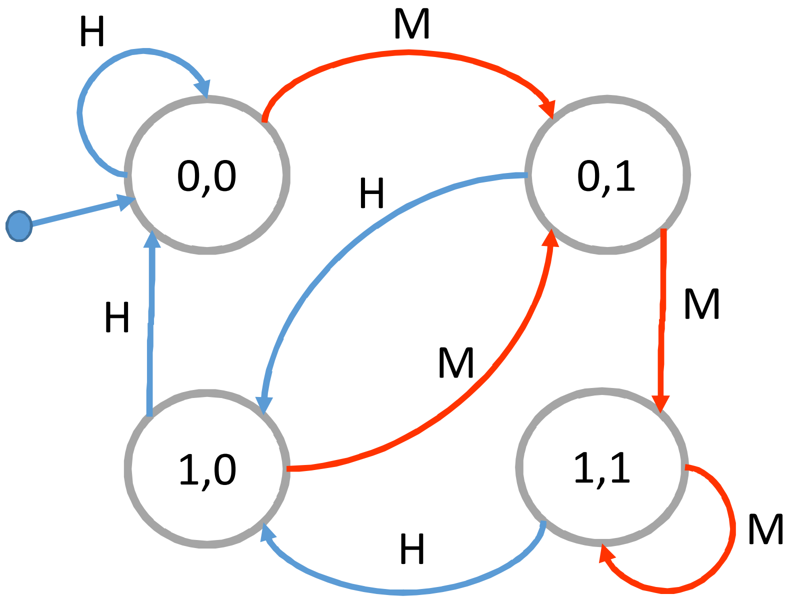
More details in

Paolo Pazzaglia, Luigi Pannocchi, Alessandro Biondi, and Marco Di Natale,
"Beyond the Weakly Hard Model: Measuring the Performance Cost of Deadline Misses",
Proceedings of the 30th Euromicro Conference on Real-Time Systems (ECRTS 18), Barcelona, Spain , July 3-6, 2018.

Performance analysis

- How performance changes for different patterns of H/M deadlines?
- H/M patterns are mapped to state trajectories of the system

State update equation: $\xi[k+1] = \Phi(\Delta_p, \Delta_c)\,\xi[k]$



Example

$$\xi[0] = \xi_0$$

H

$$\xi[1] = \Phi(0,0)\,\xi_0$$

M

$$\xi[2] = \Phi(0,1)\xi[1] = \Phi(0,1)\Phi(0,0)\,\xi_0$$

M

$$\xi[3] = \Phi(1,1)\Phi(0,1)\Phi(0,0)\,\xi_0$$

H

$$\xi[4] = \Phi(1,0)\Phi(1,1)\Phi(0,1)\Phi(0,0)\,\xi_0$$

General state trajectory equation: $\xi[k+1] = \Phi_k\Phi_{k-1} \dots \Phi_0\,\xi_0$